HIERARCHICAL DESIGN OF LOGIC CONTROLLERS USING SIGNAL INTERPRETED PETRI NETS

Georg Frey

Abstract: Logic Controllers are often used to control continuous or hybrid processes. In these cases, the connection to the plant is realized by implementing pre- and post-processors for the conversion between analog and binary data. Hence, the controller remains purely discrete. However, the pre- and post-processors can not be included in the controller analysis. To overcome this problem, Signal Interpreted Petri Nets, a Petri net model for the specification of PLC programs, are extended to include non-binary signals. In this contribution hierarchical SIPN are presented. The analysis of the hierarchical net is based on the analysis of its subnets.

Copyright © 2003 IFAC

Keywords: Petri Nets, Logic Control, Hierarchy, Verification

1 INTRODUCTION

To design a controller there are basically two approaches, model-based and non-model-based. In model-based approaches a model of the process under control (the plant) is used in the design process. Synthesis and verification can be performed model-based. In the synthesis approach a model of the uncontrolled process is built and based on formal specifications a controller is derived mathematically, e.g. (Moor et al. 1998, Hanisch et al. 1998). Verification combines models of the uncontrolled process and the controller to a model of the controlled process. Using this model, formal specifications can be verified mathematically, e.g. (Kowalewski and Preußig 1996). Verification can also be done non-model-based. These approaches verify properties of a controller under minimal or no assumptions about the process under control.

Petri nets are able to express the causality as well as the concurrency of a control algorithm. To model non-autonomous behavior Interpreted Petri Nets (IPN) have been introduced (Moalla et al. 1978, König and Quäck 1988). IPN are ordinary Petri nets with binary markings extended by means for the explicit description of input/output facilities. In Signal IPN (SIPN) the influence of the environment on the system is based on signals instead of events used in other IPN approaches. In SIPN transitions are associated with a firing condition given as a Boolean function of the input signals. The places of an SIPN specify output signals. In (Frey 2000) a definition for SIPN is given that is not restricted to binary I/O-signals but allows the use of signals from various domains. Another extension of the SIPN framework is the use of hierarchy (Frey 2002). In the following, the two ideas are combined resulting in hybrid, hierarchical SIPN.

The rest of the paper is organized as follows. In Section 2 hybrid SIPN (hSIPN) are presented. Section 3 gives an overview on properties for formal correctness of controllers and relates them to the properties defined for hSIPN. In Section 4 a hierarchy concept is introduced and methods for the analysis of hierarchical SIPN based on the subnets are presented.

2 HYBRID SIGNAL INTERPRETED PETRI NETS

The hybrid SIPN is a direct extension of the basic SIPN definition, adding means for the use of non-binary I/O-signals. Note, the use of timers and the assignment of intervals to a signal as presented in
A hybrid Signal Interpreted Petri Net is given by an ordinary Petri net with places $P$, transitions $T$, arcs $F$, and binary initial marking $m_0$, with $|P|, |T|, |F| > 0$

2.1 Dynamic behavior

The dynamic behavior of an hSIPN is given by the flow of tokens through the net i.e. the change of its marking. This flow is realized by the firing of transitions $t_i$, arcs $F$, and puts a token on each of its post-places (places $p_j$) with $(t_i, p_j) \in F$ and removes a token from each of its pre-places (places $p_i$) with $(p_i, t_i) \in F$. For the firing there are four rules:

1. A transition is enabled, if all its pre-places are marked and all its post-places are unmarked.
2. An enabled transition fires immediately, when its firing condition is fulfilled: $\phi(t_i) = \text{True}$.
3. All transitions that can fire and are not in conflict with other transitions fire simultaneously. Note: Conflicts are treated as design errors in SIPN, there are no rules for conflict resolution.
4. The firing process is iterated until a stable marking is reached (i.e. until under the current setting of input signals no more transitions can fire).

After the firing of transitions, the output signals are recalculated by applying $\Omega$ to the marking.

2.2 Output Function

An output signal $o$ in an hSIPN can have a defined value out of its domain or may be left undefined $o \notin v(o) \cup \{\cdot\}$. During the work with these signals the following points are of interest:

1. A signal may be influenced by several places of the hSIPN at the same time and we are interested in the resulting effect, i.e. the product or the conjunction of these influences.
2. We are interested in summarizing the influence of an hSIPN on a signal at different times. This leads to the sum or disjunction of the influences.

Assume two places influencing an output signal at the same time. The product of the two values specified by those places, gives the resulting value assigned to the output signal. Given two values $v_1$ and $v_2$, the combinations according to Table 1 are possible. The leftmost column gives the possible values of the first term in the product and the top row gives the possible values of the second term. In this table, $a$ and $b$ are two different numerical values from the corresponding domain, $-\cdot$ means unspecified, and $c$ stands for contradiction.

<table>
<thead>
<tr>
<th>$v_1 \cdot v_2$</th>
<th>a</th>
<th>b</th>
<th>$\cdot$</th>
<th>c</th>
<th>d</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>$\cdot$</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>d</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>o</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
</tr>
</tbody>
</table>

If an output signal is influenced at different times, the sum according to Table 2 results. Again, $a$ and $b$ are two different numerical values from the corresponding domain.

<table>
<thead>
<tr>
<th>$v_1 + v_2$</th>
<th>a</th>
<th>b</th>
<th>$\cdot$</th>
<th>c</th>
<th>d</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>$\cdot$</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>

For the application of the defined algebra in hierarchical SIPN (Section 4), the product has to be extended by the additional symbols introduced as results for the sum. Because there, for the purpose of analysis, a subnet may be abstracted as a single place with an output-function that describes all possible outputs of the subnet. Hence, this output-function can contain values like for example $o$ or $d$. The calculation of the resulting outputs of states where such an abstracted place is marked necessitates the extension of the product presented in Table 3.

<table>
<thead>
<tr>
<th>$v_1 + v_2$</th>
<th>a</th>
<th>b</th>
<th>$\cdot$</th>
<th>c</th>
<th>d</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>$\cdot$</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
</tbody>
</table>
marked under $m$ has to be built. Let $P_{\omega}(m)$ be the set of places marked under $m$ then the following formula for the output results:

$$\Omega(m) = \prod_{\omega \in P_{\omega}(m)} \omega(p).$$

Let $RS_{SIPN}$ be the reachability set of the SIPN (i.e. the set of all markings reachable during the nets evolution). The possible output produced by an SIPN during its evolution is denoted by $\Omega_S$. $\Omega_S$ is the sum over the output of all markings reachable in the SIPN. $\Omega_S(SIPN)(o_i)$ gives the possible output produced for the signal $o_i$:

$$\Omega_S(SIPN) = \sum_{m \in RS_{SIPN}} \Omega(m).$$

3 SIPN VERIFICATION

In this section, an informal specification of standard functional properties for control algorithms is presented. All the properties are essential for the correct functioning of an implemented controller and they are independent of the control problem. The properties deal with the formal correctness (Frey 2002) of the algorithm. The first property, is the deterministic behavior of the controller:

(P1) Every control algorithm has to be deterministic. If it was not, its behavior in a given situation would depend on implementation aspects. This of course cannot be the aim of a correct design. In detail, this means that in every state of the controller (a) the reaction on possible input signals is defined and (b) a non-contradictory value for each output signal is specified. Properties formulated similar to P1 are widely accepted as mandatory and can be found in many works on formal methods in control. (König and Quäck 1988) use a type of Logic Control Interpreted Petri Net and define an algorithm fulfilling P1 as „implementation-worthy“. The concept of a „sound Grafce“ defined in (David and Alla 1992) is also based on deterministic behavior.

In addition to deterministic behavior three optional standard functional properties can be specified:

(P2) Every part of a controller should work in the sense that it is always possible that the respective code is executed again. When a part of a control algorithm does not work anymore, there is—in general—an error in the design of the control algorithm. There are cases, where this property is not desired.

(P3) P3 is a weakened version of P2: A controller should never get completely stuck. A controller that should run in a cyclic fashion after a start-up phase does not fulfill the property P2, because the start-up part is never activated again. However, it should fulfill P3. There are controllers that should get stuck in defined states. For these, P3 is not desired.

(P4) A control algorithm should be able to reach its initial state again. This property implies that automatic error recovery is possible. As with P2 and P3, there are cases where this property is not desired.

To verify these properties they have to be formalized. In the next sub-section formal properties of an hSIPN are presented. In sub-section 3.2, the properties P1 to P4 given above are related to the formal properties.

3.1 Analysis of hybrid SIPN

First of all properties defined for ordinary Petri nets can be extended to hSIPN: An hSIPN is

1. conflict-free if there is no reachable marking such that under any possible setting of input signals two transitions are in conflict.
2. deadlock-free if it contains no deadlocks.
3. live if all its transitions are live. A transition is live if it can always fire again.
4. reversible if for every reachable marking, there exists a sequence of input signal settings such that the initial marking is reached again.

For hSIPN there are some additional properties to be checked:

1. An hSIPN terminates if there is no cycle without a stable marking.
2. An output signal is said to be specified if in all states of the hSIPN at least one place assigns a value to it: $\forall i: \Omega_S(o_i) \in v(o_i) \cup \{d, c\}$.
3. An output signal is said to be non-contradictory if in no state of the hSIPN it is assigned contradictory values: $\forall i: \Omega_S(o_i) \in v(o_i) \cup \{d, -\}$.
4. An output signal is called formally correct if it is always specified and non-contradictory: $\forall i: \Omega_S(o_i) \in v(o_i) \cup \{d\}$.

These properties can be verified either by using the SIPN reachability graph as defined in (Frey 2002) or by model-checking as shown in (Klein et al. 2002).

3.2 Criteria for Formal Correctness

There are various threats to the deterministic behavior specified in property P1. First, two transitions that are in conflict in an SIPN lead to a behavior that is not determined. Of course, an implemented algorithm will solve this non-determinism but it cannot be the aim of a correct controller design, to leave the decision of how to react on an input in a given situation to implementation aspects.

Another source of non-deterministic behavior is the existence of endless loops in an SIPN, i.e. an SIPN that does not terminate. An endless loop may even lead to the hang-up of the implemented controller. Further problems can arise due to the distributed assignment of values to the output signals in SIPN. On the one side, it is possible that an output signal is assigned no value at all (not specified output). On the other side, different values can be assigned in different places of the SIPN in the same reachable marking (contradictory output).

The following figures show examples of SIPN algorithms that are not deterministically defined. The algorithm in Fig. 1 is not conflict-free: If in the initial state the input signal $i_1$ is set to one it is not clear whether transition $t_1$ or $t_2$ should fire. Fig. 2 shows an SIPN that gets into an endless loop as soon as the input signal $i_1$ is set to one.
For the net in Fig. 3 the calculation of possible outputs results in $\Omega_0(\text{SIPN}) = (c, d, -)$ meaning that the outputs are contradictory for $o_1$ in at least one reachable state and not specified for $o_3$ in all reachable states.

The other three properties can be directly related to specific SIPN properties (cf. Table 4). Since the verification of the properties does not include any knowledge about the process under control, the reaction of the controller on unknown behavior like for example sensor failures is included in the calculation. If the algorithm is proved to be deterministic in the presented sense, then the controller will behave deterministically whatever process it is connected to. If the algorithm is not live, deadlock-free, or reversible in this sense then the controller cannot be live, deadlock-free, or reversible regardless what process it is connected to.

Table 4: Criteria for formal correctness.

<table>
<thead>
<tr>
<th>Informal Property</th>
<th>SIPN Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1 (mandatory)</td>
<td>Conflict-freeness</td>
</tr>
<tr>
<td></td>
<td>Termination</td>
</tr>
<tr>
<td></td>
<td>Specified output</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>P2 (optional)</td>
<td>Liveness</td>
</tr>
<tr>
<td>P3 (optional)</td>
<td>Deadlock-freeness</td>
</tr>
<tr>
<td>P4 (optional)</td>
<td>Reversibility</td>
</tr>
</tbody>
</table>

4 HIERARCHICAL SIPN

For real-world programs, SIPN controllers tend to be large and difficult to handle like for most visual languages. Furthermore, methods of analysis are no longer applicable because of the exponential growth of the systems state-space. Last but not least, the code for large SIPN gets very slow on the target system (PLC or other control hardware). However, in most cases subnets can be identified which realize certain subtasks. Therefore, the SIPN can be replaced by an abstract SIPN where single places are used instead of these subnets; the subnets are subordinated to these hierarchical places. Subnets can in turn contain hierarchical places and so on. Hierarchy does not only enhance the readability, the analyzability, and the resulting code of an SIPN. It is also a valuable means for structured design. With a hierarchical structure bottom-up as well as top-down and mixed approaches to design and analysis are possible. Top-down design and analysis is done by refining components in a net, whereas in bottom-up design, tested components are connected via a common SIPN on a higher level. In practical applications a combination of both techniques is the most appropriate: The general structure of an algorithm is designed top-down and at some level down the hierarchy pre-programmed and tested modules, often containing hierarchical structures themselves, are included.

Petri Nets can be refined hierarchically by putting a subnet in places and/or transitions (Reisig 1982). In SIPN only the refinement of places is used, because places describe a partial state or process in the controller that can take some time whereas transitions by definition take no time. The two SIPN in Fig. 4 show the same dynamic behavior. In the hierarchical SIPN with place refinement place $p_2$ receives a token whenever place $p_{23}$ receives one and transition $t_1$ can only fire if place $p_3$ is marked.

For the definition given here, the following three considerations have been essential:

1. For an effective implementation of the hierarchical SIPN, a subnet should be passive in the sense that it does not influence the outputs of the hierarchical SIPN and its transitions cannot fire while it is not activated by the corresponding hierarchical place.

2. To allow the application of analysis methods defined for „flat“ SIPN, the behavior of an unfolded, i.e. „flatted“, hierarchical SIPN should be identical to the behavior of the hierarchical SIPN. Unfolding is done by replacing hierarchical places with a structure putting the corresponding subnet in parallel to the hierarchical place (cf. Fig. 4). Note that it is not sufficient to replace the hierarchical place with the underlying subnet because the hierarchical place has an output function that would get lost in this case.

3. A subnet definition should be self-contained to allow an analysis of its properties without taking the other elements of the SIPN into account.

![Fig. 1. SIPN that is not conflict-free.](image1)

![Fig. 2. SIPN that does not terminate.](image2)

![Fig. 3. SIPN with unspecified and contradictory defined output signals.](image3)

![Fig. 4. Unfolding of a hierarchical SIPN.](image4)
A hierarchical SIPN, called SIPN\textsuperscript{H}, is an SIPN extended by a mapping \( \eta \) associating the places of the net with subnets.

A hierarchical SIPN is given by a 11-tuple \( \text{SIPN}\textsuperscript{H} = (P, T, F, m_0, I, O, \varphi, \omega, \Omega, \nu, \eta) \) with

- the sub-tuple \( (P, T, F, m_0, I, O, \varphi, \omega, \Omega, \nu) \) is an hSIPN and
- \( \eta \) is a mapping associating places \( p_i \in P \) with subnets \( \eta(p_i) \), \( \eta \) is not defined \( (\eta(p_i) = \text{nill}) \) for places containing no subnet.

A subnet associated with a place \( p_i \) in an SIPN\textsuperscript{H} denoted by SIPN\textsubscript{sub} = \( \eta(p_i) \) is itself an SIPN\textsuperscript{H} with the following restrictions:

- There exists exactly one input-place \( p_{in} \) and exactly one output-place \( p_{out} \).
- The subnet is passive, i.e. while the hierarchical place associated with the subnet is not marked, no transition inside the subnet is enabled and the subnet does not influence any output signals.
- The sets of input and output signals of the subnet are subnets of the respective sets in the SIPN\textsuperscript{H}.

A subnet SIPN\textsubscript{sub} associated with a hierarchical place \( p_i \in P \) in an hierarchical SIPN \( \text{SIPN}\textsuperscript{H} = (P, T, F, m_0, I, O, \varphi, \omega, \Omega, \nu, \eta) \) is given by \( \eta(p_i) = \text{SIPN}\textsubscript{sub} = (P_s, T_s, F_s, m_{0_s}, I_s, O_s, \varphi_s, \omega_s, \Omega_s, \nu_s, \eta_s, P_{in}, P_{out}) \) with the following properties:

- \( \exists p_{in} \in P_s \) with \( \bullet p_{in} = \emptyset \) (input-place)
- \( \exists p_{out} \in P_s \) with \( p_{out} \bullet = \emptyset \) (output-place)
- \( m(p_i) = 0 \Rightarrow \forall t \in T_s; (\exists p \in t: m(p) = 0) \Rightarrow \forall p \in t; m(p) = 1 \) (passivity, transitions)
- \( I_s \subseteq I \) (passivity, outputs)
- \( P_{in} \cap P = \emptyset \) (input-place via a reset transition, a marked idle place, and set transition. The firing condition for the set and reset transitions are defined in a way that avoids dynamic synchronization. The possible output \( \Omega_s \) of the extended net is derived by summing the output over all reachable states, excluding those containing the idle place. Fig. 5 shows the extended net for the net in Fig. 6. The corresponding reachability graph is drawn in Fig. 7. The gray shaded states are not used in the calculation of \( \Omega_N \) because they contain the idle place.

4.1 Analysis of Hierarchical SIPN

In general, the analysis of an SIPN\textsuperscript{H} can be done on the corresponding unfolded SIPN. The unfolded SIPN corresponding to a SIPN\textsuperscript{H} is derived by putting the unfolded subnets in parallel to their corresponding hierarchical places in the main SIPN. The problem with unfolding is that the resulting unfolded net can get very large. Fortunately, conditions for the formal correctness of the SIPN\textsuperscript{H} can be derived based on all the single nets contained in the hierarchical structure. To allow an analysis on these nets they have to be converted into SIPN that do not depend on the other nets in the structure. These nets are called extended SIPN.

With the extended SIPN a test for passivity can be given and finally criteria for the correctness of the hierarchical SIPN based on the properties of all the extended nets corresponding to the nets contained in the hierarchical structure are given.

The extended net corresponding to the main, i.e. highest level, net in an SIPN\textsuperscript{H} is derived by removing the underlying subnets and adding their possible output \( \Omega_N \) (built over all reachable states that do not contain the idle-state, see below) to the corresponding hierarchical place.

The extended net corresponding to a subnet in an SIPN\textsuperscript{H} is also derived by removing the underlying subnets and adding their possible output to the corresponding hierarchical place. However, in addition to that the output place of a subnet is connected to its input place via a reset transition, a marked idle place, and a set transition. The firing condition for the set and reset transitions are defined in a way that avoids dynamic synchronization. The possible output \( \Omega_s \) of the extended net is derived by summing the output over all reachable states, excluding those containing the idle place. Fig. 6 shows the extended net for the net in Fig. 5. The corresponding reachability graph is drawn in Fig. 7. The gray shaded states are not used in the calculation of \( \Omega_N \) because they contain the idle place.
The definition of a subnet demands its passivity. With the extended SIPN a simple test for passivity can be given: A subnet $\text{SIPN}_{\text{sub}}$ of a hierarchical SIPN is passive if all its own subnets are passive and for the reachability graph of its extended net the following conditions hold:

1. The output of all states containing $p_{\text{idle}}$ is undefined for all output variables.
2. All arc notations originating from states containing $p_{\text{idle}}$ are a logical conjunction with the term $\neg\text{Reset}$.

The set of all extended nets derived from an SIPN$^H$. The extended net of the main net, the extended nets of all its subnets, there subnets,... is called Extended Set $\mathcal{ES}(\text{SIPN}^H)$.

4.2 Analysis based on the Extended Set

In the following, conditions for the validity of formal correctness criteria in an hierarchical SIPN based on properties of the SIPN and its extended set are given. For most criteria there is a distinction between live and non-life SIPN$^H$. This is because if the SIPN$^H$ is not live, a certain subnet that violates a property may never be activated.

Conflicts, and Deadlocks are local properties of an SIPN therefore an SIPN$^H$ is conflict-free, or deadlock-free if all SIPN $\in \mathcal{ES}(\text{SIPN}^H)$ fulfill the respective properties. In the general case, this is a sufficient condition, for live SIPN$^H$ the condition is necessary and sufficient.

Liveness: An SIPN$^H$ is live if and only if all SIPN $\in \mathcal{ES}(\text{SIPN}^H)$ are live.

Reversibility: An SIPN$^H$ is reversible if all SIPN $\in \mathcal{ES}(\text{SIPN}^H)$ are reversible. In the general case, this is a sufficient condition, for live SIPN$^H$ the condition is necessary and sufficient.

Termination: An SIPN$^H$ terminates if all SIPN $\in \mathcal{ES}(\text{SIPN}^H)$ terminate. In the general case, this is a sufficient condition, for live SIPN$^H$ the condition is necessary and sufficient.

Specified output: If the outputs are specified in the Extended Net of the top-level SIPN in an SIPN$^H$ then the outputs are specified in the SIPN$^H$. This does not necessarily mean that the outputs are specified in all SIPN $\in \mathcal{ES}(\text{SIPN}^H)$. Specified outputs in all nets of a hierarchical SIPN are sufficient but not necessary for the specified output in the SIPN$^H$.

Non-contradictory output: If the outputs are non-contradictory in the Extended Net of the top-level SIPN in an SIPN$^H$ then the outputs are non-contradictory in the SIPN$^H$. This does not necessarily mean that the outputs are specified in all SIPN $\in \mathcal{ES}(\text{SIPN}^H)$. Non-contradictory outputs in all nets of a hierarchical SIPN are in the general case neither necessary nor sufficient for the non-contradictory output in the SIPN$^H$. It is not sufficient, because even if the output of all subnets is non-contradictory, the combination of the nets may lead to a contradiction. It is not necessary because a subnet containing contradictions may not be activated. For live SIPN$^H$, non-contradictory outputs in all subnets are necessary but not sufficient for non-contradictory outputs of SIPN$^H$.

Formally correct output: Formally correct means specified and non-contradictory. From the discussion on specified and non-contradictory outputs above it follows that the outputs of an SIPN$^H$ are formally correct if the outputs of the extended net of the main net are formally correct. The formal correctness of the outputs of all SIPN $\in \mathcal{ES}(\text{SIPN}^H)$ is neither necessary nor sufficient for this.

5 Conclusions

In this contribution, two extensions of Signal Interpreted Petri Nets are combined. First, the inclusion of non-binary signals as inputs and outputs of the net. And second, the concept of hierarchy. Based of the defined hierarchy structure, methods for the analysis of standard properties of the hierarchical SIPN based on the single nets in the hierarchical structure are presented.

References


