ALGORITHMS FOR INTERPRETED HIERARCHICAL HIGH LEVEL PETRI NETS

Marius BREZO VAN
University of Craiova
str. Lapus, nr. 5
marius.brezovan@comp-craiova.ro

Dumitru Dan BURDESCU
University of Craiova
str. Lapus, nr. 5
burdescu@topedge.com

Abstract. This paper proposes a new class of Petri nets called Interpreted Hierarchical High Level Petri Nets (IHHLPN), and an algorithm for IHHLPN semantics implementation. IHHLPN are Hierarchical High Level Petri Nets augmented with an interaction component between a modeled system and its environment. Both the formal syntax and dynamic semantics of the proposed formalism are defined. For semantics implementation of IHHLPN, a token-player optimized algorithm is proposed.

Keywords: hierarchical Petri nets, algebraical specifications, interpreted Petri nets.

Introduction

To specify complex and large systems, many high level Petri nets formalisms have been developed. The main net classes are Predicate/Transition Nets, Coloured Petri Nets and Algebraic Nets [4, 7, 8, 10, 11, 12]. By introducing of modularity and hierarchy concepts, new classes of Petri nets have been developed [5, 6, 8, 1].

All these Petri net classes allow to model discrete event systems. To describe the control level of a physical system, the Petri nets must specify the interaction between the control system and its controlled system.

This paper is based on a formalism called Hierarchical High Level Petri Nets [1], which is extended by including the interaction concept, resulting a new class of nets, called Interpreted Hierarchical High Level Petri Nets.

The semantics of Interpreted Hierarchical High Level Petri Nets is implemented by using an optimised token-player algorithm [13].

Interpreted Hierarchical High Level Petri Net Definition

Interpreted Hierarchical High Level Petri Net are defined as a Hierarchical High Level Petri Net with the property that each transition has associated a set of operations. For simplicity the interpreted nets are called control nets, while non-interpreted nets are called modeling nets. In the following the set of all Interpreted Hierarchical High Level Petri Nets is denoted by IHHLPN.

A hierarchical net contains two types of nodes: elementary nodes and non-elementary nodes, which represent hierarchical subnets. These nodes are called supernodes. Because several supernodes can have associated a same subnet, the notion of page will be used instead of subnet. A page is a net having a set of input and output nodes, a set of input and output ports, and a set of channels connecting ports to the input/output nodes. In the following, the set of all pages by PAGE, the set of all elementary nodes by ENODE, and the set of all composed nodes by CNODE.

The association of pages to supernodes is made by using an assignment map denoted by $\sigma$:

$\sigma$: NODE $\rightarrow$ PAGE $\cup$ $\{\lambda\}$,

$\sigma(n) = \lambda$, if $n \in$ ENODE,

$\sigma(n) = P \in$ PAGE, if $n \in$ CNODE.

Definition 1. A page is a tuple:

$P_g = (Hlhp, KI, KO, IN, OUT, ch, \varepsilon)$, where:
(1) HHlpn ∈ IHHLPN is an Interpreted Hierarchical High Level Petri Net;
(2) KI and KO represent the set of input and output nodes of the net HHlpn:
    KI ⊆ {n ∈ P(HHlpn) ∪ T(HHlpn) | °n = Φ}
    KO ⊆ {n ∈ P(HHlpn) ∪ T(HHlpn) | n° = Φ}
(3) IN and OUT represents the set of input and output ports of the page Pg:
    IN ⊆ {#n | n ∈ P(HHlpn) ∪ T(HHlpn) 
    OUT ⊆ {#n | n ∈ P(HHlpn) ∪ T(HHlpn) }
   The set of all ports of all pages is denoted by PORT.
(4) ch: IN ∪ OUT → KI ∪ KO, is a map, which associates each port of the page Pg to an input/output
   node of the net HHlpn where the port is connected.
(5) The map ε associates each input/output supernode to a bijection as follow:
    ε: (KI ∪ KO) ∩ NODE → BIJ,
    ε(n) ∈ BIJ(IN ∪ OUT, IN(σ(HHlpn)(n)) ∪ OUT((σ(HHlpn)(n))))

Remark. In the above definition:

- T(HHlpn) and P(HHlpn) represent the sets of transitions and places respectively of the
  Interpreted Hierarchical High Level Petri Net HHlpn;
- σ(HHlpn) represents the map of page assignment of HHlpn.

The function ε is used when an input/output node of a page is itself a composed node. This map
associates each supernode that is an input/output node of a page, to a bijection between input/output
ports of the page containing the supernode, and input/output ports of the page associated to the
supernode. In this definition BIJ(A, B) denotes the set of all bijective functions from A to B.
The map ch defines a set of channels connecting input/output ports to the input/output nodes of a page.

Definition 2. Let HHlpn be a Interpreted Hierarchical High Level Petri Net.

(1) A node n is membership of HHlpn, iff one of the following conditions are satisfied:
   (a) n ∈ P(HHlpn) ∪ T(HHlpn);
   (b) ∃m ∈ (P(HHlpn) ∪ T(HHlpn)) ∩ NODE, 
      so that n is membership of Net(σ(m)).
(2) The set of all nodes contained into HHlpn is denoted by Node(HHlpn), and defined as:
   Node(HHlpn) =
   {n | n is membership of HHlpn };
In a similar way, the following notions can be defined: the set of all places contained into HHlpn, the
set of all transitions, the set of all elementary places, the set of all elementary transitions, and the set of
all elementary nodes contained into HHlpn:
   Node(HHlpn) = Place(HHlpn) ∪ Trans(HHlpn)
   EPlace(HHlpn) = Place(HHlpn) ∩ ENODE,
   ETrans(HHlpn) = Trans(HHlpn) ∩ ENODE,
   ENode(HHlpn) = EPlace(HHlpn) ∪ ETrans(HHlpn)
   Cnode(HHlpn) = Node(HHlpn) – ENode(HHlpn).

Another map used in definition of IHHLPN is called port assignment map. It is defined only for
supernodes and associates adjacent nodes of a supernode to the input/output ports of the page
associated to the supernode:
   η: (P ∪ T) ∩ NODE → BIJ,
   η(n) ∈ BIJ(#n, IN(σ(n)) ∪ OUT((σ(n))))

The definition of IHHLPN uses some notations from algebraical specifications [3, 11, 17] to annotate
the elements of his hierarchical structure. A natural signature with variables Sig=(S,O,V) is associated
with a Hierarchical High Level Petri Net, with variables having a corresponding Sig-Algebra, H =
(S_H, O_H). There are three functions used for net annotation:
• Type: EPlace(Hhlpn)→SH is a map, which associates a type to each simple place;

• AA: A→BTERM(O∪V), is a map which associates each arc to a multiset of terms compatible with the type of the place adjacent to the arc:
  \[ \forall a \in A, a=(p,t) \text{ or } a=(t,p) \text{ with Type}(p)=H_s \cdot \]
  \[ AA(p) \in BTERM(O\cup V)_s; \]

• TC: ETrans(Hhlpn)→TERM(O∪V)_bool, is a map, which associates each simple transition to a logical formula.

Because it is desired to define the nets for system control, the interaction between the control system and its controlled system must be described [14, 16]. To realise this operation, each transition has additionally an interaction component with the environment. This process represents an interpretation operation of Petri nets.

Definition 3. Let t be a transition of a Interpreted Hierarchical High Level Petri Net.

1) an input operation associated to transition t is a syntactic construction of the form:
   \[ \langle \text{read} \rangle ::= \text{input}(\langle \text{identifier} \rangle, \{\langle \text{identifier} \rangle \}^*) \]
where identifiers are output variables of t:
   \[ \{ \forall v \in V \mid \exists a \in \text{post}(t) \cdot v \in AA(a) \} – \]
   \[ \{ \forall v \in V \mid \exists a \in \text{pre}(t) \cdot v \in AA(a) \} \]

2) an output operation associated to transition t is a syntactic construction of the form:
   \[ \langle \text{write} \rangle ::= \text{output}(\langle \text{parameter} \rangle, \{\langle \text{parameter} \rangle \}^*) \]
   \[ \langle \text{parameter} \rangle ::= \langle \text{identifier} \rangle \mid \langle \text{constant} \rangle \]
where identifiers are input variables of t:
   \[ \{ \forall v \in V \mid \exists a \in \text{pre}(t) \cdot v \in AA(a) \} – \]
   \[ \{ \forall v \in V \mid \exists a \in \text{post}(t) \cdot v \in AA(a) \} \]

3) an assignment operation associated to transition t is a syntactic construction of the form:
   \[ \langle \text{assignment} \rangle ::= \langle \text{identifier} \rangle \leftarrow \langle \text{expression} \rangle, \]
where identifiers are output variables of t, while expressions are terms compatible with the type of the left side variable:
   if \( \exists s \in S \cdot \langle \text{identifier} \rangle \in V_s \),
   then \( \langle \text{expression} \rangle \in \text{TERM}(O\cup V)_s \).

Let assig be an assignment operation. The operation identifier is denoted by ident(assig), while the expression of the operation is denoted by expr(assig).

4) the set of all input operations is denoted by INP, the set of all output operations is denoted by OUT and the set of all assignment operations is denoted by ASSIG.

The transitions, which have an associated input operation, represent a distinct class of transitions. They are called external observable transitions, and are always associated to external events. To determine uniquely the associated external event, this must be specified in a distinct mode.

Remark: In the following, EVENT denotes the set of all external events.

Definition 4. A Interpreted Hierarchical High Level Petri Net is a tuple:
   \[ \text{Hhlpn} = (\text{PG}, \text{HN}, \text{Sig}, \text{AN}, \text{INT}, M_0), \]
where:

1) PG is a finite set of pages: \( \text{PG} \subset \text{PAGE} \);

2) HN = (P, T, A, σ, η) is a net with a hierarchical structure:
   - P is a finite set of places and T is a finite set of transitions with condition that:
     \[ P \cap T = \emptyset ; \]
     \[ (P\cap \text{CNODE}) \cup (T\cap \text{CNODE}) \neq \emptyset ; \]

Let $A \subseteq (P \times T) \cup (T \times P)$ be a finite set of arcs with condition that:
\[ \forall a \in A, a = (n_1, n_2) \cdot \neg ((n_1 \in \text{CNODE}) \land (n_2 \in \text{CNODE})); \]

- $\sigma: P \cup T \rightarrow PG \cup \{\lambda\}$, is the map of page assignment, which verifies the following conditions:
  \[ \forall n \in (P \cup T) \cap \text{CNODE} \cdot H\text{hlpn} \cap \text{Net}(\sigma(n)) = \Phi; \]

- $\eta: (P \cup T) \cap \text{CNODE} \rightarrow \text{BJ}$, is the port assignment map for supernodes;

(3) $\text{Sig} = (S,O,V)$ is a Natural-Boolean signature with variables having a corresponding Sig-Algebra, $H = (S_H,O_H)$.

- $\forall n \in (P \cup T) \cap \text{CNODE} \cdot V \cap \text{Var}(\sigma(n)) = \Phi,$

(4) $\text{AN} = (\text{Type,AA,TC})$ is the net annotation:
  - $\text{Type}: P \cap \text{ENODE} \rightarrow S_H$ associates a type to each simple place;
  - $\text{AA}: A \rightarrow \text{BTERM}(O \cup V)$ associates each arc to a multiset of terms compatible with the type of the place adjacent to the arc:
    \[ (\forall a \in A, a = (p,t) \text{ or } a = (t,p) \cdot \text{Type}(p) = H_a) \cdot \text{AA}(p) \in \text{BTERM}(O \cup V)_s; \]
  - $\text{TC}: T \cap \text{ENODE} \rightarrow \text{TERM}(O \cup V)_\text{bool}$ associates each simple transition to a logical formula, called the transition condition.

(5) $\text{INT} = (\text{ev,in,out})$ is the net interpretation:
  - $\text{ev}: T \rightarrow \text{EVENT} \cup \Phi$, is a map, which associates each transition to at most one external event; a transition $t$ for which $\text{ev}(t) \neq \Phi$ is called an observable transition;
  - $\text{in}: T \rightarrow \text{INP} \cup \Phi$, is a map, which associates each transition to at most one input operation; the input operation can be associated only to observable transitions:
    \[ \forall t \in T \cap \text{ENODE} \cdot \text{in}(t) \neq \Phi \Rightarrow \text{ev}(t) \neq \Phi \]
  - $\text{out}: T \rightarrow \text{OUT} \cup \Phi$, is a map, which associates each transition to at most one output operation. In addition, the supertransitions can not have associated any operation:
    \[ \forall t \in T \cap \text{ENODE} \cdot \text{ev}(t) = \text{in}(t) = \text{out}(t) = \Phi; \]

(6) $M_0$ represents the initial marking of the net:
\[ M_0: P \cap \text{ENODE} \rightarrow \bigcup_{p \in P \cap \text{ENODE}} (\mu \text{Type}(p)), \]
\[ \forall p \in P \cap \text{ENODE}, M_0(p) = \mu \text{Type}(p). \]

**Dynamic Semantics of Interpreted Hierarchical High Level Petri Nets**

Whereas a superplace represents an abstraction of a low level net, its state is given by the states of places of the associated net. In this way, the tokens are distributed between elementary nodes of the Hierarchical net.

**Definition 5.** A marking $M$ of an Interpreted Hierarchical High Level Petri Net is a map
\[ M: \text{EPlace}(H\text{hlpn}) \rightarrow \bigcup_{p \in \text{EPlace}(H\text{hlpn})} (\mu \text{Type}(p)), \]
with the property that:
\[ \forall p \in \text{EPlace}(H\text{hlpn}) \cdot M(p) \in \mu \text{Type}(p). \]
On one can define the following recursive functions:

If \( \delta \) denoting a bijective map, which associates the nodes adjacent to supernode \( n \) to the internal input/output nodes of the associated page:

\[
\delta: \text{NODE} \times \text{CNODE} \rightarrow \text{NODE},
\]

\[
\delta(x, n) = \text{ch}(\eta(n)(x)), \text{ where } n \in \text{CNODE and } x \in \eta^o(n).
\]

If \( \delta(x, n) \in \text{CNODE} \), that means the input/output node of the page is itself a supernode, and the previous process can be repeated.

One can define the following recursive functions:

\[
\eta^1(n)(x) = \eta(n)(x), \quad \delta^1(x, n) = \delta(x, n),
\]

\[
\eta^k(n)(x) = \varepsilon(\delta^{k-1}(x, n))(\eta^{k-1}(n)(x)), \quad k > 1,
\]

\[
\delta^k(x, n) = \delta(\eta^{k-1}(n)(x)), \quad \delta^{k-1}(x, n)), \quad k > 1.
\]

The functions \( \eta^k \) determine the input/output port from the \( k \)-th nested page connected to the node \( x \); the functions \( \delta^k \) determine the input/output node from the \( k \)-th nested page connected to the node \( x \).

Let \( k \) the smallest nonnegative number for which \( \delta^k(x, n) \in \text{NODE}_E \). This number exists by the definition of Interpreted Hierarchical High Level Petri Nets. Denoting \( \delta^k(x, n) \) by \( \delta^k \), the map \( \delta^k \) represents the transitive closure of the function \( \delta \).

In the case that an elementary node is an input/output node of a page, then there exists another elementary node connected to that by input/output channels. For each input/output channel of a page, the elementary node to which the channel is connected must be determined.

Let \( \gamma \) be a map defined as:

\[
\gamma: \text{NODE} \times \text{PORT} \rightarrow \text{NODE} \cup \text{PORT},
\]

\[
\gamma(n, k) = \gamma(\text{up}(n))^1(k), \text{ where } k \in \text{ch}^1(n).
\]

If \( \gamma(n, k) \) is also a port in the page \( \sigma(\text{up}(\text{up}(n))) \), the above operation is repeated with \( \text{up}(\text{up}(n)) \) in place of \( \text{up}(n) \). The following recursive functions can be defined:

\[
\text{up}^0 = \text{id}, \quad \text{up}^0(x) = \text{up}(\text{up}^{p-1}(x));
\]

\[
\gamma^0 = \gamma;
\]

\[
\gamma^p(n, k) = \gamma^p(n, k), \quad p > 1.
\]

Let \( n \) be the smallest nonnegative number for which \( \gamma^p(n, k) \in \text{ENode}(H\text{hlpn}) \). This number exists by definition of Interpreted Hierarchical High Level Petri Nets. Denoting \( \gamma^p(n, k) \) by \( \gamma^p \), the map \( \gamma^p \) is the transitive closure of the function \( \gamma \).

For every node inside of a page associated to a supernode, the name of the supernode can be uniquely determined. The map \( \text{up} \) is defined as follows:

\[
\text{up}: \text{NODE} \rightarrow \text{NODE}_C \cup \{ \lambda \},
\]

if \( \exists n \in \text{NODE} \) so that \( x \in \mathbb{P}(\sigma(n)) \cup T(\sigma(n)) \)

then \( \text{up}(x) = n \);

if \( \exists n \in \text{NODE} \) so that \( x \in \mathbb{P}(\sigma(n)) \cup T(\sigma(n)) \) then \( \text{up}(x) = \lambda \).

Let \( \text{up}^+(n, k) \) denoting \( \text{up}^0(n, k) \) for which \( \gamma^p(n, k) \in \text{ENODE} \). The function \( \text{up}^+(n, k) \) determines the node at the same level as \( \gamma^p(n, k) \) to which \( \gamma^p(n, k) \) is connected.

**Definition 6.** Let \( n \) be an elementary node of a hierarchical net, and \( k \) be an adjacent node to \( n \) if \( n \in \text{KI}(\sigma(\text{up}(n))) \cup \text{KO}(\sigma(\text{up}(n))) \), or a port connected to \( n \), if \( n \in \text{KI}(\sigma(\text{up}(n))) \cup \text{KO}(\sigma(\text{up}(n))) \).

1. The elementary node connected to \( n \) through \( k \) is denoted \( \text{link}(n, k) \), and is defined by:

\[
\text{link}(n, k) = k, \text{ if } k \in \eta^o(n) \cap \text{ENODE};
\]
(2) The multiset of terms that annotate the arc connecting the nodes \( n \) and \( k \) (for the first case), or \( \text{up}^+(n,k) \) and \( \gamma^+(n,k) \) (for the second case) is denoted \( \text{arc}(n,k) \), and is defined by:

\[
\text{arc}(n,k) = \begin{cases} 
\text{AA}(n,k), & \text{if } n \notin K_I(\sigma(\text{up}(n))) \cup K_O(\sigma(\text{up}(n))), \\
\text{AA}(\gamma^+(n,k), \text{up}^+(n,k)), & \text{if } n \notin K_I(\sigma(\text{up}(n))) \\
\text{AA}(\text{up}^+(n,k), \gamma^+(n,k)), & \text{if } n \in K_O(\sigma(\text{up}(n))).
\end{cases}
\]

By extension, if \( k \notin \circ n \) and \( k \notin \text{ch}^{-1}(n) \), it is assumed that \( \text{arc}(n,k) = \Phi \).

**Definition 7.** Let \( t \) be a transition of Interpreted Hierarchical High Level Petri Net. An assignment \( \alpha \) of the set of variables \( \text{Var}(t) \) is called occurrence mode if there exists an evaluation \( \text{Val}_\alpha \) in \( \text{BTERM}(O \cup V) \) so that the following statements hold:

(a) the selector \( \text{TC}(t) \) can be evaluated;

(b) all multisets associated to incident arcs of \( t \), that is the set \{ \( \text{AA}(a) \mid a \in \text{pre}(t) \cup \text{post}(t) \) \}, can be evaluated;

(c) the functions \( \text{in}, \text{out} \) and \( \text{assig} \) of the transition \( t \) can be evaluated; in addition: \( \forall a \in \text{assig}(t) \cdot \text{Val}_\alpha(\text{ident}(a)) = \text{Val}_\alpha(\text{expr}(a)) \).

The main condition for the transition \( t \) to be enabled is that all input elementary relevant places contain sufficiently tokens.

**Definition 8.** Let \( Hhlpn \) be a Interpreted Hierarchical High Level Petri Net, \( M \) a marking of \( Hhlpn \), \( t \) an elementary transition in \( Hhlpn \) and \( \alpha \) an occurrence of \( t \). The transition \( t \) is \( M \)-enabled with occurrence \( \alpha \), and it is denoted \( M[t:\alpha] \), iff the following condition is satisfied:

\[
\forall p \in \text{EPlace}(Hhlpn) \cdot \text{Val}_\alpha(\text{arc}(p,t)) \leq M(\text{link}(t,p)), \quad \text{Val}_\alpha(\Phi) = \Phi.
\]

**Definition 9.** Let \( Hhlpn \) be a Interpreted Hierarchical High Level Petri Net and \( t \) an elementary transition \( M \)-enabled with occurrence \( \alpha \). The firing of \( t \) leads to a following marking \( M^1 \), denoted by \( M[t:\alpha]M^1 \), and defined as follows:

\[
\forall p \in \text{EPlace}(Hhlpn) \cdot M^1(\text{link}(t,p)) = M(\text{link}(t,p))-\text{Val}_\alpha(\text{arc}(p,t))+\text{Val}_\alpha(\text{arc}(t,p)),
\]

where \( \text{Val}_\alpha(\Phi) = \Phi \), the empty multiset.

**Control Algorithms for Interpreted Hierarchical High Level Petri Net**

A token-player control algorithm represents a net interpreter, which tests and fires the enabled transitions, transporting tokens between different nodes of the net [2, 9, 13].

The main structure of a token-player algorithm has two distinct parts: (1) an external part, which updates the state of the controled system when it receives messages from the system, and (2) an internal part, which represents a sequence of controller state updating, and where the transitions proper to control system are fired.

The stable state of a IHHLPN is the state of waiting for an external event, that is the waiting for some data associated of an input operation of a transition. When an external event appears, all transitions waiting for the associated message must be detected.
The main disadvantage of this algorithm is that at each iteration all elementary transitions of the net must be tested. An optimal alternative uses the launch places [15]. Every transition has assigned an input place, which is tested: if this place has a non-empty marking, the corresponding transition is possible to be enabled, else the transition is certainly non-enabled. For this, another attribute associated to transitions, launch, is considered to specify the launch place for each transition:

\[\text{launch: } T \rightarrow \mathcal{P} \cup \{\lambda\}\]

The control algorithm uses an additionally list, l-list, to store the launch places of the elementary transitions. An initialization procedure is used to construct the list l-list and to initialize two lists, o-enabled and i-enabled, representing the possible observable and non-observable enabled transitions respectively. The lists o-enabled and i-enabled, are modified dynamically when the algorithm runs. A description of the algorithm is presented in procedure TokenPlayer:

```plaintext
procedure TokenPlayer
  InitIhh1pn
  ExternalCycle
end
```

The procedure InitIhh1pn generates the list of launch places, and initializes the lists of possible enabled transitions under initial marking. The procedure ExternalCycle realizes the external cycle:

```plaintext
procedure ExternalCycle
  repeat
    until ExternalEvent(a)
    t-list ← GetWaitingEventTransitions(a)
    if t-list=Φ then
      Error
      return
    else
      t ← SearchEnabledTransition(t-list, a)
      if t≠λ then
        InternalCycle(t)
      else
        if Card(t-list)>1 then
          Error
          return
        fi
      fi
    fi
  end
```

The procedure ExternalEvent is used to interface the environment and its control net, taking a message from environment representing a list of values, together the name of external event:

\[a = (e; a_1, ..., a_n)\]

The function GetWaitingEventTransitions searches into the set of observable transitions a subset of transitions having attached the event specified in external message. The function SearchEnabledTransition selects one transition from the list generated by GetWaitingEventTransitions, which is enabled. If no transitions are enabled, the value λ is
returned. This is the first type of non-determinism of the token-player algorithm, and its associated function must implement a selecting strategy.

Using the functions link and arc described above, the function Enabled can be described as follows:

```plaintext
function Enabled(t, a)
    if ¬ VariableAssign(t, a) then
        return false
    fi
    if t∈KI(σ(up(t))) then
        for *∀ k∈ch⁻¹(t) do
            if ¬(Eval(arc(t,k)) ⊆ M(link(t,k))) then
                return false
            fi
        od
    else
        for *∀ p∈°t do
            if ¬(Eval(arc(p,t)) ⊆ M(link(t,p))) then
                return false
            fi
        od
    fi
    if ¬ Eval(TC(t)) then
        return false
    fi
    return true
end
```

The function VariableAssign determines the occurrence mode α of a transition, assigning values to the variables linked to the transition. The function Eval realizes the evaluation operation for expressions which appear in net annotation, implementing the map Val_α.

The procedure InternalCycle implements internal cycle of algorithm:

```plaintext
procedure InternalCycle(t)
    FiringTransition(t)
    while i-enabled ≠ Φ do
        t_1 ← SearchEnabledTransition(
            i-enabled, Φ)
        if t_1 ≠ λ then
            FiringTransition(t_1)
        fi
    od
    ExternalCycle
end
```

The procedure for transition firing has two types of action: the typical actions to transition firing, and the updating of the list of possible enabled transitions under the new marking.

```plaintext
procedure FiringTransition(t)
    if t∈KI(σ(up(t))) then
        for *∀ k∈ch⁻¹(t) do
            DelTokens(M(link(t,k),
                Eval(arc(k,t)))
            if (∃ a∈l-list) ∧
```
if ObservableTransition(Transition(a)) then
    o-enabled ← o-enabled - {Transition(a)}
else
    i-enabled ← i-enabled - {Transition(a)}
fi
fi

for ∀p ∈ t do
  DelTokens(M(link(t,p), Eval(arc(p,t)))
  if (∃a ∈ l-list) ∧ (Place(a)=link(t,p)) ∧ (M(link(t,p))=Φ) then
    if ObservableTransition(Transition(a)) then
        o-enabled ← o-enabled - {Transition(a)}
    else
        i-enabled ← i-enabled - {Transition(a)}
    fi
  fi
od
fi

for ∀p ∈ t do
  DelTokens(M(link(t,p), Eval(arc(p,t)))
  if (∃a ∈ l-list) ∧ (Place(a)=link(t,p)) then
    if ObservableTransition(Transition(a)) then
        o-enabled ← o-enabled - {Transition(a)}
    else
        i-enabled ← i-enabled - {Transition(a)}
    fi
  fi
od
fi

if t ∈ KO(σ(up(t))) then
  for ∀k ∈ ch⁻¹(t) do
    AddTokens(M(link(t,k)), Eval(arc(t,k)))
    if (∃a ∈ l-list) ∧ (Place(a)=link(t,k)) then
      if ObservableTransition(Transition(a)) then
        o-enabled ← o-enabled ∪ {Transition(a)}
      else
        i-enabled ← i-enabled ∪ {Transition(a)}
      fi
    fi
  od
else
  for ∀p ∈ t do
    AddTokens(M(link(t,p)), Eval(arc(t,p)))
    if (∃a ∈ l-list) ∧ (Place(a)=link(t,p)) then
      if ObservableTransition(Transition(a)) then
        o-enabled ← o-enabled ∪ {Transition(a)}
      else
        i-enabled ← i-enabled ∪ {Transition(a)}
      fi
    fi
  od
else
\[
i\text{-enabled} \leftarrow i\text{-enabled} \cup \{\text{Transition}(a)\}
\]

\[
\text{if } \text{OutputExists}(t) \text{ then } \\
\text{OutputMessage}(t) \\
\text{fi}
\]

Conclusions

In this paper a class of Petri nets was defined, Interpreted Hierarchical High Level Petri Nets, which are based on the standard of high level Petri nets, and the class of Hierarchical High Level Petri Nets. Interpreted Hierarchical High Level Petri Nets represent an extension of that class by introducing the concept of environment interaction. For Interpreted Hierarchical High Level Petri Nets, both the syntax and dynamic semantics are presented. Also, a method for implementing the semantics of Interpreted Hierarchical High Level Petri Nets was introduced by using a token-player algorithm. The manner of definition of dynamic semantics of Interpreted Hierarchical High Level Petri Nets allows a simple way to represent the dynamic evolution of systems, and a simple way to model and control these systems, comparing by other proposals.

References