Potential Arc: A Modeling Mechanism For Conflict Control In Multi-Agent Systems

Jiexin Lian and Sol M. Shatz
Department of Computer Science
University of Illinois at Chicago
851S. Morgan Street, Chicago, IL ,60612
{jlian1, shatz}@cs.uic.edu

ABSTRACT

Multi-Agent Systems (MAS) have become one of the most important topics in distributed systems research. Although there have been some system modeling techniques to support MAS design and automatic analysis, most state-of-the-art techniques haven't distinguished potential conflicts from real conflicts during the design stage. In this paper, a new concept, called potential arc, is added into colored Petri nets to support the modeling of MAS. This modeling mechanism eliminates the unnecessary resolution of potential conflicts since such conflicts might not result in real conflicts at runtime. The proposed technique also separates the global conflict scheme from the local agent behavior, so as to provide greater flexibility for conflict resolution and system design.

KEYWORDS

Multi-Agent-System, Colored Petri Net, Conflict Control, Potential Arc.

1. INTRODUCTION

In a Multi-Agent System (MAS), multiple agents may work together to perform tasks or solve problems. Conflicts may occur in the runtime when multiple agents compete for external resource. Generally speaking, most state-of-the-art research follows one of the following two approaches to handle run-time conflicts:

1) In the design stage of MAS, every agent is assigned a local plan by a certain agent designer, and the local plan models the behavior of the agent. After all the agents have been assigned local plans, a certain algorithm would be chosen to integrate local plans to generate a global plan, which models the behavior of the MAS. Through analyzing local plans, the integration algorithm may request some local plans be updated in order to generate conflict-free global plans – a global plan being conflict-free means that no run-time conflict is possible to occur in the MAS described and developed based on this global plan. The drawback of this approach is that the local plans are dependent on each other, so that revising one local plan may force other local plans to be revised, in order to guarantee that the global plan remains conflict-free. As a result, an individual agent cannot be designed independently. Papers [1,2,9] describe some research using this approach.

2) Individual agents are designed independently, and there is no guarantee that run-time conflicts can be avoided. Whenever a run-time conflict occurs, involved agents may need to halt their current activities, and negotiate using a predefined protocol to resolve the conflict. Waiting for negotiation results can be expected to make the agents idle (blocked) during runtime, and some conflicts might not be resolved at all. Papers [6,12] are examples for research that follows this approach.

Typically, techniques based on the first approach have the characteristic that agents cannot be designed independently because such techniques only model the static properties of the MAS, and these models are not designed to explicitly simulate the execution of a MAS. We call such a model a *static model*. Henceforth, algorithms applied to static models identify conflicts that are possible at run-time, without considering any run-time events that would trigger such conflicts, and eliminate these conflicts by imposing various constraints during the modeling stage. However, a run-time conflict only occurs when triggered by a run-time event corresponding to some competition for an external resource. The actual occurrence of such an event is unknown in the MAS design stage. For example, consider two agents *A1* and *A2* that model two different airplanes, and each airplane requires a runway for landing. A conflict occurs only when both airplanes request to use the same runway in the same run-time moment. Using a static model, which does not support run-time events, we would avoid the conflict by creating agent models that assign different runways for different airplanes. In this case, neither *A1* nor *A2* can be designed independently without the risk of violating the constraint that the design be conflict free. The key point is that we need to distinguish a *potential conflict* from a real conflict by considering the appropriate resource-competition event that triggers the conflict.

It is a benefit to distinguish the representation of potential conflicts and real conflicts in the modeling stage, so that we can tolerate potential conflicts in the system design stage, while still guarantee the avoidance of real run-time conflicts. Our proposal is to incorporate some new features in traditional colored Petri net (CPN) models to explicitly support the description of potential conflicts. Here we select CPN as the modeling tool because CPN-

based MAS models can support the simulation of dynamic execution of a MAS and the token driven mechanism in CPN can be used to describe the event triggering mechanism of conflicts. We assume that the readers are generally familiar with the basics of colored Petri net theory [3].

The new feature we introduce in this paper is called a *potential arc*, which serves to extend the traditional CPN model with an explicit support for MAS modeling. By adopting potential arcs, the design procedure of a MAS can be divided into two basic steps: 1) design local plans for individual agents using CPN with potential arcs, and 2) concatenate local plans to form the MAS model. For each agent in a MAS, there can be many different agent actions. For example, "landing" and "taxiing" are two different actions of an airplane agent. For each action, the agent designer may design several alternative paths based on the access or release of resources. Each path may contain potential arcs to model such access/release of resources. The agent designer doesn't need to get the paths approved in the design stage, because these paths only contain potential conflicts, and those potential conflicts will not become real conflicts (at runtime) unless appropriate resource-competition events occur. For an implementation based on our design, real conflicts will be detected automatically at run time and an agent will become aware of these detected conflicts. In contrast to some techniques mentioned earlier, our approach will result in an agent that can choose any path without real conflicts to take actions and perform tasks, instead of the agent needing to wait for negotiation results. In this way, we still guarantee the conflict-free multi-agent environment.

2. MULTI-AGENT-SYSTEM BASED ON COLORED PETRI NETS

Before discussing the potential arc concept, let's generally describe how a CPN model can be used in MAS modeling [4,8,11].

To maintain conciseness, an incoming arc refers to the arc from a Petri net place to a transition, while an out-going arc refers to an arc from a transition to a place. With regard to modeling an agent's behavior, we want to emphasize the following characteristics of an agent: An agent is an autonomous entity with several properties and actions. *An agent's mental state is a possible assignment of the agent's properties, and we abstract an agent's behavior into a set of transitions between different mental states. An agent action implements some tasks and updates the agent's mental state.*

For example, a very simple agent AP describes a person's behavior. The person has two properties, *isHome* and *isOffice* – to indicate if the person is currently at home or at his or her office. The person also has one action: *go-to-work*. The action *go-to-work* moves the person from the home to the office. Both *isHome* and *isOffice* have two possible assignments: "*YES*" or "*NO*." At a specific run-time moment, *isHome* equals "*YES*" means the person is at home, while *isHome* equals "*NO*" means the person isn't home; similarly for the property *isOffice*. The tuple *(isHome, isOffice)* represents the agent's mental state. The state *(isHome, isOffice) = ("YES", "NO")* means the person is home, and the action *go-to-work* updates the agent's mental state to *("NO", "YES")*.

We call the model describing an agent's behavior the agent's *local plan*, and the agent is called the owner of its local plan. Using a CPN model, we can model an agent's local plan. In a local plan, one Petri net place is used to model one property of the owner agent, and one transition is used to represent one agent action. For each agent property, the possible assignments to that property contribute to the color set for the corresponding CPN place. Also, a default assignment means there is no colored token in the place. A Petri net arc is used to connect places and transitions, and each arc can carry an inscription. The inscription carried by an incoming (place-to-transition) arc specifies the colored tokens removed from the arc's source place when the arc's destination transition fires. The inscription carried by an out-going (transition-to-place) arc specifies the colored tokens deposited into the arc's destination place when the arc's source transition fires. The local plan can describe its owner agent's run-time behavior. The concept of "running an agent" means the agent performs actions, and the actions update the agent's mental state. From the perspective of our model, an action corresponds to firing a transition in the agent's local plan. Similarly, updating the agent's mental state corresponds to updating the state associated with that local plan.

Take agent AP as an example. We define place *Home* and *Office* for the agent's two properties *isHome* and *isOffice*, respectively; and transition *Move* for the agent's action *go-to-work*. The color set for the place *Home* can have only one element, since *isHome* has only two possible assignments, and one assignment corresponds to the default assignment of place *Home* containing no token. We define the color set as {*person*}. If the place *Home* holds the token "*person*", then *isHome* equals "*YES*". The default assignment for *isHome* can be defined as "*NO*". Place *Office* has the same color set as place *Home*. We also define arcs *(Home, Move)* and *(Move, Office)* in the local plan, and the inscriptions in arcs *(Home, Move)* and *(Move, Office)* are both "*1`person*".

In the run-time, taking the action "*go-to-work*" can be represented by firing the transition *Move* in the agent AP's local plan, which results in the token "*person*" being removed from place *Home* and a token "*person*" being deposited into place *Office*. Figure 1 shows the local plan for agent AP.

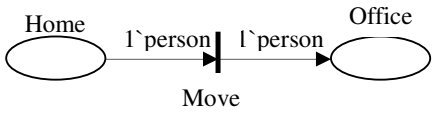


Figure 1. A local plan for a simple agent AP

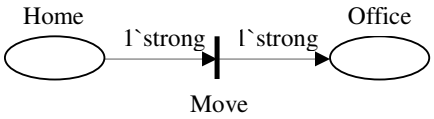


Figure 2. A revised local plan for a simple agent AP

To illustrate the role of arc inscriptions on the local plan model, let's refine the previous example. Consider Figure 2. The difference between Figure 1 and Figure 2 is that the color set and the inscription on the arcs have changed. The new color set is defined as *{strong, week}*. The inscriptions for both arcs *(Home, Move)* and *(Move, Office)* are revised to "*1`strong*". The place *Home* should hold a token "*strong*" to enable transition "*move*". A token "*weak*" in place Home cannot enable transition "*move*." Intuitively, this enabling policy can be explained as follows: A person at home should be strong enough to walk a long distance in order to be moved from home to office.

Based on the BDI agent model [5], an agent has goal, plan, knowledge base, and environment modules. The local plan of our agent corresponds to the plan module in the BDI agent model. In the BDI model, an agent's properties in the plan module also contain belief state set and goal state set. Goal, knowledge base, environment, belief state set and goal state set are used to describe a decision making policy for an agent in the BDI model. These components are omitted from our agent model since the technique proposed here is independent of any specific agent decision making policy.

A local plan only involves one agent; however, in a MAS, agents interact with each other and one agent's action may be dependent on the actions of other agents. In particular, conflicts may occur due to competition for external resources. Therefore, supporting a flexible conflict resolution mechanism is important in MAS modeling (as discussed earlier). We use the term *global plan* to refer to the model of the whole MAS. The global plan is a concatenation of local plans, and such concatenation must ensure that the resulting global plan encapsulates the behaviors of all agents. The global plan should also model the interaction between different agents and provide mechanism for conflict detection and resolution. Ideally, the resulting global plan is also a CPN model, since the local plans are CPN models.

As a summary, a CPN based MAS model should include local plans for each agent involved, and a global plan results from concatenation of local plans. As we can observe, a CPN-based local plan can't distinguish potential conflicts from real conflicts, since the model can't encapsulate events triggering run-time conflicts. In the next section, we will introduce the potential arc concept to deal with this issue.

3. POTENTIAL ARCS IN MAS MODELING

Considering the MAS environment, we can extend our description of an agent to include the concept of a path: *Within an agent, an action is taken through one of several predefined paths. Associated with each path are an agent action and a set of resources that may be acquired or released. In the runtime, an agent action can be taken along any available path, whose availability is determined by the availability of associated resources (especially external resource). The agent is provided the autonomy to select any available path.*

Now we introduce the concept of potential arc (illustrated by the dashed arcs in Figure 3, Figure 4). In the basic CPN model, each arc carries one inscription – we refer to these as *regular arcs* and *regular inscriptions*. We now add the capability for an additional inscription to describe the need for access to an external resource that is modeled "out-side" of a local plan. This new inscription is called a *potential inscription*, since the access to the resource is a potential access in terms of the view of the design model. We call an arc that includes both a regular inscription and a potential inscription a *potential arc* and we call a CPN with potential arcs a *Potential Colored Petri Net (PCPN)***.** A potential inscription carried by an incoming potential arc specifies the resource unit required to enable the arc's destination transition, while a potential inscription carried by an out-going potential arc specifies the resource unit released after the arc's source transition is fired. We can view the PCPN model as a superset of CPN models.

By adopting the concept of potential arc, a local plan can encapsulate the description of an external resource that triggers a conflict. Thus, we can distinguish a potential conflict from a real conflict in the agent model. Due to the unpredictability of external resource conflict at runtime, it is useful to include several paths for each agent action in an agent's local plan. As long as a path exists in the runtime, the corresponding agent action can be taken, resulting in a conflict-free behavior in the MAS environment. In this way, we tolerate potential conflicts in the agent design stage.

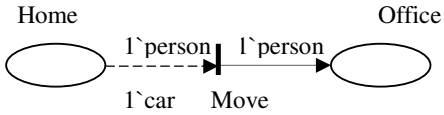


Figure 3. A potential arc from place to transition

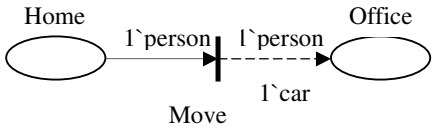


Figure 4. A potential arc from transition to place

Figure 3 is a PCPN-based local plan revised from Figure 1. Intuitively, the agent modeled by Figure 3 has the potential to fire transition *Move*. But in order to enable the transition, we not only need token "*person*" in place *Home*, but also some other agent from the outside must provide one resource unit "*car*." Figure 4 is another revision of Figure 1. In Figure 4, we need token "*person*" in *Home* to enable transition *Move*, and firing the transition would release a resource unit (*car*) that is accessible outside the scope of this local plan. The two local plans, Figure 3 and Figure 4, define different ways in which the agent of concern, AP, is able to interact with other agents.

In the design stage of an agent's local plan, the designer can use the potential arc concept to describe a potential path for one agent action, without concern for the ability of that potential path to become a real path at run-time. After creating the individual PCPN-based local plans for each agent of the MAS, the local plans will be concatenated to form a global plan. This concatenation process depends upon an appropriate interpretation scheme for the potential arcs that appear in the local plans. A potential arc is interpreted as follows: For each potential arc, we create a place node outside the local plan to produce or consume the resource unit specified by the potential inscription. Figure 5 and Figure 6 show examples of how a potential arc in a local plan is interpreted in the global plan. As we can see, Figure 5 shows how the potential arc in Figure 3 is interpreted in a global plan. The potential arc *(Home, Move)* in Figure 3 is replaced by a regular arc, and place *R1*, outside the local plan model, is added to control the enablement of *Move*. *R1* needs to produce the token "*car*" so that transition *Move* can be enabled. Similarly, Figure 6 shows how the potential arc in Figure 4 is interpreted in a global plan.

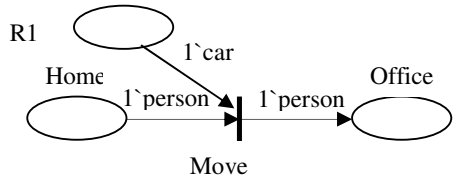


Figure 5. The interpretation of Figure 3

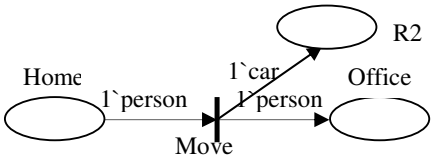


Figure 6. The interpretation of Figure 4

The concatenation procedure interprets all the potential arcs from different local plans by the above semantics, and also develops a unit, modeled as a CPN, called a *coordinator*. The final step involves connecting the coordinator with all the local plans to form the global plan. An important observation is that with our approach, an agent's local plan can be modified in a way that is transparent to other agents. This is because a modified local plan can be reconnected to an existing global plan as long as the potential arcs of the modified local plan remain the same as in the pre-modified local plan..

4. A MAS DESIGN PROCEDURE

After adopting the concept of potential arc, the MAS design procedure can be divided into four stages. Tasks of the four different stages are briefly defined as follows. (1) Specify the type and the quantity of agents, and the communication language in the MAS system. A communication language is a color set defined for agents to specify the external resource unit requested or released. (2) Design the local plan for each agent. (3) Interpret all potential arcs to develop the coordinator. (4) Concatenating local plans and the coordinator to generate the global plan.

When designing a MAS model, we need to make sure all the agents in the MAS use the same communication language so that a resource unit requested or released by one agent can be recognized and processed by others. An agent communication language has an analogy with human language. In human society, two people speaking two different languages can't communicate with each other. Likewise, in the agent society, two agents representing their resources in different communication languages can't interact with each other, because there is no way for one agent to recognize and process the resource requested or demanded from the other. The external resource can be represented as potential inscription, so we use *PI* to denote such a potential inscription. In our discussion, the CPN ML language [13] is used to specify the communication language. According to the definition of the CPN ML language, *PI* is a CPN ML expression that evaluates to a multi-set or a single element of the communication language. For example, the following statement written using CPN ML defines color set RESOURCE, which serves as our communication language:

colset ResType = with car | train | bus;

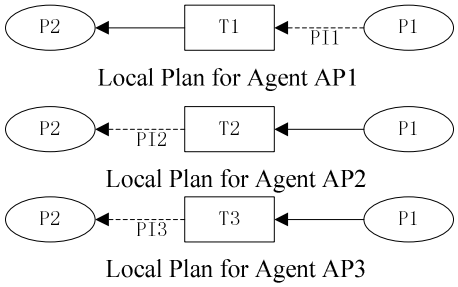
colset Loc = with home | office | park | school | hospital;

*colset RESOURCE = product ResType * Loc;*

An element on RESOURCE is a 2-tuple (*x*, *y*), which defines one color token. For example, *(car, office)* is a single element on RESOURCE; it's a colored token. *2`(car, office) + 3`(train, office)* is a CPN ML expression that evaluates to a multi-set on RESOURCE. Both *(car, office)* and *2`(car, office) + 3`(train, office)* are valid *PIs*. We can also use variables in *PI*. For example, *x* is a variable whose type is color set *Loc* defined above; so

1`(car, x) + 2`(train, office) is also a valid *PI*. Note that our design procedure is independent of any specific representation of a communication language, as long as the communication language guarantees that a *PI* can be recognized and processed by all the agents.

When designing a coordinator, first we create a Petri net place for each type of external resource that appears in potential inscriptions, and create a transition corresponding to each source or destination transition of a potential arc in a local plan. Also, the agent id is added as a prefix for transitions in the coordinator to ensure unique naming. Then we match the transitions and places in the coordinator and connect them together. In the coordinator, a transition *T* matches a place *P* if and only if *T* is created from a local plan transition that connects to a potential arc, whose *PI* contains the resource represented by the place *P*. Since a *PI* may contain variables, *T* may match multiple places in the coordinator. Connect *T* to different matching places will lead to different coordinators. If a transition in the coordinator is created for a source transition of a potential arc in a local plan, an arc will be drawn in the coordinator from transition to place, while place to transition arc will be drawn for the transition created for a destination transition of a potential arc in a local plan. Figure 8 provides two different coordinators (*Coordinator1* and *Coordinator2*) for the local plans in Figure 7. Transition *AP1.T1* matches both places *(Car, blue)*, and *(Car, red)*, and it's connected to *(Car, blue)* in *Coordinator1* and *(Car, red) Coordinator2*. The step-by-step coordinator generation procedure is omitted due to lack of space.



PI1: *1`(car, x)*, PI2: *1`(car, blue)*, PI3: *1`(car, red)*
Figure 7. Local plans for agent AP1, AP2, and AP3

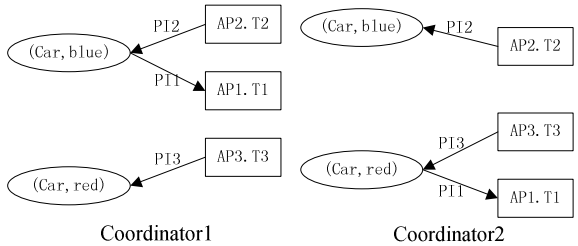


Figure 8. Two different Coordinators for local plans in **Figure 7.**

Concatenation of local plans and a coordinator can be realized by merging corresponding transitions from the coordinator and local plans. When two transitions are merged, the arcs connected to a transition in a local plan would be connected to the corresponding transition in the coordinator. Also, the agent id is added as a prefix for those places and transitions in local plans to ensure the unique naming in the resulting global plan. For instance, Figure 9 shows the global plan generated by merging transitions from Coordinator1 in Figure 8 and local plans in Figure 7.

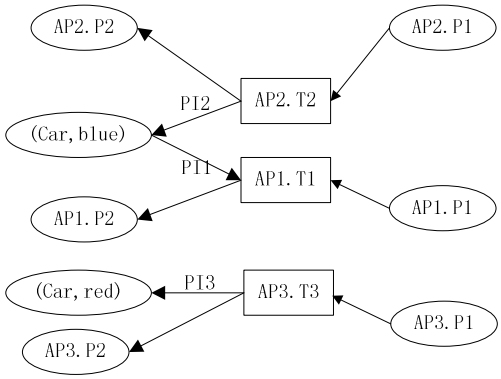


Figure 9. The global plan generated using local plans and *Coordinator1*

5. CASE STUDY

To put the pieces together, and illustrate our design procedure, we now apply the procedure discussed in Section 4 to a simple scenario and show the final result.

5.1 A Simple Scenario

A family lives in the suburb of Chicago. The father needs to work on every weekday morning, so he needs to commute from home to office, and in the afternoon, the father may go to the park for sports if the weather allows. In the case of bad weather, the father returns home directly from the office when the weather doesn't permit out-of-door sports. The son is a college student, and he needs to work part time every evening in a hospital to earn some money for his tuition. He sleeps in the morning, and goes to school every afternoon for class. The office, the school and the hospital locate downtown, while the park lies in the suburb of the city. The home locates in the other side of the city, also a suburb area.

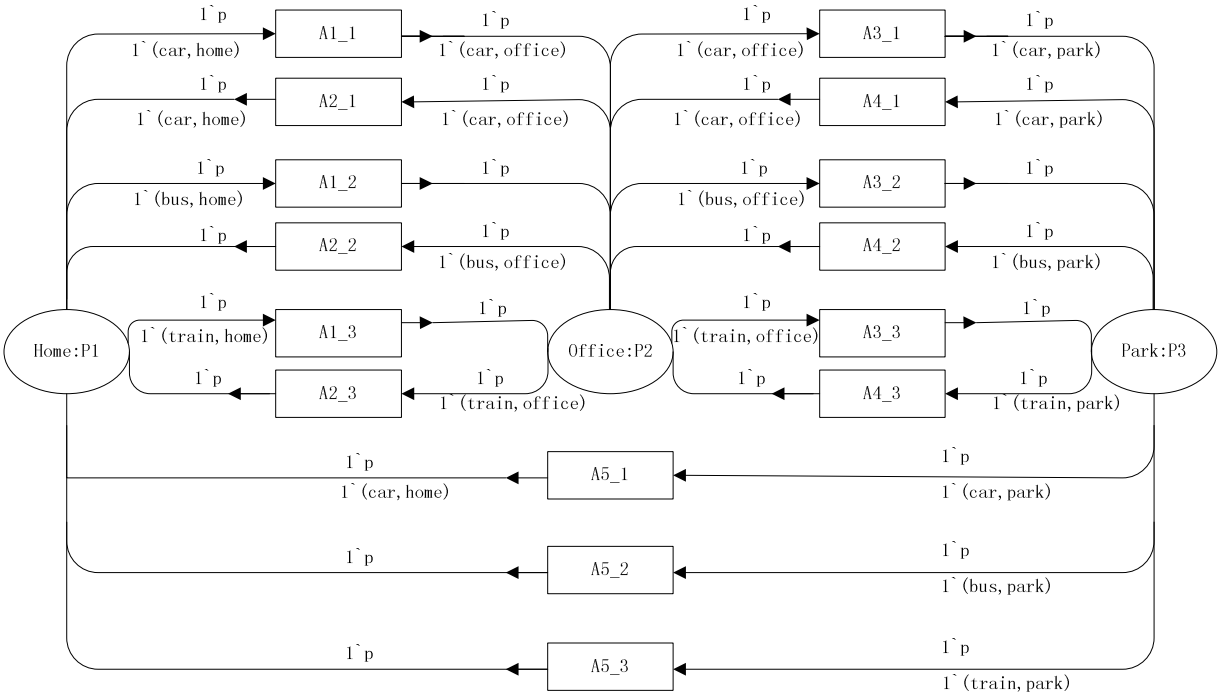
For the transportation between two locations, there're three choices: by car, by train, or by CTA bus. However, not all the three options are available between every two locations. For example, there is no train between the school and the hospital. The family has only one car, so that if the father drives the car to work, the son can't use the car before his father drive the car back. The

availability of trains and buses isn't fully predictable, although it roughly follows some schedule.

Based on the scenario above, we can develop a MAS model to describe the commuting behavior of the family, helping them to choose an appropriate approach at a specific moment. Here the car, trains and CTA buses are described as external resources, and their availability changes dynamically in the runtime. We assume that the weather conditions are not reliably predictable. These impact on the father's behavior, and the availability of the car consequently. Therefore, the resource availability is unknown at the design stage.

5.2 System Design

We design four agent types in our system: *father*, *son*, *train dispatcher* and *bus dispatcher*. The father agent models the father's commuting behavior. The son agent is responsible for modeling the son's behavior. The bus dispatcher agent and the train dispatcher agent model the dispatching of buses and trains at different locations according to some mechanism, respectively. For this simple example, we use one agent of each type and we adopt the color set RESOURCE, defined in Section 4, as the communication language. We follow the design procedure in Section 4 to generate a MAS system, including local plans for each agent, a coordinator, and a global plan. The detailed design procedure is omitted due to the lack of space, but Figure 10 to Figure 13 illustrate the local plans for the four different agents. Remember, inscriptions below arcs are potential inscriptions, while inscriptions above arcs are regular inscriptions.



Actions: *A1*: Travel from P1 (Home) to P2 (Office). *A2*: Travel from P2 (Office) to P1 (Home).
A3: Travel from P2 (Office) to P3 (Park). *A4*: Travel from P3 (Park) to P2 (Office) *A5*: Travel from P3 (Park) to P1 (Home).
Path representation: Ai_j : the jth path for action Ai.

Figure 10. The father's local plan

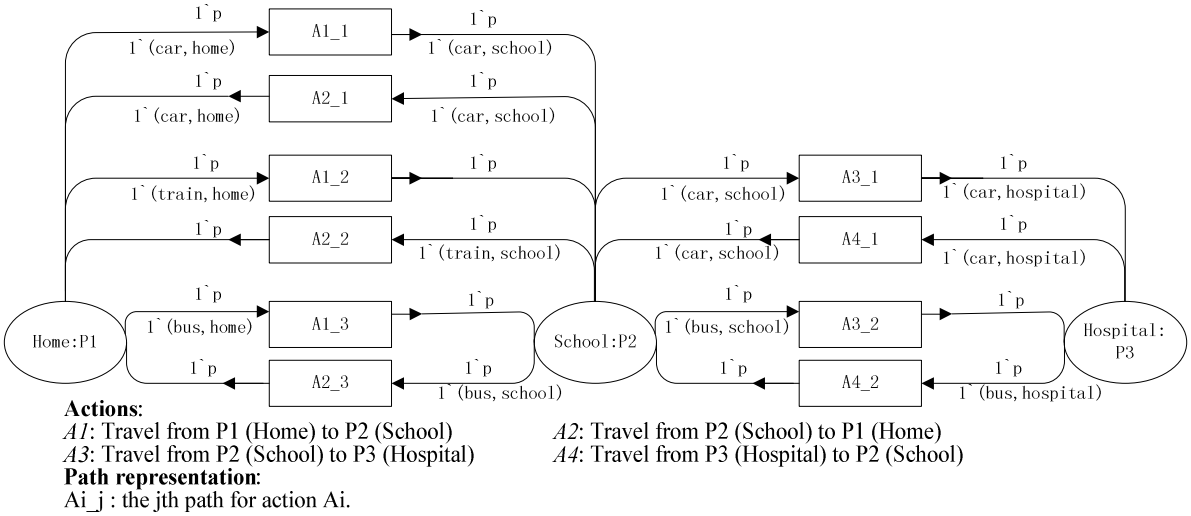


Figure 11. The son's local plan

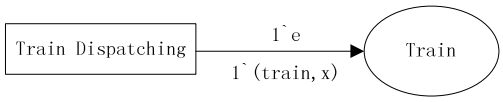


Figure 12. *The train dispatcher's local plan*

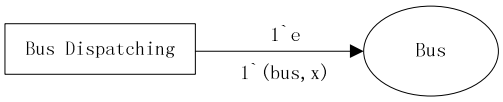


Figure 13. *The bus dispatcher's local plan*

The train dispatcher's local plan has only one action with one path: dispatch a train to a location specified by variable *x*. Similarly, we get the Bus Dispatcher's local plan. Since the coordinator is quite large, we only show parts as an example. Figure 14 shows the transitions connected to the resource *(car, home)*, and Figure 15 shows transitions connected to the resource *(bus, home)*. From Figure 14 and Figure 15, we can clearly capture the interactions surrounding a specific resource without concern for other parts of the model. Finally, generating the global plan means merging the transitions from local agents and the coordinator. We also omit the global plan due to the lack of space and the size of the global plan.

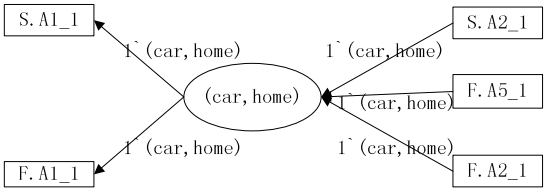


Figure 14. Part of the coordinator related to the car

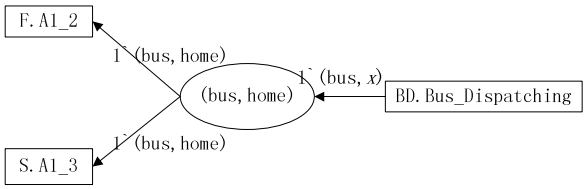


Figure 15. Part of the coordinator related to buses

6. CONCLUSION

Briefly speaking, the potential arc concept expands the modeling power of CPN to distinguish the representation of potential conflicts from real conflicts in MAS design. This feature also leads to a modular design procedure where individual agent designs are independent.

REFERENCES

[1] K. S. Barber, T. H. Liu and A. Goel C. E. Martin.1999."Conflict Representation and Classification in a Domain Independent Conflict Management Framework." Proceedings of the Third International Conference on Autonomous Agents.

[2] K. S. Barber, T. H. Liu and S. Ramaswamy.2001 "Conflict Detection During Plan-Integration for Multi-Agent Systems." IEEE Transactions on Systems, Man, and Cybernetics, Volume 31, Number 4, August.

[3] International Standard. 2000. "High-level Petri Nets - Concepts, Definitions and Graphical Notation. Final Draft." International Standard ISO/IEC 15909, Version 4.7.1, October 28.

[4] Tom Holvoet, 1995."Agents and Petri Nets." Petri Net Newsletters, Number 49.

[5] David Kinny, Michael Georgeff, and Anand Rao. 1996. " A methodology and modeling technique for systems of BDI agents." Proceedings of the Seventh European Workshop on Modeling Autonomous Agents in a Multi-Agent World.

[6] Iouri Loutchko and Frank Teuteberg. 2005. "An agent-based electronic job marketplace: conceptual foundations and fuzzyMAN prototype." International Journal of Computer Systems Science & Engineering.

[7] Robin Milner, Mads Tofte, Robert Harper and David MacQueen.1997." The Definition of Standard ML – Revised. "The MIT Press, May.

[8] Daniel Moldt and Frank Wienberg. 1997. "Multi-Agent-Systems based on Coloured Petri Nets." 18th International Conference on Application and Theory of Petri Nets.

[9] Pavlos Moraïtis and Alexis Tsoukiàs. 1996. "A Multi-criteria Approach for Distributed Planning and Conflict Resolution for Multi-Agent Systems." 1996 International Conference on Multi Agent Systems.

[10] T. Murata.1989. "Petri Nets: Properties, Analysis and Application." Proceeding of the IEEE, Volume 77, Number 4, April, pp. 541-580.

[11] Yaov Shoham.1993. "Agent-Oriented Programming." AI, Vol. 60, 51-92.

[12] John A Sillince. 1994. "Multi-agent conflict resolution: a computational framework for an intelligent argumentation program." Knowledge Based Systems, Volume 7, Number 2, June.

[13] Haiping Xu and Sol M. Shatz.2003 "A Framework for Model-Based Design of Agent-Oriented Software." IEEE Transactions on Software Engineering, Volume 29, Issue 1, January.