Game Strategy in the NPCs using Interpreted Petri Nets

Alejandra Santoyo-Sanchez[1], Luis Isidro Aguirre-Salas[2],
Carlos Alberto De Jesús-Velásquez[3] and M.V. Alvarez-Ureña[4],

[1]Department of Computing, Universidad de Guadalajara – CUCEI, Guadalajara, Jalisco, México
[2]Department of Engineering, Universidad de Guadalajara – CUCSUR, Autlán de Navarro, Jalisco, México
[3]Compatibility Validation, Intel Tecnología de México S.A., Tlaquepaque, Jalisco, México
[4]Department of Industrial Engineering, Universidad de Guadalajara – CUCEI, Guadalajara, Jalisco, México

Phone (33) 33485900 E-mail: alejandra.santoyo@cucei.udg.mx

Abstract. A common limitation of adventure and strategy games is that players quickly learn the positions and behavior of the characters programmed or Non-Players Characters (NPCs). In this work, is explored using the feedback supervisory control for Discrete Event Systems (DES) to control the actions of the NPCs based on predicting where the player is and decide what to do next. That is, to locate and plan their actions for themselves. Thus, the NPC may present a different behavior every time the player plays.

Keywords: Feedback supervisory control, Non-player characters, Petri Net, Artificial Intelligence.

1 Introduction

In the sort of video games of adventures and strategies, most NPCs are limited by a restricted range of reactions programmed by game designers. In particular, when a player meets the same NPC again after a certain period time, the NPC takes same responses or only reactions that have been manually programed a priori. This model has attracted increasing interest in the use of computational intelligence techniques to control the actions of the NPCs instead of relying on simple heuristics or rules-based systems, as Artificial intelligence (AI) [1], finite state machines [2], [3], search algorithms like A * [4], [5], [6], methods based on software engineering [7], [8], dynamic tables of probability [9] among others.

Manual editing of intelligent behaviors for Non Player Characters (NPCs) of games is a cumbersome task that needs experienced designers, our research aims to assist designers in this task. In [10] Interpreted Petri Nets (IPNs) are used to model and to specify all the possible behaviors of NPCs to avoid blocking situations. The model allows capturing behaviors like: causal relationship, synchronizations, asynchronies, exclusions, concurrence or parallelism, among others. In addition, they

have a mathematical support that makes suitable for analysis of qualitative and quantitative properties. While as in this paper, we focus on dynamic retrieval and selection of behaviors taking into account the current state and the underlying goals (reach the user player). The global behavior of the NPC is dynamically built at runtime the feedback supervisory control for Discrete Event Systems (DES). This paper is organized as follows. Section 2 reviews Petri nets (PN) and IPN notation and concepts used in this article. Section 3 presents the contribution of this paper methodology for the NPC design; i.e. the game strategy for NPC's. Next section 4 presents a case of study to illustrate the use of the game of strategy in the design of a video game. Finally, section 5 provides conclusions and future work.

2 Petri nets and Interpreted Petri nets concepts and properties

This section presents a review the main concepts of the PN and IPN formalism used in this paper. An interested reader can consult [11], [12] and [13] for more details.

2.1 Petri nets

Definition 1. A PN system is a pair (N,M_0) where $N=(P,T,I,O)$ is a bipartite digraph that specifies the net structure and $M_0 : P \to Z^+$ is the initial marking. Each element of N is defined as follows $P=\{p_1,p_2,\ldots,p_n\}$ is a finite set of places; $T=\{t_1,t_2,\ldots,t_m\}$ is a finite set of transitions; $I : P\times T \to Z^+$ and $O : P\times T \to Z^+$ are functions representing the weighted arcs going from places to transitions and from transitions to places, respectively. The initial marking of PN M_0 is a function that assigns to each place of N a non-negative number of tokens, depicted as black dots inside the places.

A PN structure N can be represented by its incidence matrix $C=[c_{i,j}]_{n\times m}$, where $c_{i,j}=O(p_i,t_j)-I(p_i,t_j)$. The sets ${}^\bullet t_j = \{ p_i \mid I(p_i,t_j)\neq 0 \}$ and $t_j{}^\bullet = \{ p_i \mid O(p_i,t_j)\neq 0 \}$ are the set of input and output places of a transition t_j respectively, which are denominated predecessors and successors of t_j respectively. Analogously, the sets of input and output transitions of a place p_i are ${}^\bullet p_i = \{ t_j \mid I(p_i,t_j)\neq 0 \}$ and $p_i{}^\bullet = \{ O(p_i,t_j)\neq 0 \}$ respectively. In a PN system, a self-loop is a relation where $c_{i,j}=O(p_i,t_j)-I(p_i,t_j)=0$ and $O(p_i,t_j)\neq 0$, $I(p_i,t_j)\neq 0$. In this work the self-loop structure is represented by the matrix $F=[f_{i,j}]_{r\times m}$, where $f_{i,j}=O(p_i,t_j)\wedge I(p_i,t_j)$, and r is the number of places with self-loops.

A vector often represents the marking at the k-th instant $M_k=[M_k(p_1)\ \ M_k(p_2)\ \ \cdots\ \ M_k(p_n)]^T$. Hereafter, a marking M can be represented by a list $M=[1^{M(p_1)},2^{M(p_2)},\cdots,i^{M(p_i)},\cdots,n^{M(p_n)}]$ where i-th item is omitted if

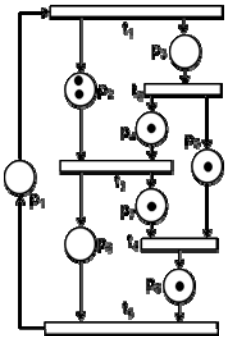
$M(p_i)=0$ and exponents $M(p_i)=1$ are also omitted. For example, a marking $M=[2\ \ 0\ \ 1\ \ 1]^T$ can be represented by the list $M=[1^2,3,4]$.

A transition t_j is enabled at marking M_k if $\forall p_i \in P$, $M_k(p_i) \geq I(p_i,t_j)$; when an enabled transition t_j is fired, then a new marking M_{k+1} is reached. This new marking is computed as $M_{k+1}=M_k+Cv_k$, where v_k is an m-entry firing vector whit $v_k(j)=1$ when t_j is fired once and $v_k(i)=0$ if $i \neq j$ and t_j is not fired, v_k is called Parikh vector; the equation $M_{k+1}=M_k+Cv_k$ is called the PN state equation.

A firing sequence of a PN system (N,M_0) is a transition sequence $\sigma=t_it_j\ldots t_k$ such as $M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} \cdots \xrightarrow{t_k} M_k$.. The firing language of (N,M_0) is the set $L(N,M_0) = \{\sigma | \sigma = t_it_j\ldots t_k \wedge M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} \cdots \xrightarrow{t_k} M_k$, while the Parikh vector $\bar{\sigma}: T \rightarrow (Z^+)^m$ of σ maps every $t \in T$ to the number of occurrences of t in σ. The fact of reaching M_k from M_0 by firing an enabled sequence σ is denoted by $M_0 \xrightarrow{\sigma} M_k$. The set of all reachable markings from M_0 is $R(N,M_0) = \{M_k | M_0 \xrightarrow{\sigma} M_k$ and $\sigma \in L(N,M_0)\}$ and it is called reachability set.

Definition 2: A p-invariant *Y* of a PN is a rational solution to equation $Y^TC=\bar{0}$. Support p-invariant Y_i is set $|Y_i| = \{p_j | Y_i(p_j) \neq 0\}$.

Example 1: Consider the PN of Fig. 1a. The net consists of 8 places $P=\{p_1,p_2,\ldots,p_8\}$ and 5 transitions $T=\{t_1,t_2,\ldots,t_5\}$. The incidence matrix is illustrated in Fig. 1b. The sets of input and output places of t_1 are $\bullet t_1 = \{p_1\}$ and $t_1 \bullet = \{p_2,p_3\}$ respectively. The initial marking is $M_0=[0\ \ 2\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1]^T$ or $M_0=[2^2,4,5,7,8]$. The set of enabled transitions at M_0 is $E(M_0)=\{t_3,t_4\}$. When transition t_3 fires the net reaches the marking $M_1=[5,6,7^2,8]$. A p-invariant of the system A is $Y_0=[2\ \ 1\ \ 0\ \ 0\ \ 0\ \ 2\ \ 0\ \ 0]$ and its support is $|Y_0| = \{p_1, p_2, p_6\}$.■



$$C=\begin{bmatrix} -1 & 0 & 0 & 0 & 1 \\ 2 & 0 & -2 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

(a) **(b)**

Fig. 1. a) Petri Net System A, **b)** Incidence matrix of Petri Net System A

2.2 Interpreted Petri nets

Definition 3. An IPN system is a 6-tuple $Q = (N', \Sigma, \Phi, \lambda, \Psi, \varphi)$ where $N' = (N, M_0)$ is a PN system; $\Sigma = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the input alphabet, where α_i is an input symbol; $\Phi = \{\delta_1, \delta_2, \ldots, \delta_s\}$ is the output alphabet of the net, where δ_i is an output symbol; $\lambda : T \to \Sigma \cup \{\varepsilon\}$ is a function that assigns an input symbol to each transition of the net, with the following constraint: $\forall t_j, t_k \in T$, $j \neq k$, if $\forall p_i \quad I(p_i, t_j) = I(p_i, t_k) \neq 0\}$ and both $\lambda(t_j) \neq \varepsilon$, $\lambda(t_k) \neq \varepsilon$, then $\lambda(t_j) \neq \lambda(t_k)$. In this case, ε represents an internal system event. If $\lambda(t_i) \neq \varepsilon$ then transition t_i is said to be controlled, otherwise uncontrolled. T_c and T_u are the sets of controlled and uncontrolled transitions, respectively. $\Psi : P \to \Phi \cup \{\varepsilon\}$ is a labeling function of places that assigns an output symbol or the null event ε to each place of the net as follows: $\Psi(p_i) = \delta_k$ if p_i represents an output signal, in otherwise $\Psi(p_i) = \varepsilon$. In this case $P_m = \{p_i \mid \Psi(p_i) \neq \varepsilon\}$ is the measurable place set and $q = \mid P_m \mid$ is the number of measured places. While $P_{nm} = P \setminus P_m$ is the set of non-measured places.

Finally, $\varphi : R(N, M_0) \to (Z^+)^q$ is a function that associates an output vector to every reachable marking of the net as follows: $\varphi(M_k) = M_k \mid_{Pm}$, where $M_k \mid_{Pm}$ is the projection of M_k over P_m i.e. if $M_k = [M_k(p_1) \quad M_k(p_2) \quad \cdots \quad M_k(p_n)]^T$ and $P_m = \{p_i, p_j, \ldots, p_h\}$ then $M_k \mid_{Pm} = [M_k(p_i) \quad M_k(p_j) \quad \cdots \quad M_k(p_h)]^T$. Notice that function φ is linear and can be represented as a matrix $\varphi = [\varphi_{ij}]_{q \times n}$, where each row $\varphi(k, \bullet)$ of this matrix is an elementary vector where $\varphi(k, i) = 1$ if place p_i is the *k-th* measured place and otherwise $\varphi(k, i) = 0$ and it is called non- measured.

In this paper, a measured place is depicted as an unfilled circle, while a non-measured place is depicted as a filled circle. Similarly, uncontrollable transitions are depicted by filled bars and controllable transitions are depicted by unfilled bars. Also, (Q, M_0) will be used instead of $Q = (N', \Sigma, \Phi, \lambda, \Psi, \varphi)$ to emphasize the fact that there is an initial marking in an IPN.

Example 2: Consider the IPN shown in Fig. 2. The input and output alphabets are $\Sigma = \{a, b\}$ and $\Phi = \{\delta_1, \delta_2, \delta_3\}$ respectively. Functions Ψ and λ are given by:

i	1	2	3	4	5	6	7	8
$\Psi(p_i)$	δ_1	ε	ε	ε	δ_2	δ_3	ε	ε

(**1**)

k	1	2	3	4	5
$\lambda(t_k)$	a	ε	ε	ε	b

(**2**)

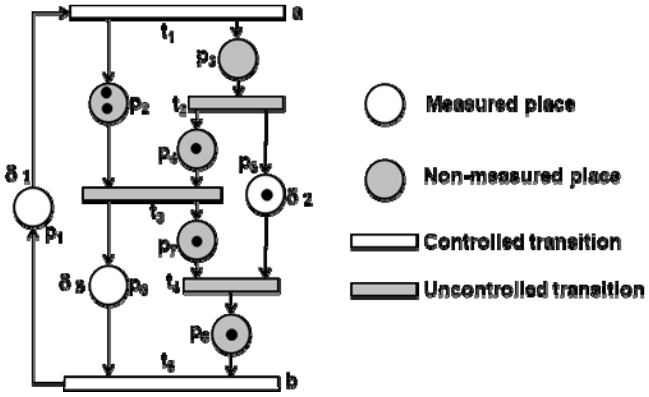


Fig. 2. Interpreted Petri Net System A.

Thus, the controlled transitions are $T_c = \{t_1, t_5\}$ and the uncontrolled ones are $T_u = \{t_2, t_3, t_4\}$. The measured places are $P_m = \{p_1, p_5, p_6\}$ and the non-measured are $P_{nm} = \{p_2, p_3, p_4, p_7, p_8\}$. In this case, the output function is the matrix:

$$\varphi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (3)$$

The initial output is $y_0 = \varphi(M_0) = [0 \quad 1 \quad 0]^T$. ■

Similarly to a PN, in an IPN system, a transition t_j is enabled at making M_k if $\forall p_i \in P$; $M_k(p_i) \geq I(p_i, t_j)$ however t_j has fire conditions. When t_j is enabled and t_j is controllable for that t_j fire, it is necessary that the input signal $\lambda(t_j) \neq \varepsilon$ must be given as input. Otherwise when enabled transition t_j is uncontrollable, then it can be fired. In both cases, when a transition t_j is fired a new marking $M_{k+1} = M_k + C(\bullet, t_j)$ is reached and the output symbol $y_{k+1} = \varphi(M_{k+1})$ is observed. Also, the firing sequence and firing language is computed and denoted as in PN system to enhance the fact that there exists an initial marking in an IPN $Q = (N', \Sigma, \Phi, \lambda, \Psi, \varphi)$ in this paper it is denoted as (Q, M_0).

3 Game Strategy Description

In this paper, Step Sate-Feedback Supervisory Control [14] and Error Petri Net are used for compute game strategy for each NPC. Based on these mechanisms and

according to Supervisory Control Theory from [15], our Step Sate-Feedback Supervisory Control consists of three elements (Fig. 3):

1) A Dynamic Discrete event system (DES), representing the system to be controlled; in this work the system itself is composed by the opponents.
2) The required behavior for the system, called specification; in this case all behavior possible of the player (the user game).
3) An external agent, called supervisor, which restricts the behavior of the system to the behavior of the specification by the manipulation of controllable events.

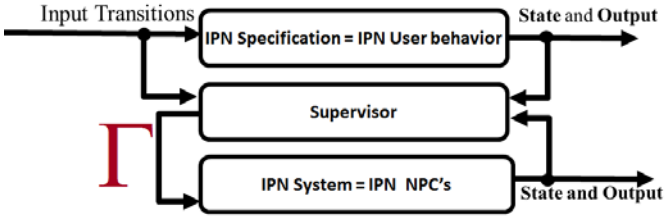


Fig. 3. Game Strategy Scheme.

In this work, the model system (NPC's) and specification (user game) are IPN models, where it's output information indicates which output information the system must reach. Then the enabling rule of events Γ determines which subset of inputs events are enabled to occur into the system model (to generate the behavior of these adversaries), based in the input string generated by the user game, and the state and outputs of both system and specification model.

A novel technique for model NPC's based on IPN and controlled place is presented in [10]. The procedure includes a method to capture restriction operation in term of the linear equations. The IPN model obtained with that methodology is used in this approach to establish the control law based Step Sate-Feedback Supervisory Control to simulate intelligence in the NPC's as follow.

3.1 Modeling the specification: user behavior

In this paper all behavior possible of the player (the user game) and opponents (NPC's) are modeled using the propose methodology in [10]. That proposal based on IPN and controlled place, and the model obtained is composed by all the IPN model of the elements: player and opponents, also the function φ is the identity matrix, due to the game is composed only by measured places. In the strategy games the player and opponents do not always have a complete knowledge of the system state; in these cases, the function φ represents only the places that each player and opponent can measure. In this paper all behavior possible of the player (the user game) is modeled using the follow algorithm.

Algorithm 1: Modeling the specification: user behavior.

Inputs: A model in IPN computed with the algorithm [10].
Outputs: Specification (IPN model of the player).
Procedure:

1. **Identify the player of the game and generate "system specification".** In this step the player is defined as the system specification, this is done by selecting the places and transitions of the net which correspond to the player, each NPC will try to achieve the same token marking as the specification.
2. **Defining the knowledge places for the player.** Each one of these places is selected according to the game specification. It's defined a matrix φ_i. Consider for example the state variable *position* for the player, in this case the relevant values are: $room_1$, $room_2$, …, $room_7$ where there can be non-measurable rooms for the player. Thus the IPN of player is defined as: (Q, M_0^{player})

3.2 Modeling the system: NPC behavior

In a similar form, to generate the behavior of these adversaries, the IPN of the close loop system with the supervisor proposed in [10] is analyzed.

Algorithm 2: Modeling the system.

Inputs: A model in IPN of the close loop system with the supervisor [10].
Outputs: The system to be controlled (IPN model of the opponent).
Procedure:

1. **Composition of the system.** In this step the system is defined, this is done by selecting the places and transitions of the net that do not correspond to the player.
2. **Defining the knowledge places for each NPC.** Each one of these places is selected according to the game specification. It´s defined a matrix φ for each NPC. Consider for example the state variable *position* for an opponent, in this case the relevant values are: $room_1$, $room_2$, …, $room_7$ where there can be non-measurable rooms for this NPC only. In this case the IPN for each NPC is defined as (Q, M_0^{NPCi}).

The next algorithm uses the individual marking of each part of the system and specification to define the strategy game.

Algorithm 3: Measuring the system and specification.

Inputs: Specification (IPN model of the player) and a system to be controlled (IPN model of the opponent)
Outputs: A vector containing the observable information and a transition-firing vector.
Procedure:

1. **Selecting the NPC.** It's used the φ_i of the NPC whose information is required and the system marking that are necessary for next steps.
2. **Obtaining the output function of the system.** Define the output for φ_i for the system and the output for the specification. By applying the equation: $\varphi_i(M_j^{NPCi})$.
3. **Obtain the output function of the specification**. Multiply the φ_i and the player marking M_k^{player} for the system and the output for the specification. By applying the equation: $\varphi_i(M_k^{player})$
4. **Obtaining the vector of information.** Subtract the system output from the specification output. By applying the equation: $\varphi_i(M_k^{player}) - \varphi_i(M_j^{NPCi}) = M_{sig}$.
5. **Calculating the firing vector.** Select the route for the NPC to follow by calculating the firing vector that allows the NPC to reach the desired destination. Let $M_k \rightarrow M_{(k+1)}$ executing the transition t_i be the marking of the system and $M_l \rightarrow M_{(l+1)}$ executing the firing vector w_i be the specification marking where $w_i = \{t_a, t_b, t_c, \ldots, t_s\}$. Then $M_k \rightarrow M_{k'} by\, t_a, M_{k'} \rightarrow M_{k''} by\, t_b, \ldots, \rightarrow M_{(k+1)} by\, t_s$. Calculate the firing vector w_i for the system solving the next equation system: $CV = M_{sig}$, where C is the incidence matrix of the NPC_i and $V = [v_1, v_2, \ldots, v_n]$ where V_i is the amount of firing times necessary of t_i to reach the desired state if the player remains static, in other case is necessary to apply this algorithm again until the player is captured by an opponent or the player meets its goal objectives.

If M_{sig} is a vector 0, means that the player and the NPC_i both are in the same room or both are in not observable rooms, if $\exists \alpha_j \in M_{sig} \mid a_j > 0$ then the player is visible for the NPC_i and in any other case the player is not visible for the NPC_i.

The application from all algorithms shown in this section is illustrated in the next case study.

4 Case of study strategy game

Fig. 4 represents the strategy game considered in this case of study proposed in [10]. It is a dungeon environment, which is composed by seven rooms (h_1, h_2, …, h_7). Inside a dungeon there are the following elements: a player (j_1), two opponents (o_A and o_B), and a reward. The behavior of each element inside dungeon is the following. Player (j_1) has the goal of finding the reward and exit from the dungeon. Also, player should achieve its goal just by avoid being in the same room as any of opponents. This strategy game is indicated by the user game. Two opponents (o_A and o_B) have the goal of finding the player. In this case its strategy game consists in moving and watching between dungeon's rooms, which is computed by an algorithm.

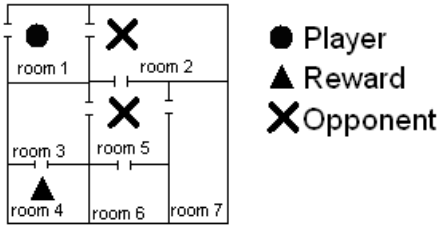


Fig. 4. Component of the game [10].

Fig. 5 show IPN model game computed using the algorithm proposed in [10]. Where the input and output alphabets are: $\Sigma = \{a_{1-2}, a_{2-1}, a_{2-5}, a_{3-4}, a_{3-5}, a_{4-3}, a_{5-2}, a_{5-3}, a_{5-6}, a_{5-7}, a_{6-5}, a_{7-5}\}$ and $\Phi = \{\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \delta_6, \delta_7\}$ respectively. Functions Ψ and λ are given by:

i	1	2	3	4	5	6	7
$\Psi(p_i)$	δ_1	δ_2	δ_3	δ_4	δ_5	δ_6	δ_7

(**4**)

k	*1, 13, 25*	*2, 14, 26*	*3, 15, 27*	*4, 16, 28*	*5, 17, 29*	*6, 18, 30*	*7, 19, 31*	*8, 20, 32*	*9, 21, 33*	*10, 22, 34*	*11, 23, 35*	*12, 24, 36*
$\lambda(t_k)$	a_{1-2}	a_{2-1}	a_{2-5}	a_{5-}	a_{3-5}	a_{5-3}	a_{4-3}	a_{3-4}	a_{5-6}	a_{6-5}	a_{7-5}	a_{5-7}

(**5**)

All transitions are controlled and all places are measured. Thus the output function φ is an identity matrix. In the final step it is necessary to synchronize the state variables, in this case it step is not necessary.

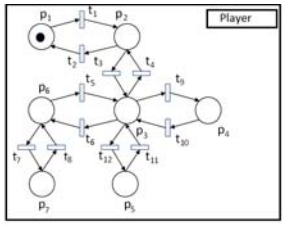


Fig. 5.a) Player IPN [10]

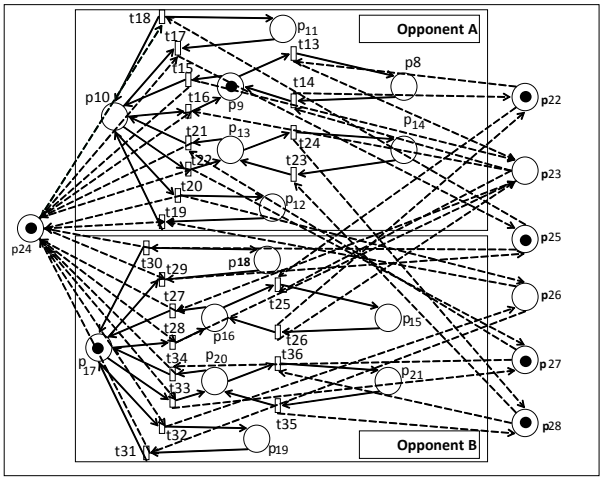


Fig. 5.b) IPN Control places among opponet A and opponent B [10].

Fig. 5. IPN model computed with [10].

While as, the Fig. 5a and Fig. 5b show IPN model of player and the two opponents, which are obtained using the algorithms 1 y 2 respectively. In this case, the output function for the player and opponents are the matrix:

$$\varphi_{NPCA} = \begin{bmatrix} 1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix} \quad (6)$$

$$\varphi_{NPCB} = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \end{bmatrix} \quad (7)$$

The marking are:

$$M_0^{player} = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$$

$$M_0^{NPCA} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T \quad (8)$$

$$M_0^{NPCB} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$$

Now the algorithm 3 is used, in this case for the opponent A $\varphi_{NPCA}(M_0^{player}) = [1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]^T$ and $\varphi_{NPCA}(M_0^{NPCA}) = [0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0]^T$; then

$\varphi_{NPCA}(M_0^{player}) - \varphi_{NPCA}(M_0^{NPCA}) = M_{sig} = [1 \quad -1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$. The firing vector computed is: $\sigma = t_{14}$. In similar form is computed by the opponent B. Then the enabling rule of events Γ determines $\lambda(t_{14}) = a_{2-1}$, thus is defined the behavior for each NPC´s.

5 Conclusions

This paper presents a novel technique for dynamic retrieval and selection of behaviors taking into account the current state and the underlying goals (reach the user player). The global behavior of the NPC is dynamically built at runtime the feedback supervisory control for Discrete Event Systems (DES). In proposal, the use of Interpreted Petri Nets (IPN) which is an extension of Petri Nets (PN) for design the Intelligence in the NPCs is explored. The proposal is based creating a relationship between input signals and output signals for IPN models of each NPCs to define its behavior.

As future work this methodology will analyze to establish the control law based in Feedback Supervisory Control and Partial Observation to simulate intelligence in the NPC inside a collaboration schema.

6 References

[1] M. van Lent, "Game Smarts", *Computer*, Vol. 40, pp 99-101, April 2007.

[2] W. M. Wonhan, and P.J. Ramadge, "On the Supremal Controllable Sublanguage of a Given Language", *SIAM Journal on Control and Optimization*, Vol. 25, No.3, pp. 635-659, 1987.

[3] J.E. Hopcroft and J.D. Ullman. "Introduction to automata theory, languages, and computation", Ed. Addison-Wesley, 1979 [4] P.E. Hart, L.J. Nilsson and B. Raphael. "A formal basis for the heuristic determination of minimum cost paths", *IEEE Transactions on Systems Science and Cybernetic*s, Vol. 4, No. 2, pp. 100-107, 1968.

[5] P.E. Hart, L.J. Nilsson, and B. Raphael. Correction to "A formal basis for the heuristic determination of minimum cost paths", *SIGART Newsletter*, No. 37, pp. 28-29, 1932.

[6] V. Kumar, and D. S. Nau. "A general branch-and-bound formulation for AND/OR graph and game tree search", *Search in Artificial Intelligence*, pp. 91-130, Ed. Springer-Verlag, 1988.

[7] M. McNaughton, M. Cutumisu, D. Szafron, J. Schaeffer, J.Redford, and D. Parker. "ScriptEase: Generating Scripting Code for Computer Role-Playing Games", *in Proc.19th IEEE International Conference on Automated Software Engineering ASE´04*, 2004, Linz, Austria, pp. 386 – 387.

[8] L. Wei-Po, L. Li-Jen, and C. Jeng-An; "A Component-Based Framework to Rapidly Prototype Online Chess Games for Home Entertainment"; *in Proc. IEEE International Conference on Systems, Man, and Cybernetics* SMC'06, 2006, Taipei, Taiwan, Vol. 5, pp 4011-4016, 2006.

[9] W. Yingxu. "Mathematical models and properties of games". *in Proc. Of the Fourth IIEEE Conference Cognitive Informatics ICCI´05*, 2005, Irvine, CA, USA, pp. 294-300.

[10] Santoyo-Sanchez, A.; Pérez-Martinez, M.A.; De Jesús-Velásquez, C.; Aguirre-Salas, L.I.; Alvarez-Ureña, M.A.; "Modeling methodology for NPC's using interpreted Petri Nets and feedback control", *in Proc. 7th International Conference on IEEE Electrical Engineering Computing Science and Automatic Control (CCE)*, 2010, Tuxtla Gutiérrez, Chiapas, México, pp.369 – 374, 2010.

[11] T. Murata, "Petri nets: Properties, analysis, and application", *in Proc. Of the IEEE*, Vol. 77, No.4, pp. 541-580, 1989.

[12] J. Desel, J. Esparza and C. J. van Rijsbergen, *Free choice Petri nets*, pp. 1-5, Ed. Cambridge University Press, 2005.

[13] M. E. Meda, A. Ramirez and A. Malo, "Identification in discrete event systems", *in Proc.1998 IEEE International Conference Systems, Man and Cybernetics*, *SMC 1998*,*San Diego CA*., pp. 740-745.

[14] A. Santoyo-Sanchez, A. Ramírez-Treviño, C. De Jesús Velásquez, L.I. Aguirre-Salas, "Step State-feedback Supervisory Control of Discrete Event Systems using Interpreted Petri Nets", *in Proc.13th IEEE International Conference on Emerging Technologies and Factory Automation*, *ETFA 2008*, Hamburg Germany, pp. 926 – 933.

[15] P.J. Ramadge, and W. M. Wonhan, "Supervisory Control of a Class of Discrete Event Processes", *SIAM Journal on Control and Optimization*, Vol. 25, No.1, pp. 206-230, 1987.

[16] De-Jesus, C.A.; Ramirez-Trevino, A.; "Controller and observer synthesis in discrete event systems using stability concepts", *in Proc. IEEE International Conference on Systems, Man, and Cybernetics*, 2001, Tucson, AZ, USA, Vol. 1, pp. 664 – 668.