sensors

MDPI

Article

Extended Hierarchical Fuzzy Interpreted Petri Net

Michał Markiewicz , Lesław Gniewek and Dawid Warchoł *

Department of Computer and Control Engineering, Faculty of Electrical and Computer Engineering, Rzeszów University of Technology, W. Pola 2, 35-959 Rzeszów, Poland; mmarkiewicz@prz.edu.pl (M.M.); lgniewek@prz.edu.pl (L.G.)
* Correspondence: dawwar@prz.edu.pl; Tel.: +48-17-865-1592

Abstract: Petri nets (PNs) have many advantages such as graphical representation, formal description, and the possibility of sequential and concurrent control. An important aspect of using PNs is hierarchical modeling, which may be provided in different ways. In this paper, a new concept and definition of the hierarchical structure for Fuzzy Interpreted Petri Net (FIPN) are proposed. The concept of macroplace with several input, output, and input-output places is introduced to the net. The functionality of the macroplace instances and the hierarchy graph are also proposed. They are implemented in a computer simulator called HFIPN-SML. In this study, FIPN is employed since it allows the use of analogue sensors directly for process control. Better visualization and more precise control are among advantages of the introduced approach.

Keywords: Petri nets; fuzzy nets; interpreted nets; hierarchical nets; computer simulation; control systems

check for updates

Citation: Markiewicz, M.; Gniewek, L.; Warchoł, D. Extended Hierarchical Fuzzy Interpreted Petri Net. *Sensors* **2021**, *21*, 8433. https://doi.org/10.3390/s21248433

Academic Editor: Antonio Guerrieri

Received: 16 November 2021
Accepted: 14 December 2021
Published: 17 December 2021

1. Introduction

Petri nets (PNs) are increasingly used in the development of information systems whose high complexity requires a new approach to design and implementation. Therefore, new concepts of PN description, analysis, and presentation have appeared, among which an important place takes introducing the hierarchy (modularization) to the net structure. It allows for the presentation of the developed systems in different levels of abstraction and facilitates the determination of the properties of nets modeling these systems. There is, however, no unified approach to using the hierarchy in PNs, and the main research directions are described in the following subsections.

1.1. Refinement of Places and Transitions

One of the first works related to the hierarchy in PNs was presented by Valette [1], who proposed a concept of macrotransition with a single input and output. The author defined a concept of a well-formed block and examined the properties of the hierarchical net, such as boundedness, safeness, and liveness. Suzuki and Murata [2] continued Valette's research by extending the concept of the well-formed block to a k-well-behaved block. Vogler [3] proposed an extension of the concepts described in [1,2] by replacing a transition having several inputs and outputs with a subnet taking into account certain constraints of the structure. Brauer et al. [4] described the refinement of transitions for a subnet with one initial and one final transition as well as a method for creating macroplaces.

Bernardinello and De Cindio [5] presented and compared different approaches to the refinements of place and transition. The majority of these concepts use the composition of synchronized subnets with a state machine structure. Van Der Aalst [6] described a method to validate the workflow using a hierarchical net, which preserves properties such as safeness, liveness, boundedness, free-choice, and well-structuredness. The net was used to model the flow control of different aspects of business processes. Huang et al. [7] described a method for refinement providing nineteen system properties. In the literature, a refinement of actions and operations is often considered [4,8–12]. Such an approach also allows representing a system hierarchically and its analysis in different levels of abstraction.

1.2. Reduction

Another approach to the modeling of the hierarchy in PNs is a reduction technique. It allows determining static and dynamic properties of a net based on its smaller counterpart to which it has been transformed. The first research on this topic appeared in the seventies [13–16]. These works have been generalized by Berthelot [17,18], who proposed transformations providing properties such as liveness, reachable marking, proper termination, home state, deadlock freeness, unavoidable states, and abstraction. The following transformations can be distinguished: transformation of places, fusion of transitions, S-decomposition, and T-decomposition, which are based mainly on the static properties. Lee et al. [19,20] proposed a hierarchical reduction method that uses a decomposition. The method, based on a net structure, allows preserving properties such as liveness, boundedness, and proper termination.

Desel, Best and Esparza [21–24] described the reduction of free choice nets. In [21–23], they proposed the use of four types of reduction: P-reduction, T-reduction, F-reduction, and A-reduction. Desel and Esparza [24] summarized the results related to free choice nets and their properties. Jiao et al. [25,26] continued the research of Desel, Best, and Esparza and studied the properties of asymmetric free choice net. In the literature, the application of reduction rules for time PNs were also described [27–29].

1.3. Formal Description of the Hierarchical PNs

Introducing the hierarchy in PNs by a formal description and a net definition allows imposing certain initial constraints (such as the number of net inputs and outputs) and creating subnets in any way and with any properties.

One of the first works presenting the definition of a hierarchical net is the study of Fehling [30], in which a combination of a hierarchical PN with a morphism is presented. One of the most complete concepts is, however, the Colored Petri Net (CPN) described by Jensen [31]. The author proposed a definition, according to which a net can consist of separate subnets (called pages) that are related to transitions. Instances of places, transitions, arcs, and subnets were also described. Additionally, it is possible to define the so-called fusion of places. The relations between the subnets can be visualized with a hierarchy graph.

Holvoet and Verbaeten [32] presented an alternative definition of a hierarchical net. A recursion is used in the formal description. He [33] introduced the formal definition of hierarchical PN for modeling large, complex, and parallel distributed systems which uses the concept of macrotransitions and macroplaces. In the proposed solution, the hierarchical net has a tree structure. Andrzejewski [34] presented a formal specification of a reactive system based on a hierarchical interpreted time PN, which has a state memory in different levels of the hierarchy. In turn, Pan and Sun [35] proposed a hierarchical fuzzy PN, which can be used for decision systems modeling. In the net, both the abstract places and transitions (macroplaces and macrotransitions) are defined.

1.4. Hierarchical Nets in Industrial Standards

The formalism of PNs along with their hierarchy structure has found its application in programming standards of industrial controllers. The use of PNs to write programs for PLC controllers is described in the international standard IEC 60848:2013 [36], that proposes a Grafcet language. Based on Grafcet, the Sequential Function Chart (SFC) is defined [37] as one of the five programming languages for PLC controllers. The theoretical basis of Grafcet are described by David and Alla [38].

In the Grafcet standard, it is possible to use several functionalities based on the hierarchy. The first of them is the concept of the macroplace called macrostep. It assumes that a token, which is moved to a macroplace node, is automatically moved to the input step of a subnet assigned to this macrostep. Another concept is the so-called enclosure, in which a subnet is assigned to one enclosure step. If a token moves to the enclosure step, the initial steps of the subnet are activated, and when the token leaves the enclosure step, all subnet steps are deactivated. A different functionality is forcing subnet states: pausing and resuming a subnet work as well as activating/deactivating of the chosen steps.

In the IEC 61131-3 standard, the hierarchical structure is not directly defined for the SFC language. However, the way in which the actions assigned to steps are defined indirectly introduces the hierarchical nesting. They can be implemented using SFC and other languages of this standard.

1.5. Methodology of Dealing with Complex Systems

Modular PNs can be used for the creation of methodologies that describe dealing with complex systems, e.g., manufacturing systems. This approach is usually given in the form of a general verbal description supplemented with examples and formal descriptions.

Silva and Valette [39] proposed the modeling of Flexible Manufacturing Systems (FMS) using hierarchical PNs. Stepwise place/transition refinements and modular composition were applied. Valavanis [40] described a methodology of FMS modeling using an extended PN. In the approach, a decomposition (top-down) and composition (bottom-up) are used. Zhou et al. [41] proposed a hybrid methodology that ensures the appropriate properties of modules, such as boundedness, liveness, and reversibility. In the described concept, the decomposition and composition are employed for the determination of unshared and shared resources between modules.

Der Jeng and DiCesare [42] described the use of PNs in the modeling of shared-resource automated manufacturing systems. The merging of the modules is performed through transitions shared by different subnets. The methodology allows the preservation of liveness and boundedness. Zhou [43] showed a practical example of modular modeling of a semiconductor manufacturing system using time PN. A reduction technique is used to facilitate the analysis of properties of the entire system.

1.6. Recent Studies

Recent studies indicate that the hierarchy/modularization in PNs can be very useful. One of the subjects of research is decomposition. Ye et al. [44] showed the method for the creation of the decentralized control system using PNs. In the proposed solution, the subnets have a state-machine structure. Wisniewski et al. proposed a new algorithm of decomposition for edge systems modeled using Petri nets [45]. Formal proof of the correctness of the algorithm and comparison to a similar method were also given. For the modeling of practical problems, hierarchical Colored Petri Nets (HCPN) are widely used. The application of HCPN for the implementation of a traffic signal control model was proposed by An et al. [46]. Sicchar et al. [47] presented the application of HCPN in the control system for load-balance procedure in low voltage grid. Gozhyj et al. [48] presented the use of HCPN to model the interaction between web services using CPN tools.

Fuzzy hierarchical PNs and formal definitions of hierarchy are also considered. Li et al. [49] presented a layered fuzzy Petri net with a hierarchical structure for risk assessment. The fault diagnosis approach using time hierarchical fuzzy Petri net was proposed by Yuan et al. [50]. The combination of fuzzy and colored PNs for runtime veryficaton of pacemaker was presented by Majma et al. [51]. Padberg [52] proposed a formal definition of hierarchical reconfigurable PN and its conversion to a flat net. Different models and problems related to hierarchical PNs can also be found in [53–56].

1.7. Paper Scope and Organization

As can be seen from the previous subsections, the hierarchy in PNs can be realized in different ways. The boundary between the discussed research areas is ambiguous because they often overlap and can complement each other. For example, an important methodology for dealing with large systems, such as FMS, can be the reduction technique [43,57,58].

In this paper, a new concept and a formal description of hierarchical FIPN (HFIPN) is presented. The theoretical basis of FIPN are described in [59], where the structure, behavior, and algebraic representation of this net are shown. In [60], a coverabality graph for FIPN was proposed, which can be used in the study of net properties and as a diagnostic system. The properties of FIPN can also be determined using a reachability graph. Research related to the FIPN implementation is also conducted. The works [61,62] describe the FIPN software model and its implementation for PLC controllers, while a method of code generation for FPGA circuits is presented in [63]. As part of the research, a software tool called HFIPN-SML was developed based on (H)FIPN model.

This paper proposes a new hierarchical structure for FIPN and is a continuation of [64]. The contributions of this study are as follows:

(a) Concept of a macroplace that can have several input, output and input-output places;
(b) Functionality of macroplace instances;
(c) Formal definition of HFIPN;
(d) Concept and a definition of a hierarchy graph displaying a hierarchical structure and allowing a quick access to all subnets in an implementation version;
(e) Formal algebraic representation of HFIPN;
(f) Conversion of HFIPN to its flat version;
(g) Formal way to sum any two subnets.

The paper is organized as follows. First, a comparison of our approach to similar solutions is presented (Section 1.8). Then, the theoretical basis of FIPN is described (Section 2). Next, the new HFIPN concept is presented (Section 3). Then, an exemplary application is shown (Section 4). Finally, a summary of the research and potential future studies are given (Section 5).

1.8. Comparison to Similar Solutions

Due to previous research, HFIPN is compared to PNs used in PLC controllers. In Table 1 general characteristics of Signal Interpreted Petri Net (SIPN) [65–68], Grafcet [36], SFC [37], and other fuzzy PNs [69–71] are presented. The first three were chosen since they seem to be the most comprehensive because of the hierarchy structure, software tool support, and automatic executable code generation. The others, like HFIPN, allow the use of analogue signals, but they do not have the advantages shown in Table 1 that facilitate the practical application.

SIPN is a binary net with a formal description. It allows the use of a hierarchy based on a macroplace concept with one input and output place. Software based on SIPN enables the automatic conversion of the graphical net representation to executable code in Instruction List (IL) language and the automatic investigation of the net properties. All functionalities are implemented in one computer tool. Grafcet and SFC are binary nets allowing the use of a hierarchical structure (described in Section 1.4) and automatic executable code generation based on a net graph. Although they are proposed as international standards, there are some issues related to their informal description. There is no unified approach to the formal analysis of their properties and their operation depends on implementation. They can be converted to the finite machine [72–74], timed automata [72,75–78], or formal PNs [79–83].

In comparison to the mentioned solutions, the main advantage of HFIPN is the possibility of using analogue signals for direct control of a process. Conditions assigned to a transition may not be binary (it is explained in the next section). Contrary to Grafcet, SFC, and SIPN, HFIPN also allows resource modeling in the structure of a net since the weight of arcs and the capacity of places can be greater than one. In comparison to SIPN, HFIPN offers more possibilities in the use of a hierarchical structure. However, HFIPN has some gaps that need to be filled as shown in Table 1. As with SIPN, SFC, and Grafcet, HFIPN permits automatic generation of executable code [64], but unlike the others, only for a non-hierarchical net structure. Another issue is the investigation of properties. For HFIPN, like for Grafcet and SFC, there are created methods to investigate properties [60], but they are not integrated with HFIPN-SML. Clearing both gaps is planned for future studies.

To conclude this section, it can be observed that there is no other solution combining PNs and PLC programming to create control systems and offering the possibility of using analogue sensors, software tool support, resource modeling by the structure of a net, and automatic generation of executable code. However, some new functionalities need to be implemented to see HFIPN and its simulator as a complete solution.

Table 1. Comparison of HFIPN to similar nets.

	HFIPN	Grafcet/SFC	SIPN	Other Fuzzy PNs
hierarchy/modularization	✓	✓	✓	✗
analogue signals	✓	✗	✗	✓
software tool support	✓	✓	✓	✗
modeling of resources using a net structure	✓	✗	✗	✗
automatic executable code generation	✓/✗	✓	✓	✗
automatic investigation of properties	✓/✗	✓/✗	✓	✗

2. The Formal Basis and the Concept of FIPN

Three definitions describe the formal basis of FIPN. The first presents the net construction.

Definition 1. *The fuzzy interpreted Petri net is the system [59]:*
FIPN $= (P, T, \Omega, \Psi, R, \Delta, K, W, \Gamma, \Theta, M_0, e)$, *where:*
$P = P' \cup P''$ *is a nonempty finite set of places, where:*
$P' = \{p'_1, p'_2, ..., p'_{a'}\}$ *is a set of places for processes modeling,*
$P'' = \{p''_1, p''_2, ..., p''_{a''}\}$ *is a set of places for resources modeling;*
$T = \{t_1, t_2, ..., t_b\}$ *is a nonempty finite set of transitions;*
$\Omega = \{\omega_1, \omega_2, ..., \omega_{a'+a''}\}$ *is a nonempty finite set of statements;*
$\Psi = \{\psi_1, \psi_2, ..., \psi_b\}$ *is a nonempty finite set of conditions;*
P, T, Ω, Ψ *are disjoint sets;*
$R \subseteq (P \times T) \cup (T \times P)$ *is the incidence relation that assigns a place to each transition* t_i $(1 \leqslant i \leqslant b)$ *where there is the place* $p' \in P'$, *such that* $(p', t_i) \in R$ *or* $(t_i, p') \in R$;
$\Delta\colon P \to \Omega$ *is the function assigning a statement to each place;*
$K\colon P' \to \{1\}$ *and* $P'' \to \aleph \backslash \{1\}$ *are the functions assigning a capacity to each place, where* $\aleph = \{1, 2, ...\}$;
$\Gamma\colon T \to \Psi$ *is the function assigning a condition to each transition;*
$\Theta\colon T \to [0, 1]$ *is the function defining the degree to which the condition corresponding to the each transition is satisfied;*
$W\colon R \to \aleph$ *is the weight function which* $\forall p \in P$ *and* $\forall t \in T$ *meets two conditions:*
$W(p, t) \leq K(p)$,
$W(t, p) \leq K(p)$;
$M_0\colon P' \to \{0, 1\}$ *and* $P'' \to \mathbb{W}_+$ *are the initial marking functions, where:*
$M_0(p''_j) = z_j / K(p''_j)$,
$z_j \in \aleph \cup \{0\}$,
$z_j \leq K(p''_j)$,
$1 \leq j \leq a''$,
$\mathbb{W}_+$ *is a set of non-negative rational numbers;*
e *is an event that synchronizes the work of all transitions.*

FIPN can be represented as a bipartite graph. An exemplary net and a system controlled by it (in different states) are presented in Figure 1. There exist two types of places in the net: p'-type—p_1 and p_2 (for process modeling) and p''-type—p_3 (for resource modeling). They are shown as circles. For both types, the marking is a real number in the range [0, 1] located inside the circle. However, the marking of p''-type places is presented as a fraction and can hold more than one token. The denominator holds the capacity of a place, while the numerator holds the number of tokens.

Moreover, statements can be assigned to p'-type places to set the value of process variables. In Figure 1 actions *start* and *stop* are assigned to places p_1 and p_2 for the control of a dispensing process. Transitions are represented by rectangles and can be related to binary or analogue signals. Transition t_1 is synchronized by an analogue signal A_1 representing the weight of the material transferred to the truck (Figure 1). In FIPN, analogue signals are normalized into the range [0, 1]. As can be seen in the figure, the transfer of a token across the fired transition is a process with a duration longer than one clock cycle which synchronizes the net operation. This duration depends on the increment of the sensor value. Such operation of transitions allows for better visualization as well as for more precise control in comparison to discrete PNs. Additionally, some logic conditions (an example is shown in Figure 3) can be assigned to transitions.

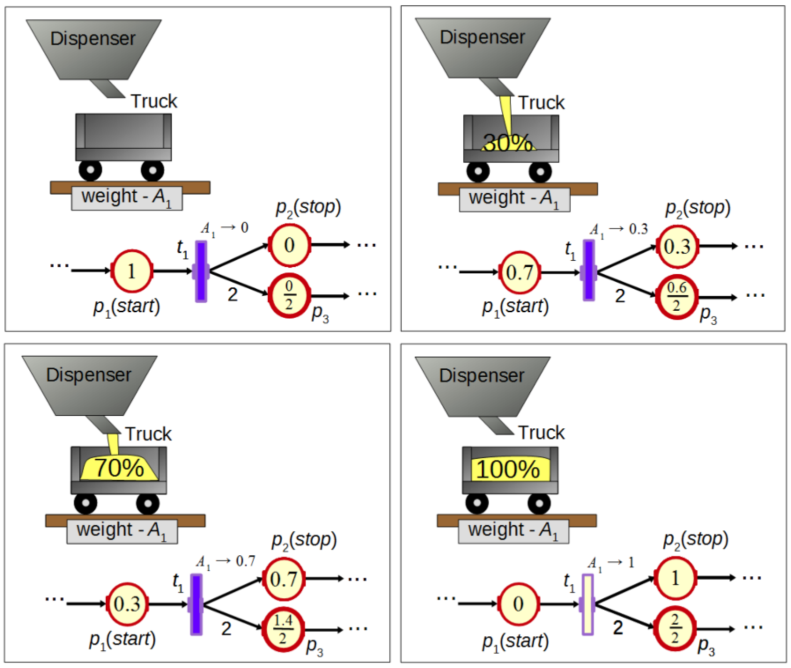

Figure 1. An algorithm in the form of a FIPN graph to control the dispensing process.

The basic conditions for transition activation and deactivation are described in Definition 2.

Definition 2. *The transition $t \in T$ with marking $M\colon P \to [0,1]$ is enabled when the degree to fulfill the condition $\Theta(t) = \theta$, assigned to the transition, is greater than zero and the following conditions are satisfied [59]:*

$$\forall p \in {}^{\bullet}t, M(p) \geq W(p,t)/K(p) \tag{1}$$

and

$$\forall p \in t^{\bullet}, M(p) \leq 1 - W(t,p)/K(p), \tag{2}$$

until:

$$\exists p' \in {}^{\bullet}t, M(p') = 0 \tag{3}$$

or

$$\exists p' \in t^{\bullet}, M(p') = 1, \tag{4}$$

where:

$^{\bullet}t = \{p \in P | (p,t) \in R\}$ *is the set of input places of* t,

$t^{\bullet} = \{p \in P | (t,p) \in R\}$ *is the set of output places of* t.

The transfer of tokens from input to output places across the transition can begin when conditions (1) and (2) of Definition 2 are satisfied, while it ends when condition (3) or (4) is fulfilled.

The change of the marking for places connected to the enabled transition depends on the increment of the degree to which the condition corresponding to the transition is satisfied. The method for calculating the new marking is described by Definition 3. The transition remains active until the tokens are transferred from the input places to the output places.

Definition 3. *Let M be the marking for which the transition $t \in T$ is enabled, the degree $\Theta(t) = \theta$ (where $\theta \in [0,1]$), to which the condition corresponding to the enabled transition is satisfied, will be changed by $\Delta\theta \geq 0$ and there will be an event e synchronizing the work of all transitions. The new marking of the net M' can be computed by the following rule [59]:*

$$M'(p) = \begin{cases} M(p) - \Delta\theta \cdot \dfrac{W(p,t)}{K(p)} \text{ for } p \in {}^{\bullet}t \setminus t^{\bullet}, \\ M(p) + \Delta\theta \cdot \dfrac{W(t,p)}{K(p)} \text{ for } p \in t^{\bullet} \setminus {}^{\bullet}t, \\ M(p) - \Delta\theta \cdot \dfrac{W(p,t) - W(t,p)}{K(p)} \text{ for } p \in t^{\bullet} \cap {}^{\bullet}t, \\ M(p) \text{ for } p \notin t^{\bullet} \cup {}^{\bullet}t. \end{cases} \tag{5}$$

The increment $\Delta\theta < 0$ does not introduce any changes in the net marking.

It has been mentioned that FIPN gives better visualization effect in comparison to discrete PNs. This advantage can be seen in Figure 2 which presents the discrete PN graph and the FIPN graph (in different states) used for the control of heating, measuring time and filling the tank. Discrete PN displays only the actions that are currently performed, while FIPN additionally presents the progress of these actions.

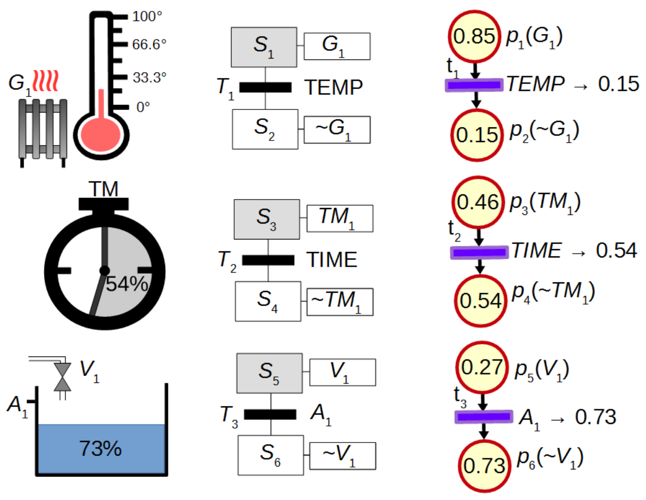

Figure 2. The advantage of FIPN visualization in comparison to discrete PNs.

Another advantage of FIPN in comparison to discrete PNs is the possibility of the dynamic control in response to changing values of analogue signals or fuzzy marking of places. An example of such control is shown in Figure 3. If the temperature is too low after 70% of the measured time (logic condition assigned to transition t_4), the additional heater

G_2 is turned on (place p_8). If the temperature increases with enough speed (t_5), no action is needed (p_9). Such dynamic control is also presented in Section 4 (macroplace mp_4).

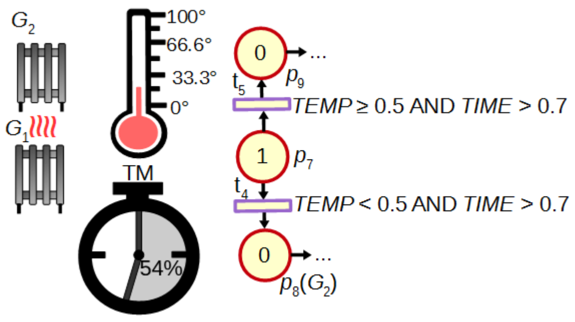

Figure 3. A dynamic control of heating.

3. New HFIPN Concept and Its Definitions

In this section, a new concept and definitions of HFIPN are presented. The proposed net enables the creation of subnets with multiple input, output, and input-output places as well as the use of macroplace instances. Moreover, based on a HFIPN graph, a hierarchy graph is created. All of these functionalities allow for good adaptation of HFIPN to practical needs. The way of summing subnets and the conversion of HFIPN to its flat version are also described. Finally, a new algebraic representation and a brief introduction of HFIPN-SML features are given.

3.1. The Concept of HFIPN

In this subsection, the conception of HFIPN is presented. Figure 4 shows an example of the hierarchical net consisting of two subnets N_0 and N_1. The subnet N_0 contains the macroplace mp_1 which includes the subnet N_1. The macroplace mp_1 is an internal macroplace of the subnet N_0 and is overriding for the subnet N_1. In the hierarchical net, a macroplace can have several input, output, and input-output places that are visible in its overriding and internal subnets. Input places can only have input arcs in the overriding subnet of a macroplace, while output places can only have output arcs. Input-output places can have both types of arcs in the overriding subnet. The macroplace mp_1 (Figure 4) contains two input places (p_4 and p_5), two output places (p_9 and p_{10}), and one input-output place (p_6). The place p_6 has an input and output arc in the subnet N_0 and two output arcs in the subnet N_1. In the subnet N_1, the place p_6 can also have an input arc.

HFIPN enables the application of macroplace instances, which facilitates the use of modularization in a net. This functionality is analogous to that used in CPN [31]. It allows creating several macroplaces (subnets) that have the same structure: they have the same number of places with the same capacity, the same number of transitions and arcs of the same weights connecting the corresponding nodes. In the proposed concept, macroplaces that are instances of the same type do not share nodes and arcs. Therefore, the places, transitions, and arcs of one macroplace are always separate from the elements of other macroplaces, and in the algebraic representation, they are represented by different numbers in all vectors and an incidence matrix. However, the concept of macroplace instances allows for simultaneous modification of macroplaces of the same type.

The concept of macroplace instances entails the introduction of place, transition and arc instances, but only for those elements that are inside macroplaces. Places that are instances of the same type may be associated with the same statements, while transitions may be associated with the same conditions. Therefore, instances of places and transitions belonging to macroplace instances can be created in two ways in HFIPN. In the first approach, the statements assigned to the places and conditions associated with the transitions are shared between nodes of the same type, while in the second approach they are separate. In Figure 5 the first type of place and transition instances is presented, in which statements and conditions are shared. There are two macroplaces mp_1 and mp_2 in the net, that are of the same type $mtype_1$. The places p_1 and p_2 from the subnet N_1 are functionally the same places as p_3 and p_4 from the subnet N_2. They set the variable O_1. The same applies to the transitions t_1 and t_2, whose degrees of the conditions fulfillment are synchronized with the variable A_1. In Figure 6, the second type of macroplace instances with separate statements and conditions is presented. The macroplaces mp_1 and mp_2 do not share the output variables assigned to places. The places p_1 and p_2 set the variable O_1, and the places p_3 and p_4 set O_2. The variables synchronizing the degrees to which the conditions corresponding to the transitions are satisfied for transitions t_1 (variable A_1) and t_2 (A_2) are also not shared between macroplaces. The concept of macroplace instances with shared variables can be used when a certain part of a system is controlled by several modules in the same way—Figure 7a. The second approach with unshared variables is used when similar parts of a system are controlled in the same way by different macroplaces—Figure 7b.

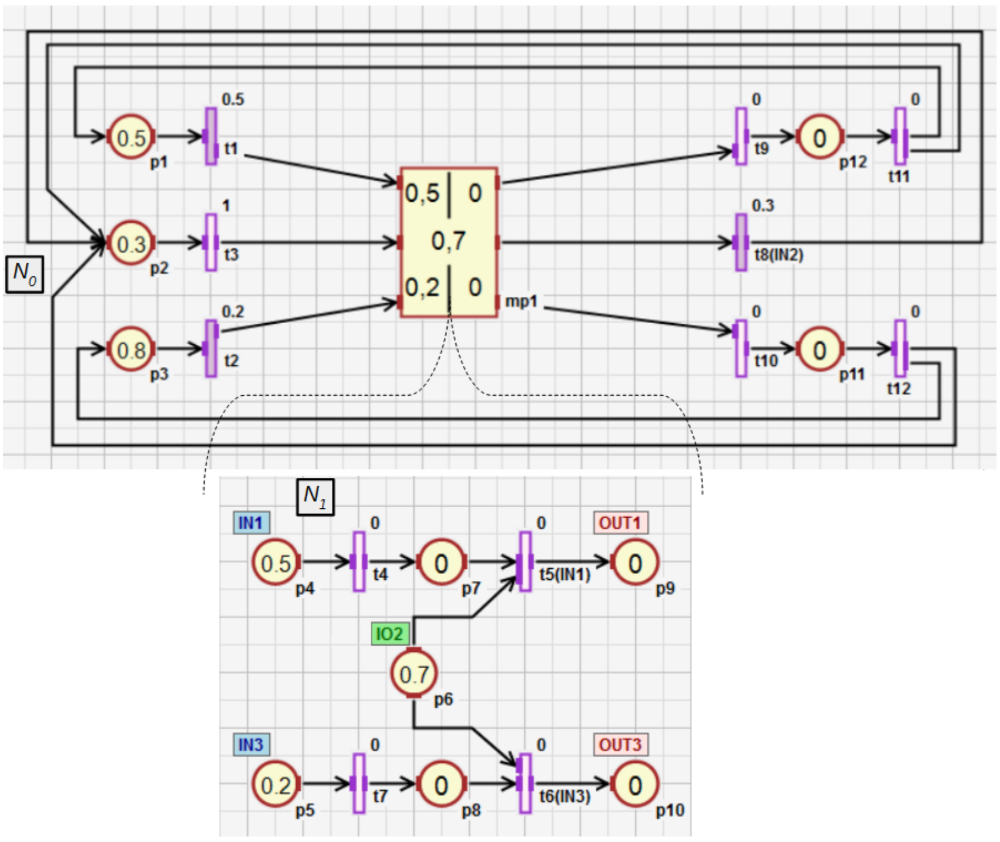

Figure 4. HFIPN graph with the macroplace having two input places, two output places and one input-output place.

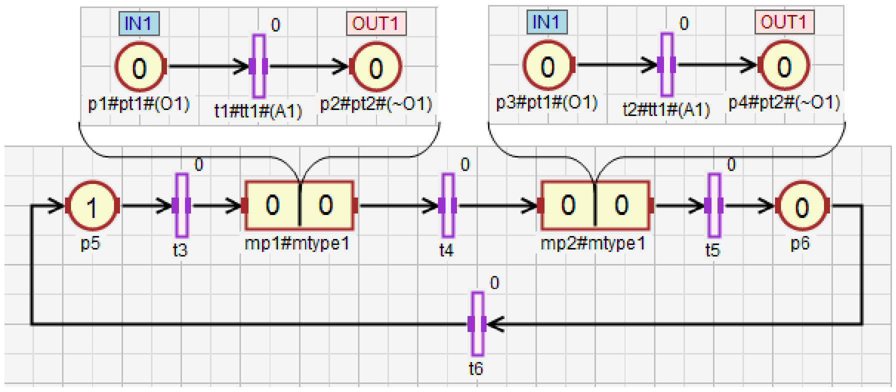

Figure 5. The net with two macroplaces of the same type and shared input and output variables.

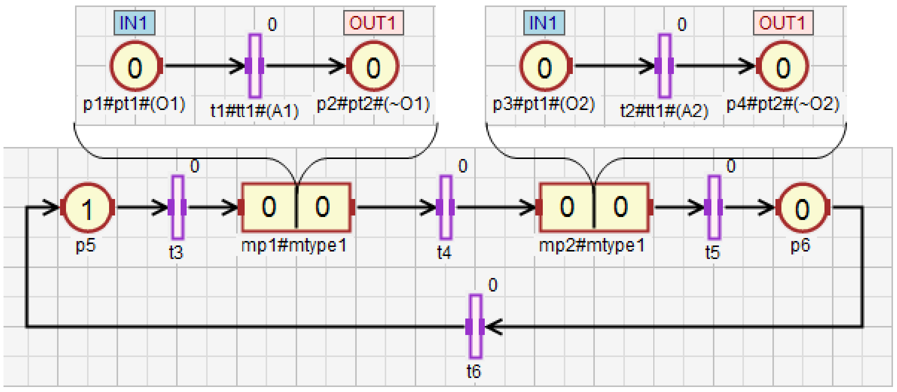

Figure 6. The net with two macroplaces of the same type, in which input and output variables are not shared.

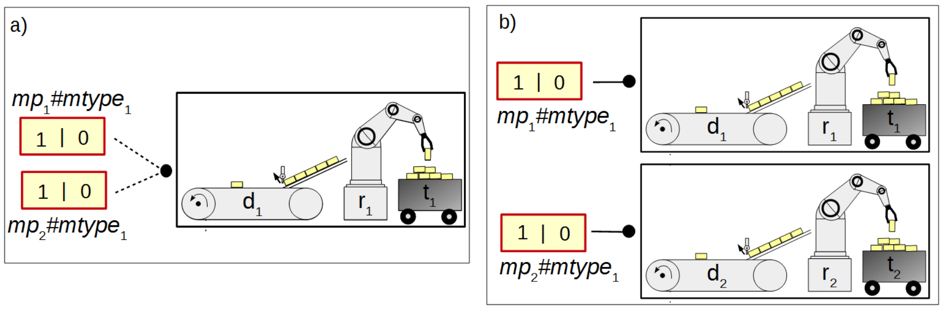

Figure 7. The application of two concepts of macroplace instances: (**a**) shared variables, (**b**) unshared variables.

3.2. Formal Description of HFIPN

Three definitions specify the structure and operation of HFIPN. The first of them, the main definition of a whole net, is as follows.

Definition 4. *The hierarchical FIPN is the system:*
$HFIPN = (N, MP, IO, INS)$*, where:*
$N = \{N_0, N_1, ..., N_d\}$ *is a nonempty finite set of subnets described in Definitions 5 and 6;*
$MP = \{mp_1, mp_2, ..., mp_d\}$ *is an finite set of macroplaces, such that* $\forall mp_i \in MP$*:*
$mp_i = \{N_i, P_{mp(i)}\}$*, where:*
$N_i \in N (1 \le i \le d)$ *is a subnet, for which the macroplace* mp_i *is overriding (the subnet* N_i *is internal for the macroplace* mp_i*);*
$P_{mp(i)} = P^{in}_{mp(i)} \cup P^{out}_{mp(i)} \cup P^{io}_{mp(i)}$ *is a nonempty finite set of input, output and input-output places of a macroplace;*
$IO\colon P \to \{in, out, inout, normal\}$ *is a function determining the place type;*
INS *is the function that returns the instance type for any node or arc:*

$$INS = \begin{cases} MP \to MTYPE, \\ P \to PTYPE, \\ T \to TTYPE, \\ R \to RTYPE, \end{cases}$$

where:
$MTYPE = \{mtype_1, mtype_2, ...\}$ *is a set of instance types of macroplaces,*
$PTYPE = \{ptype_1, ptype_2, ...\}$ *is a set of instance types of places,*
$TTYPE = \{ttype_1, ttype_2, ...\}$ *is a set of instance types of transitions,*
$RTYPE = \{rtype_1, rtype_2, ...\}$ *is a set of instance types of arcs.*

By Definition 4, each subnet except N_0 has exactly one overriding macroplace. Moreover, HFIPN is divided into $d + 1$ subnets. If $d = 0$, the net is flat, and if $d > 0$, the net is hierarchical. The subnet N_0 is the main or top level subnet. The remaining subnets are called lower level subnets or macronets. Definition 4 also shows that the elements of a macroplace are sets of input, output, and input-output places. To distinguish the types of places in the net, the global function IO was added to the main definition, which allows specifying the type of each place in the net, whether it is an input, output, input and output of a macroplace or a normal place. Each macroplace in HFIPN is an instance of a given type. The remaining elements (places, transitions, and arcs) are instances if they occur inside macroplaces (e.g., the place p_1 in Figure 5 is an instance of type pt_1). However, the elements are not instances if they occur directly in the main subnet N_0 (e.g., the place p_5 in Figure 5).

Definition 5 describes the structure of each subnet in HFIPN.

Definition 5. *Each subnet $N_i, 0 \leq i \leq d$ is defined as follows:*
$N_i = (MP_i, P_i, T_i, \Omega_i, \Psi_i, R_i, \Delta_i, K_i, W_i, \Gamma_i, \Theta_i, M_{0(i)}, e)$, *where:*
$MP_i \subseteq MP$ *is a finite set of internal macroplaces of the subnet N_i (N_i is an overriding for these macroplaces);*
$P_i \subseteq P$ *and* $P_i = P_i' \cup P_i'' \cup P_i^{in} \cup P_i^{out} \cup P_i^{io}$ *is a nonempty finite set of places (in the subnet N_i), where:*
$P_i' \subseteq P'$ *is a set of places of the type p',*
$P_i'' \subseteq P''$ *is a set of places of the type p'',*
$P_{mp(i)} \subseteq P_i' \cup P_i''$ *for $i \neq 0$,*
$P_i^{in} = P_i'^{in} \cup P_i''^{in}$ *is a set of input places (of the type p' or p'') of internal macroplaces from the set MP_i,*
$P_i^{out} = P_i'^{out} \cup P_i''^{out}$ *is a set of output places (of the type p' or p'') of internal macroplaces from the set MP_i,*
$P_i^{io} = P_i'^{io} \cup P_i''^{io}$ *is a set of inputoutput places (of the type p' or p'') of internal macroplaces from the set MP_i,*
$P_i', P_i'', P_i'^{in}, P_i''^{in}, P_i'^{out}, P_i''^{out}, P_i'^{io}, P_i''^{io}$ *any two of them have no common elements;*
$T_i \subseteq T$ *is a nonempty finite set of transitions;*
$\Omega_i \subseteq \Omega$ *is a nonempty finite set of statements;*
$\Psi_i \subseteq \Psi$ *is a nonempty finite set of conditions;*
$R_i \subseteq R$ *and* $R_i \subseteq ((P_i' \cup P_i'' \cup P_i^{out} \cup P_i^{io}) \times T_i) \cup (T_i \times (P_i' \cup P_i'' \cup P_i^{in} \cup P_i^{io}))$ *is a subnet incidence relation;*
$\Delta_i \colon P_i \to \Omega_i$ *is the function assigning a statement to each place;*
$K_i \colon (P_i' \cup P_i'^{in} \cup P_i'^{out} \cup P_i'^{io}) \to \{1\}$ *and* $(P_i'' \cup P_i''^{in} \cup P_i''^{out} \cup P_i''^{io}) \to \aleph \backslash \{1\}$ *are the functions assigning a capacity to each place;*
$\Gamma_i \colon T_i \to \Psi_i$ *is the function assigning a condition to each transition;*
$\Theta_i \colon T_i \to [0,1]$ *is the function defining the degree to which the condition corresponding to the transitions is satisfied;*
$W_i \colon R_i \to \aleph$ *is the weight function, which for $\forall p \in P_i$ and $\forall t \in T_i$ meets two conditions:*
$W_i(p,t) \leq K_i(p)$,
$W_i(t,p) \leq K_i(p)$;
$M_{0(i)} \colon (P_i' \cup P_i'^{in} \cup P_i'^{out} \cup P_i'^{io}) \to \{0,1\}$ *and* $(P_i'' \cup P_i''^{in} \cup P_i''^{out} \cup P_i''^{io}) \to \mathbb{W}_+$ *are the initial marking functions, where for each* $p'' \in (P_i'' \cup P_i''^{in} \cup P_i''^{out} \cup P_i''^{io})$ *exists* $z \in \aleph \cup \{0\}$, *such that:*
$z \leq K_i(p'')$,
$M_{0(i)}(p'') = z / K_i(p'')$;
e is the global event which synchronizes the work of all transitions.

By Definition 5 each set of places P_i, the transitions T_i, the conditions Ψ_i, the statements Ω_i and the arcs R_i, which belongs to the subnet N_i, is a subset of the flat net set: P, T, Ψ, Ω and R respectively. From Definitions 4 and 5, it follows that the main subnet may have internal macroplaces, which are overriding macroplaces for some subnet, which, in turn, may contain further macroplaces, etc. The level of maximum depth in HFIPN is not limited in the definition. Subnets/macroplaces only in a special case may have common elements (places) with other subnets/macroplaces, which is described in Definition 6 specifying the relationship between sets of subnets in HFIPN.

Definition 6. *For any two subnets N_i, $N_j \in N$ for $0 \le i \le d$, $0 \le j \le d$ and $i \neq j$ the relationship between the subsets of places, transitions, statements, macroplaces, conditions assigned to transitions, arcs is defined as follows:*

$$P_i \cap P_j = \begin{cases} P_{mp(i)} \text{ for } mp_i \in MP_j \text{ and } i \neq 0, \\ P_{mp(j)} \text{ for } mp_j \in MP_i \text{ and } j \neq 0, \\ \varnothing \text{ in other cases,} \end{cases} \tag{6}$$

$$T_i \cap T_j = \varnothing, \tag{7}$$

$$\Omega_i \cap \Omega_j = \varnothing, \tag{8}$$

$$\Psi_i \cap \Psi_j = \varnothing, \tag{9}$$

$$R_i \cap R_j = \varnothing, \tag{10}$$

$$MP_i \cap MP_j = \varnothing, \tag{11}$$

and for additionally fulfilled conditions $d > 1$ and $i, j \neq 0$:

$$P_{mp(i)} \cap P_{mp(j)} = \varnothing. \tag{12}$$

Definition 6 shows that for any two subnets, for which the overriding macroplace of one of them is the internal macroplace of the other, the common part is the set of input, output and input-output places of this macroplace (Equation (6)). The example of such a set in Figure 4 is the set $P_{mp(1)} = \{p_4, p_5, p_6, p_9, p_{10}\}$. Places from this set belong to the subnets N_0 and N_1. However, the introduced restrictions do not allow the input and output place of the first macroplace/subnet to be the input and output place of the second macroplace/subnet (Equation (12)). This prevents the possibility of the creation of an infinite number of macroplaces for a finite number of places.

3.3. The Hierarchy in HFIPN

This subsection presents a hierarchy graph, allowing to define nesting relations between macroplaces (subnets) in HFIPN and facilitates navigation to various areas of the net. The terms defining the degree of the nesting relation: ancestor, direct ancestor, descendant, and direct descendant are also introduced. The special function $*()$ is used to formulate the concept of ancestor and descendant, and in the next subsection to define an algebraic representation.

First, the function $*()$ is introduced. For each subnet $N_i \in N$, $N_i = (MP_i, P_i, T_i, \Omega_i, \Psi_i, R_i, \Delta_i, K_i, W_i, \Gamma_i, \Theta_i, M_{0(i)}, e)$ the function $*(N_i)$, abbreviated $*N_i$, is defined as follows:

$$*\, N_i = \begin{cases} N_i \quad \text{for} \quad MP_i = \varnothing, \\ N_i \cup \sum\limits_{\forall mp_j \in MP_i} *\, N_j \quad \text{otherwise.} \end{cases} \tag{13}$$

Definition 7 specifies what is the sum of two subnets.

Definition 7. *For any two subnets $N_i, N_j \in N$ for $0 \le i \le d$, $0 \le j \le d$ and $i \ne j$ their sum N_k is calculated as follows:*
$N_k = N_i \cup N_j = (MP_k, P_k, T_k, \Omega_k, \Psi_k, R_k, \Delta_k, K_k, W_k, \Gamma_k, \Theta_k, M_{0(k)}, e)$, *where:*

$$MP_k = \begin{cases} (MP_i \cup MP_j) \backslash \{mp_i\} \text{ for } mp_i \in MP_j \text{ and } i \ne 0, \\ (MP_i \cup MP_j) \backslash \{mp_j\} \text{ for } mp_j \in MP_i \text{ and } j \ne 0, \\ MP_i \cup MP_j \text{ in the other cases;} \end{cases} \tag{14}$$

$$P_k = P'_k \cup P''_k \cup P^{in}_k \cup P^{out}_k \cup P^{io}_k, \textit{where}: \tag{15}$$

$$P'_k = P'_i \cup P'_j, \tag{16}$$

$$P''_k = P''_i \cup P''_j, \tag{17}$$

$$P^{in}_k = \begin{cases} (P^{in}_i \cup P^{in}_j) \backslash P^{in}_{mp(i)} \text{ for } mp_i \in MP_j \text{ and } i \ne 0, \\ (P^{in}_i \cup P^{in}_j) \backslash P^{in}_{mp(j)} \text{ for } mp_j \in MP_i \text{ and } j \ne 0, \\ P^{in}_i \cup P^{in}_j \text{ in the other cases,} \end{cases} \tag{18}$$

$$P^{out}_k = \begin{cases} (P^{out}_i \cup P^{out}_j) \backslash P^{out}_{mp(i)} \text{ for } mp_i \in MP_j \text{ and } i \ne 0, \\ (P^{out}_i \cup P^{out}_j) \backslash P^{out}_{mp(j)} \text{ for } mp_j \in MP_i \text{ and } j \ne 0, \\ P^{out}_i \cup P^{out}_j \text{ in the other cases,} \end{cases} \tag{19}$$

$$P^{io}_k = \begin{cases} (P^{io}_i \cup P^{io}_j) \backslash P^{io}_{mp(i)} \text{ for } mp_i \in MP_j \text{ and } i \ne 0, \\ (P^{io}_i \cup P^{io}_j) \backslash P^{io}_{mp(j)} \text{ for } mp_j \in MP_i \text{ and } j \ne 0, \\ P^{io}_i \cup P^{io}_j \text{ in the other cases;} \end{cases} \tag{20}$$

$$T_k = T_i \cup T_j; \tag{21}$$

$$\Omega_k = \Omega_i \cup \Omega_j; \tag{22}$$

$$\Psi_k = \Psi_i \cup \Psi_j; \tag{23}$$

$$R_k = R_i \cup R_j; \tag{24}$$

$$\Delta_k \colon P_k \to \Omega_k; \tag{25}$$

$$K_k \colon (P'_k \cup P'^{in}_k \cup P'^{out}_k \cup P'^{io}_k) \to 1 \text{ and } (P''_k \cup P''^{in}_k \cup P''^{out}_k \cup P''^{io}_k) \to \aleph \backslash \{1\}; \tag{26}$$

$$\Gamma_k \colon T_k \to \Psi_k; \tag{27}$$

$$\Theta_k \colon T_k \to [0,1]; \tag{28}$$

$$W_k \colon R_k \to \aleph; \tag{29}$$

$$M_{0(k)} \colon (P'_k \cup P'^{in}_k \cup P'^{out}_k \cup P'^{io}_k) \to \{0,1\} \text{ and } (P''_k \cup P''^{in}_k \cup P''^{out}_k \cup P''^{io}_k) \to \mathbb{W}_+. \tag{30}$$

By Definition 7, the sum of the subnets N_i and N_j is the subnet N_k, which consists of subsets (transitions, statements, conditions, and arcs), which are the sums of the corresponding subsets from the subnets N_i and N_j. The subsets of macroplaces and places can be summed in a different way. If, for instance, the macroplace mp_i is internal for the subnet N_j, then mp_i is removed from the result subset of internal macroplaces of N_k (Mp_k). Similar operations occur for the set of places P_k (if mp_i is internal for N_j). The input places $P^{in}_{mp(i)}$, the output places $P^{out}_{mp(i)}$ and the input-output places $P^{io}_{mp(i)}$ of mp_i are removed from the sets P^{in}_k, P^{out}_k and P^{io}_k, respectively. These places are still the elements of the normal place sets of the subnet N_k, i.e., $P'_k \cup P''_k$.

In the HFIPN-SML simulator, there is no need to continuously add subnets to create the algebraic representation described in Section 3.4. For each subnet, the vectors and the incidence matrix are updated on-the-fly when places, transitions, and arcs are added to the subnet. The summation operation is used when a user wants to undo adding a new macroplace based on a selected fragment of the subnet.

Definition 7 and the function (13) allow for the transformation of a hierarchical net (HFIPN) into its flat equivalent (FIPN). This conversion involves the addition of all subnets of the hierarchical net. Some sets of the obtained net, i.e., the set of internal macroplaces and the sets of input, output, and input-output places of the internal macroplaces are empty:

$$FIPN = *N_0 \tag{31}$$

Based on Definition 7, the following functions for subsets of subnets can be created: $*(N_i, P_i)$, $*(N_i, MP_i)$, $*(N_i, T_i)$, $*(N_i, \Omega_i)$, $*(N_i, \Psi_i)$ and $*(N_i, R_i)$ (abbreviated $*P_i$, $*MP_i$, $*T_i$, $*\Omega_i$, $*\Psi_i$ and $*R_i$). They are defined as follows:

$$*P_i = \begin{cases} P_i & \text{for} \quad MP_i = \varnothing, \\ P_i \cup \sum\limits_{\forall mp_j \in MP_i} *P_j & \text{otherwise,} \end{cases} \tag{32}$$

$$*MP_i = \begin{cases} MP_i & \text{for} \quad MP_i = \varnothing, \\ MP_i \cup \sum\limits_{\forall mp_j \in MP_i} *MP_j & \text{otherwise,} \end{cases} \tag{33}$$

$$*T_i = \begin{cases} T_i & \text{for} \quad MP_i = \varnothing, \\ T_i \cup \sum\limits_{\forall mp_j \in MP_i} *T_j & \text{otherwise,} \end{cases} \tag{34}$$

$$*\Omega_i = \begin{cases} \Omega_i & \text{for} \quad MP_i = \varnothing, \\ \Omega_i \cup \sum\limits_{\forall mp_j \in MP_i} *\Omega_j & \text{otherwise,} \end{cases} \tag{35}$$

$$*\Psi_i = \begin{cases} \Psi_i & \text{for} \quad MP_i = \varnothing, \\ \Psi_i \cup \sum\limits_{\forall mp_j \in MP_i} *\Psi_j & \text{otherwise,} \end{cases} \tag{36}$$

$$*R_i = \begin{cases} R_i & \text{for} \quad MP_i = \varnothing, \\ R_i \cup \sum\limits_{\forall mp_j \in MP_i} *R_j & \text{otherwise.} \end{cases} \tag{37}$$

Functions $*()$ (Equations (32)–(37)) were defined for subsets of places, macroplaces, transitions, statements, conditions, and arcs. Their notation used without parentheses, e.g., $*P_i$ for $*(N_i, P_i)$ can be treated as the set returned by them, which simplifies their definition. After defining the functions $*()$, the formal definitions of nesting relations between subnets can be formulated.

Definition 8. *The subnet N_i ($N_i \in N$) is a direct ancestor for the subnet N_j ($N_j \in N$) if the overriding macroplace mp_j ($mp_j = \{N_j, P_{mp(j)}\}$) of the subnet N_j belongs to internal macroplaces of the subnet N_i ($mp_j \in MP_i$).*

Definition 9. *The subnet N_i ($N_i \in N$) is a direct descendant for the subnet N_j ($N_j \in N$) if the overriding macroplace mp_i ($mp_i = \{N_i, P_{mp(i)}\}$) of the subnet N_i belongs to internal macroplaces of the subnet N_j ($mp_i \in MP_j$).*

Definition 10. *The subnet N_i ($N_i \in N$) is a ancestor for the subnet N_j ($N_j \in N$) if the overriding macroplace mp_j ($mp_j = \{N_j, P_{mp(j)}\}$) of the subnet N_j belongs to the set $*MP_i$ created according to Equation (33).*

Definition 11. *The subnet N_i ($N_i \in N$) is a descendant for the subnet N_j ($N_j \in N$) if the overriding macroplace mp_i ($mp_i = \{N_i, P_{mp(i)}\}$ of the subnet N_i belongs to the set $*MP_j$ created according to Equation* (33).

In Definitions 8–11, the nesting relations between subnets were defined. Nesting relations for macroplaces can be defined analogously. Therefore, the concepts of descendant, ancestor, direct descendant, and direct ancestor can also be used for macroplaces. The nesting relations and the function $*()$ will be additionally shown on the example of a hierarchical net, which is presented in Figure 8. It contains three macroplaces. The macroplaces mp_1 and mp_2 belong to the subnet N_0, and the macroplace mp_3 belongs to the internal subnet of the macroplace mp_1. The macroplaces mp_2 and mp_3 are instances of the same type. For the subnet N_0 the set $*N_0$ has the form $N_0 \cup N_1 \cup N_2 \cup N_3$, i.e., it contains this subnet and all of its descendant subnets. In turn, the direct descendants of the subnet N_0 are the subnets N_1 and N_2. A direct ancestor of the subnet N_3 is the subnet N_1, and all the ancestors of N_3 are subnets N_0 and N_1. The subnet N_3 has no descendants. For the main subnet N_0, the set $*P_0$ consists of all places in the net $\{p_1, p_2, ..., p_{13}\}$, i.e., in the subnet N_0 and in its descendant subnets N_1, N_2 and N_3, while the set $*P_3$ has the form $\{p_9, p_{10}, p_{11}, p_{12}\}$. The set $*T_0$ consists of all the transitions $\{t_1, t_2, ..., t_{13}\}$ in the net, while $*T_1$ consists of the transitions $\{t_8, t_9, ..., t_{13}\}$ from the subnet N_1 and all of its descendant subnets (N_3 in this case).

After defining the concepts related to the degree of nesting and function $*()$, the HFIPN hierarchy graph is formalized.

Definition 12. *The hierarchy graph of HFIPN (HG_HFIPN) is a directed tree, such that:*
HG_HFIPN $= (mp_0^{hg}, MP^{hg}, R^{hg})$, *where:*
mp_0^{hg} *is a root of the tree representing the main subnet N_0 from Definition 4 marked as main;*
$MP^{hg} = \{mp_1^{hg}, mp_2^{hg}, ..., mp_d^{hg}\}$ *is a set of remaining nodes of the tree representing macroplaces from the set MP from Definition 4;*
R^{hg} *is a set of arcs connecting nodes, consisting of the elements such that $\forall mp_i^{hg}, mp_j^{hg} \in MP^{hg} \cup \{mp_0^{hg}\}$, $0 \leq i, j \leq d$, $i \neq j$ exists the ordered pair of nodes $(mp_i^{hg}, mp_j^{hg}) \in R^{hg}$ if and only if the macroplace mp_j is an internal macroplace of the subnet N_i ($mp_j \in MP_i$).*

A hierarchy graph is a directed graph in which nodes represent subnets (macroplaces) described in Definition 4. A hierarchy graph for the net from Figure 8 is shown in Figure 9. The graph allows for quick reading and understanding of subnets/macroplaces nesting, the number of instances and their nesting in HFIPN. A special node of the graph is the root representing the main subnet N_0, whose name is *main* since N_0 does not have an overriding macroplace. If there is an internal macroplace in a given subnet, the node representing it in the hierarchy graph is connected to the node representing the overriding macroplace for this subnet. The direction of the arcs is visible in the graphical representation by arranging the nodes. The node, to which the arc leads (e.g., mp_3 in Figure 9), is located below and to the right of the node (mp_1) which is the starting point of the arc. Arcs in the hierarchy graph are unlabeled. The advantage of the graph is the presentation of the hierarchical structure of the net: the level of depth and the indication of the way of macroplace nestings in the net. It also allows determining the number of instances of a given macroplace type, because the labels of nodes (except *main*) are represented by a pair consisting of the macroplace name and the instance type of this macro. Moreover, the software version of the hierarchy graph in the HFIPN-SML simulator allows a user to click on any node to move inside the macroplace (subnet) represented by that node.

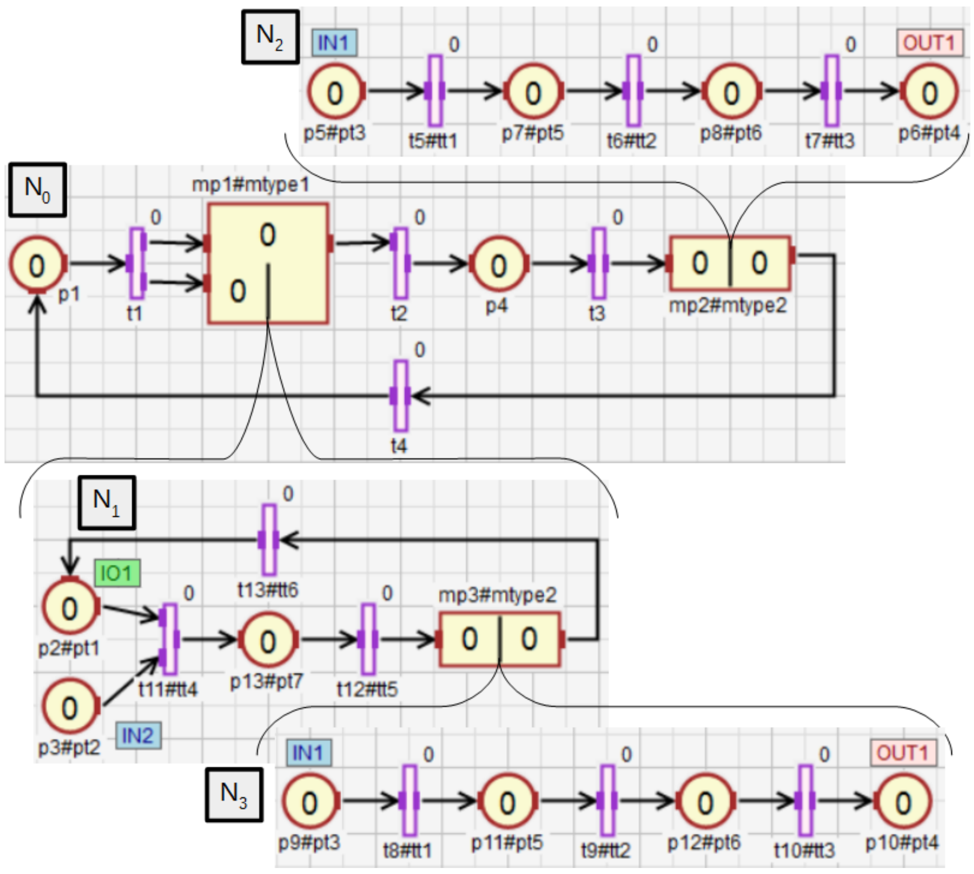

Figure 8. The net with three macroplaces, two of which are of the same type.

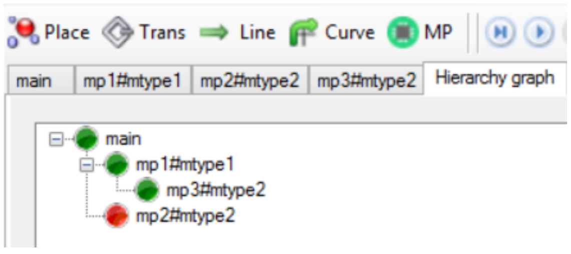

Figure 9. The hierarchy graph for the subnet from Figure 8.

3.4. The Algebraic Representation

In this subsection, the algebraic representation of HFIPN is presented. First, the auxiliary functions are defined and discussed:

$$*\Delta_i\colon *P_i \to *\Omega_i, \tag{38}$$

$$\begin{aligned} *K_i\colon (*P_i' \cup *P_i'^{in} \cup *P_i'^{out} \cup *P_i'^{io}) &\to \{1\} \text{ and} \\ (*P_i'' \cup *P_i''^{in} \cup *P_i''^{out} \cup *P_i''^{io}) &\to \aleph\backslash\{1\}. \end{aligned} \tag{39}$$

$$*\Gamma_i\colon *T_i \to *\Psi_i, \tag{40}$$

$$*\Theta_i\colon *T_i \to [0,1], \tag{41}$$

$$*W_i\colon *R_i \to \aleph, \tag{42}$$

$$\begin{aligned} *M_{0(i)}\colon (*P_i' \cup *P_i'^{in} \cup *P_i'^{out} \cup *P_i'^{io}) &\to \{0,1\} \text{ and} \\ (*P_i'' \cup *P_i''^{in} \cup *P_i''^{out} \cup *P_i''^{io}) &\to \mathbb{W}_+. \end{aligned} \tag{43}$$

Functions (38)–(43) extend the following functions: assigning statements to places $\Delta_i()$, returning the places capacity $K_i()$, assigning conditions to transitions $\Gamma_i()$, determining the degrees to which the conditions corresponding to the transitions $\Theta_i()$, as well as weight functions $W_i()$ and initial marking functions $M_{0(i)}()$, which are all defined within a subnet (Definition 5). The extended functions $*\Delta_i()$, $*K_i()$, $*\Gamma_i()$, $*\Theta_i()$, $*W_i()$, $*M_{0(i)}()$ defined for a subnet N_i can take as arguments elements that belong to N_i directly or to its descendant subnets. These functions are explained using the exemplary net shown in Figure 10 and the function returning the capacity of places. The other extended functions work in the same way. For the subnet N_0, the function $K_0()$ arguments can only be places that are directly in the subnet, i.e., p_1–p_4. However, for the same subnet, the function $*K_0()$ can additionally return capacity for the places p_5 and p_6, i.e., all places of descendant subnets for N_0 (N_1 in this case). If the subnet has no descendant subnets (N_1 in Figure 10), then the functions $K_i()$ and $*K_i()$ work in the same way. The functions $K_1()$ and $*K_1()$ can only return capacity of the places p_3–p_6.

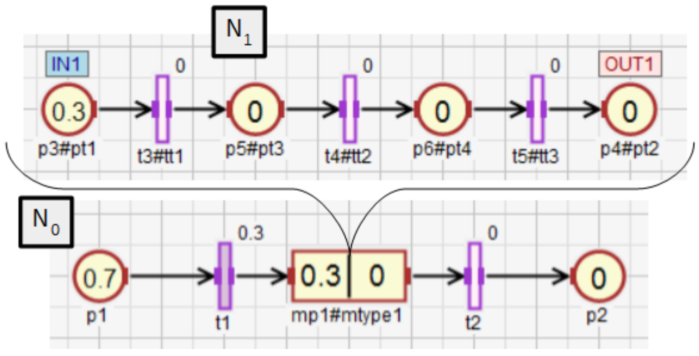

Figure 10. HFIPN with the single macroplace mp_1.

Each subnet N_i in HFIPN may be represented by its own algebraic representation $N_i = (*\boldsymbol{C_i}, *\Omega_i, *\Psi_i, *\Delta_i, *K_i, *\Gamma_i, *\Theta_i, *M_{0(i)})$. The algebraic representation of the subnet N_0 is used to simulate the work of the entire net. However, for the remaining subnets, it can be used to test the properties or to simulate the work of the subnet, separately. The most important element of the algebraic representation is the incidence matrix $*\boldsymbol{C}_i$ defined analogously to classical Petri nets, but taking the hierarchy into account. It has the following form.

Definition 13. *Let* HFIPN $= (N, MP, IO, INS)$ *be the hierarchical fuzzy interpreted Petri net, and the subnet* $N_i = \{MP_i, P_i, T_i, \Omega_i, \Psi_i, R_i, \Delta_i, K_i, W_i, \Gamma_i, \Theta_i, M_{0(i)}, e\}$ *one of its subnets* ($N_i \in N$)*, for which* $card(*T_i) = *b_i$ *and* $card(*P_i) = *a_i$. *The incidence matrix* $*\boldsymbol{C_i} = *\boldsymbol{C_i^+} - *\boldsymbol{C_i^-}$ *is defined as follows by the matrices:* $*\boldsymbol{C_i^+} = \{*c_{jk}^+\}$ *and* $*\boldsymbol{C_i^-} = \{*c_{jk}^-\}$*, both with a dimension* $*b_i \times *a_i$*:*

$$*\,c_{jk}^{+} = \begin{cases} *W_i(t_j, p_k) & \text{for} \quad (t_j, p_k) \in *R_i, \\ 0 \quad \text{for} \quad (t_j, p_k) \notin *R_i, \end{cases} \tag{44}$$

$$*\,c_{jk}^{-} = \begin{cases} *W_i(p_k, t_j) & \text{for} \quad (p_k, t_j) \in *R_i, \\ 0 \quad \text{for} \quad (p_k, t_j) \notin *R_i, \end{cases} \tag{45}$$

for $j = 1, 2, ..., *b_i$ *and* $k = 1, 2, ..., *a_i$.

The HFIPN algebraic representation enables the computation of a new marking in the net in response to transition activation and token movement between places. It uses vector and matrix operations, such as addition (46), multiplication (47), division (48), and minimum of two numbers (49):

$$D = [d_{ij}]_{n\times m} = A + B, \quad d_{ij} = a_{ij} + b_{ij}, \tag{46}$$

$$E = [e_{ij}]_{n\times m} = A \cdot B, \quad e_{ij} = a_{ij} \cdot b_{ij}, \tag{47}$$

$$F = [f_{ij}]_{n\times m} = \frac{A}{B}, \quad f_{ij} = \frac{a_{ij}}{b_{ij}}, \tag{48}$$

$$G = [g_{ij}]_{n\times m} = A \wedge B, \quad g_{ij} = a_{ij} \wedge b_{ij}, \tag{49}$$

where: i = 1, 2, ..., n and j = 1, 2, ..., m.

Therefore, for each subnet $N_i \in N$, the algebraic representation takes the following form:

$$*\boldsymbol{M_i'} = *\boldsymbol{M_i} + \frac{*\boldsymbol{U_i} \wedge \boldsymbol{\Delta *\Theta_i} \cdot (*\boldsymbol{C_i^+} - *\boldsymbol{C_i^-})}{*\boldsymbol{K_i}}, \tag{50}$$

where:
$*\boldsymbol{M_i}$ is the vector of the length $1 \times *a_i$, holding the current marking of places, where $*a_i = card(*P_i)$;
$*\boldsymbol{M_i'}$ is the vector of the length $1 \times *a_i$, holding the new marking of places;
$*\boldsymbol{U_i}$ is the vector of the length $1 \times *b_i$, in which the given coefficient is equal to one if it corresponds to the enabled transition by the marking $*\boldsymbol{M_i}$ in the subnet, where $*b_i = card(*T_i)$;
$\boldsymbol{\Delta *\Theta}_i$ is the vector of the length $1 \times *b_i$, in which the coefficient $\Delta\theta_j$ $(1 \le j \le *b_i)$ describes the increment in the degree to which the condition corresponding to the transition t_j $(t_j \in *T_i)$ is satisfied;
$*\boldsymbol{K_i}$ is the vector of the length $1 \times *a_i$, holding the places capacity.

For the subnet N_0 from Figure 10, the incidence matrix $*\boldsymbol{C_0}$ and the vectors $*\boldsymbol{M_0}$, $*\boldsymbol{K_0}$, $*\boldsymbol{U_0}$, $*\boldsymbol{\Theta_0}$ take the following form:

$$*\boldsymbol{C_0} = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{array}{c} \begin{array}{cccccc} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \end{array} \\ \begin{pmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix} \end{array}, \tag{51}$$

$$*\boldsymbol{M_0} = \begin{array}{c} \begin{array}{cccccc} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \end{array} \\ \begin{pmatrix} 0.7 & 0 & 0.3 & 0 & 0 & 0 \end{pmatrix} \end{array}, \tag{52}$$

$$*\boldsymbol{K_0} = \begin{array}{c} \begin{array}{cccccc} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \end{array}, \tag{53}$$

$$*\boldsymbol{U_0} = \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}, \tag{54}$$

$$*\boldsymbol{\Theta_0} = \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{pmatrix} 0.3 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}. \tag{55}$$

If the increment of the degree to which the condition corresponding to the transition t_1 is satisfied changes by $\Delta\theta_1 = 0.1$, and the event e, synchronizing the work of all transition, occurs, then the vector $\boldsymbol{\Delta\Theta_0}$ takes the following form:

$$\boldsymbol{\Delta} * \boldsymbol{\Theta_0} = \begin{array}{c} \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \end{array} \\ \begin{pmatrix} 0.1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}. \tag{56}$$

According to (50), the vector of the new marking $*\boldsymbol{M_0'}$ takes the following form:

$$*\boldsymbol{M_0'} = \begin{array}{c} \begin{array}{cccccc} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \end{array} \\ \begin{pmatrix} 0.6 & 0 & 0.4 & 0 & 0 & 0 \end{pmatrix} \end{array}. \tag{57}$$

After that, the vector $*\boldsymbol{\Theta_0}$ changes as follows:

$$*\boldsymbol{\Theta_0} = \begin{array}{c} \\ \end{array}\begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \\ (0.4 & 0 & 0 & 0 & 0\,), \end{array} \tag{58}$$

and the vector $\boldsymbol{\Delta * \Theta_0}$, which contains the increments of the degrees to which the conditions corresponding to the transitions are satisfied, is reset to zero at each calculation cycle. Therefore, the vector $*\boldsymbol{\Theta_0}$ is used for storing the current values of these degrees and to calculate their increments at the beginning of each cycle.

3.5. HFIPN-SML Tool

In this subsection HFIPN-SML is briefly described. Most of the figures presented in this and the next section are created using screenshots from this application. Although the tool has not yet been published, it implements some functionalities that can be useful in practical modeling. In the current version, HFIPN-SML enables:

(a) the creation of a net graph,
(b) the use of a hierarchical structure including macroplace instances,
(c) the automatic code generation in Structured Text (ST) language for PLC controllers based on a non-hierarchical graph,
(d) the hierarchy graph presentation,
(e) displaying all vectors and matrices from algebraic representation,
(f) automatic and step simulations of a net operation,
(g) monitoring the operation of a program generated based on a non-hierarchical net.

All these features can make HFIPN-SML useful for practical modeling from the control system design to its practical implementation, and even then for monitoring purposes. However, some gaps need to be filled, i.e., automatic code generation module based on a hierarchical net and automatic analysis of properties integrated with HFIPN-SML.

4. Exemplary Application

In this section, an example of using HFIPN is shown, including the concepts of macroplace instances and macroplaces that can have multiple input, output and input/output places. Figure 11 shows a system for mixing three components, while Figures 12–16 illustrate an algorithm in the form of an HFIPN graph for controlling this system. First, three portions of each of the liquids $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$, measured in the tanks $\boldsymbol{Tk_1}$ and $\boldsymbol{Tk_2}$, are transferred to the mixer. At the same time, two soluble bricks are fed into the mixer. Then all ingredients are mixed for some time represented by the variable $\boldsymbol{TimeVal}$. The default time value is stored in $\boldsymbol{tx_1}$. Moreover, the measured time is converted to a value from the range [0, 1] and stored in $\boldsymbol{TIME1}$. Finally, after obtaining the liquid of the desired consistency, the mixer is emptied and the whole process is repeated. In addition, if the liquids $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ are not transferred to the mixer at a similar speed, the mixing time can be extended even up to twelve times by the value of the variable $\boldsymbol{tx_2}$ or the entire process may stop.

Figure 12 shows the main subnet with four macroplaces (modules). The macroplace $\boldsymbol{mp_1}$ controls the transport of the bricks, while the macroplace $\boldsymbol{mp_2}$ controls the transfer of the liquids $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ to the mixer. The macroplace $\boldsymbol{mp_4}$ serves as a diagnostic module that checks the uniformity of filling by both liquids. The macroplace $\boldsymbol{mp_3}$ controls the mixing of all ingredients and emptying the mixer. The whole process starts with pressing the $\boldsymbol{START}$ button (transition $\boldsymbol{t_1}$), and the red diode is turned on, while the green one is turned off (place $\boldsymbol{p_2}$), what signals the operation of the system. The modules $\boldsymbol{mp_1}$ and $\boldsymbol{mp_2}$ are activated.

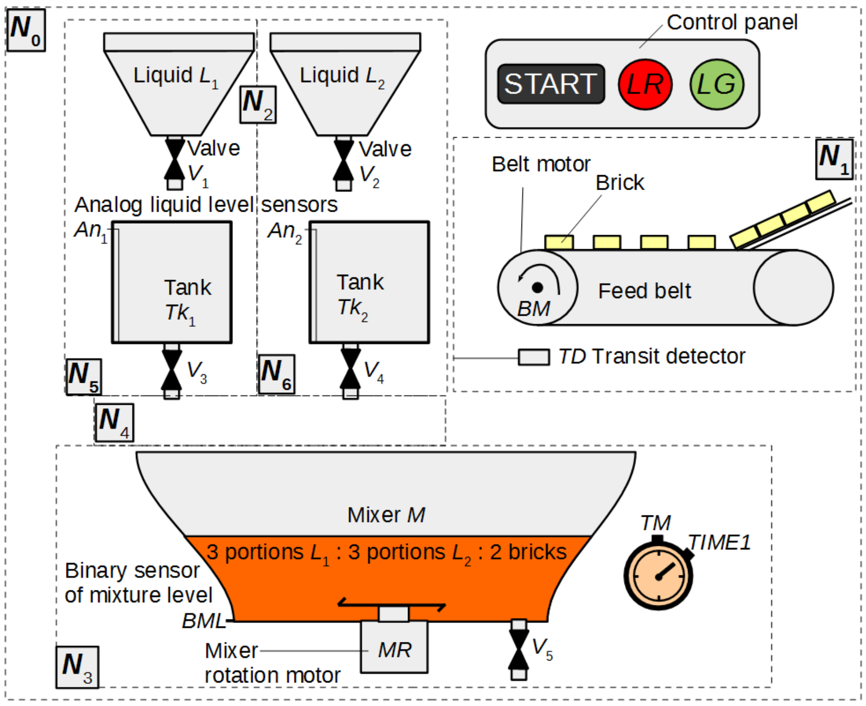

Figure 11. The scheme of the system to mix three components: two liquids and soluble bricks. Based on [64].

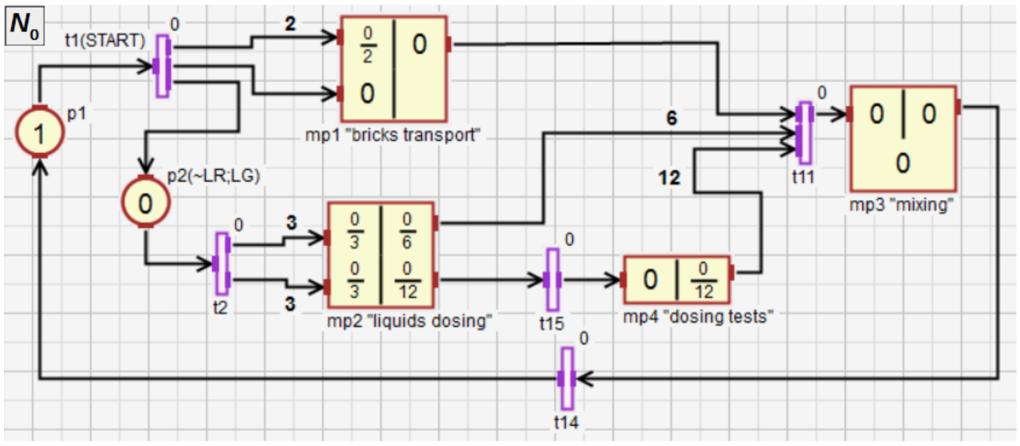

Figure 12. The main subnet of the control algorithm in the form of HFIPN graph.

The macroplace mp_1 (Figure 13) turns on the belt motor (place p_{10}), which causes that the two bricks are transported to the mixer. After the detection of the second brick by the binary transit detector TD the feed belt is turned off (p_{13}). The input place p_9 of type p'' represents the number of bricks that need to be transferred to the mixer. Moreover, this place enables to observe the state of the macroplace mp_1 internal subnet (N_1) from the main subnet (N_0) without going inside the subnet. If the marking of this place equals 2, it means that the two bricks will soon be transported to the mixer. If it equals 1, a single brick has been transported and the other one will be moved soon. If the marking of this place is 0, and the marking of the output place p_{13} equals 1, it means that there are two bricks in the mixer. In the other case, the brick transport module is not working.

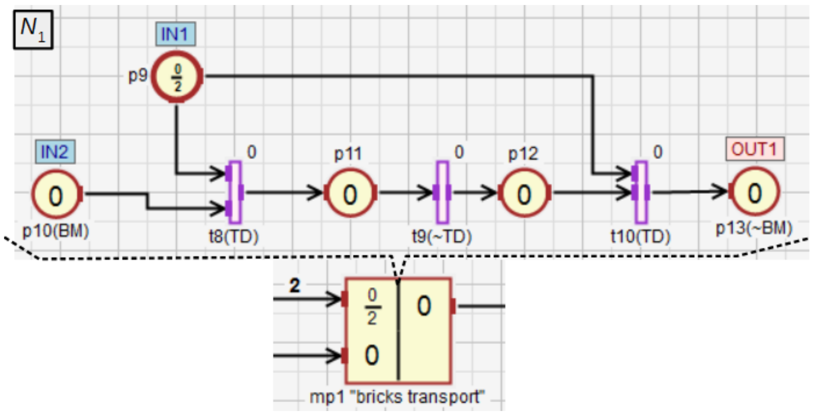

Figure 13. The brick transport module to the mixer.

In the macroplace mp_2 (Figure 14), the transfer of the liquids L_1 and L_2 to the tanks Tk_1 and Tk_2 is controlled using macroplace mp_5 and mp_6, that are instances of the same type. The dosing of both liquids with the use of the valves V_1 (place p_3 in mp_5) and V_2 (p_6 in mp_6) is monitored by the analogue sensors An_1 (transition t_3) and An_2 (t_4). It ends when the value of the signal provided by the sensors equals 1. Then, both measured liquid portions are transferred from Tk_1 and Tk_2 to the mixer through the valves V_3 (p_{17}) and V_4 (p_{18}). The valve V_3 (V_4) is closed by the action assigned to the place p_3 (p_6) when the value from the liquid level sensor An_1 (An_2) equals 0. Then, successive portions of liquids are measured.

The number of portions of the liquids L_1 and L_2 that need to be transferred to the mixer are represented by the input places p_5 and p_8. Both liquids have to be delivered at a similar speed due to the technological requirements. The sum of both portions of the liquids transferred into the mixer is represented by the output place p_{19}. Thus, at any time during the work of the liquid dosing module, its state can be determined based on the marking of the places p_5, p_8 and p_{19} (analogously to mp_1). If, for instance, the marking of the place p_5 equals 1.9, p_8 equals 1.87, and p_{19} is 2, it means that the transfer of the first portions of both liquids is complete, while the second are being transferred into the dispensers. During the transport of the liquids to the mixer, twelve tests are performed to determine whether the speed of transferring the liquids L_1 and L_2 is similar, as long as the process has not been interrupted earlier. Through the output place p_{20} the token is sent to the test module (the macroplace mp_4). The test is performed every half of the portion obtained from both liquids (a total portion number is six).

The module comparing the speeds of the liquids dosing into the mixer (macroplace mp_4) is shown in Figure 15. First, the numeric variables x_1 and x_2 are calculated (place p_{21}) based on the marking of places that are inside the macroplace mp_2 to determine how many portions of L_1 and L_2 were transferred into the mixer. If x_2 equals 0 (t_{22}), the whole process is stopped (p_{24}). This most likely means the dispensing failure of L_2. If x_2 is greater than 0, the quotient of x_1 and x_2 is assigned to the variable x. Then, based on x, specific control actions are taken by activating one of the transitions t_{17}–t_{19}. Conditions excluding any conflict between these transitions are assigned to them. If L_1 and L_2 are transferred into the mixer evenly ($x \in [0.9, 1.1]$), no additional control action is taken. If one of the liquids is transferred to the mixer too fast compared to the other, but their proportion is still sufficient ($x \in [0.8, 0.9) \cup (1.1, 1.2]$), the mixing time of all components $TimeVal$ is extended by tx_2 (p_{23}). If L_1 and L_2 are transferred unevenly (t_{19}), the whole system is stopped (p_{24}).

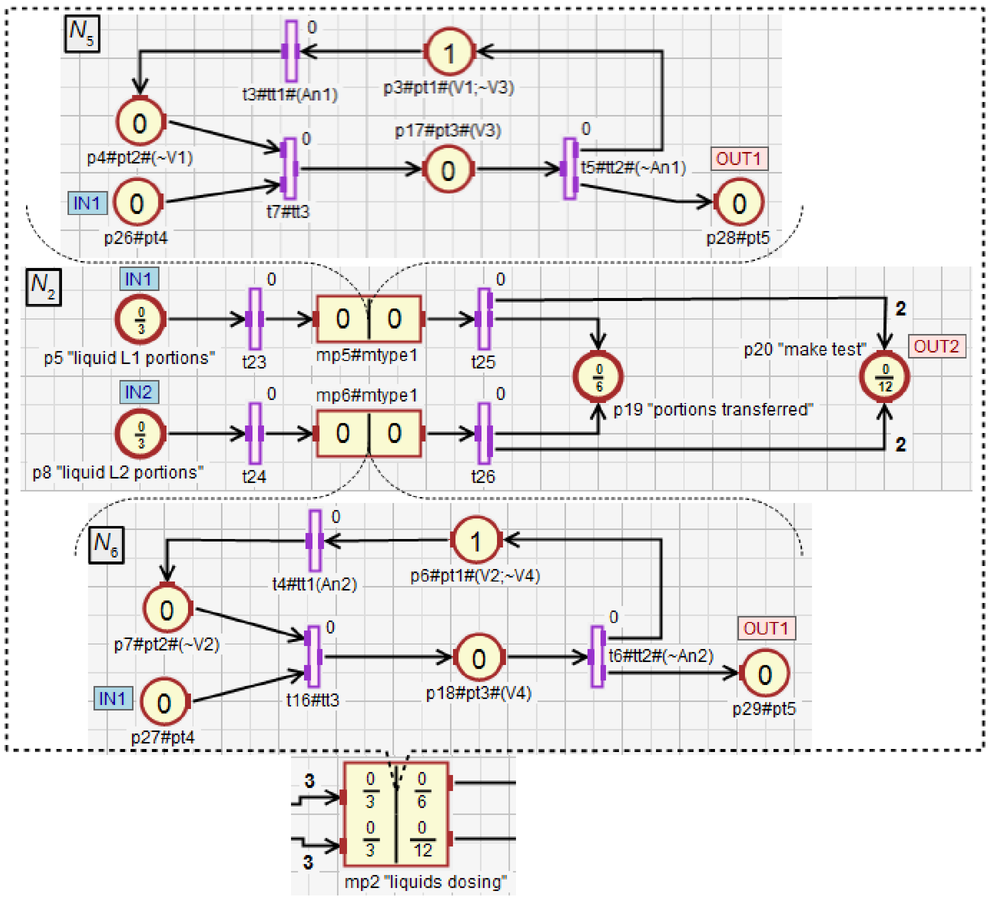

Figure 14. Dosing module for the liquids L_1 and L_2.

In the macroplace mp_3 (Figure 16), the mixing of all ingredients using the rotary engine MR is controlled. First, the operation of MR and the external timer TM are initialized (p_{14}). After mixing for a period of time defined by $TimeVal$ and obtaining the liquid of the desired consistency, the mixer is emptied through the valve V_5 (p_{15}). The binary level sensor BML detects when the mixer is completely empty (t_{13}). After that, the mixing time $TimeVal$ of the components is restored to the base value (p_{16}). Then, if the $START$ button is on, the next cycle of the whole system starts. The place p_{15} is an input-output place and allows for better visualization of the module from the main subnet.

The presented example shows that the use of HFIPN can improve the visualization of the process and facilitate communication between different modules in the net. Moreover, in the macroplace mp_4 an example of dynamic control in response to fuzzy marking of places is presented.

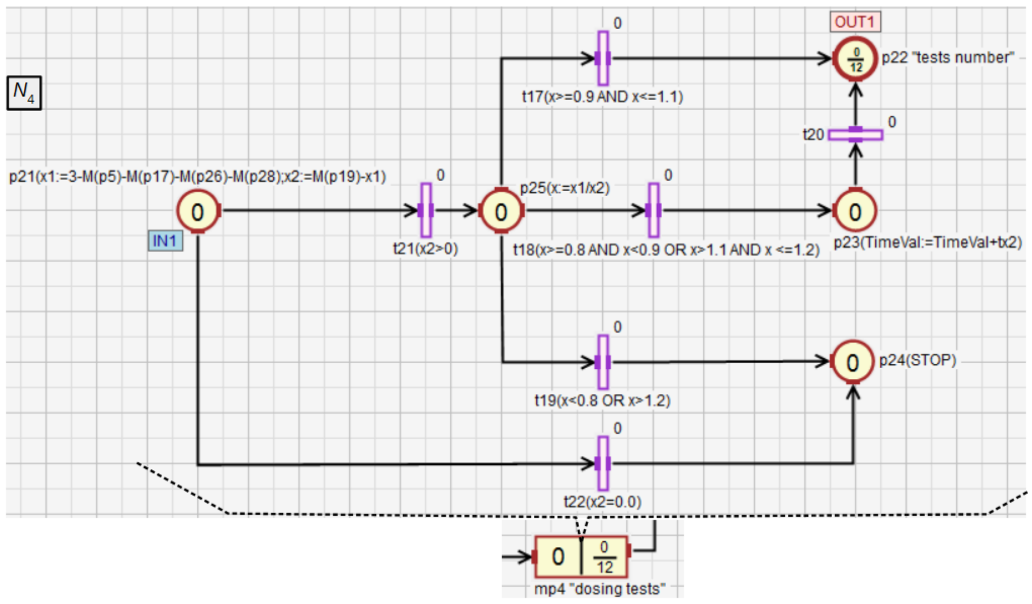

Figure 15. The module checking whether the liquids L_1 and L_2 are transferred to the mixer at a similar speed.

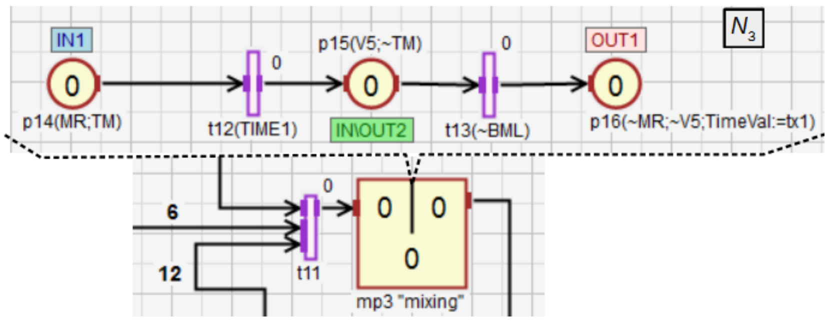

Figure 16. The module for mixing and emptying the mixer.

5. Conclusions

In this work, a new concept of HFIPN has been introduced. The contributions of this study are as follows:

(a) Concept of a macroplace that can have several input, output and input-output places;
(b) Functionality of a macroplace instance;
(c) Formal definition of HFIPN;
(d) Concept and a definition of a hierarchy graph displaying a hierarchical structure and allowing a quick access to all subnets in an implementation version;
(e) Formal algebraic representation of HFIPN;
(f) Conversion of HFIPN to its flat version;
(g) Formal description of the combination of any two subnets in the hierarchical net.

The proposed macroplaces with several inputs and outputs can be useful for the implementation of modules that require asynchronous communication and facilitate resource modeling as illustrated in the example presented in Section 4. The concept of macroplace instances allows for the creation of several modules with the same structure and operation, their use in different areas of a net, and their simultaneous modification by introducing changes to one of them.

The nesting relations have also been defined, as well as the hierarchy graph which presents graphically the hierarchical structure of the net: the number of macroplaces and instances, and how they are nested in the net. The implementation of the graph in the HFIPN-SML simulator allows for quick movement between modules. An important part of the described research is an algebraic representation of HFIPN and the conversion method of a hierarchical net to the corresponding flat FIPN.

However, further studies on HFIPN are needed. In the future work, we are going to focus on the conversion of its graph to executable code.

Author Contributions: Conceptualization, M.M. and L.G.; methodology, M.M. and L.G; software, M.M.; formal analysis, M.M. and L.G.; writing—original draft preparation, D.W. and M.M.; writing—review and editing, D.W.; visualization, M.M.; supervision, L.G.; funding acquisition, L.G.; literature review and bibliography organizing, M.M. and D.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded byPB23.EA.21.001.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Valette, R. Analysis of Petri nets by stepwise refinements. *J. Comput. Syst. Sci.* **1979**, *18*, 35–46. [CrossRef]
2. Suzuki, I.; Murata, T. A method for stepwise refinement and abstraction of Petri nets. *J. Comput. Syst. Sci.* **1983**, *27*, 51–76. [CrossRef]
3. Vogler, W. Behaviour preserving refinements of Petri nets. In *Graph-Theoretic Concepts in Computer Science*; Tinhofer, G., Schmidt, G., Eds.; Number 246 in Lecture Notes in Computer Science, Springer: Berlin/Heidelberg, Germany, 1986; pp. 82–93. [CrossRef]
4. Brauer, W.; Gold, R.; Vogler, W. A survey of behaviour and equivalence preserving refinements of Petri nets. In *Advances in Petri Nets 1990*; Rozenberg, G., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1989; pp. 1–46. [CrossRef]
5. Bernardinello, L.; De Cindio, F. A survey of basic net models and modular net classes. In *Advances in Petri Nets 1992*; Springer: London, UK, 1992; pp. 304–351.
6. Van Der Aalst, W.M. Workflow verification: Finding control-flow errors using Petri-net-based techniques. In *Business Process Management*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 161–183.
7. Huang, H.; Jiao, L.; Cheung, T.Y. *Property-Preserving Petri Net Process Algebra in Software Engineering*; World Scientific: Singapore, 2012.
8. Van Glabbeek, R.; Goltz, U. Refinement of actions in causality based models. In *Stepwise Refinement of Distributed Systems Models, Formalisms, Correctness*; Springer: Berlin/Heidelberg, Germany, 1989; pp. 267–300.
9. Best, E.; Devillers, R.; Kiehn, A.; Pomello, L. Concurrent bisimulations in Petri nets. *Acta Inform.* **1991**, *28*, 231–264. [CrossRef]
10. Vogler, W. Bisimulation and action refinement. *Theor. Comput. Sci.* **1993**, *114*, 173–200. [CrossRef]
11. Van Glabbeek, R.; Goltz, U. Refinement of actions and equivalence notions for concurrent systems. *Acta Inform.* **2001**, *37*, 229–327. [CrossRef]
12. Jiao, L. Refining and verifying regular Petri nets. *Int. J. Syst. Sci.* **2008**, *39*, 17–27. [CrossRef]
13. Hack, M. Analysis of Production Schemata by Petri Nets. Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1972.
14. Lipton, R.J. Reduction: A method of proving properties of parallel programs. *Commun. ACM* **1975**, *18*, 717–721. [CrossRef]
15. Kwong, Y.S. On reduction of asynchronous systems. *Theor. Comput. Sci.* **1977**, *5*, 25–50. [CrossRef]
16. Kowalk, W.; Valk, R. On reductions of parallel programs. In *Automata, Languages and Programming*; Maurer, H.A., Ed.; Number 71 in Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1979; pp. 356–369. [CrossRef]
17. Berthelot, G. Checking properties of nets using transformations. In *European Workshop on Applications and Theory in Petri Nets*; Springer: Berlin/Heidelberg, Germany, 1985; pp. 19–40.
18. Berthelot, G. Transformations and decompositions of nets. In *Petri Nets: Central Models and Their Properties*; Brauer, W., Reisig, W., Rozenberg, G., Eds.; Number 254 in Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1987; pp. 359–376. [CrossRef]
19. Lee, K.H.; Favrel, J. Hierarchical reduction method for analysis and decomposition of Petri nets. *IEEE Trans. Syst. Man Cybern.* **1985**, *SMC-15*, 272–280. [CrossRef]
20. Lee, K.H.; Favrel, J.; Baptiste, P. Generalized Petri net reduction method. *Syst. Man Cybern. IEEE Trans.* **1987**, *17*, 297–303.
21. Desel, J. Reduction and design of well-behaved concurrent systems. In *CONCUR'90 Theories of Concurrency: Unification and Extension*; Springer: Berlin/Heidelberg, Germany, 1990; pp. 166–181.

22. Best, E.; Desel, J. Partial order behaviour and structure of Petri nets. *Form. Asp. Comput.* **1990**, *2*, 123–138. [CrossRef]
23. Esparza, J. Reduction and synthesis of live and bounded free choice Petri nets. *Inf. Comput.* **1994**, *114*, 50–87. [CrossRef]
24. Desel, J.; Esparza, J. *Free Choice Petri Nets*; Cambridge University Press: Cambridge, UK, 2005; Volume 40.
25. Jiao, L.; Cheung, T.Y.; Lu, W. Characterizing liveness of Petri nets in terms of siphons. In *International Conference on Application and Theory of Petri Nets*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 203–216.
26. Jiao, L.; Cheung, T.Y.; Lu, W. On liveness and boundedness of asymmetric choice nets. *Theor. Comput. Sci.* **2004**, *311*, 165–197. [CrossRef]
27. Sloan, R.H.; Buy, U. Reduction rules for time Petri nets. *Acta Inform.* **1996**, *33*, 687–706. [CrossRef]
28. Wang, J.; Deng, Y.; Zhou, M. Compositional time Petri nets and reduction rules. *IEEE Trans. Syst. Man Cybern. Part B* **2000**, *30*, 562–572. [CrossRef]
29. Juan, E.; Tsai, J.; Murata, T.; Zhou, Y. Reduction methods for real-time systems using Delay Time Petri Nets. *IEEE Trans. Softw. Eng.* **2001**, *27*, 422–448. [CrossRef]
30. Fehling, R. A concept of hierarchical Petri nets with building blocks. In *Advances in Petri Nets 1993*; Rozenberg, G., Ed.; Number 674 in Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1991; pp. 148–168. [CrossRef]
31. Jensen, K. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 1997; Volume 1.
32. Holvoet, T.; Verbaeten, P. Petri charts: An alternative technique for hierarchical net construction. In Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics, Intelligent Systems for the 21st Century, Vancouver, BC, Canada, 22–25 October 1995; Volume 3, pp. 2688–2693.
33. He, X. A formal definition of hierarchical predicate transition nets. In *Application and Theory of Petri Nets 1996*; Billington, J., Reisig, W., Eds.; Number 1091 in Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1996; pp. 212–229. [CrossRef]
34. Andrzejewski, G. Hierarchical Petri nets for digital controller design. In *Design of Embedded Control Systems*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 27–36.
35. Pan, H.; Sun, J. Complex knowledge system modeling based on hierarchical fuzzy petri net. In Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops, Fremont, CA, USA, 2–5 November 2007; pp. 31–34.
36. *International Standard IEC 60848:2013: Grafcet Specification Language for Sequential Function Charts Approach*; Technical Report; International Electrotechnical Commission: Geneva, Switzerland, 2013.
37. *International Standard IEC 61131-3: PROGRAMMABLE Controllers—Part 3: Programming Languages*; International Standard; International Electrotechnical Commission: Geneva, Switzerland, 2013.
38. David, R.; Alla, H. *Discrete, Continuous, and Hybrid Petri Nets*; Springer: Berlin/Heidelberg, Germany, 2010.
39. Silva, M.; Valette, R. Petri nets and flexible manufacturing. In *European Workshop on Applications and Theory in Petri Nets*; Springer: Berlin/Heidelberg, Germany, 1988; pp. 374–417.
40. Valavanis, K.P. On the hierarchical modeling analysis and simulation of flexible manufacturing systems with extended Petri nets. *IEEE Trans. Syst. Man, Cybern.* **1990**, *20*, 94–110. [CrossRef]
41. Zhou, M.; DiCesare, F.; Desrochers, A.A. A hybrid methodology for synthesis of Petri net models for manufacturing systems. *IEEE Trans. Robot. Autom.* **1992**, *8*, 350–361. [CrossRef]
42. Jeng, M.D.; DiCesare, F. Synthesis using resource control nets for modeling shared-resource systems. *IEEE Trans. Robot. Autom.* **1995**, *11*, 317–327. [CrossRef]
43. Zhou, M. Modeling, analysis, simulation, scheduling, and control of semiconductor manufacturing systems: A Petri net approach. *IEEE Trans. Semicond. Manuf.* **1998**, *11*, 333–357. [CrossRef]
44. Ye, J.; Zhou, M.; Li, Z.; Al-Ahmari, A. Structural decomposition and decentralized control of Petri nets. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *48*, 1360–1369. [CrossRef]
45. Wiśniewski, R.; Karatkevich, A.; Wojnakowski, M. Decomposition of distributed edge systems based on the Petri nets and linear algebra technique. *J. Syst. Archit.* **2019**, *96*, 20–31. [CrossRef]
46. An, Y.; Wu, N.; Zhao, X.; Li, X.; Chen, P. Hierarchical colored petri nets for modeling and analysis of transit signal priority control systems. *Appl. Sci.* **2018**, *8*, 141. [CrossRef]
47. Sicchar, J.R.; Da Costa, C.T.; Silva, J.R.; Oliveira, R.C.; Oliveira, W.D. A load-balance system design of microgrid cluster based on hierarchical petri nets. *Energies* **2018**, *11*, 3245. [CrossRef]
48. Gozhyj, A.; Kalinina, I.; Gozhyj, V.; Vysotska, V. Web service interaction modeling with colored petri nets. In Proceedings of the 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Metz, France, 18–21 September 2019; Volume 1, pp. 319–323.
49. Li, W.; He, M.; Sun, Y.; Cao, Q. A novel layered fuzzy Petri nets modelling and reasoning method for process equipment failure risk assessment. *J. Loss Prev. Process Ind.* **2019**, *62*, 103953. [CrossRef]
50. Yuan, C.; Liao, Y.; Kong, L.; Xiao, H. Fault diagnosis method of distribution network based on time sequence hierarchical fuzzy petri nets. *Electr. Power Syst. Res.* **2021**, *191*, 106870. [CrossRef]
51. Majma, N.; Babamir, S.M.; Monadjemi, A. Runtime verification of pacemaker functionality using hierarchical fuzzy colored Petri-Nets. *J. Med. Syst.* **2017**, *41*, 1–21. [CrossRef]

52. Padberg, J. Subtyping for hierarchical, reconfigurable Petri nets. *arXiv* **2018**, arXiv:1802.04698.
53. Figat, M.; Zieliński, C. Methodology of designing multi-agent robot control systems utilising hierarchical petri nets. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–4 June 2019; pp. 3363–3369.
54. Silva, J.M.; Silva, J.R. A new hierarchical approach to requirement analysis of problems in automated planning. *Eng. Appl. Artif. Intell.* **2019**, *81*, 373–386. [CrossRef]
55. Wisniewski, R.; Grobelna, I.; Karatkevich, A. Determinism in cyber-physical systems specified by interpreted petri nets. *Sensors* **2020**, *20*, 5565. [CrossRef]
56. López, J.; Sánchez-Vilariño, P.; Sanz, R.; Paz, E. Implementing autonomous driving behaviors using a message driven petri net framework. *Sensors* **2020**, *20*, 449. [CrossRef] [PubMed]
57. Proth, J.M.; Xie, X. *Petri Nets: A Tool for Design and Management of Manufacturing Systems*; John Wiley & Sons: Hoboken, NJ, USA, 1996; Volume 6.
58. Uzam, M. The use of the Petri net reduction approach for an optimal deadlock prevention policy for flexible manufacturing systems. *Int. J. Adv. Manuf. Technol.* **2004**, *23*, 204–219. [CrossRef]
59. Gniewek, L. Sequential control algorithm in the form of fuzzy interpreted Petri net. *IEEE Trans. Syst. Man Cybern. Syst.* **2013**, *43*, 451–459. [CrossRef]
60. Gniewek, L. Coverability graph of fuzzy interpreted Petri net. *IEEE Trans. Syst. Man Cybern. Syst.* **2014**, *44*, 1272–1277. [CrossRef]
61. Markiewicz, M.; Surdej, Ł.; Gniewek, L. Transformation of a fuzzy interpreted Petri net diagram into structured text code. In Proceedings of the 2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR), Miedzyzdroje, Poland, 29 August–1 September 2016; pp. 94–99.
62. Markiewicz, M.; Gniewek, L. A program model of fuzzy interpreted Petri net to control discrete event systems. *Appl. Sci.* **2017**, *7*, 422. [CrossRef]
63. Hajduk, Z.; Wojtowicz, J. FPGA implementation of fuzzy interpreted Petri net. *IEEE Access* **2020**, *8*, 61442–61452. [CrossRef]
64. Markiewicz, M.; Gniewek, L. Conception of hierarchical fuzzy interpreted Petri net. *Stud. Inf. Control* **2017**, *26*, 151–160. [CrossRef]
65. Frey, G. Automatic implementation of Petri net based control algorithms on PLC. In Proceedings of the 2000 American Control Conference, ACC (IEEE Cat. No. 00CH36334), Chicago, IL, USA, 28–30 June 2000; Volume 4, pp. 2819–2823.
66. Minas, M.; Frey, G. Visual PLC-programming using signal interpreted Petri nets. In Proceedings of the 2002 American Control Conference, Anchorage, USA, 8–10 May 2002; Volume 6, pp. 5019–5024.
67. Klein, S.; Frey, G.; Minas, M. PLC programming with signal interpreted Petri nets. In *International Conference on Application and Theory of Petri Nets*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 440–449.
68. Frey, G. Hierarchical design of logic controllers using signal interpreted Petri nets. *IFAC Proc. Vol.* **2003**, *36*, 361–366. [CrossRef]
69. Andreu, D.; Pascal, J.C.; Valette, R. Fuzzy Petri net-based programmable logic controller. *IEEE Trans. Syst. Man Cybern. Part B* **1997**, *27*, 952–961. [CrossRef]
70. Gniewek, L.; Kluska, J. Hardware implementation of fuzzy Petri net as a controller. *IEEE Trans. Syst. Man Cybern. Part B* **2004**, *34*, 1315–1324. [CrossRef]
71. Venkateswaran, P.; Bhat, J. Fuzzy Petri net algorithm for flexible manufacturing systems. *ACSE J.* **2006**, *6*, 1–5.
72. Bauer, N.; Engell, S.; Huuck, R.; Lohmann, S.; Lukoschus, B.; Remelhe, M.; Stursberg, O. Verification of PLC programs given as sequential function charts. In *Integration of Software Specification Techniques for Applications in Engineering*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 517–540.
73. Roussel, J.M.; Lesage, J.J. Validation and verification of Grafcets using state machine. In *IMACS-IEEE "CESA'96"* July, Lille, 1996; pp. 758–764.
74. Provost, J.; Roussel, J.M.; Faure, J.M. A formal semantics for Grafcet specifications. In Proceedings of the 2011 IEEE International Conference on Automation Science and Engineering, Trieste, Italy, 24–27 August 2011; pp. 488–494.
75. Remelhe, M.; Lohmann, S.; Stursberg, O.; Engell, S.; Bauer, N. Algorithmic verification of logic controllers given as sequential function charts. In Proceedings of the 2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No. 04CH37508), New Orleans, LA, USA, 26 April–1 May 2004; pp. 53–58.
76. Lohmann, S.; Stursberg, O.; Engell, S. Comparison of event-triggered and cycle-driven models for verifying SFC programs. In Proceedings of the 2007 American Control Conference, New York, NY, USA, 9–13 July 2007; pp. 3606–3611.
77. L'Her, D.; Le Parc, P.; Marcé, L. Proving sequential function chart programs using timed automata. *Theor. Comput. Sci.* **2001**, *267*, 141–155. [CrossRef]
78. Stursberg, O.; Lohmann, S. Analysis of logic controllers by transformation of SFC into timed automata. In Proceedings of the 44th IEEE Conference on Decision and Control, Seville, Spain, 15–25 December 2005; pp. 7720–7725.
79. Wightkin, N.; Buy, U.; Darabi, H. Formal modeling of sequential function charts with time Petri nets. *IEEE Trans. Control Syst. Technol.* **2010**, *19*, 455–464. [CrossRef]
80. Sogbohossou, M.; Vianou, A. Formal modeling of Grafcets with time Petri nets. *IEEE Trans. Control Syst. Technol.* **2015**, *23*, 1978–1985. [CrossRef]

81. Peng, S.; Zhou, M. Design and analysis of sequential function charts using sensor-based stage Petri nets. In Proceedings of the SMC'03 Conference Proceedings, 2003 IEEE International Conference on Systems, Man and Cybernetics, Conference Theme-System Security and Assurance (Cat. No. 03CH37483), Washington, DC, USA, 5–8 October 2003; Volume 5, pp. 4748–4753.
82. Fujino, K.; Imafuku, K.; Yuh, Y.; Hirokazu, N. Design and verification of the SFC program for sequential control. *Comput. Chem. Eng.* **2000**, *24*, 303–308. [CrossRef]
83. Schumacher, F.; Fay, A. Transforming time constraints of a Grafcet graph into a suitable Petri net formalism. In Proceedings of the 2013 IEEE International Conference on Industrial Technology (ICIT), Cape Town, South Africa, 25–28 February 2013; pp. 210–218.