Infinite Petri Nets: Part 2, Modeling Triangular, Hexagonal, Hypercube and Hypertorus Structures

Dmitry A. Zaitsev
Vistula University
Warsaw, Poland

Ivan D. Zaitsev
Ershov Institute of Informatics Systems
Novosibirsk, Russia

Tatiana R. Shmeleva
National Academy of Telecommunications
Odessa, Ukraine

A composition and analysis technique was developed for investigation of infinite Petri nets with regular structure introduced for modeling networks, clusters and computing grids that also concerns cellular automata and biological systems. A case study of a hypercube structure composition and analysis is presented; particularities of modeling other structures are discussed: triangular and hexagonal structures on a plane and a hypertorus in a multidimensional space. Parametric description of Petri nets, parametric representation of infinite systems for the calculation of place/transition invariants and solving them in parametric form allow the invariance proof for infinite Petri net models. Complex deadlocks are disclosed and a possibility of the network blocking via ill-intentioned traffic revealed. Prospective directions for future research of infinite Petri nets are formulated and hypotheses advanced.

1. Introduction

Petri nets find wide application in automated manufacture [1, 2], business [3], telecommunications [4], computing [5], cellular automata [6, 7] and systems biology [8]. Classical Petri net theory was developed for finite Petri nets. The first book on the subject was published by Peterson [9], then a series of books appeared on various aspects of Petri net theory and application. A review of them with the summary of the current state of Petri net theory was published by Murata [10]. Recent developments include particular extensions of Petri nets, such as colored Petri nets [11] and their application for business process management [3]. Moreover, specializations of process-resource Petri nets were developed and applied for finding deadlocks in manufacturing systems [1, 2].

The verification of telecommunication protocols involving an unlimited number of devices is a significant scientific problem. The majority of known works study communication processes in pairs of communicating devices [12, 13]. But anomalies may occur that involve an arbitrary number of communicating devices, as shown in the present paper.

The first paper where infinite Petri nets were introduced and applied to solving practical tasks appeared in 2006, as cited in [14]; it solved the problem of Marsan [15] of verification of an Ethernet network with the common bus architecture. Then, as the result of accomplishing a project on a NATO grant, a series of papers was published. Shmeleva applied infinite Petri nets with a tree-like structure to analyze a switched Ethernet network. Then the approach was applied for analysis of square computing grids [16]. Finally, a hypercube structure was analyzed [17]. It was proven that there exists a possibility of blocking a grid or network by ill-intentioned traffic. Recent developments include studying of various edge conditions of grids [18], a dual parametric description of a grid [19] and software generators of grids' models [20] with a given size.

The majority of Petri net examples studied in the literature are rather simplified models of real-life systems and processes. A certain gap is formed between large-scale nets, employed in real-life projects, and illustrative examples, found in articles and monographs. Moreover, various simulating systems, which are counted in dozens and hundreds, have been tested and presented on simplified examples of nets. Thus, there is a definite deficiency of both realistic models and simulating systems that are able to analyze large-scale nets in admissible time. In many cases, detailed models, close to enterprise-class specifications, are a trade secret, and simulating systems, which are able to analyze them, are only available commercially. So certain difficulties arise when developing formal methods of Petri net analysis and the corresponding software, induced by the lack of actual nets. In many cases, the application of random Petri nets does not give the appreciated result, since artificial (industrial-level) systems possess a series of particularities—for instance, they are decomposed into a few functional subnets [14], while random nets are close to indecomposable.

A demand exists for a library of large-scale Petri nets that are either models or specifications of real-life systems, as well as for facilities for automated construction (synthesis) of specific Petri nets with given characteristics: number of vertices, density and localization of arcs, connectivity and so on. Owing to the lack of appropriate actual nets, it is difficult to formulate a set of characteristics and their ranges for various standard models: a parallel program; a computing, production, transport or other system. While there is no objective precondition for solving general tasks of Petri net synthesis, it is possible to construct special generators of Petri nets [20] for separate application domains [19].

Thus, a basis of infinite Petri nets theory was developed that is described in the present paper, as well as the directions for future work to accomplish the development of infinite Petri nets theory. Moreover, a case study of Petri net software generator construction for modeling computing grids is presented for the first time in English. The described technique could be employed in a wide range of Petri net application domains, including automated manufacture, business processes and programming.

The first problem that researchers encountered when verifying networking protocols via Petri nets was the problem of exponential computation complexity of the majority of known analysis methods. To solve this problem, the compositional analysis of Petri nets [14] was offered based on the decomposition of a net into the set of its functional subnets (clans), solving tasks for each clan and then, either simultaneous or sequential composition of functional subnets. Solving a few systems with considerably lesser dimension (size) under the condition of exponential complexity allowed an exponential speedup of computations and verification of known protocols, such as ECMA, BGP, TCP and IOTP, in a reasonable time [14].

The second problem arose when investigating protocols of the Ethernet network with common bus architecture [15]. It was rather simple to construct a model of a single device. The majority of protocols stipulate interaction of two systems, for example, as protocol TCP. Electronic commerce protocol IOTP [21] considers a few interacting systems, but their number is constant: Customer, Merchant, Payment Handler, Delivery Handler and Merchant Customer Care Provider. An Ethernet segment with common bus architecture could contain a priori an unknown number of computers, which is reasonable to not limit in research in spite of definite physical limitations stated in standards.

To solve this problem, infinite Petri nets with regular structure were introduced for the first time as a linear composition of the workstation models [14]. The further progress of research was indicated by the number of dimensions and the structure of devices' connections: tree-like structures for analysis of switched Ethernet; triangular, rectangular, and hexangular grids for analysis of distributed computations, radio and television broadcasting, and cellular communications [19], as well as various edge conditions [18], which define a connection of a model with its environment. In the most general form, results were obtained for a hypercube with an arbitrary size and an arbitrary number of dimensions [17].

The practical value of the obtained result on deadlocks disclosure within the mentioned structures and their classification consists in revealing possibilities of network blocking via ill-intentioned traffic of special form [16, 17, 22].

The present paper generalizes models studied in [23] of multidimensional spaces and structures of various shapes: triangular, hexagonal; it also specifies a technique of software generators [20] for models with regular structure.

2. Plain Grids of Different Shapes and Trees

The communication device model [23] (represented with equation (1)), under various values of the parameter *np*, could be involved in composition of models of triangular and hexangular grids, trees and networks of an arbitrary structure. From the models of devices in the forms of a triangle and hexagon (Figure 1), models of triangular and hexangular grids are composed (Figure 2).

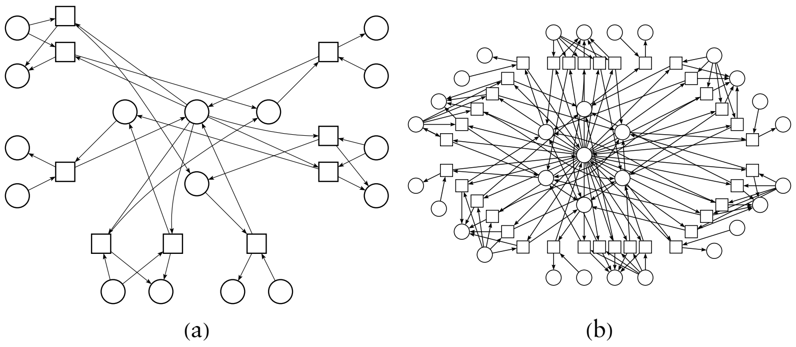

Figure 1. Forms of devices' models: (a) triangle; (b) hexagon.

Triangular grids correspond with the forms of the coverage area in modern systems of radio and television broadcasting. Hexangular grids find their wide application when modeling cellular communication systems and networks.

Application of infinite models, constructed on the base of the device model of form [23] (equation (1)), historically started from tree-like structures, which are the model of modern switched networks of the technology Ethernet. In Figure 3, a binary tree is represented, which is convenient for theoretical research; modern Ethernet switches contain, as a rule, 8, 16, 24 and more ports.

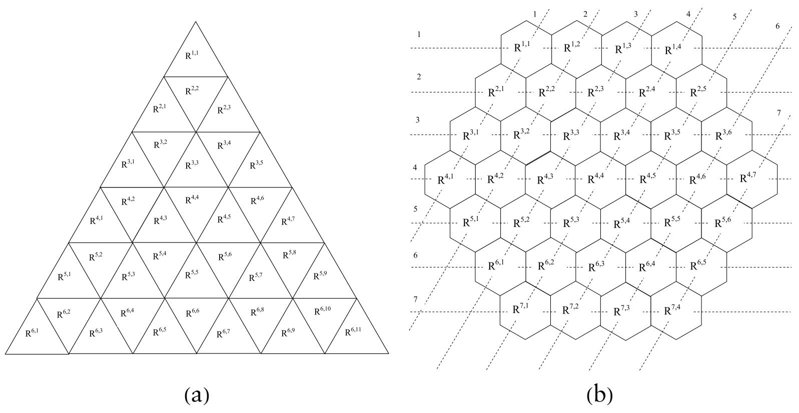

Figure 2. Composition of grids: (a) triangular; (b) hexangular.

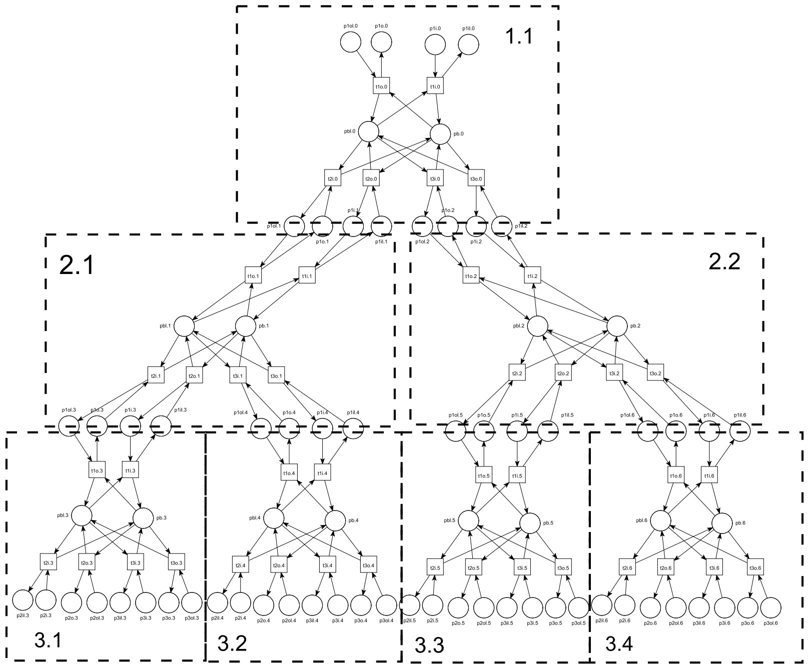

Figure 3. Composition of tree-like structures.

In the conducted research, a single basic fragment was employed for composition of either trees or grids. A prospective direction is working with a few basic fragments (a finite set) and arbitrary schemes of their connection.

3. Cubic Grid

All early studied models were constructed on a plane. However, for solving boundary value problems in space, structures of cubes as a pattern of processor (computers) connection is widely applied.

To construct the corresponding model, each device is represented by a cube of unit size with ports situated on its facets (Figure 4(a)). For generalization on an arbitrary number of dimensions, when constructing the cube model, a new system of the ports' enumeration was chosen. Recall that within the rectangular grid model, the ports were enumerated clockwise starting from the upper side.

Within a cube, the device ports (facets) have two indices: the first is the number of the dimension, and the second is the number of the direction. The number of the dimension corresponds to the coordinate axis along which (perpendicular to which) opposite facets are situated; the direction to the beginning of coordinates is designated as 1 and the direction to infinity is designated as 2. In Figure 4(b), a graphical representation of the parametric description of the device model in equation (1) from [23] is shown, taking into consideration the system of the ports' notation.

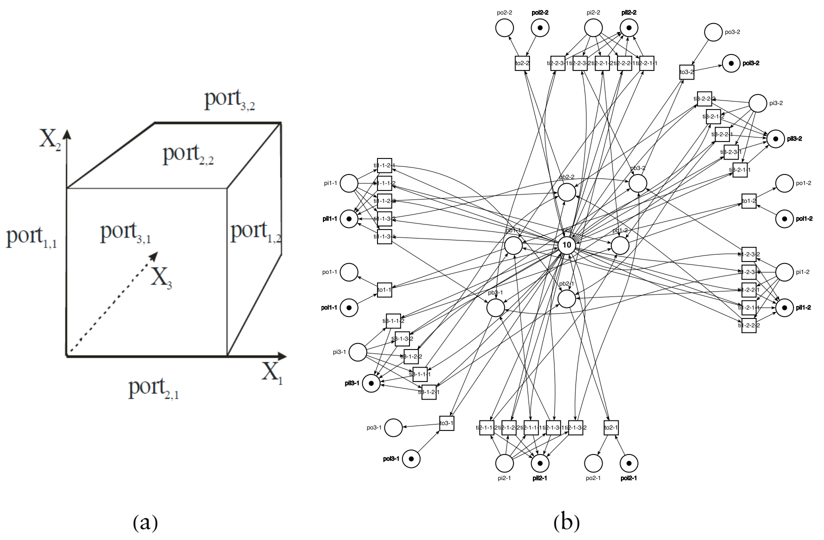

Figure 4. Model of a cube device.

A scheme of the cubic grid model composition is represented in Figure 5. Neighboring devices, the same as for plane grids, are connected via fusion (union) of the ports' contact places.

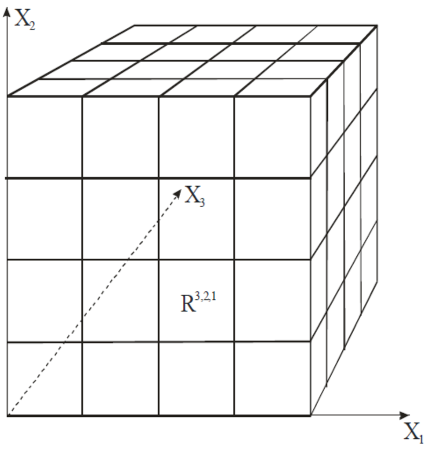

Figure 5. Composition of a cube.

4. Hypercube Grid Structure

The model of a device in the form of a cube (Figure 4(b)) does not possess a distinction of obviousness. That is why, in the further constructions, a preference is given to the parametric description. The previous constructions were implemented with the increment of the number of dimensions: 1–line, 2–plane, 3–space; let us generalize the results on an arbitrary number of dimensions.

4.1 Parametric Model of Hypercube Communication Device

Let us consider a d-dimensional space, where $d = 1, 2, \ldots$. Each communication device is represented by a hypercube of size 1 in d-dimensional space. The communication structure composed by connected communication devices constitutes a hypercube of size k, where $k = 1, 2, \ldots$. So the total number of devices is $N_{\text{dev}} = k^d$. Each device $R^{i_1, i_2, \ldots, i_d}$ has its index $(i_1, i_2, \ldots, i_d)$, where $i_u = \overline{1, k}$, $u = \overline{1, d}$. The model of the hypercube communication structure is denoted as $H_{d,k}$. Next we describe the Petri net model of a device and then the composition of a communication structure model via connections of a device with its neighbors.

The model of a hypercube device is denoted as $H_{d,1}$ (number of dimensions equals d, size of the structure equals 1). On each facet of a hypercube device $R^{i_1, i_2, \ldots, i_d}$ in d-dimensional space a port is situated. So each device has $N_{\text{port}} = 2 \cdot d$ ports; two ports for each dimension are situated at the opposite facets of the hypercube. To denote opposite facets for a dimension j ($j = \overline{1, d}$), the number of the direction is used. The direction is denoted by the variable n; the value $n = 1$ is used for the direction to zero in the corresponding dimension and the value $n = 2$ is used for the opposite direction to infinity. So the ports may be denoted with the following indices: $\mathrm{port}_{j,n}^{i_1,i_2,\ldots,i_d}$, where $i_u = \overline{1, k}$, $u = \overline{1, d}$, $j = \overline{1, d}$, $n = 1, 2$. Each port is represented by the two channels (input, output); each channel is represented by a pair of places: one place for the packets buffer, the other for the buffer capacity. So each port of the device $R^{i_1,i_2,\ldots,i_d}$ is represented by the four following contact places:

$pi_{j,n}^{i_1,i_2,\ldots i_d}$	input buffer of packets;
$pil_{j,n}^{i_1,i_2,\ldots i_d}$	capacity of input buffer (equals 1);
$po_{j,n}^{i_1,i_2,\ldots i_d}$	output buffer of packets;
$pol_{j,n}^{i_1,i_2,\ldots i_d}$	capacity of output buffer (equals 1).

The inside of the device contains $N_{\mathrm{port}} + 1$ following places. The packets redirected to the port $\mathrm{port}_{j',n'}^{i_1,i_2,\ldots,i_d}$ are stored in the corresponding place $pb_{j',n'}^{i_1,i_2,\ldots i_d}$, and one place $pbl^{i_1,i_2,\ldots i_d}$ contains the capacity of the internal buffer, where $j' = \overline{1, d}$, $n' = 1, 2$. Notice that the internal buffer is represented by the set of places $pb_{j',n'}^{i_1,i_2,\ldots i_d}$ (one place for each port) to distinguish the number of the destination port given by indices j', n'.

The transitions of the device $R^{i_1,i_2,\ldots,i_d}$ provide the redirection of the input packets from an input port buffer place $pi_{j,n}^{i_1,i_2,\ldots i_d}$ into one of the internal buffer places $pb_{j',n'}^{i_1,i_2,\ldots i_d}$, $j' \neq j$, $n' \neq n$ and then the transmission of the packets from the internal buffer place $pb_{j',n'}^{i_1,i_2,\ldots i_d}$ to the output buffer of the target port $po_{j',n'}^{i_1,i_2,\ldots i_d}$. Moreover, the limitations of the buffers' capacities should be taken into consideration: check and decrease the buffer size at putting the packet into the buffer; increase the buffer size at getting the packet from the buffer. So each port $\mathrm{port}_{j,n}^{i_1,i_2,\ldots,i_d}$ of the device $R^{i_1,i_2,\ldots,i_d}$ is supplied by $N_{\mathrm{port}} = (N_{\mathrm{port}} - 1) + 1$ following transitions:

1. one transition for the output channel $to_{j,n}^{i_1,i_2,\ldots i_d}$ with the input arcs from places $pb_{j,n}^{i_1,i_2,\ldots i_d}$, $pol_{j,n}^{i_1,i_2,\ldots i_d}$ and the output arcs to places $po_{j,n}^{i_1,i_2,\ldots i_d}$, $pbl^{i_1,i_2,\ldots i_d}$

2. $N_{\text{port}} - 1$ transitions $ti_{j,n,j',n'}^{i_1,i_2,\ldots i_d}$, $j' = \overline{1,d}$, $n' = 1, 2$, $j' \neq j$, $n' \neq n$ for the input channel with the input arcs from places $pi_{j,n}^{i_1,i_2,\ldots i_d}$, $pbl^{i_1,i_2,\ldots i_d}$ and the output arcs to places $pb_{j',n'}^{i_1,i_2,\ldots i_d}$, $pil_{j,n}^{i_1,i_2,\ldots i_d}$

The formal parametric description of the net $H_{d,1}$ is the following:

$$\left(\begin{pmatrix} \left(ti_{j,n,j',n'} : pi_{j,n}, pbl \to pb_{j',n'}, pil_{j,n}\right); \\ j' = \overline{1,d}, n' = 1, 2, j' \neq j, n' \neq n; \\ \left(to_{j,n} : pb_{j,n}, pol_{j,n} \to po_{j,n}, pbl\right) \end{pmatrix}, j = \overline{1,d}, n = \overline{1,2}\right). \tag{1}$$

If the net $H_{d,1}$ is considered as a model of the device $R^{i_1,i_2,\ldots,i_d}$ in the hypercube structure, the upper indices of its hypercube cell should be added:

$$\left(\begin{pmatrix} \left(ti_{j,n,j',n'}^{i_1,i_2,\ldots i_d} : pi_{j,n}^{i_1,i_2,\ldots i_d}, pbl^{i_1,i_2,\ldots i_d} \to \right. \\ \left. \quad \mathrm{pb}_{j',n'}^{i_1,i_2,\ldots i_d}, pil_{j,n}^{i_1,i_2,\ldots i_d}\right), \\ j' = \overline{1,d}, n' = 1, 2, j' \neq j, n' \neq n; \\ \left(to_{j,n}^{i_1,i_2,\ldots i_d} : pb_{j,n}^{i_1,i_2,\ldots i_d}, pol_{j,n}^{i_1,i_2,\ldots i_d} \to \right. \\ \left. \quad \mathrm{po}_{j,n}^{i_1,i_2,\ldots i_d}, pbl^{i_1,i_2,\ldots i_d}\right) \end{pmatrix}, j = \overline{1,d}, n = \overline{1,2}\right).$$

The net represented by equation (1) is called a parametric Petri net because its description has the parameter d for the calculation of the elements' indices. The size of the net $H_{d,1}$ is unlimited and represented by the parameter d. The parametric model $H_{d,1}$ is illustrated by the example for the concrete number of dimensions $d = 3$ shown in Figure 4. It is rather difficult to visualize models for larger numbers of dimensions.

4.2 P-Invariants of Hypercube Communication Device Model

Using the parametric description in equation (1) of the communication device model $H_{d,1}$ given in the previous section, the following system was constructed for the calculation of p-invariants:

$$\begin{cases} to_{j,n} : xpb_{j,n} + xpol_{j,n} = xpo_{j,n} + xpbl, \\ ti_{j,n,j',n'} : xpi_{j,n} + xpbl = xpb_{j',n'} + xpil_{j,n}, \\ j = \overline{1,d}, n = 1, 2, j' = \overline{1,d}, n' = 1, 2, j' \neq j, n' \neq n. \end{cases} \tag{2}$$

Notice that the system of equation (2) has a parametric form; its parameter is the number of dimensions d. The system was constructed directly on the description of equation (1), using the usual rule that each equation corresponds to transition and contains sums for its input and output arcs, which are equal. Sums should be calculated using the multiplicities of arcs, but all the arcs of equation (1) have the multiplicity equaling to unit. The total number of the system in equation (2) equations is $N^{t}_{d,1} = 4 \cdot d^2$. The total number of system variables in equation (2) is $N^{p}_{d,1} = 10 \cdot d + 1$.

To study p-invariants of the model for any number of dimensions, the system of equation (2) should be solved in the parametric form. The obtained parametric solution of the system in equation (2) has the following form:

$$\begin{pmatrix} \left(pi_{j,n}, pil_{j,n}\right), j = \overline{1, d}, n = 1, 2; \\ \left(po_{j,n}, pol_{j,n}\right), j = \overline{1, d}, n = 1, 2; \\ \left(pbl, \left(pb_{j,n}, j = \overline{1, d}, n = 1, 2\right)\right) \\ \left(\left(pb_{j,n}, pi_{j,n}, po_{j,n}\right), j = \overline{1, d}, n = 1, 2\right) \\ \left(pbl, \left(\left(pil_{j,n}, pol_{j,n}\right), j = \overline{1, d}, n = 1, 2\right)\right) \end{pmatrix}. \qquad (3)$$

The way solutions are described here is common enough for sparse vectors and especially for the Petri net theory. Only nonzero components are mentioned by the names of the corresponding places. The nonzero multiplier 1 is omitted; in case it is not the unit, the notation $p * x$ is used, where x is the value of the invariant for place p. Such notation is adopted in the Tina software [24], which was used for obtaining the Petri net figures in this paper. A line of the matrix in equation (3) gives us a set of lines according to the used indices i, j, n, except the last two lines, which contain a variable number of components given by indices.

A heuristic algorithm was employed for the construction of the matrix in equation (3), but with Lemma 1 the proof is presented that equation (3) is a solution of equation (2). The fact that equation (3) is the basis solution is not required for the conclusion about the p-invariance of $H_{d,1}$. The total number of solutions in the matrix in equation (3) is $N^{\text{pinv}}_{d,1} = 4d + 3$.

Lemma 1. Each line of the matrix in equation (3) is a solution of the system in equation (2).

Proof. Let us substitute each parametric line of equation (3) into each parametric equation of the system in equation (2). It gives us the correct statement. At the substitution, the different names of indices are chosen. For instance, let us substitute the fourth line of equation (3)

$$\left((pb_{l,m}, pi_{l,m}, po_{l,m}), l = \overline{1, d}, m = 1, 2\right)$$

into the second equation of (2)

$$\begin{gathered} xpi_{j,n} + xpbl = xpb_{j',n'} + xpil_{j,n}, j = \overline{1, d}, \\ n = 1, 2, j' = \overline{1, d}, n' = 1, 2, j' \neq j, n' \neq n. \end{gathered}$$

For each concrete equation given by valid tuple (j, n, j', n'), the solution contains $pi_{j,n}$ at $l = j$, $m = n$ and $pb_{j',n'}$ at $l = j'$, $m = n'$; moreover, the other variables of the equation $xpbl, xpil_{j',n'}$ are not mentioned in the solution. So we obtain $1 + 0 = 1 + 0$, reducing further to $1 = 1$ for each equation.

The first two solutions of equation (3) are slightly different: they represent a series of lines given by their indices. Let us substitute the first parametric line of equation (3)

$$(pi_{l,m}, pil_{l,m}), l = \overline{1, d}, m = 1, 2;$$

into the second parametric equation of equation (2)

$$\begin{gathered} xpi_{j,n} + xpbl = xpb_{j',n'} + xpil_{j,n}, j = \overline{1, d}, \\ n = 1, 2, j' = \overline{1, d}, n' = 1, 2, j' \neq j, n' \neq n. \end{gathered}$$

We obtain:

when $l \neq j$ or $m \neq n$: $0 + 0 = 0 + 0$, reducing further to $0 = 0$;

when $l = j$ and $m = n$: $1 + 0 = 0 + 1$, reducing further to $1 = 1$.

In the same way, all the 5×2 combinations are checked. □

Theorem 1. The net $H_{d,1}$ is a p-invariant Petri net for an arbitrary natural number d.

Proof. Let us consider the sum of the fourth and the fifth lines of the matrix in equation (3), which represents the solutions of the system in equation (2) according to Lemma 5:

$$\left((pb_{j,n}, pi_{j,n}, po_{j,n}), j = \overline{1, d}, n = 1, 2\right)$$

plus

$$\left(pbl, \left((pil_{j,n}, pol_{j,n}), j = \overline{1, d}, n = 1, 2\right)\right)$$

equals

$$\begin{gathered} \left(pbl, \left((pil_{j,n}, pol_{j,n}, pb_{j,n}, pi_{j,n}, po_{j,n}),\right.\right. \\ \left.\left. j = \overline{1, d}, n = 1, 2\right)\right). \end{gathered} \tag{4}$$

As all the $N^p_{d,1} = 10 \cdot d + 1$ places are mentioned in this invariant, the net $H_{d,1}$ is a p-invariant Petri net for an arbitrary natural number d. Moreover, as each component of equation (4) is equal to 1, the net $H_{d,1}$ is a safe and bounded Petri net for an arbitrary natural number d. □

4.3 Composition of Hypercube Model

The connections of communication devices in the hypercube are provided by the fusion (union) of the corresponding contact places of neighbor devices.

Let us consider an internal communication device $R^{i_1,\ldots i_j,\ldots,i_d}$, $i_u = \overline{2, k-1}$, $u = \overline{1, d}$, $j = \overline{1, d}$:

1. Places of $\text{port}^{i_1,\ldots,i_j,\ldots,i_d}_{j,1}$ are fused with the corresponding places of $\text{port}^{i_1,\ldots,i_j-1,\ldots,i_d}_{j,2}$, device $R^{i_1,\ldots i_j-1,\ldots,i_d}$ in such a way that place $po^{i_1,\ldots,i_j,\ldots i_d}_{j,1}$ is fused with $pi^{i_1,\ldots,i_j-1,\ldots i_d}_{j,2}$, place $pol^{i_1,\ldots,i_j,\ldots i_d}_{j,1}$—with $pil^{i_1,\ldots,i_j-1,\ldots i_d}_{j,2}$, place $pi^{i_1,\ldots,i_j,\ldots i_d}_{j,1}$—with $po^{i_1,\ldots,i_j-1,\ldots i_d}_{j,2}$, place $pil^{i_1,\ldots,i_j,\ldots i_d}_{j,1}$—with $pol^{i_1,\ldots,i_j-1,\ldots i_d}_{j,2}$.
2. Places of $\text{port}^{i_1,\ldots,i_j,\ldots,i_d}_{j,2}$ are fused with the corresponding places of $\text{port}^{i_1,\ldots,i_j+1,\ldots,i_d}_{j,1}$, device $R^{i_1,\ldots i_j+1,\ldots,i_d}$ in such a way that place $po^{i_1,\ldots,i_j,\ldots i_d}_{j,2}$ is fused with $pi^{i_1,\ldots,i_j+1,\ldots i_d}_{j,1}$, place $pol^{i_1,\ldots,i_j,\ldots i_d}_{j,2}$—with $pil^{i_1,\ldots,i_j+1,\ldots i_d}_{j,1}$, place $pi^{i_1,\ldots,i_j,\ldots i_d}_{j,2}$—with $po^{i_1,\ldots,i_j+1,\ldots i_d}_{j,1}$, place $pil^{i_1,\ldots,i_j,\ldots i_d}_{j,2}$—with $pol^{i_1,\ldots,i_j+1,\ldots i_d}_{j,1}$.

To avoid duplication, the names of the places for the zero direction ports $n = 1$ will be considered with respect to the current device, and for the infinity direction ports $n = 2$ with respect to the neighbor devices and their zero direction ports $n = 1$. So the names of the fusion places have only the indices of the zero direction ports $n = 1$. Moreover, to simplify further notations, the places with the indices of the infinity direction ports $n = 2$ on the facets (borders) of the communication hypercube are named with respect to nonexistent devices with the indices equal to $k + 1$. So the names of ports with the indices of the infinity direction $n = 2$ do not appear in the hypercube. The communication hypercube structure described is denoted as $H_{d,k}$. An example of $H_{d,k}$ for $d = 3$, $k = 4$ is represented in Figure 5.

The formal description of $H_{d,k}$ composition is given with the following:

$$\left(\begin{pmatrix} pi_{j,1}^{i_1,\ldots,i_j+1,\ldots i_d} := \\ \quad \mathrm{po}_{j,2}^{i_1,\ldots,i_j,\ldots i_d} \bigcup pi_{j,1}^{i_1,\ldots,i_j+1,\ldots i_d} \\ pil_{j,1}^{i_1,\ldots,i_j+1,\ldots i_d} := \\ \quad \mathrm{pol}_{j,2}^{i_1,\ldots,i_j,\ldots i_d} \bigcup pil_{j,1}^{i_1,\ldots,i_j+1,\ldots i_d} \\ po_{j,1}^{i_1,\ldots,i_j+1,\ldots i_d} := \\ \quad \mathrm{pi}_{j,2}^{i_1,\ldots,i_j,\ldots i_d} \bigcup po_{j,1}^{i_1,\ldots,i_j+1,\ldots i_d} \\ pol_{j,1}^{i_1,\ldots,i_j+1,\ldots i_d} := \\ \quad \mathrm{pil}_{j,2}^{i_1,\ldots,i_j,\ldots i_d} \bigcup pol_{j,1}^{i_1,\ldots,i_j+1,\ldots i_d} \end{pmatrix}, i_u = \overline{1, k-1}, u = \overline{1, d}, j = \overline{1, d}\right),$$

$$\left(\begin{pmatrix} pi_{j,1}^{i_1,\ldots,i_j+1,\ldots i_d} := po_{j,2}^{i_1,\ldots,i_j,\ldots i_d} \\ pil_{j,1}^{i_1,\ldots,i_j+1,\ldots i_d} := pol_{j,2}^{i_1,\ldots,i_j,\ldots i_d} \\ po_{j,1}^{i_1,\ldots,i_j+1,\ldots i_d} := pi_{j,2}^{i_1,\ldots,i_j,\ldots i_d} \\ pol_{j,1}^{i_1,\ldots,i_j+1,\ldots i_d} := pil_{j,2}^{i_1,\ldots,i_j,\ldots i_d} \end{pmatrix}, \begin{matrix} i_u = \overline{1, k-1}, u = \overline{1, d}, \\ j = \overline{1, d}, u \neq j, i_j = k \end{matrix}\right).$$

The union sign $\bigcup$ denotes the fusion of places; the left column gives new names of places.

4.4 P-Invariants of Hypercube Model

Using the abstract description of the communication hypercube model $H_{d,k}$ given in the previous section, the following system is constructed for the calculation of p-invariants:

$$\begin{cases} to_{j,1}^{i_1,\ldots,i_d} : xpb_{j,1}^{i_1,\ldots,i_d} + xpol_{j,1}^{i_1,\ldots,i_d} = xpo_{j,1}^{i_1,\ldots,i_d} + xpbl^{i_1,\ldots,i_d}, \\ ti_{j,1,j',n'}^{i_1,\ldots,i_d} : xpi_{j,1}^{i_1,\ldots,i_d} + xpbl^{i_1,\ldots,i_d} = xpb_{j',n'}^{i_1,\ldots,i_d} + xpil_{j,1}^{i_1,\ldots,i_d}, \\ to_{j,2}^{i_1,\ldots,i_j,\ldots,i_d} : xpb_{j,2}^{i_1,\ldots,i_j,\ldots,i_d} + xpil_{j,1}^{i_1,\ldots,i_j+1,\ldots,i_d} = \\ \qquad xpi_{j,1}^{i_1,\ldots,i_j+1,\ldots,i_d} + xpbl^{i_1,\ldots,i_j,\ldots,i_d}, \\ ti_{j,2,j',n'}^{i_1,\ldots,i_j,\ldots,i_d} : xpo_{j,1}^{i_1,\ldots,i_j+1,\ldots,i_d} + xpbl^{i_1,\ldots,i_j,\ldots,i_d} = \\ \qquad xpb_{j',n'}^{i_1,\ldots,i_j,\ldots,i_d} + xpol_{j,1}^{i_1,\ldots,i_j+1,\ldots,i_d}, \\ \quad j = \overline{1, d}, j' = \overline{1, d}, n' = 1, \\ \quad 2, j' \neq j, n' \neq n, i_u = \overline{1, k}, u = \overline{1, d}. \end{cases} \tag{5}$$

The total number of system equations in equation (5) is $N_{d,k}^t = 4 \cdot d^2 \cdot k^d$. The total number of system variables in equation (5) is $N^{p}_{d,k} = \left(6 \cdot d + 1\right) \cdot k^{d} + 4 \cdot d \cdot k^{d-1}$. The obtained parametric solution has the following form:

$$\left(\begin{array}{l}
\left(pi_{j,1}^{i_1,\ldots,i_j,\ldots,i_d}, pil_{j,1}^{i_1,\ldots,i_j,\ldots,i_d}\right),\ j = \overline{1,d},\\
\left(i_u = \overline{1,k},\ u = \overline{1,d},\ u \neq j\right),\ i_j = \overline{1,k+1};\\
\left(po_{j,1}^{i_1,\ldots,i_j,\ldots,i_d}, pol_{j,1}^{i_1,\ldots i_j,\ldots,i_d}\right),\ j = \overline{1,d},\\
\left(i_u = \overline{1,k},\ u = \overline{1,d},\ u \neq j\right),\ i_j = \overline{1,k+1};\\
\left(pbl^{i_1,\ldots,i_d}, \left(pb_{j,n}^{i_1,\ldots,i_d}, j = \overline{1,d},\ n = 1,2\right)\right),\\
i_u = \overline{1,k},\ u = \overline{1,d};\\
\left(\left(pb_{j,n}^{i_1,\ldots,i_d}, n = 1,2, j = \overline{1,d}, i_u = \overline{1,k}, u = \overline{1,d}\right),\right.\\
\quad \left(\left(pi_{j,1}^{i_1,\ldots,i_d}, po_{j,1}^{i_1,\ldots,i_d}\right), j = \overline{1,d}, i_u = \overline{1,k}, u = \overline{1,d}\right),\\
\left(\left(pi_{j,1}^{i_1,\ldots,i_j,\ldots,i_d}, po_{j,1}^{i_1,\ldots,i_j,\ldots,i_d}\right), j = \overline{1,d},\right.\\
\qquad \left.\left. i_u = \overline{1,k},\ u = \overline{1,d},\ u \neq j,\ i_j = k+1\right)\right)\\
\left(\left(pbl^{i_1,\ldots,i_d}, \left(\left(pil_{j,1}^{i_1,\ldots,i_d}, pol_{j,1}^{i_1,\ldots,i_d}\right), j = \overline{1,d}\right),\right.\right.\\
\qquad \left. i_u = \overline{1,k},\ u = \overline{1,d}\right),\\
\left(\left(pil_{j,1}^{i_1,\ldots,i_j,\ldots,i_d}, pol_{j,1}^{i_1,\ldots,i_j,\ldots,i_d}\right), j = \overline{1,d},\right.\\
\qquad \left.\left. i_u = \overline{1,k},\ u = \overline{1,d},\ u \neq j,\ i_j = k+1\right)\right)
\end{array}\right). \tag{6}$$

The total number of solutions is

$$N_{d,k} = \left(1 + 2 \cdot d\right) \cdot k^{d} + 2 \cdot d \cdot k^{d-1} + 2.$$

Lemma 2. Each line of the matrix in equation (6) is a solution of the system in equation (5).

Theorem 2. The net $H_{d,k}$ is a p-invariant Petri net for arbitrary natural numbers d, k.

The proofs of Lemma 2 and Theorem 2 were done in the same way as for the net $H_{d,1}$.

4.5 Adding Models of Terminal Devices

The communication devices are attached to each other, constituting a communication structure, but they are created only for packet transmission among the terminal devices: workstations and servers. In this paper, the client-server technique of interaction is not studied, so the types of terminal devices are not distinguished. An abstract terminal device provides at least two basic functions: send packet and receive packet. These basic functions are implemented in the model represented in Figure 7 of [23].

The model contains an internal buffer of the packets *qb*; transition *si* models the receiving of the packets, while transition *so* models the sending. The model keeps the balance of input and output packets; the limitation of buffer *qbl* size is not considered.

The terminal devices shown in Figure 7 of [23] are attached to the border ports of the hypercube structure; the corresponding model is denoted as $HT_{d,k}$. The terminal device in hypercube structure is denoted as $Ai_1, \ldots i_j, \ldots, i_d$, where $i_u = \overline{1, k}$, $u = \overline{1, d}$, $j = \overline{1, d}$, $u \neq j$, $i_j \in \{1, k\}$, and attached to the communication device $R^{i_1, \ldots i_j, \ldots, i_d}$. The formal description of $HT_{d,k}$ composition using $HT_{d,k}$, $Ai_1, \ldots i_j, \ldots, i_d$ is given with the following:

$$\left(\left(\begin{array}{l} pi_{j,1}^{i_1, \ldots, i_j, \ldots i_d} := \\ \quad qoi_1, \ldots, i_j, \ldots i_d \cup pi_{j,1}^{i_1, \ldots, i_j, \ldots i_d} \\ pil_{j,1}^{i_1, \ldots, i_j, \ldots i_d} := \\ \quad qoli_1, \ldots, i_j, \ldots i_d \cup pil_{j,1}^{i_1, \ldots, i_j, \ldots i_d} \\ po_{j,1}^{i_1, \ldots, i_j, \ldots i_d} := \\ \quad qii_1, \ldots, i_j, \ldots i_d \cup po_{j,1}^{i_1, \ldots, i_j, \ldots i_d} \\ pol_{j,1}^{i_1, \ldots, i_j, \ldots i_d} := \\ \quad qili_1, \ldots, i_j, \ldots i_d \cup pol_{j,1}^{i_1, \ldots, i_j, \ldots i_d} \end{array}\right), \begin{array}{l} i_u = \overline{1, k}, u = \overline{1, d}, \\ j = \overline{1, d}, u \neq j, i_j = 1 \end{array}\right),$$

$$\left(\left(\begin{array}{l} pi_{j,1}^{i_1, \ldots, i_j+1, \ldots i_d} := \\ \quad qii_1, \ldots, i_j, \ldots i_d \cup pi_{j,1}^{i_1, \ldots, i_j+1, \ldots i_d} \\ pil_{j,1}^{i_1, \ldots, i_j+1, \ldots i_d} := \\ \quad qili_1, \ldots, i_j, \ldots i_d \cup pil_{j,1}^{i_1, \ldots, i_j+1, \ldots i_d} \\ po_{j,1}^{i_1, \ldots, i_j+1, \ldots i_d} := \\ \quad qoi_1, \ldots, i_j, \ldots i_d \cup po_{j,1}^{i_1, \ldots, i_j+1, \ldots i_d} \\ pol_{j,1}^{i_1, \ldots, i_j+1, \ldots i_d} := \\ \quad qoli_1, \ldots, i_j, \ldots i_d \cup pol_{j,1}^{i_1, \ldots, i_j+1, \ldots i_d} \end{array}\right), \begin{array}{l} i_u = \overline{1, k}, u = \overline{1, d}, \\ j = \overline{1, d}, u \neq j, i_j = k \end{array}\right).$$

In the same way as in [23], it was proven that $HT_{d,k}$ is a p-invariant Petri net for any given natural numbers d and k.

4.6 T-Invariants and Deadlocks of Hypercube

The same approach is applied for calculating the t-invariants. The only difference is that for t-invariants, each equation corresponds to place and variables correspond to transitions. It is an advantage that the Petri net $H_{d,k}$ is not t-invariant, but it is quite trivial because the modeled system is open, since the terminal devices are not attached. It was proven that the model of closed system with attached terminal devices $HT_{d,k}$ is a t-invariant Petri net for arbitrary natural numbers d, k. But the consistency of the model does not imply its liveness.

Each pair of neighbor communication devices can fall into a local deadlock, for instance, when the device $R^{i_1,\ldots,i_j,\ldots,i_d}$ gets l packets directed to the device $R^{i_1,\ldots,i_j+1,\ldots,i_d}$ and the device $R^{i_1,\ldots,i_j+1,\ldots,i_d}$ gets l packets directed to the device $R^{i_1,\ldots,i_j,\ldots,i_d}$ and, moreover, the input and output buffers of their common port are occupied with the packets, where l is the limitation of the internal buffer size (marking of places $pbl^{i_1,\ldots,i_j,\ldots,i_d}$, $pbl^{i_1,\ldots,i_j+1,\ldots,i_d}$). Such a situation constitutes the t-dead marking for the transitions of both devices, while other transitions of the net $HT_{d,k}$ are potentially live.

But the structure of all the possible deadlocks is more sophisticated. We show that deadlocks occur either in cycles (chains) of blockings involving a few communicating devices (where the pair is a particular case) or because of isolation with surrounding deadlocks.

For the description of complex deadlocks of the net $HT_{d,k}$, the graph $GH_{d,k}$ of connections is constructed. In the graph $GH_{d,k}$, each node corresponds to a communication device and has arcs directed to its neighbors. An example of internal node connections for $GH_{3,k}$ is shown in Figure 6. An arc with two arrows denotes two arcs of opposite directions.

A directed simple cycle in the graph $GH_{d,k}$ represents a deadlock of the communication hypercube $HT_{d,k}$. In a deadlock cycle, each arc connecting a pair of neighbor devices $R^{i_1,\ldots,i_u,\ldots,i_d}$, $R^{i_1,\ldots,i'_u,\ldots,i_d}$, $\left|i_u - i'_u\right| = 1$ means that $R^{i_1,\ldots,i_u,\ldots,i_d}$ blocks itself if and only if it got l packets directed to $R^{i_1,\ldots,i'_u,\ldots,i_d}$, its output buffer of the port connecting $R^{i_1,\ldots,i_u,\ldots,i_d}$ with $R^{i_1,\ldots,i'_u,\ldots,i_d}$ contains a packet, and the device $R^{i_1,\ldots,i'_u,\ldots,i_d}$ is blocked also. When the cycle ends, the last device blocks itself because the first device is blocked and cannot receive packets.

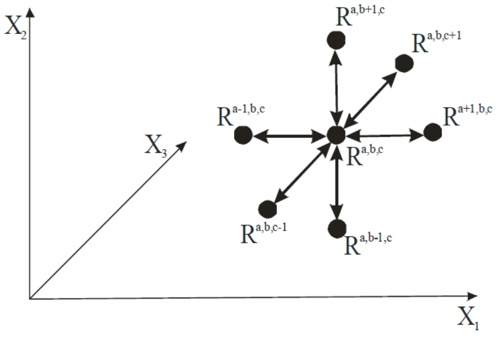

Figure 6. Node connections of the graph $GH_{3,k}$.

Let us prove that all the transitions of a blocked device $R^{i_1,\ldots,i_u,\ldots,i_d}$ are dead. For distinctness we denote

$$r = \begin{cases} 1, \; i_u - i'_u = -1, \\ 2, \; i_u - i'_u = 1. \end{cases}$$

All the transitions $ti_{j,n,j',n'}^{i_1,\ldots,i_d}$ are dead because marking of their input place $pbl^{i_1,\ldots,i_d}$ equals zero, so the device cannot receive packets. All the transitions $to_{j,n}^{i_1,\ldots,i_d}$, $n \neq r$ are dead because each of their input places $pb_{j,n}^{i_1,\ldots,i_d}$ has zero marking. The transition $tou_{u,r}^{i_1,\ldots,i_d}$ is dead because marking of its input place $pol_{u,r}^{i_1,\ldots,i_d}$ is zero. So the device cannot send packets. Notice that marking of $pol_{u,r}^{i_1,\ldots,i_d}$ cannot be changed because $R^{i_1,\ldots,i'_u,\ldots,i_d}$ is blocked.

Nonsimple cycles of $GH_{d,k}$ represent deadlocks also. For instance, if device $R^{i_1,\ldots,\ldots,i_u,\ldots,i_v,\ldots,i_d}$ belongs to two simple cycles and it got two output arcs directed to $R^{i_1,\ldots,\ldots,i'_u,\ldots,i_v,\ldots,i_d}$ and $R^{i_1,\ldots,\ldots,i_u,\ldots,i'_v,\ldots,i_d}$, it means that $R^{i_1,\ldots,\ldots,i_u,\ldots,i_v,\ldots,i_d}$ blocks itself and $R^{i_1,\ldots,\ldots,i'_u,\ldots,i_v,\ldots,i_d}$, $R^{i_1,\ldots,\ldots,i_u,\ldots,i'_v,\ldots,i_d}$ are blocked also. In this case, $R^{i_1,\ldots,\ldots,i_u,\ldots,i_v,\ldots,i_d}$ blocks itself, having a_u packets directed to $R^{i_1,\ldots,\ldots,i'_u,\ldots,i_v,\ldots,i_d}$ and a_v packets directed to $R^{i_1,\ldots,\ldots,i_u,\ldots,i'_v,\ldots,i_d}$, where $a_u + a_v = l$, and moreover, each corresponding output port buffer of $R^{i_1,\ldots,\ldots,i_u,\ldots,i_v,\ldots,i_d}$ contains a packet when $a_u > 0$ ($a_v > 0$). Inductive reasoning gives the proof for d simple cycles.

The other kind of deadlock is induced by the isolation of a device by deadlocks containing all its neighbor devices. It can be done with one simple cycle as well. For instance, in $GH_{3,k}$, the following cycle R^{i_1-1,i_2,i_3}, R^{i_1-1,i_2+1,i_3}, R^{i_1,i_2+1,i_3}, R^{i_1,i_2+1,i_3+1}, R^{i_1,i_2,i_3+1},

R^{i_1+1,i_2,i_3+1}, R^{i_1+1,i_2,i_3}, R^{i_1+1,i_2-1,i_3}, R^{i_1,i_2-1,i_3}, R^{i_1,i_2-1,i_3-1}, R^{i_1,i_2,i_3-1}, R^{i_1-1,i_2,i_3-1}, R^{i_1-1,i_2,i_3} contains all the neighbors of R^{i_1,i_2,i_3} (R^{i_1-1,i_2,i_3}, R^{i_1,i_2+1,i_3}, R^{i_1,i_2,i_3+1}, R^{i_1+1,i_2,i_3}, R^{i_1,i_2-1,i_3}, R^{i_1,i_2,i_3-1}), so the device R^{i_1,i_2,i_3} is blocked because of isolation. The isolation of a node can be generalized on the blocking of a simple chain by the isolation of its last node.

So a deadlock is a chain of blockings where the last node is blocked because:

1. It coincides with the first node.
2. It belongs to another deadlock.
3. It is isolated by another deadlock.

It is very significant that deadlocks that have occurred create more possibilities for new deadlocks occurring. So the process has avalanche-like character. A full deadlock involving all the devices (and all the transitions) occurs when cycles (chains) contain all the devices in the hypercube. It requires at least $(l+1)\cdot k^d$ packets provided by the terminal devices. But if isolations of devices occur, a small number of packets is required.

In spite of the fact that rather sophisticated hypercube communication structures were studied, the described deadlocks in the chains (cycles) of blockings and isolations are rather common for real-life communication graphs, where devices with the compulsory buffering are used. We believe that these deadlocks may be purposely inflicted by the specially situated generators of the particular traffic. In real-life networks, the blocking of the devices is overcome by the timeout mechanisms causing the cleaning of the buffers, but this leads to a considerable decrease in network performance as soon as the situation is repeated by the special generators of ill-intentioned traffic.

A hypertorus structure, common for (thermo) nuclear physics applications, is composed in a similar way by connecting opposite sides of a hypercube structure [18]. The software generators of hypertorus models are available at http://github.com/dazeorgacm/htgen.

5. Software Generators of Grid Models

During development of plugins for the known simulating system Tina, named Deborah and Adriana [14] and destined for Petri nets decomposition into clans and compositional calculation of invariants, respectively, software generators of Petri nets [20] were applied. They produced large-scale Petri nets that were thereafter used as tests. These nets had the structure of a simple chain, a simple loop and a sequence of connected basic fragments. Thus, the possibility of testing program modules on nets with many thousands of vertices was provided. Analogous generators were used for verification of Ethernet protocols and analysis of computing grids [25, 26], with the aim of creating an inductive base for generalizing obtained results on structures of an arbitrary size.

This section represents, in essence, a case study of software generators of grid model construction, and the described technique could be employed in a wide range of Petri net application domains, including automated manufacture, business processes and programming.

5.1 Formats of Files

The simulating system Tina accepts two formats of files describing a Petri net: logical (.net) and graphical (.ndr). For constructing generators (Figure 7), either of the formats could be used. Since for big Petri nets their visual representation becomes inessential, in the present paper, a logical format is employed substantially. When required, visualization of a Petri net is done by embedded facilities of Tina that provide automatic construction of files with a graphical format (drawing net).

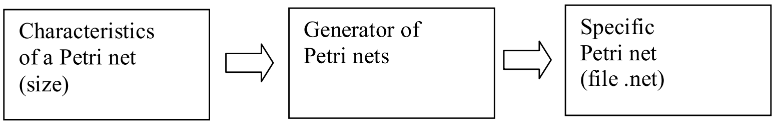
Characteristics of a Petri net (size)
Generator of Petri nets
Specific Petri net (file .net)

Figure 7. General scheme of a Petri net generator operation.

A file with a logical format (.net) contains an abstract description of a Petri net without information regarding its visual representation. A Petri net graph is described as a list of transitions and their input and output arcs; the initial marking is described as a list of places with values of their marking. In addition, Tina employs information such as auxiliary labels of vertices, types of arcs, transitions' priorities and times of transitions' firing. Such additional information is not used in the present paper for generating a classical Petri net, namely its graph required for structural analysis via calculation of places' and transitions' invariants. In the simplified form, the file format is described as follows:

```
tr <t-name> <p-name>[*<weight>],... -> <p-name>$[*<weight>],...
...
pl <p-name> (<marking>)
...
net <net-name>
```

The transition description begins with the reserved word "tr", then the list of the transition's incoming arcs follows, and after a delimiter "->" the list of outgoing arcs follows. An arc is described by a place name and the arc's multiplicity (weight) is indicated after a delimiter "*"; multiplicity equal to unit is omitted. Compound names of vertices are parenthesized in curly brackets. An example of a file is shown in Listing 2.

5.2 Software Implementation of a Generator

The parametric description of grids via the form [23] (represented with equation (3)) is source data for software implementation of Petri net generators. The generator of the open square grid models in programming language C is shown in Listing 1. The value of parameter *k* of the square grid size is inputted from the command line as argv[1]. Then, on each of the variables *i*, *j* used in [23] (equation (3)), the corresponding loop is organized; because of the small range of variable *v*, as an alternative to additional loops, the direct enumeration of all variants is employed.

```
/* generate open square grid on plane */
#include <stdio.h>

main( int argc, char * argv[] )
{
 int k,i,j;
 if( argc < 2 )
 {
  printf("*** USAGE: g2o k\n");
  return 2;
 }
 else k = atoi( argv[1] );

 for(i=1; i<=k; i++)
  for(j=1; j<=k; j++)
  {
   printf("tr {to_1^%d,%d} {pol_1^%d,%d} {pb_1^%d,%d} ->
{po_1^%d,%d} {pbl^%d,%d}\n", i,j, i,j, i,j, i,j, i,j );
   printf("tr {ti_1,2^%d,%d} {pi_1^%d,%d} {pbl^%d,%d} ->
{pil_1^%d,%d} {pb_2^%d,%d}\n", i,j, i,j, i,j, i,j, i,j );
   printf("tr {ti_1,3^%d,%d} {pi_1^%d,%d} {pbl^%d,%d} ->
{pil_1^%d,%d} {pb_3^%d,%d}\n", i,j, i,j, i,j, i,j, i,j );
   printf("tr {ti_1,4^%d,%d} {pi_1^%d,%d} {pbl^%d,%d} ->
        {pil_1^%d,%d} {pb_4^%d,%d}\n", i,j, i,j, i,j, i,j, i,j );
   printf("tr {to_4^%d,%d} {pol_4^%d,%d} {pb_4^%d,%d} ->
{po_4^%d,%d} {pbl^%d,%d}\n", i,j, i,j, i,j, i,j, i,j );
   printf("tr {ti_4,1^%d,%d} {pi_4^%d,%d} {pbl^%d,%d} ->
{pil_4^%d,%d} {pb_1^%d,%d}\n", i,j, i,j, i,j, i,j, i,j );
   printf("tr {ti_4,2^%d,%d} {pi_4^%d,%d} {pbl^%d,%d} ->
{pil_4^%d,%d} {pb_2^%d,%d}\n", i,j, i,j, i,j, i,j, i,j );
```

```
   printf("tr {ti_4,3^%d,%d} {pi_4^%d,%d} {pbl^%d,%d} ->
{pil_4^%d,%d} {pb_3^%d,%d}\n", i,j, i,j, i,j, i,j, i,j );
   printf("tr {to_2^%d,%d} {pil_4^%d,%d} {pb_2^%d,%d} ->
{pi_4^%d,%d} {pbl^%d,%d}\n", i,j, i,j+1, i,j, i,j+1, i,j );
   printf("tr {ti_2,1^%d,%d} {po_4^%d,%d} {pbl^%d,%d} ->
{pol_4^%d,%d} {pb_1^%d,%d}\n", i,j, i,j+1, i,j, i,j+1, i,j );
   printf("tr {ti_2,3^%d.%d} {po_4^%d,%d} {pbl^%d,%d} ->
{pol_4^%d,%d} {pb_3^%d,%d}\n", i,j, i,j+1, i,j, i,j+1, i,j );
   printf("tr {ti_2,4^%d,%d} {po_4^%d,%d} {pbl^%d,%d} ->
{pol_4^%d,%d} {pb_4^%d,%d}\n", i,j, i,j+1, i,j, i,j+1, i,j );
   printf("tr {to_3^%d,%d} {pil_1^%d,%d} {pb_3^%d,%d} ->
{pi_1^%d,%d} {pbl^%d,%d}\n", i,j, i+1,j, i,j, i+1,j, i,j );
   printf("tr {ti_3,1^%d,%d} {po_1^%d^%d} {pbl^%d,%d} ->
{pol_1^%d,%d} {pb_1^%d,%d}\n", i,j, i+1,j, i,j, i+1,j, i,j );
   printf("tr {ti_3,2^%d,%d} {po_1^%d,%d} {pbl^%d,%d} ->
{pol_1^%d,%d} {pb_2^%d,%d}\n", i,j, i+1,j, i,j, i+1,j, i,j );
   printf("tr {ti_3,4^%d,%d} {po_1^%d,%d} {pbl^%d,%d} ->
{pol_1^%d,%d} {pb_4^%d,%d}\n", i,j, i+1,j, i,j, i+1,j, i,j );
  }
 printf("net n2o%d\n", k);
}
```

Listing 1. Generator of the open square grid model.

In Listing 2, a Petri net generated by the program, shown in Listing 1, at the parameter value $k = 2$, is represented; the net completely corresponds to the graphical representation of the grid model shown in [23, Figure 6].

```
tr {to_1^1,1} {pol_1^1,1} {pb_1^1,1} -> {po_1^1,1} {pbl^1,1}
tr {ti_1,2^1,1} {pi_1^1,1} {pbl^1,1} -> {pil_1^1,1} {pb_2^1,1}
tr {ti_1,3^1,1} {pi_1^1,1} {pbl^1,1} -> {pil_1^1,1} {pb_3^1,1}
tr {ti_1,4^1,1} {pi_1^1,1} {pbl^1,1} -> {pil_1^1,1} {pb_4^1,1}
tr {to_4^1,1} {pol_4^1,1} {pb_4^1,1} -> {po_4^1,1} {pbl^1,1}
tr {ti_4,1^1,1} {pi_4^1,1} {pbl^1,1} -> {pil_4^1,1} {pb_1^1,1}
tr {ti_4,2^1,1} {pi_4^1,1} {pbl^1,1} -> {pil_4^1,1} {pb_2^1,1}
tr {ti_4,3^1,1} {pi_4^1,1} {pbl^1,1} -> {pil_4^1,1} {pb_3^1,1}
tr {to_2^1,1} {pil_4^1,2} {pb_2^1,1} -> {pi_4^1,2} {pbl^1,1}
tr {ti_2,1^1,1} {po_4^1,2} {pbl^1,1} -> {pol_4^1,2} {pb_1^1,1}
tr {ti_2,3^1.1} {po_4^1,2} {pbl^1,1} -> {pol_4^1,2} {pb_3^1,1}
tr {ti_2,4^1,1} {po_4^1,2} {pbl^1,1} -> {pol_4^1,2} {pb_4^1,1}
tr {to_3^1,1} {pil_1^2,1} {pb_3^1,1} -> {pi_1^2,1} {pbl^1,1}
tr {ti_3,1^1,1} {po_1^2^1} {pbl^1,1} -> {pol_1^2,1} {pb_1^1,1}
tr {ti_3,2^1,1} {po_1^2,1} {pbl^1,1} -> {pol_1^2,1} {pb_2^1,1}
tr {ti_3,4^1,1} {po_1^2,1} {pbl^1,1} -> {pol_1^2,1} {pb_4^1,1}
tr {to_1^1,2} {pol_1^1,2} {pb_1^1,2} -> {po_1^1,2} {pbl^1,2}
tr {ti_1,2^1,2} {pi_1^1,2} {pbl^1,2} -> {pil_1^1,2} {pb_2^1,2}
tr {ti_1,3^1,2} {pi_1^1,2} {pbl^1,2} -> {pil_1^1,2} {pb_3^1,2}
tr {ti_1,4^1,2} {pi_1^1,2} {pbl^1,2} -> {pil_1^1,2} {pb_4^1,2}
tr {to_4^1,2} {pol_4^1,2} {pb_4^1,2} -> {po_4^1,2} {pbl^1,2}
tr {ti_4,1^1,2} {pi_4^1,2} {pbl^1,2} -> {pil_4^1,2} {pb_1^1,2}
tr {ti_4,2^1,2} {pi_4^1,2} {pbl^1,2} -> {pil_4^1,2} {pb_2^1,2}
tr {ti_4,3^1,2} {pi_4^1,2} {pbl^1,2} -> {pil_4^1,2} {pb_3^1,2}
tr {to_2^1,2} {pil_4^1,3} {pb_2^1,2} -> {pi_4^1,3} {pbl^1,2}
tr {ti_2,1^1,2} {po_4^1,3} {pbl^1,2} -> {pol_4^1,3} {pb_1^1,2}
tr {ti_2,3^1.2} {po_4^1,3} {pbl^1,2} -> {pol_4^1,3} {pb_3^1,2}
tr {ti_2,4^1,2} {po_4^1,3} {pbl^1,2} -> {pol_4^1,3} {pb_4^1,2}
tr {to_3^1,2} {pil_1^2,2} {pb_3^1,2} -> {pi_1^2,2} {pbl^1,2}
tr {ti_3,1^1,2} {po_1^2^2} {pbl^1,2} -> {pol_1^2,2} {pb_1^1,2}
tr {ti_3,2^1,2} {po_1^2,2} {pbl^1,2} -> {pol_1^2,2} {pb_2^1,2}
tr {ti_3,4^1,2} {po_1^2,2} {pbl^1,2} -> {pol_1^2,2} {pb_4^1,2}
tr {to_1^2,1} {pol_1^2,1} {pb_1^2,1} -> {po_1^2,1} {pbl^2,1}
tr {ti_1,2^2,1} {pi_1^2,1} {pbl^2,1} -> {pil_1^2,1} {pb_2^2,1}
tr {ti_1,3^2,1} {pi_1^2,1} {pbl^2,1} -> {pil_1^2,1} {pb_3^2,1}
tr {ti_1,4^2,1} {pi_1^2,1} {pbl^2,1} -> {pil_1^2,1} {pb_4^2,1}
tr {to_4^2,1} {pol_4^2,1} {pb_4^2,1} -> {po_4^2,1} {pbl^2,1}
tr {ti_4,1^2,1} {pi_4^2,1} {pbl^2,1} -> {pil_4^2,1} {pb_1^2,1}
tr {ti_4,2^2,1} {pi_4^2,1} {pbl^2,1} -> {pil_4^2,1} {pb_2^2,1}
tr {ti_4,3^2,1} {pi_4^2,1} {pbl^2,1} -> {pil_4^2,1} {pb_3^2,1}
tr {to_2^2,1} {pil_4^2,2} {pb_2^2,1} -> {pi_4^2,2} {pbl^2,1}
tr {ti_2,1^2,1} {po_4^2,2} {pbl^2,1} -> {pol_4^2,2} {pb_1^2,1}
tr {ti_2,3^2.1} {po_4^2,2} {pbl^2,1} -> {pol_4^2,2} {pb_3^2,1}
tr {ti_2,4^2,1} {po_4^2,2} {pbl^2,1} -> {pol_4^2,2} {pb_4^2,1}
tr {to_3^2,1} {pil_1^3,1} {pb_3^2,1} -> {pi_1^3,1} {pbl^2,1}
tr {ti_3,1^2,1} {po_1^3^1} {pbl^2,1} -> {pol_1^3,1} {pb_1^2,1}
tr {ti_3,2^2,1} {po_1^3,1} {pbl^2,1} -> {pol_1^3,1} {pb_2^2,1}
tr {ti_3,4^2,1} {po_1^3,1} {pbl^2,1} -> {pol_1^3,1} {pb_4^2,1}
tr {to_1^2,2} {pol_1^2,2} {pb_1^2,2} -> {po_1^2,2} {pbl^2,2}
tr {ti_1,2^2,2} {pi_1^2,2} {pbl^2,2} -> {pil_1^2,2} {pb_2^2,2}
tr {ti_1,3^2,2} {pi_1^2,2} {pbl^2,2} -> {pil_1^2,2} {pb_3^2,2}
tr {ti_1,4^2,2} {pi_1^2,2} {pbl^2,2} -> {pil_1^2,2} {pb_4^2,2}
tr {to_4^2,2} {pol_4^2,2} {pb_4^2,2} -> {po_4^2,2} {pbl^2,2}
tr {ti_4,1^2,2} {pi_4^2,2} {pbl^2,2} -> {pil_4^2,2} {pb_1^2,2}
tr {ti_4,2^2,2} {pi_4^2,2} {pbl^2,2} -> {pil_4^2,2} {pb_2^2,2}
tr {ti_4,3^2,2} {pi_4^2,2} {pbl^2,2} -> {pil_4^2,2} {pb_3^2,2}
tr {to_2^2,2} {pil_4^2,3} {pb_2^2,2} -> {pi_4^2,3} {pbl^2,2}
tr {ti_2,1^2,2} {po_4^2,3} {pbl^2,2} -> {pol_4^2,3} {pb_1^2,2}
tr {ti_2,3^2.2} {po_4^2,3} {pbl^2,2} -> {pol_4^2,3} {pb_3^2,2}
tr {ti_2,4^2,2} {po_4^2,3} {pbl^2,2} -> {pol_4^2,3} {pb_4^2,2}
tr {to_3^2,2} {pil_1^3,2} {pb_3^2,2} -> {pi_1^3,2} {pbl^2,2}
tr {ti_3,1^2,2} {po_1^3^2} {pbl^2,2} -> {pol_1^3,2} {pb_1^2,2}
tr {ti_3,2^2,2} {po_1^3,2} {pbl^2,2} -> {pol_1^3,2} {pb_2^2,2}
tr {ti_3,4^2,2} {po_1^3,2} {pbl^2,2} -> {pol_1^3,2} {pb_4^2,2}
net n2o2
```

Listing 2. Generated model of the open square grid of size 2 (.net format).

5.3 Application of Generated Nets

The main application area of specific grid models, obtained as the result of running generators, is the formation of a database of actual nets for further inductive conclusions regarding the properties of infinite Petri nets with regular structure.

On the basis of calculation and analysis of place invariants for a sequence of grid models with definite sizes, the general parametric description of invariants was obtained and the p-invariance of a Petri net for an arbitrary value of parameter k was proven [23]. Analogous results were obtained in Section 4 for hypercube structures of an arbitrary size with an arbitrary number of dimensions. For instance, the parametric description of place invariants of the grid (represented here with equation (2)) has the form shown in [23, equation (15)].

Parametric representation [23, equation (15)] of the basis invariants matrix is rather sophisticated because it is valid for any given magnitude of parameter k. Only nonzero elements are listed, and in the example of a square grid, all of them are equal to unit. There are two types of parametric rows: each of rows 1 through 5 describes a set of rows with a few nonzero elements (two for rows 1 through 4 and five for row 5); each of rows 6 and 7 describes a single row containing a set of nonzero elements. To distinguish the difference, brackets are used.

Invariants calculated by the system Tina for definite values of parameter k coincide with invariants generated from [23, equation (15)]. Thus, the described technique could be used for generating net invariants on their parametric description. For instance, the place invariants of the net, shown in Figure 6 of [23], are represented in Listing 3 in an explicit form.

In some cases, the automatic visualization of generated nets is useful for evaluating general patterns of layout and doing an additional check of composition rules. In Figure 8, a model of a grid of size 4 is represented in graphical form, which is created automatically by simulating the system in Tina.

Moreover, generated Petri net models could be applied as tests in Petri net simulating and analysis systems development, especially when the properties of studied nets are known a priori. For instance, the considered models of grids are safe according to the composition rules because each transition has exactly two input and two output places. It allows debugging software on large dimension nets.

5.4 Generating Petri Nets in Graphical Form

Automatic visualization of grid models is not distinguished by clearness and allows visual evaluation of the general pattern only. Some applications require working with graphical formats, for instance, for watching a token game and studying transition firing sequences. A model could be generated in graphical format as well.

```
({pi_ 1 ^ 1, 1}, {pil_ 1 ^ 1, 1})   ({po_ 1 ^ 1, 1}, {pol_ 1 ^ 1, 1})
({pi_ 1 ^ 1, 2}, {pil_ 1 ^ 1, 2})   ({po_ 1 ^ 1, 2}, {pol_ 1 ^ 1, 2})
({pi_ 1 ^ 2, 1}, {pil_ 1 ^ 2, 1})   ({po_ 1 ^ 2, 1}, {pol_ 1 ^ 2, 1})
({pi_ 1 ^ 2, 2}, {pil_ 1 ^ 2, 2})   ({po_ 1 ^ 2, 2}, {pol_ 1 ^ 2, 2})
({pi_ 1 ^ 3, 1}, {pil_ 1 ^ 3, 1})   ({po_ 1 ^ 3, 1}, {pol_ 1 ^ 3, 1})
({pi_ 1 ^ 3, 2}, {pil_ 1 ^ 3, 2})   ({po_ 1 ^ 3, 2}, {pol_ 1 ^ 3, 2})

({pi_ 4 ^ 1, 1}, {pil_ 4 ^ 1, 1})   ({po_ 4 ^ 1, 1}, {pol_ 4 ^ 1, 1})
({pi_ 4 ^ 1, 2}, {pil_ 4 ^ 1, 2})   ({po_ 4 ^ 1, 2}, {pol_ 4 ^ 1, 2})
({pi_ 4 ^ 1, 3}, {pil_ 4 ^ 1, 3})   ({po_ 4 ^ 1, 3}, {pol_ 4 ^ 1, 3})
({pi_ 4 ^ 2, 1}, {pil_ 4 ^ 2, 1})   ({po_ 4 ^ 2, 1}, {pol_ 4 ^ 2, 1})
({pi_ 4 ^ 2, 2}, {pil_ 4 ^ 2, 2})   ({po_ 4 ^ 2, 2}, {pol_ 4 ^ 2, 2})
({pi_ 4 ^ 2, 3}, {pil_ 4 ^ 2, 3})   ({po_ 4 ^ 2, 3}, {pol_ 4 ^ 2, 3})

({pb_ 1 ^ 1, 1}, {pb_ 2 ^ 1, 1}, {pb_ 3 ^ 1, 1}, {pb_ 4 ^ 1, 1}, {pbl ^ 1, 1})
({pb_ 1 ^ 1, 2}, {pb_ 2 ^ 1, 2}, {pb_ 3 ^ 1, 2}, {pb_ 4 ^ 1, 2}, {pbl ^ 1, 2})
({pb_ 1 ^ 2, 1}, {pb_ 2 ^ 2, 1}, {pb_ 3 ^ 2, 1}, {pb_ 4 ^ 2, 1}, {pbl ^ 2, 1})
({pb_ 1 ^ 2, 2}, {pb_ 2 ^ 2, 2}, {pb_ 3 ^ 2, 2}, {pb_ 4 ^ 2, 2}, {pbl ^ 2, 2})

({pil_ 1^1,1},{pol_ 1^1,1},{pil_ 4^1,1},{pol_ 4^1,1},{pbl^1,1},{pil_ 1^1,2},
{pol_ 1^1,2},{pil_ 4^1,2},{pol_ 4^1,2},{pbl^1,2},{pil_ 1^2,1},{pol_ 1^2,1},
{pil_ 4^2,1},{pol_ 4^2,1},{pbl^2,1},{pil_ 1^2,2},{pol_ 1^2,2},{pil_ 4^2,2},
{pol_ 4^2,2},{pbl^2,2},{pil_ 1^3,1},{pol_ 1^3,1},{pil_ 1^3,2},{pol_ 1^3,2},
{pil_ 4^1,3},{pol_ 4^1,3},{pil_ 4^2,3},{pol_ 4^2,3})

({pi_ 1^1,1},{po_ 1^1,1},{pi_ 4^1,1},{po_ 4^1,1},{pb_ 1^1,1},{pb_ 2^1,1},
{pb_ 3^1,1},{pb_ 4^1,1},{pi_ 1^1,2},{po_ 1^1,2},{pi_ 4^1,2},{po_ 4^1,2},
{pb_ 1^1,2},{pb_ 2^1,2},{pb_ 3^1,2},{pb_ 4^1,2},{pi_ 1^2,1},{po_ 1^2,1},
{pi_ 4^2,1},{po_ 4^2,1},{pb_ 1^2,1},{pb_ 2^2,1},{pb_ 3^2,1},{pb_ 4^2,1},
{pi_ 1^2,2},{po_ 1^2,2},{pi_ 4^2,2},{po_ 4^2,2},{pb_ 1^2,2},{pb_ 2^2,2},
{pb_ 3^2,2},{pb_ 4^2,2},{pi_ 1^3,1},{po_ 1^3,1},{pi_ 1^3,2},{po_ 1^3,2},
{pi_ 4^1,3},{po_ 4^1,3},{pi_ 4^2,3},{po_ 4^2,3})
```

Listing 3. Place invariants of the open square grid with size 2 [23, Figure 6].

The graphical file format (.ndr) of the system Tina contains such extra information, compared with the logical format (.net) described in Section 5.1, as coordinates of vertices on a plane and arcs' curves description:

```
t <xpos> <ypos> <transition-name> <options>
p <xpos> <ypos> <place-name> <marking> <options>
e <vertex1-name> <vertex2-name> <weight> <options>
e <vertex1-name> <ang> <rad> <vertex2-name> <ang> <rad> <options>
```

Lines of types p and t describe places and transitions, respectively. Each row contains node coordinates (xpos, ypos) on a plane and its name; the place description also includes its initial marking. Each arc is described separately with a line of type e. It contains names of the source and target vertices and multiplicity (weight) for a straight arc; for a curved arc, two pairs of additional parameters (ang, rad) define an angle and radius of arc curve for both its ends.

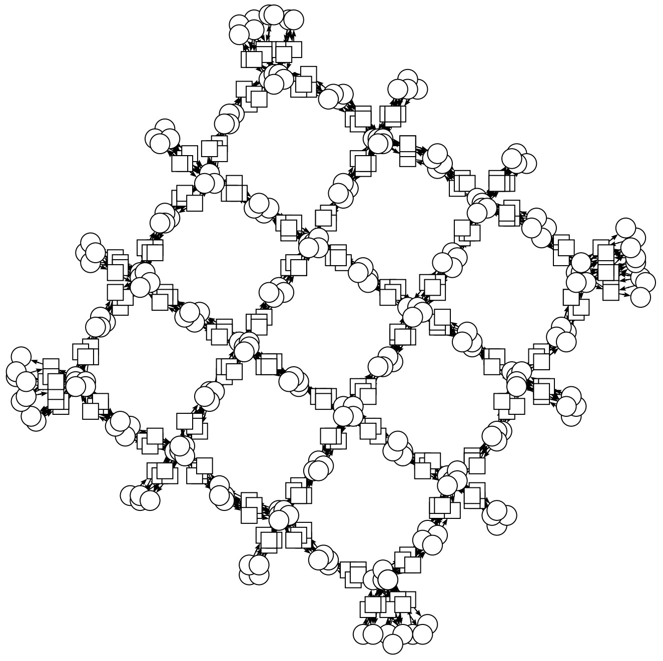

Figure 8. Automatic visualization of a generated grid model in Tina.

The technique of generating .ndr files starts with drawing a model of a grid node in Tina (Figure 4) and saving the corresponding .ndr file. An example of a net fragment description containing a place, a transition and an arc connecting them, for the grid node model shown in Figure 4, looks like this:

```
p 200.0 50.0 {po_1} 0 n
t 240.0 130.0 {to_1} 0 w n
e {to_1} {po_1} 1 n
```

Then the .ndr file obtained is used as a pattern to fill in the main loop of the generator, whose overall organization is similar to Listing 1. Only descriptions of vertices contain coordinates, which should be recalculated for each node of the grid on its indices; the relative format of the arcs' curvature description does not require their correction. Based on an element description, a format string of the corresponding printf operator is created. Instead of definite coordinates (xpos, ypos) it contains format "%.1f %.1f" for printing recalculated coordinates and a format "∧%d.%d" is inserted into vertices' names for printing the current node index within the grid:

```
printf("p %.1f %.1f {po_1^%d.%d} 0 n\n",(i-1)*DI+200.0, (j-1)*DJ+50.0, j, i);
printf("t %.1f %.1f {to_1^%d.%d} 0 w n\n",(i-1)*DI+240.0, (j-1)*DJ+130.0, j, i);
printf("e {to_1^%d.%d} {po_1^%d.%d} 1 n\n",j, i, j, i);
```

Coordinates of vertices are calculated based on vertical and horizontal grid node offsets DI and DJ, respectively, which are equal to the maximal coordinates of a single grid node description. Then the local offset of a vertex within the grid node is added. Reverse order of indices for coordinates regarding names is explained by the fact that in the grid model a matrix order of indexing is used: first index (i)—number of a row, second index (j)—number of a column. But in Tina, the first coordinate axis (x) is horizontal and the second (y) is vertical.

Series of grid model generators, working in graphical format, were developed and employed in investigation of infinite Petri nets' properties [23]. An example of a generated open square grid model of size 4 is shown in Figure 9. The generators of the square grid model are available at http://github.com/dazeorgacm/sq.

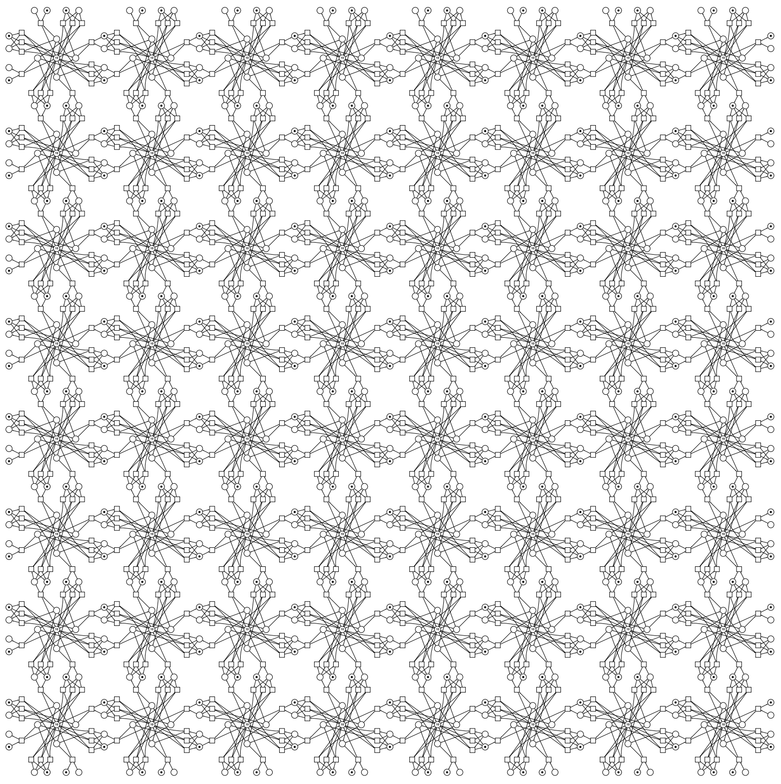
Figure 9. An example of the open square grid model of size 4 generated in graphical format.

Recently, for generating a canvas of a generalized neighborhood of cellular automata [7] in the form of a Petri net, software was developed, available at http://github.com/dazeorgacm/hmn. An example of the model obtained for the hypertorus cellular automaton is shown in Figure 10.

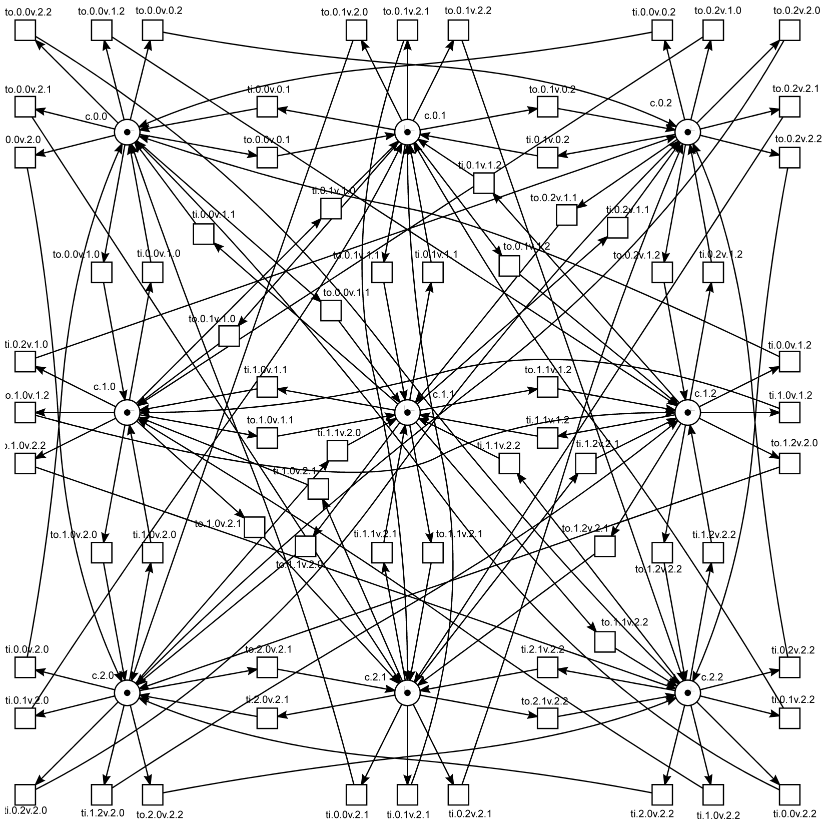

Figure 10. A hypertorus finite cellular automata model.

6. Results and Hypothesis

A basis of infinite Petri nets theory was developed for modeling computing grids. The described technique could be employed in a wide range of Petri net application domains, including automated manufacture, business processes, programming and systems biology.

Infinite Petri nets and their invariants were represented in parametric form. For each of the obtained parametric descriptions of the invariants of infinite Petri nets with regular structure, it was proven that they are solutions of the corresponding infinite system of equations. To prove the Petri net invariance, it was not required to prove that the obtained set of solutions form a basis; an invariant, having all natural elements, was constructed explicitly in each case. For an extra validation of results, the calculation of invariants on series of definite grids of certain size was implemented in the environment of the system Tina using ad hoc software generators of models; and also the fact was used that all the models are conservative, and closed models are consistent Petri nets because of the manner of their composition: each vertex has the same number of incoming and outgoing arcs.

The two following statements have been proven:

Statement 1. Grid models are invariant Petri nets for an arbitrary grid size and an arbitrary number of dimensions.

Statement 2. Grid models are not live Petri nets for an arbitrary grid size and an arbitrary number of dimensions.

The three following hypothesis have been advanced:

Hypothesis 1. Occurrence of deadlocks increases the probability of new deadlocks creation; thus, the growth of the number of deadlocks has an avalanche-like character.

Hypothesis 2. Deadlocks could be caused by ill-intentioned traffic of a special form.

Hypothesis 3. For infinite systems of the form in equation (6), the three following situations are possible: it has no solution except the trivial; it has solutions and they can be specified in a finite form; it has solutions but they cannot be specified in a finite form.

To acknowledge Hypotheses 1 and 2, colored Petri nets [11] could be employed in a way similar to that described in [22]. The proof of Hypothesis 3 requires advances in modern mathematics theory.

All the models presented in this paper were constructed and analyzed in the system Tina [12] supplied with plugins Deborah and Adrian [14], available at http://member.acm.org/~daze.

7. Conclusion

The search for new application domains, such as manufacturing and transportation systems, will allow the generalization of the problematic infinite Petri nets with regular structure and make it more independent from the terminology of computer networks, clusters and grids. Among such generalizations, we advise using a set of basic fragments and formalizing the models' composition rules.

When constructing the grid models, more attention could be paid to the description of computational aspects of information processing as well as to the architectural peculiarities of modern networking devices.

A general method of solving infinite systems of linear Diophantine equations in non-negative integer numbers, which the invariant analysis of the models is reduced to, is unknown. That is why the search for such methods is significant. It could be possible to construct separate methods for subclasses of nets and also prove that the obtained set of the parametric solutions forms a basis.

Heterogeneous infinite systems of equations and inequalities could be used to check the marking reachability conditions in infinite nets, to search the siphons and traps using an analogy with the compositional analysis of finite nets.

Not considered were methods of Petri net analysis via graphs of reachable and coverable markings, which could be modified for work with infinite nets.

Acknowledgments

The work was partially supported by NATO grant ICS.NUKR. CLG982689 and presented at IEEE seminar http://meetings.vtools.ieee.org/m/47179.

References

[1] Z. W. Li and A. M. Al-Almari (eds.), *Formal Methods in Manufacturing Systems: Recent Advances*, Hershey, PA: Engineering Science Reference, 2013.

[2] Z. W. Li and M. C. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*, London: Springer Verlag Ltd., 2009.

[3] W. van der Aalst and C. Stahl, *Modeling Business Processes: A Petri Net-Oriented Approach*, Cambridge, MA: MIT Press, 2011.

[4] C. Girault and R. Valk, *Petri Nets for Systems Engineering: A Guide to Modeling, Verification, and Applications*, New York: Springer, 2003.

[5] E. Best, R. Devillers and M. Koutny, *Petri Net Algebra*, New York: Springer, 2001.

[6] D. A. Zaitsev, "Simulating Cellular Automata by Infinite Petri Nets," *Journal of Cellular Automata*, **13**(1–2), 2018 121–144.

[7] D. A. Zaitsev, "A Generalized Neighborhood for Cellular Automata," *Theoretical Computer Science*, **666**(1), 2017 pp. 21–35. doi:10.1016/j.tcs.2016.11.002.

[8] I. Koch, W. Reisig and F. Schreiber (eds.), *Modeling in Systems Biology: The Petri Net Approach*, New York: Springer, 2011.

[9] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Englewood Cliffs, NJ: Prentice-Hall, 1981.

[10] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, **77**(4), 1989 pp. 541–580. doi:10.1109/5.24143.

[11] K. Jensen and L. M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*, New York: Springer, 2009 p. 384.

[12] G. Berthelot and R. Terrat, "Petri Nets Theory for the Correctness of Protocols," *IEEE Transactions on Communications*, **30**(12), 1982 pp. 2497–2505. doi:10.1109/TCOM.1982.1095452.

[13] M. Diaz, "Modeling and Analysis of Communication and Cooperation Protocols Using Petri Net Based Models," *Computer Networks*, **6**(6), 1982 pp. 419–441. doi:10.1016/0376-5075(82)90112-X.

[14] D. A. Zaitsev, *Clans of Petri Nets: Verification of Protocols and Performance Evaluation of Networks*, Lambert Academic Publishing, 2013 p. 292.

[15] A. M. Marsan, G. Chiola and A. Fumagalli, "An Accurate Performance Model of CSMA/CD Bus LAN," in *Advances in Petri Nets 1987 (APN 1986)*, (G. Rozenberg, ed.), 1987 pp. 146–161. doi:10.1007/3-540-18086-9_ 24.

[16] T. R. Shmeleva, D. A. Zaitsev and I. D. Zaitsev, "Verification of Square Communication Grid Protocols via Infinite Petri Nets," *10th Middle Eastern Simulation Multiconference (MESM 2009)*, Beirut, Lebanon, 2009 pp. 53–59.

[17] D. A. Zaitsev and T. R. Shmeleva, "Verification of Hypercube Communication Structures via Parametric Petri Nets," *Cybernetics and Systems Analysis*, **46**(1), 2010 pp. 105–114. doi:10.1007/s10559-010-9189-y.

[18] D. A. Zaitsev, "Verification of Computing Grids with Special Edge Conditions by Infinite Petri Nets," *Automatic Control and Computer Sciences*, **47**(7), 2013 pp. 403–412. doi:10.3103/S0146411613070262.

[19] D. A. Zaitsev, I. D. Zaitsev and T. R. Shmeleva, "Infinite Petri Nets as Models of Grids," in *Encyclopedia of Information Science and Technology*, 3rd ed., (M. Khosrow-Pour, ed.), Hershey, PA: IGI Global, 2014 pp. 187–204. doi:10.4018/978-1-4666-5888-2.ch019.

[20] D. A. Zaitsev, "Generators of Petri Net Models," *Computer Communication & Collaboration*, **2**(2), 2014 pp. 12–25.

[21] D. Burdett, Internet Open Trading Protocol—IOTP, Version 1.0E, RFC 2801, April, 2000 p. 290.

[22] D. A. Zaitsev, T. R. Shmeleva, W. Retschitzegger and B. Pröll, "Security of Grid Structures under Disguised Traffic Attacks," *Cluster Computing*, **19**(3), 2016 pp. 1183–1200. doi:10.1007/s10586-016-0582-9.

[23] D. A. Zaitsev, I. D. Zaitsev and T. R. Shmeleva, "Infinite Petri Nets: Part 1, Modeling Square Grid Structures," *Complex Systems*, **26**(2), 2017 pp. 157–195. www.complex-systems.com/pdf/26-2-4.pdf.

[24] B. Berthomieu, P.-O. Ribet and F. Vernadat, "The Tool TINA: Construction of Abstract State Spaces for Petri Nets and Time Petri Nets," *International Journal of Production Research*, **42**(14), 2004 pp. 2741–2756. doi:10.1080/00207540412331312688.

[25] Information Resources Management Association, *Grid and Cloud Computing: Concepts, Methodologies, Tools and Applications*, Hershey, PA: IGI Global, 2012.

[26] N. P. Preve (ed.), *Grid Computing: Towards a Global Interconnected Infrastructure*, London: Springer-Verlag, 2011 p. 312.