Modeling methodology for NPC's using interpreted Petri Nets and feedback control*

A. Santoyo-Sanchez[1], M. A. Pérez-Martínez[1],
C. De Jesús-Velásquez[2], L.I. Aguirre-Salas[3], M.A. Alvarez-Ureña[4]
[1]Department of Computing, Universidad de Guadalajara – CUCEI, Guadalajara, Jalisco, México
[2]Compatibility Validation, Intel Tecnología de México S.A., Tlaquepaque, Jalisco, México
[3]Department of Engineering, Universidad de Guadalajara – CUCSUR, Autlán de Navarro, Jalisco, México
[4]Department of Industrial Engineering, Universidad de Guadalajara – CUCEI, Guadalajara, Jalisco, México

Phone (33) 36398038 E-mail: alejandra.santoyo@cucei.udg.mx

Abstract —**In the context of video games design, Artificial Intelligence (AI) a broad of techniques have been used to generate the behavior of Non-Players Characters (NPC's). In this document, is explored the use of Supervisory control in Discrete Event Systems for design the behavior of NPC's, where the interaction between them it is not blocking. The modeling tool used is Interpreted Petri Nets (IPN) which are an extension of Petri Nets (PN) allow to relate input signals and output signals for PN models.**

Keywords — **Feedback control, Non-player characters, Petri Nets, Supervisor using p-invariant**.

I. INTRODUCTION

In the sorts of video games of adventures and strategies, the hero and/or main character is controlled generally by the user of the game. Whereas the rest of the characters are programmed, some of them have the roll to play like opponents; when these simulate to be an intelligent character NPC's are used.

For the developers of NPC's Artificial Intelligence (AI) has come to mean the broad range of used techniques to generate the behavior of these adversaries, the units of equipment, battlefield, or any other thing that simulate intelligence [1]. Some of these techniques, such as 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, have been used in the last years.

According to [1] in the most basic level, the machines of finite states are used to model and to specify the possible behaviors of a character. Thus, a series of conditions define the events that allow the NPC to change its state, whereas a code fragment allows that the NPC acquires certain conduct associated to the state.

Later, the game strategies that will execute the NPC are designed. In this case the planning routes are used generally to specify the set of behaviors that will have the NPC, where the objective is to arrive from the point A to point B. Being the A* search algorithms the most used.

Although finite state machine are decomposed in a very efficient way, all the behavior of the NPC, its application is explicitly must consider all the NPC´s states, consequently resulting in huge models. Also the complexity is increased when the simultaneous behavior of all NPC´s is considered in the video game environment. Making very difficult to validate the behavior between players in order to avoid blocking situations and search situations using A* algorithm.

In order to avoid these disadvantages, in this paper we use IPN to model the behavior of NPC's, which allow to capture behaviors like: causal relationship, synchronizations, asynchronies, exclusions, concurrence or parallelism, among others, in a compact way avoiding to enumerate the large amount of states that arise in the automaton. In addition, they have a mathematical support that makes suitable for analysis of qualitative and quantitative properties.

In other to cope the issues mentioned before, this paper is based on the use of state feedback control for Discrete Event Systems (DES) [10], [11] where the modeling tool used is IPN during the design of the NPC's through two main stages. 1) To design the individual behavior of the NPC; and 2) To design the behavior prohibited between NPC's, to avoid blocking through the implementation of a feedback control based on control places.

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 contributions of this paper, i.e. the proposal methodology for the NPC design, whereas the section 4 presents a case of study (game of strategy) to illustrate the use of the methodology in the design of a video game. Finally, section 5 provides conclusions and future work.

II. 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 [12], [13] and [14] for more details.

A. 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 $M_0 : P \rightarrow Z^+$ 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; $I : P \times T \rightarrow Z^+$ and $O : P \times T \rightarrow Z^+$ are functions representing the weighted arcs going form places to

*Supported by project PROMEP 103.5/09/1158

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.

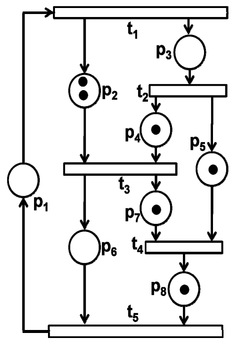
The marking at the k-th instant is often represented by a vector $M_k = [M_k(p_1) \;\; M_k(p_2) \;\; \cdots \;\; M_k(p_n)]^T$. Hereafter, a marking M can be represented by a list $M = [1^{M(p_1)}, 2^{M(p_2)}, \cdots, i^{M(p_i)}, \cdots, n^{M(p_n)}]$ where i-th item is omitted if $M(p_i) = 0$ and exponents $M(p_i) = 1$ 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 $\sigma = t_i t_j \ldots t_k$ 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 $L(N, M_0) = \{\sigma \mid \sigma = t_i t_j \ldots t_k \wedge M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} \cdots \xrightarrow{t_k} M_k$, while the Parikh vector $\vec{\sigma} : T \to (Z^+)^m$ 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 $R(N, M_0) = \{\, M_k \mid M_0 \xrightarrow{\sigma} M_k$ 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 $Y^T C = \vec{0}$. Support p-invariant Y_i is set $|Y_i| = \{p_j \mid Y_i(p_j) \neq 0\}$.

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



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

(a) (b)

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

¶

$T = \{t_1, t_2, \ldots, t_5\}$. 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 $|Y_0| = \{p_1, p_2, p_6\}$.■

B. 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 $\forall p_i$ $I(p_i,t_j) = I(p_i,t_k) \neq 0\}$ 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 $q = |P_m|$ 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: $\varphi(M_k) = M_k|_{P_m}$, where $M_k|_{P_m}$ is the projection of M_k over P_m i.e. if $M_k = [M_k(p_1) \;\; M_k(p_2) \;\; \cdots \;\; M_k(p_n)]^T$

and $P_m = \{p_i, p_j, \ldots, p_h\}$ then $M_k|_{P_m} = [M_k(p_i)\ \ M_k(p_j)\ \ \cdots\ \ M_k(p_h)]^T$. 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)

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 $y_0=\varphi(M_0)=[0\ \ 1\ \ 0]^T$. ■

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).

Next some concepts related to controllability and control useful in the development of the modeling methodology and a case of study appear.

III. Modeling Description

In this paper, feedback control proposed by [10] is used

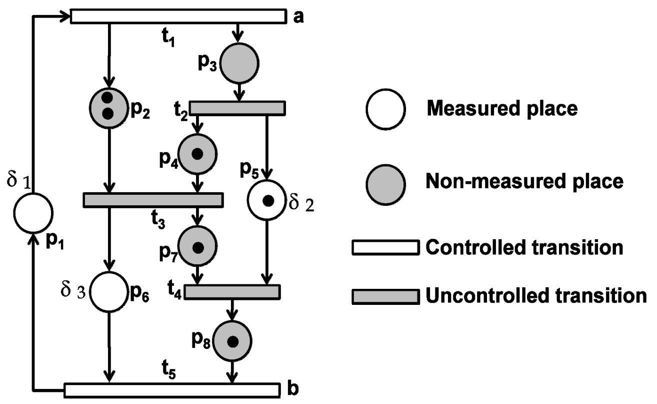


Fig. 2) Interpreted Petri Net System A.

¶

for compute a controller blocking states between NPC's. According to Supervisory control theory [15], feedback supervisory control [16] consists of three elements: 1) A Dynamic Discrete event system (DES), representing the system to be controlled; 2) the required behavior for the system, called specification; and 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.

In the context of NPC's the model description has the following elements.

A. Modeling the game

In this paper the NPC's are modeled with IPN's adapting the propose methodology in [17], in this proposal the permissive relationship between state variables are not considered, since these represent behavior restrictions and in this work they comprise the model of the specification to avoid the blockings.

Algorithm 1: Modeling the game.

Inputs: Any.
Outputs: IPN model of the game.
Procedure:

1. **Identify each player of the game and generate "sate variables".** In this step the state variable are defined for each player. The state variable are related with the behavior of each player for example, a player it is usual consider the variables: position, movement, bullets, etc. In general a state variable is denoted as sv_j^i.
2. **Code each state variable into relevant values.** Each one of these code values will be called a "state" of the state variable. In general form to player i and the state variable j with p "states" is coded as: $values_{sv_j^i}=\{val_1^{ij},val_2^{ij},\ldots,val_p^{ij}\}$. Consider for example the state variable *position* for a player, in this case the relevant values are: $room_1$, $room_2$, …, $room_7$.
3. **Code each relevant value into a Petri net place.** If the

ISBN: 978-1-4244-7314-4

value of state variable $values_{sv_j^i}$ is coded into p states, then there must exist p Petri nets places, each one to represent each different state of the state variable i.e. the set $p_{sv_j^i} = \{p_1^{ij}, p_2^{ij}, \ldots, p_p^{ij}\}$ is defined, where $|values_{sv_j^i}| = |p_{sv_j^i}|$. For example, if for state variable *position* the relevant values are: room$_1$, room$_2$, …, room$_7$ then seven Petri net place will be need to represent this codification, the set of places is $\{p_1, p_2, p_3, \ldots, p_7\}$.

4. **Define the change between the states of each state variable.** If a state variable sv_j^i changes from state val_x^{ij} to state val_w^{ij}, the Petri net transition t_{x-w}^{ij} must be added to Petri net model. An arc goring from the place p_x^{ij} which represents the state val_x^{ij} to transition t_{x-w}^{ij} must be added. Also an arc going form the added transition t_{x-w}^{ij} to the place p_w^{ij} that represents the state val_w^{ij} must be added. For example if the player can travel from de room$_1$ to room$_2$ (represented with the places p_1, p_2 respectively). Then a transition must be added, in this case suppose that it is the transition t_1. Next the arcs going from p$_1$ to t_1 and from t1 to p2 must be added. Thus, the change between states is captured into the model.
5. **Define the initial marking.** The token for each state variable must be added, in this case using the follow criteria: when the initial value of the state variable is the place p_x^{ij} then $M_0(p_x^{ij}) = 1$, in other case $M_0(p_w^{ij}) = 0$, where $x \neq w$. For example, in this case suppose that player starting in the room$_1$ then a token must be added in the place p_1, thus $M_0(p_7) = 1$ and the others place representing the position of the player the marking is zero.
6. **Define the interpretation.** In this step the functions λ and φ are defined. Consider for example, that for the transition t_1 the input signal O$_1$, i.e. open door and travel to room$_2$. While the function φ is de identity matriz, due to the game is composed only by measured places.
7. **Synchronize the state variables.** In this step the transitions representing common operations in the different state variable models are merged into a single one.

B. *Modeling the specification.*

In this paper proposes a methodology to capture the permissive and exclusion requirements between NPC's as a set of linear inequalities. In the case of IPN, a linear restriction represents a limit in the number of marks that the places and that are modeled using the following algorithm.

Algorithm 2: Modeling methodology of the specification

Inputs: The IPN model of the system.
Outputs: The matrix of restrictions L, and the vector of limit B.
Procedure:

1. **Composition permissive or relational.** The purpose of this step is to obtain the causal relationship between the state variables of the system. Basically consist in adding arcs from places to transitions and back for, using the different state variables.
2. **To define exclusions.** In this step a set of inequities l are established expressed as $\mu_i + \mu_j \le b$ where μ_i and μ_j are the tokens for places p_i and p_j between different state variables corresponding to different players. b is an integer, that works as the maximum amount of tokens that must exist simultaneously in p_i and p_j. For instance, supposed that player$_1$and player$_2$ can not be in the room$_1$ simultaneously, i.e., just one player can be inside the room, then $b=1$. Where the place p$_1$ represents that player$_1$is inside room$_1$, while the place p$_8$ represents that player p$_2$ is inside room$_1$. Then a restriction $l = l \cup \{\mu_1 + \mu_8 \le 1\}$ is added to the specification, indicating that p$_1$ and p$_8$ cannot have tokens simultaneously.
3. **Transforming the restrictions.** With the purpose of using the mathematical structure of PN, in this step the set of restrictions l is transformed in the matrix $L_{r \times n}$ and the vector B_r where r is the number of restrictions and n is the number of places of the IPN model for the video game. The step $\forall l_a \in l$ do $L(l_a, k) = 1$ if μ_k appears in the restriction l_a, in other case is 0, with $k = 1,2,\ldots,n$ and $B(a) = b_a$.

In general, there are more types of restrictions, but never the less in this work we will only consider the definition of exclusions like part of the specification. Next an algorithm proposed in [10] is synthesized for the calculation of the supervisor.

C. *Design a supervisor.*

In the supervisor design phase, a controlled place control scheme is used [10]. This technique uses the linear inequalities to avoid that system reach forbidden conditions.

Algorithm 3: Compute the supervisor.

Inputs: IPN of the model of the game, the matrix of restrictions L, and the vector of limit B.
Outputs: A model in IPN of the close loop system with the supervisor.
Procedure:

1. **Transforming restrictions in inequities.** Define $\bar{L} = [L \quad I]$ where $I_{a \times a}$ is an identity matrix and a is the

number of row in L. In this way the restrictions l_a should have the form $\mu_i + \mu_j + \mu_a \leq b$.

2. **Defining the IPN matrix under control.** Define $\overline{C} = [C \quad -\overline{L}C]^T$ where C is the incidence matrix for the IPN of the game and $\overline{C}$ represents the incidence matrix for the system under control.
3. **Defining the initial marking for system under control.** Establish the initial marking as $\overline{M}_0 = [M_0 \quad B - LM_0]^T$ where M_0 is the initial marking of the system.

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

IV. Case of Study: Strategy Game

Fig. 3 represents the strategy game considered in this case of study. 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), four opponents (o_1, o_2, o_3 and o_4), 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 avoiding to be detected by opponents. This strategy game is indicated by the user game. Four opponents (o_1, o_2, o_3 y o_4) 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. This paper only contains the model for the all behavior possible of the player and opponents, in a future work the strategy game algorithm will present.

Using the algorithm 1 the fist step is to identify each player of the game and generate "sate variables". To illustrate this step consider the player game and the state variable "position". Next, the second step is to code each state variable into relevant values, in this case "position" is coded in the values h_1, h_2, …, h_7, i.e. the player can be inside each dungeon's room. The third step each relevant value is coded into a Petri net place, it is showed in Fig. 4a in which contains the places p_1, p_2, …, p_7. In the fourth step the possible travel between rooms is defined, for example the player can travel from the room$_1$ to room$_2$ then a transition must be added, in this case the transition is t_1. Now the arcs from p_1 to t_1 and from t_1 to p_2 are added, in a similar form the rest of transitions and arcs will added.

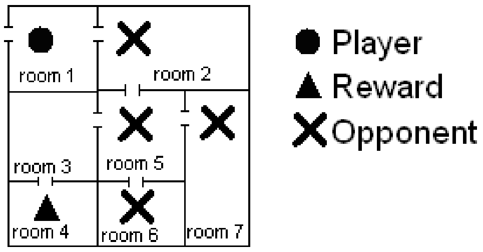


Fig. 3) Component of the game.

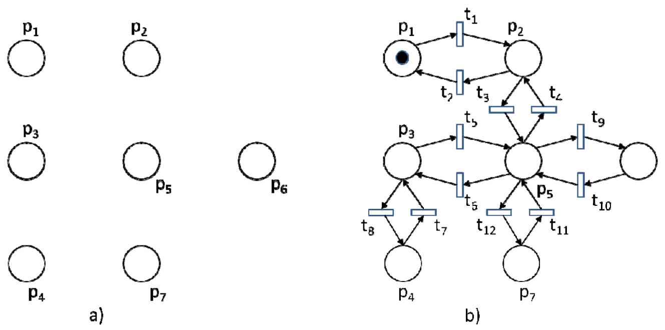


Fig. 4.a) Places for the values of the state variable "position". b) Player PN.

The Petri net model of the player is showed in the figure 4b note that this model contains the initial marking, the player is inside the room$_1$.

Next an extension to PN is defined in this case the extension consists in assigning input and output signals to PN models. IPN is defined as follows. 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	2	3	4	5
$\lambda(t_k)$	a_{1-2}	a_{2-1}	a_{2-5}	a_{5-2}	a_{3-5}

(5)

k	6	7	8	9	10
$\lambda(t_k)$	a_{5-3}	a_{4-3}	a_{3-4}	a_{5-6}	a_{6-5}

(6)

k	11	12
$\lambda(t_k)$	a_{7-5}	a_{5-7}

(7)

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.

The algorithm 1 is used to define the behavior of each opponent. Fig. 5a and Fig. 5b show IPN model by two opponents denominated A and B respectively. Finally the IPN model of the strategy game is composed by all the IPN model of the elements: player and opponents.

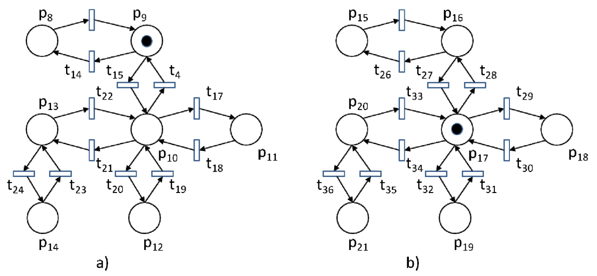


Fig. 5.a) Opponet A IPN model. b) Opponent B IPN model.

To capture the permissive and exclusion requirements between NPC's as a set of linear inequalities is used the algorithm 2. In this case the restriction is the following: "*In each room only must be an opponent*". By the algorithm 2 ¶the linear restriction is defined as:

$$l = \begin{cases} \mu_8 + \mu_{15} \le 2, \\ \mu_9 + \mu_{16} \le 2, \\ \mu_{10} + \mu_{17} \le 2, \\ \mu_{11} + \mu_{18} \le 2, \\ \mu_{12} + \mu_{19} \le 2, \\ \mu_{13} + \mu_{20} \le 2, \\ \mu_{14} + \mu_{21} \le 2 \end{cases} ; \quad B = [2\,2\,2\,2\,2\,2\,2]^T ; \tag{8}$$

$$L = \begin{bmatrix} 000000010000001000000 \\ 000000001000000100000 \\ 000000000100000010000 \\ 000000000010000001000 \\ 000000000001000000100 \\ 000000000000100000010 \\ 000000000000010000001 \end{bmatrix} \tag{9}$$

Finally using the algorithm 3 the closed-loop produced guarantees the restriction rule, which is illustrated in the Fig. 6 note that the control places are out the box for each opponent.

V. CONCLUSIONS

This paper presents a novel technique for model NPC's based on IPN and controlled place. The procedure includes a method to capture restriction operation in term of the linear equations. As future work the IPN model obtained with this methodology will analyze to establish the control law based in feedback supervisory control to simulate intelligence in the NPC's.

VI. ACKNOWLEDGMENT

This work was supported by SEP in the scope of the project PROMEP 103.5/09/1158.

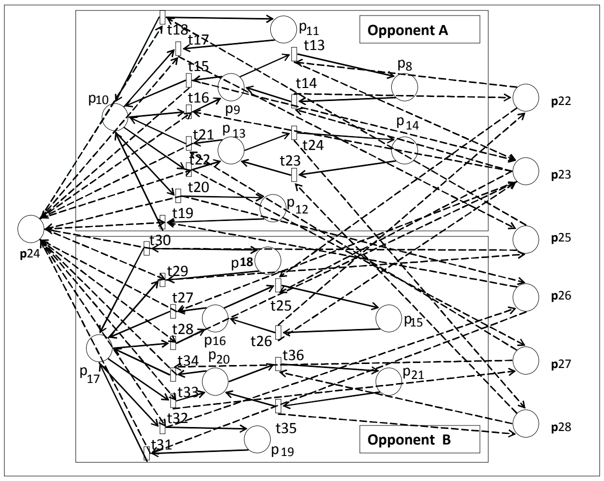


Fig. 6. IPN Control places among opponet A and opponent B.

VII. 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 Cybernetics*, 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] K. Yamalidou, J. Moody, M. Lemmon, and P. Antsaklis, "Feedback Control of Petri Nets Based on Place Invariants", *Automatica*, Vol. 32, No.1, pp. 15-28, 1996.

[11] R. Campos-Rodríguez, M. Alcaraz-Mejia, and J. Mireles-Garcia. "Supervisory Control of Discrete Event Systems using observers", in Proc. Of the 15th IEEE Mediterranean Conference on Control & Automation MED´07, 2007, Athens, Greece , pp. 1-7.

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

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

[14] 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.

[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] 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.

[17] K. L. Muñoz Pizano, "Restablecimiento de Sistemas Eléctricos con asistencia de un Sistema Experto", Master disertation, *Ed. Universidad de Guadalajara Maestrìa en Sistemas de Información*, Julio 2007.