Specifying Knowledge in Cognitive Multiagent Systems using a Class of Hierarchical Petri Nets

Eder Mateus Nunes Gonçalves
Center for Computational Sciences - C3
Federal University of Rio Grande - FURG
Rio Grande, RS, Brazil
Email: edergoncalves@furg.br

***Abstract*—Nowadays, problems demands new computational paradigms to solve them. New computational systems must consider features which before was not. Among these new features is the distributed nature of some systems. Nevertheless, even problems with a centralized nature has so high complexity that the best solution is to break them in small blocks. To deal with this problems, multiagent systems (SMA) and agents are seen as an excellent alternative as a framework to model, specify and codify solutions. However, there is not a standard framework to aid in building such systems. In this paper, it is proposed a framework based on a specific Petri Net to model, specify and codify the knowledge since the social level of the system until the agents reactive levels. This approach is based on a explicit separation between the knowledge level and the mechanisms which will manipulate it. The Petri net model proposed presents an implied specification language which permit to deal with any knowledge representation formalisms, since production rules, ontology or even database structures. Besides that, the Petri net proposed has tools to analyse and validate the system about issues like redundancy, deadlocks and conditions associated to agents tasks.**

***Index Terms*—multiagent systems, petri nets, specification, knowledge-based systems**

I. Introduction

There is a class of problems that demands new paradigms to model, specify and implement computational systems to solve them. Some problems present a distributed nature, i.e., it is composed by sub-systems whose operation is viable only by integration of its parts. Example of this kind of problems are the recomposition of an electrical network after a blackout, the monitoring and control of the natural environment of a forest park, the management of a distant education system, or the control of a team of robots that should play a football game.

In general, problems like this share the following characteristics:

- they are physically and/or conceptually distributed, in the sense that their global state is composed by the aggregation of partially independent local states,
- the tasks involved in solving these problems refer to different levels os abstractions, varying from global coordination protocols to local perception/action procedures, that use sensors to perceive the world state and effectors to act in the world.

In this case, it is natural to use an approach based on distributed systems. Special attention exists in systems carried out by multiagent systems [1], an area that integrate distributed systems with techniques from artificial intelligence, specially the agents technology. In fact, multiagent systems is classified as an area of Distributed Artificial Intelligence [2].

A multiagent system is composed by a set of agents. The power of a multiagent system is the agent integration in the sense to solve a computational problem. An agent can be seen as an entity which see its world from its perceptions and act on it by its actions. Besides that, an agent is an autonomous entity once it deliberate by yourself [3].

In the context proposed in this work, an agent has local goals and knowledge associated to local states of the problem. Also, they share global goals with other agents to determine how local states are assembled in a global state [1]. Nevertheless, the system complexity is splited into different decisions levels that can be structured inter-agents or intra-agents.

According to the abstraction levels necessary to describe a problem, one for each decision level, an agent can be designed by an hierarchical strategy based in one or more Knowledge-Based Systems (KBS). In this work, a KBS is a generic formalism to encapsulate and implement an agent which is able to manipulate and infer about any knowledge level, since deliberative planning until reactive tasks. Its only constraint is to present mechanisms to detach implementation aspects and the needed knowledge to codify a solution [4].

Each intelligence aspect of an agent, like planning, environment state classification or actions coordination, is implemented by a specific KBS. The multiple agents must coordinate their actions in the sense of acting intelligently in the environment. Besides, the agents must coordinate their actions to achieve common goals.

According to this approach a system is constituted by two main levels: a *social level*, where is defined the system collective strategies; and *individual levels*, where

This paper is based on "An Approach to Specify Knowledge in Multiagent Systems using Petri Nets," by E. M, Gonçalves, which appeared in the Proceedings of the 4th International Conference on Network and System Security (NSS), Melbourne, Australia, 2010. © 2010 IEEE.

This work was supported in part by Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) .

doi:10.4304/jsw.7.11.2405-2414

the social knowledge is instantiated according the agent tasks and roles in the environment.

In this paper, it is proposed an approach to specify the social and individual levels using the same formalism. This formalism is based on a high-level Petri Net designed to work as an interface between the expert in the domain and the framework(s) used to implement the system. The Petri net proposed presents a specification language and generic mechanisms to manipulate different knowledge representation formalism using different inference mechanisms. Besides that, the Petri net proposed permits to create and verify a formal mapping between the social and individual levels through a hierarchical formalism which integrate the knowledge of the whole system.

The methodology, basically, works in the following way. In a first level, we have a multiagent specification. The Petri Net, in this case, represents the various plans that can be executed by the agents. This specification does not have a real implementation, but it just guides the agents strategies, standing in a conceptual and operational level. From this multiagent specification, each agent instantiate its own specification, also in a Petri Net model. This instantiation consider the agent role and its possible actions in the environment. The agent can have multiple nets hierarchicly structured, one net for each intelligence aspect, or one net for each KBS. The adopted Petri Net model makes possible to use any knowledge representation formalism, such as frames, logic or production rules. The token, in a general description, is a data structure, that can assume any knowledge base form.

This paper is structured in this way. The next section presents a synthesis about the use of Petri Nets in multiagent systems and agents specification. Section III describes the specification language integrated in a high-level Petri net model to specify different levels of a multiagent system. Section IV presents some applications examples of the proposed approach and section V presents the conclusions and some future works from this stage of work.

II. Related Work

In the development of agents, Petri nets can be used to specify, model and simulate either in social or individual levels. They are a good formalism once they simple to use and understand, they have a powerful expressivity and a substantial baggage. However, there are a few works that use it in both levels in a integrated framework.

Considering the use in the multiagent specification, Petri nets are an convenient tool once that can represent parallel and synchronization activities. The main ideia is to represent the multiagent system through the composition of multiple nets, where each one represents a single agent in the environment.

In [5] and [6] Petri nets are used to represent parallel activities in multiagent plans. The main argument for this work is to represent explicit parallelism in the sense to reduce the necessary communication and coordination between the agents. The approach permits a mapping between a Planning Graph, as Graphplan [7], and a Predicate/Transition net.

In [8] is provided a formalism to specify multi-model nets. In this case, each net represents an agent, while the inter-net specifications describe the cooperation structure of the corresponding agents. System components are modeled by objects or agents, whose behavior is modeled by Petri nets. It is provided a means to abstract from "how" of agent communication, emphasizing the "what" of agent interdependencies.

In [9], Coloured Petri nets are used as a formal description model, for composite behaviors among agents. In this model, it is possible to express data and control dependency between agents explicitly. The model also permits to structure the model in a hierarchical manner.

In [10] it is presented an application in which Petri nets are used to specify the social system in a interactive learning environment, MATHEMA [11]. The Petri net is used to specify the relationship between the necessary tasks to communication process between the agents in SATA (Society of Artificial Tutoring Agents). In the MATHEMA, each SATA component is responsible by a knowledge domain, and the interaction is used to treat multidisciplinary requests. Thus, Coloured Petri nets are used to model, analyze and simulate the social system. A model more generic of this approach is presented in [12] where the multiagent system is implemented through the coordination of individual actions considering different agents architecture and a not controllable and not deterministic environment.

In the individual context of the multiagent development, i.e., in the development of the agent itself, Petri nets are used, in the most cases, in the verification and validation of the knowledges bases. However the model used to represent a rule-based system should respect some principles in the sense to avoid some inconsistencies that can cause errors in the final system.

In [13] it is proposed a rule-base system verification using Petri nets through its structural connectivity among rules clauses, that is determined through the reachability properties of the net. Intuitively, a rule in a Petri net is formed by a transition, whose input places represents the rule antecedents and the output place represents the rule consequent. However, in this model, it is introduced a partitioned set of places that represents *external* clauses, *inferred* clauses and *goal* clauses. From this partitioned model, it is introduced a new marking scheme in the net, which alters the reachability analyses. Now, according to the type of anomaly investigated is generated a submarking reachability analyses, based on one, or more than one, partition of the place set. The limitation of the approach, however, is on the model used, ordinary Petri nets. In this case, it is only permitted the use of classical logical as knowledge representation in a only one abstraction level.

An another approach for the verification of knowledge bases is PREPARE [14]. PREPARE is based on modeling a knowledge base by using a Predicate/Transition net representation, whose anomalies are detect as patterns

of the net model through a syntactic pattern recognition method. These patterns are formed through strings associated to places and transitions of the net, which by free-context grammar identify the anomalies to be detect. The main advantage of this model is the simplicity, once that the mapping between the knowledge base and Predicate/Transition net is formally defined in first-order logic. However, the model is very difficulty to use in knowledge bases with hybrid knowledge representation, and with more quantity of knowledge.

[15] propose a enhanced high-level Petri net for verification and validation of rule-based system whose differential is the possibility to represents variable and negative sentences. Just like other models, the verification and validation process is based on reachability analyses, even in the matrix representation. To represent variables and negative sentences, it is used special colors and inhibitors arcs in the net, causing little differences in the net modeling. The main advantage of the model is the formal rigor in the net construction, but equal to the other model, it is limited to representation in first-order logic. Besides that, the use of hierarchic models are not obvious once that the Colored Petri net model undergoes an extension process that change its structure.

There are a lot of work in the specification and verification of fuzzy knowledge bases using different models of fuzzy Petri nets [16] [17] [18] [19] [20] [21]. Basically, these works propose the representation of a weighted fuzzy production rules in a Petri net model that is enabled to express certainty factor of the rule, thresholds and weights. The difference among the models is the way to represent these factors. Besides that, the models also contain a reachability analyses to structural verification of the knowledge. In these cases, of course, these models are appropriated for fuzzy systems. In all of these works, there are not models that considers the insertion in a multiagent context.

The works presented until here consider the use of Petri nets just in the verification and validation phases, even in the individual or the social development of agents. The works presented below describes different roles to Petri nets in the development of intelligent systems.

In [22], it is proposed the use of high-level Petri net in task structures with the objective to verify knowledge based-system. Task structures acquire and organize domain knowledge, functional requirements, and problem solving methods around the general notion of tasks. For a requirements specification driven by task structures, pieces of the specification can be refined iteratively and verification can be performed for a single layer or between layers [23]. According the author, this approach permits an analyze in a semantic context and not only in a structural context. This is possible once that the analysis is performed in the specification level of the system requirements. According to the principles established in the Software Engineering, the verification process must happens in two different levels: model and process level. The model verification refers to static properties of the systems (domain model), meanwhile the process specification refers to dynamic properties of the system (functionality). The use of Petri nets is viable from a mapping in task structures. In this way, the model specification is represented through a constraint net, and the process specification is performed through a reachability analyses. The constraint net and the reachability analyses guarantee the system consistence about its requirements, and it is permitted a global vision of the system.

In [24], it is proposed a method to validation of a knowledge-based system specification through the transfer of its informal conceptual model in KADS in a formal operationalization model using Petri nets. Basically, it is the same approach cited above, where the task structures are switched by informal conceptual model in KADS. In this case, the operational model is used to simulate the system and, consequently, the specification validation. The operational model is obtained through the transformation of the inference model and tasks in Petri nets, whose relationship are formally established. The main drawback of the model is the possibility of a whole vision of the system. However, its formalization is limited to the use in a decentralized environment, without multiagent strategies.

The main conclusion about the related work analysis is that there is not a framework that integrates the social and individual specification. Besides that, there are specifics demand in each development level, social and individual. All the works consider only one knowledge representation formalism, normally a logic one. Nevertheless, the system made are very simple since the models do not consider hierarchic structures.

III. Petri Nets as Specification Language to Multiagent Systems and Agents

The approach presented in this section consider a development based on the explicit distinction between the knowledge level and the inference mechanisms [25], according to a generic description of a KBS. In this scenario, the approach proposed must be used to model and specify the knowledge level of a multiagent system (MAS) in all levels.

In this scenario, one can have a specific framework to formalize the MAS, and another framework to codify the agents which belong to the MAS, or, one can have different frameworks to codify agents, considering its internal architecture. Example of a framework to develop MAS is the Moise model [26], and a example of framework and/or language to develop agent itself is Jason [27].

In fact, in this model, MAS is considered an abstract formalism that must define the guidelines in the agent development. These guidelines, according to the MAS model used, can be based on concepts like goals, plans, roles and collective actions in a environment. An MAS is implemented by the agents which compose and act in the environment.

Nevertheless, an agent can present an architecture with many different decision levels, according to the environment complexity in which it will act. These decision levels

can encapsulate different aspects of intelligence necessary to percept and act in the environment.

From an initial knowledge elicited with the aid of an expert in the domain, and already structured according to the intelligence aspects of the each decision level, it is proposed a specification language that helps in the knowledge specification at the *knowledge level* [25]. This language is based on a particular Petri net model that presents some important aspects to the knowledge acquisition process:

- its graphical model is used as a diagrammatic language that help the interaction between the expert and knowledge engineering, minimizing the communication problems. It permits to specify concurrent activities, in social and individual context, and the multiple relationships between them. Besides that, the language is independent of the mechanisms used by the architecture, once the specification is made in the knowledge level.
- its mathematical model can be used to verify problems like inconsistencies, ambiguities and redundancies;
- Petri nets can be composed hierarchically, respecting the multiple levels of a multiagent system;
- it is possible to automatically translate the information contained in the net into a knowledge base, once the data structure associated with the token is adequately arranged according to the knowledge representations chosen to the KBS.

The methodology is implemented in a top-down approach, i.e., the system must be specified from the social context in the sense of the individual actions in the environment. However, inside of each level, the specification can be either in top-down or bottom-up approach. According to the model presented in figure 1, each level corresponds to a KBS, or an hierarchic Petri net model, that presents a hierarchic relation between them. As seen in the above sections, this relationship is made by the inference results of each KBS that are sent to the underling levels, in the form of goals or selected behaviors, for example.

The use of Petri net is justified once that it is a well established tool for the specification of an information system of any type, specially those that need to specify concurrency and synchronism. Besides the aspects cited above, the KBS can be seen as a discrete event system, once the state changes, or the occurrences of new facts, are not guided by time. On other hand, Petri nets is also used in the development of rule-based system with good results[1]. However, the model presented below presents extensions that guarantee its application in a broader context.

Basically, Petri net is a graphical and mathematical tool that permits to specify, model, simulate and verify any discrete event system, just like a KBS. Its graphical representation is formed by two types of nodes, *places*

[1]C.f. section II

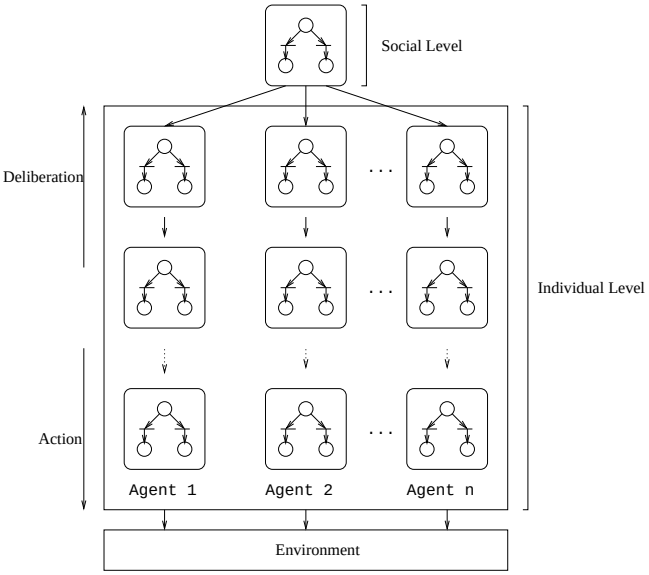

Fig. 1. A description of the development model using Petri Nets in the Expert-Coop++

and *transitions*, and *tokens* that are used to feature the system dynamic. Places are represented by circles and transitions by bars. Tokens are represented by dots inside the place, when the predicate associated with the place is true. Thus, we can define a Petri net formally by a n-tuple $R = \langle P, T, Pre, Post \rangle$ where:

- P is a finite set of *place* with size n.
- T is a finite set of *transitions* with size m.
- Pre is a *forward incidence function*: $Pre : P \times T \rightarrow \mathbb{N}$.
- $Post$ is the *backward incidence function*: $Post : T \times P \rightarrow \mathbb{N}$.

A marked Petri net is a couple $N = \langle R, M \rangle$, where:

- R is a Petri net.
- $M : P \rightarrow \mathbb{N}$ is a initial marking and $M(p)$ is the number of tokens in place p.

In order to represent a KBS using a Petri net, it is necessary to extend the expressive power of the tokens to allow the representation of the knowledge base manipulations that occur when a rule is fired. For that purpose, high level Petri net models (e.g. [28]) are well adapted. Given this model, each transition is associated with pre-conditions and pos-conditions that control its load and fire. Thus, in representing a knowledge base, each transition is associated with a rule, whose its pre-conditions represents the rule premises and its pos-conditions represents the rule conclusions and selected actions. The transition fire implies in a facts base change, that is updated by the manipulation of the tokens. The token distribution represents the state of the facts base.

From an epistemological perspective, the central component of a KBS is the *knowledge base* [3]. Informally, a knowledge base is formed by a set of fact descriptions that, in this case, can include also social facts. A generic knowledge base, independently of the adopted knowledge

representation method, can be formalized through the definition of two access functions called *Tell* and *Ask*, that allow, respectively, to include a new fact in the knowledge base and to query a given knowledge base. More formally, let KB be the set of all possible knowledge bases and ϕ an expression of the formal language used by the adopted knowledge representation method. Without loss of generality, we suppose that ϕ is a term. Let $\mathcal{V}$ be a set of variable symbols, $\mathcal{C}$ a set of names of primitive entities in the domain and $\mathcal{F}$ a set of function names. The set of all terms $\mathcal{T}$ is defined as follows:

- $\mathcal{V} \subseteq \mathcal{T}$;
- $\mathcal{C} \subseteq \mathcal{T}$;
- if $t_1, \ldots, t_n \in \mathcal{T}$ and $f \in \mathcal{F}$, then $f(t_1, \ldots, t_n) \in \mathcal{T}$.

Let also $\mathcal{S}$ be the set of all possible mappings $\mathcal{V} \rightarrow \mathcal{T}$, i.e., the set of all substitutions of variables, and $\mathcal{T}^*$ be the set of all *ground* terms, i.e., terms where no variable occurs. In this way, it is possible to define:

$$Tell : KB \times \mathcal{T}^* \rightarrow KB$$

$$Ask : KB \times \mathcal{T} \rightarrow \mathcal{S}$$

During the knowledge acquisition process, at the agent level, when the lowest abstraction level is reached, the actions associated with the Petri net transitions become actual operations in the domain. These operations usually has preconditions and effects that are registered in a knowledge base. To introduce this conditions and effects into the formalism we extends the Petri net definition in [28] as follows.

The *token* is defined as an element of the set KB, i.e., the token now represents a knowledge base. We introduce the following functions:

$$Cond : T \times KB \rightarrow \mathcal{S}$$

$$Act : T \times KB \times S \rightarrow KB$$

A $Cond$ function is associated with each transition. It receives a knowledge base $k \in KB$ and returns a substitution $\theta \in \mathcal{S}$. In the lowest abstraction level, its general form is:

$$\begin{aligned} &\theta_1 \leftarrow Ask(k, \phi_1) \\ &\vdots \\ &\theta_n \leftarrow Ask(k, \phi_n) \\ &\theta \leftarrow Combine(\theta_1, \ldots, \theta_n) \end{aligned}$$

where $\phi_i \in \mathcal{T}$ are domain dependent terms, possibly containing variables, that are used to query the knowledge base k and $Combine$ is a function that combines substitutions.

In higher abstraction levels, the $Cond$ function may contain expressions such as:

$$\theta_i \leftarrow Run(\phi, R)$$

where R is a lower level Petri net and $\phi \in \mathcal{T}$ is a domain dependent term, possibly containing variables, that is used as a query to the lower abstraction level knowledge base, after the execution of the Petri net R.

An Act function is also associated with each transition. It receives a knowledge base $k \in KB$ and a substitution $\theta \in \mathcal{S}$, and return an updated knowledge base. Its general form is:

$$Tell(Tell(k, \psi_1\theta), \ldots, \psi_n\theta)$$

where $\psi_i \in \mathcal{T}$ are domain dependent terms, possibly containing variables, that represent a generic action and $\psi_i\theta_i \in \mathcal{T}^*$ are the associated ground terms that are used to update the knowledge base k.

The semantic of this extension is the following: before a transition is fired, the $Cond$ function is applied on the knowledge base token and, if the result is a non empty substitution θ, then the function Act is executed with the substitution θ applied to all the terms that occur in it.

An hierarchical model is possible if we extend the above model. The hierarchical model used is based on [29]. According to this model, there are five ways to hierarchical a Petri net: *transitions substitution*, *places substitution*, transitions calls, transitions fusion and places fusion. In this model to specify KBS, we use only the first two.

The principle to use transitions substitution and places substitution is the same. Basically, the idea is substitute the transition or place indicated by another Petri net. In this way, it is possible to refine the action or activity associated with the element. The Petri net that extends either the transition or the place has the same syntax of the superior one.

IV. Motivating Example

In this section, it will be demonstrated a case study about the model proposed in the previous section. The intention is to demonstrate a whole example, where all the steps in the conception of the system is approached.

In [30], it was demonstrated just how to design the formalism to knowledge representation in production rules. However, in this case study, all the features of the model proposed is presented, i.e, the system conception in all levels, social and individual one, the hierarchic capability, multiple knowledge representation formalism and the generation of a knowledge base. Besides that, the whole framework is suitable to a specific agent architecture.

The agent proposed here was developed to act in the Robocup Soccerserver 2D Simulator [31]. In this environment, it must be developed a team of virtual entities to play soccer against other teams of virtual entities. To act in this environment, it is necessary to attend features as real-time control, learning, deliberative and cooperative actions.

A. The agent description

The agent proposed to attend the Soccer Server constraints is based on [32]. According to this architecture, an agent model presents three different decisions levels, in a hierarchical structure: *reactive*, *instinctive* and *cognitive*.

Each level, together with its lower levels, is intended to model a complete agent, each new level just increasing the behavior complexity. The generic agent model is instantiated by a computational architecture named Concurrent Autonomous Agent (CAA), and it is described by the figure 2.

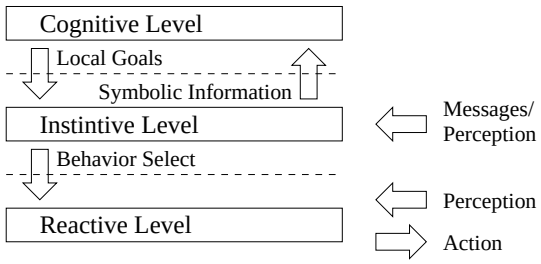

Fig. 2. The Concurrent Autonomous Agent architecture

The reactive level in CAA is responsible by the direct interaction with the environment. It is constituted by a behaviors set implemented by fuzzy controllers, one for each behavior. Just one behavior is active at time. The behaviors set of an agent is determined by its role in the environment.

The instintive level is responsible by the classification of the environment states and the execution of local goals. From this information, it chooses the reactive behavior in the lower level and provides the higher level with symbolic information. This is done by a KBS, that is executed each time the environment state changes. A local goal is reached through the execution of a reactive behaviors sequence. However, in each KBS cycle, the conditions associated with the state are verified. If this new state is not verified, the instinctive level solicit a new local goal to the cognitive level.

The cognitive level chooses the agent goals. Initially, it must coordinate with the other agents in the environment to choose a global goal. It coordinates its actions according to this social goal. This is done selecting a sequence of local goals that is sent to the instinctive level. In this sense, the basic function of this level is to plan in social and individual contexts. Just like the instinctive level, this is done by a KBS, however with two different knowledge bases. The first one is constituted by the social knowledge, responsible by the interaction with the other agents in the environment, and stores the social decisions. The second one is a local base, that according to the global goal and the symbolic information stored, selects the local goal sequence to be sent to the instintive level.

To model and specify this agent in the individual level is necessary to build one KBS for each level. In this implementation, the reactive level is build through a set of fuzzy controllers where only one is active at time. It would be possible to specify these controllers using the model with Petri nets proposed here. However, they were ready when the other levels was started.

B. The knowledge representation formalisms

Given these initial conditions, it was proposed a top-down approach to develop the team, i.e., the development was conceived from the social level to the individual ones. Inside the agent, the cognitive level was first conceived and then the instinctive level.

Before to demonstrate how to use de Petri net model proposed, it is necessary to specify the knowledge representation formalisms used in the SMA and in the agents.

The specification language presented in the previous section is intended to encapsulate any formalism to represent MAS and its agents. This include BDI formalism, ontologies, production systems, frames, logic and others.

Considering in the most basic format, each transition in the Petri Net corresponds a rule in a rule base. Thus, the Ask field corresponds to the rule premise and the $Tell$ field corresponds to the rule conclusion. The optional $Action$ field indicates the action that must be executed in the environment. It is important to consider that the premises and conclusions must respect the knowledge representation formalisms used in the KBS. The figure 3 demonstrates the mapping between a transition in the Petri Net and the rule generated[2].

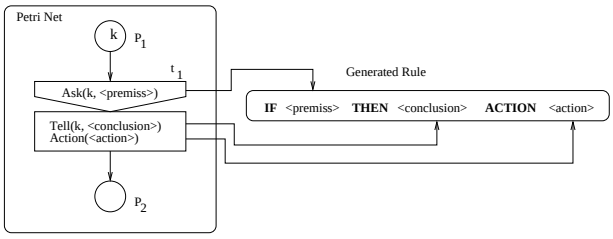

Fig. 3. Generic Rule

Once defined how to use the model in only one level, it is presented the mechanisms that permit the system conception in multiple abstraction levels. In this case, it must be used the Run function. The figure 4 describes the generic mechanism that permit the use of hierarchical conception.

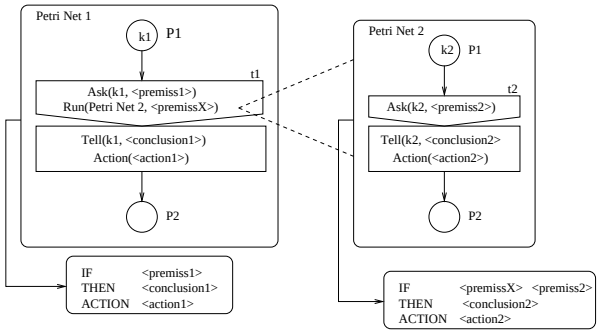

Fig. 4. Generic Hierarchic Rule

When the Run function is called, it must be passed two values: one is the Petri Net that must be instantiated, and

[2]Because the notational simplification, the substitutions and the $Cond$ function was not considered.

the second one is the premise that constrain the inferior level knowledge base. Thus, the rule specified in the Petri Net 1 is just fired if the Petri Net 2 is executed. In the $< premiss1 >$, in the Petri Net 1, it must contain an ask about a fact that is generated only by the Petri Net in the inferior level, i.e., the Petri Net 2. In the premises of the Petri net 2 must contain the one passed by the Petri Net 1.

C. The Petri Nets

According to the approach to implement the CAA, the multi-agent level serves as guidelines to model, specify and codify the cognitive level of the agents. The multi-agent level, which correspond to the social level in the generic description, is structured through sequences of goals, as description in figure 5.

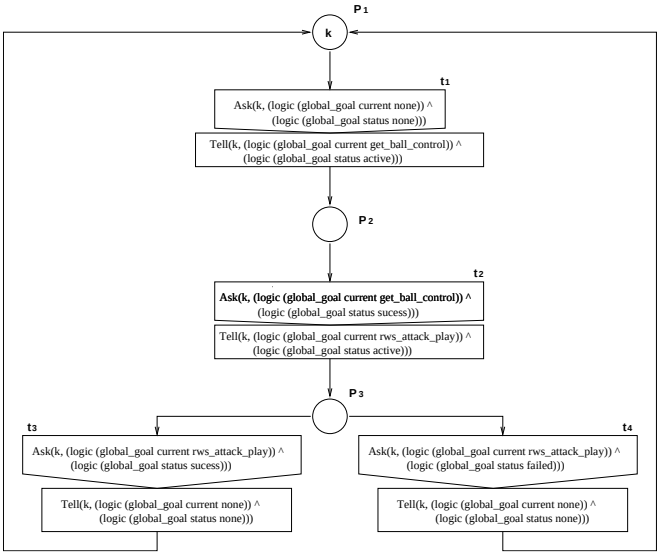

Fig. 5. Multi-agent specification using Petri nets

In the knowledge representation formalism used in the social level, the generic structures Ask and $Tell$ made references to a formalism based on *pattern logics*, which are structures in the form $< object\ atribute\ value >$. The plan specified in this Petri net considers only a specific scenario on the environment.

Once that the multi-agent specification is considered ready, the next step is to specify the social base of each agent. The sum of each social base must implement the multi-agent planning and form the social knowledge about the domain.

Each agent must instantiate the multi-agent planning according to its roles and possible actions in the environment to constitute its social base. In figure 6, it is specified the agent 9 social base according to the multi-agent planning. The link between these two nets can be formal and informal, depending on the model rigor. In a formal link, the multi-agent planning would be capable to generate semi-automaticly the individual social base, from a formal specification of the agent's roles and actions in the environment. This can be done because the individual

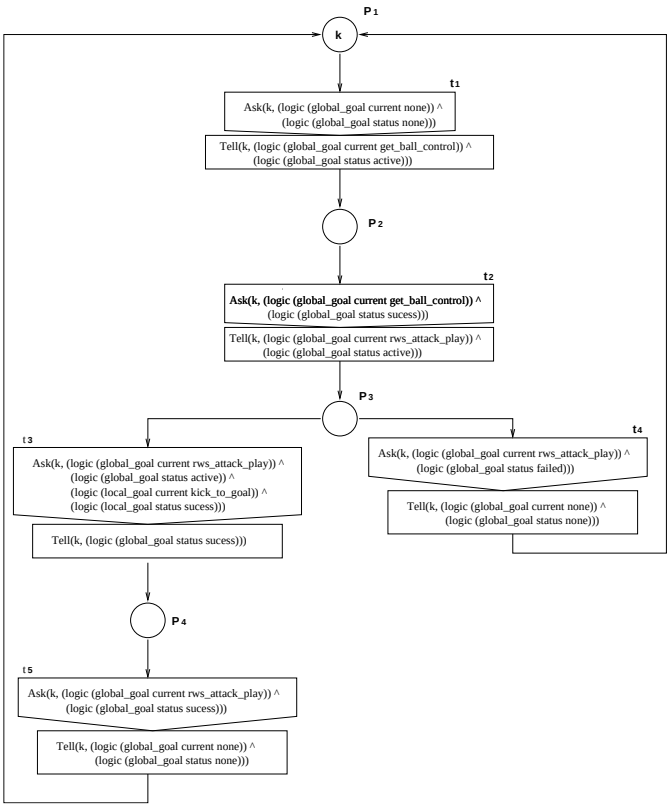

Fig. 6. Agent 9 Social Base Specification

model is inferior hierarchically than the social model, and this relation can be formally established. From this hierarchical structuring it would be possible to semi-automatically transform a social model in an individual one. Unfortunately, this formal specification is not ready yet, but it will be included in the next versions of the methodology. The current methodology permits that the multi-agent specification determine the guidelines to build the social bases in each agent.

In the net presented in figure 6, transition t_3 and place P_4 represent the contribution of agent 9 in the play, if it is compared with the multi-agent model. According to the specification, the agent is responsible by finishing the play, kicking the ball to the goal. The occurrence of the other transitions is defined in a high level description, once that the other agents can interfere with its occurrence.

Using the social base specification, the local base is modeled, according to the CAA architecture. Each global goal, i.e, each transition in the social base model can be represented by a new hierarchically inferior net. The sum of this hierarchically inferior Petri nets represents the local knowledge base of the cognitive level. This local base specifies the agent planning in the individual context.

Consider, for example, transition t_2 in agent 9 social base (see figure 6). This transition can be expanded in the Petri net of figure 7. In this net, all states that can take the agent from transition t_2 to transition t_3 in the social base Petri net are represented. In case of failure, the fired

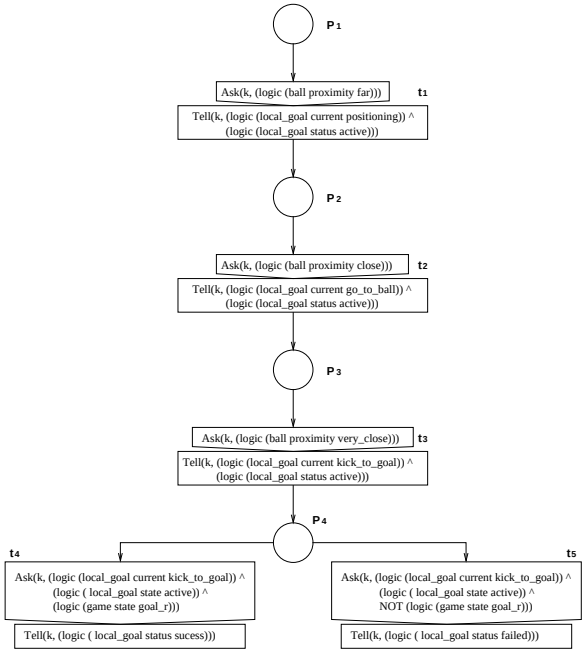

Fig. 7. Agent 9 Local Base Specification

transition can be t_4. Even if the local base mentions that the goal failed, this proposition is maintained in a high-level description in the social base (t_4) once that the others agents can detect the goal failure.

Considering the dynamics of the social base Petri net, when transition t_2 occurs, putting token k in the place P_3, the Petri net of figure 7 is instantiated. Once the token in this net leaves transition t_5, transition t_3 in the social base Petri net, that it is already loaded, is fired, and the Petri net dynamics go on.

D. The knowledge bases

Once all the logic propositions that instantiate the token in place P_3 in the transitions hierarchicly superior is true (see figure 6, i.e., the knowledge base formed by the local base Petri net should consider yet that:

```
(logic (global_goal current rws_attack_play))
(logic (global_goal status active))
```

From this propositions, the local base Petri net must specify the states that can take the social base Petri net from transition t_2 to t_3 or t_4. This states are represented by the local goals that can reach the global one, according to the agent's roles and possible actions. In other words, it represents the agent individual planning. Observe that this net is not restartable. This happens once that the net dynamic continue in the net hierarchicly superior, the social one.

Basically, this local base instance manipulates the symbolic variable *ball proximity*. It represents the ball proximity in relation to the agent. When the ball is considered in the kick area, it is inferred the local goal `kick_to_goal`. If we remember the agent architecture, this local goal is sent to the instintive level for the correct reactive behavior sequencing.

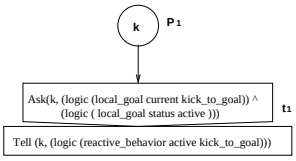

Fig. 8. Agent 9 Instintive Knowledge Base Specification

The same methodology is used to codify the instintive level knowledge base. Each transition in each local base specification can generate a hierarchicly inferior Petri net that represents a instintive knowledge base instance. The sum of all this instances forms the instintive level knowledge base.

Figure 8 presents the Petri net obtained from the expansion of transition t_3 in the local base Petri net. It represents the knowledge base formed to catch the local goal `kick_to_goal`. According to the Petri net in figure 8, this local goal is obtained just activating the reactive behavior `Kick_To_Goal`[3].

To translate logical expressions in to rules is a complex task, even computationally, but the proposed methodology permits this. Next, it is presented some rule example that were generated from the Petri net models described above.

Considering transition t_3 in the social base model, the following rule can be generated, respecting the syntax for production rules:

```
(rule_003
  (if   ( logic ( global_goal current rws_attack_play ))
        ( logic ( global_goal status active ))
        ( logic ( local_goal current kick_to_goal ))
        ( logic ( local_goal current success )))
  (then ( logic ( global_goal status success ))))
```

Also for transition t_3, but now considering the net that models the local base, it has the following production rule in knowledge local base:

```
  (rule_003
  (if   ( logic ( global_goal current rws_attack_play ))
        ( logic ( global_goal current active ))
        ( logic ( ball proximity very_close )))
  (then ( logic ( local_goal current kick_to_goal ))
        ( logic ( local_goal status active ))))
```

Finally, the Petri net in figure 8 generates the follow production rule, that constitute part of the instintive knowledge base:

```
(rule_001
  (if   ( logic ( local_goal current kick_to_goal ))
        ( logic ( local_goal status active )))
  (then ( logic ( reactive_behavior current Kick_To_Goal ))))
```

In fact, in this net, the transition condition is redundant once this proposition is true for the transition expanded in the hierarchicly superior net. In this case, the condition is merely illustrative.

The use of Petri nets as its specification language reduced the design time and the presence of errors, like redundancy and incompatible rules. Petri nets permit to visualize the knowledge base through its own interface, that is more intuitive and expressive than any other expression form. Petri nets permit yet to specify the

[3]The upper case letters are used to distinguish the reactive behaviors from the global and local goals

concurrent activity, that is an inherent characteristic of any multi-agent system.

After the Petri Nets modeling in the different levels of the system, it can be executed an analysis of its properties. This analysis can guarantee some level of syntactic correction.

The analysis is carried out through the underlying model of each Petri Net. The underlying model corresponds to the ordinary model of each Petri Net, that can be obtained considering just the places and transitions, without any data structure associated with the tokens. It is considered an initial marking of the net just with binary tokens [28].

Once obtained the underlying Petri Net of each net, it can be executed simulations of this nets with the goal to guarantee properties like liveness, boundness and restartability. Considering the proposed model, a live Petri Net indicates that all the rules specified can be reachable. A bound Petri Net indicates that the resources specified in the model are sufficient to execute the KBS. A restartable Petri Net indicates that the KBS can back in its initial state, in a new execution cycle.

However, it is important to note that this properties can not be always desirable, according to the environment contingencies. In this way, it is important to execute many simulations to guarantee the accomplishment of the system requisites.

V. Conclusion

In this paper, it is presented an unified approach to model, specify and implement MAS in all its levels, including the agents itself. The approach proposed is independent of knowledge representation formalisms, underlying architectures and has yet tools to analyse and verify the system implemented. As seen in section II, there is not any work which use Petri nets as a framework to build MAS that integrate all development levels.

The approach proposed is based on the clear separation between the knowledge level of the system and the inference mechanisms that use it. In this sense, the system is composed by a set of KBS, with a formal and hierarchical relation between them. This is done through a model composed by a high-level Petri Net with support to hierarchy and an underlying language that permits to manipulate organizational structure in a social context and knowledge representations in a individual context.

A comparison between the proposed approach and the models that exist in the literature concludes about some contributions as:

- An integrate approach that consider both levels, the social and individual one, through formal mechanisms that guarantee the correct relation between the organizational structure and the knowledge representation formalisms used;
- A language that permits to manipulate different knowledge representation mechanisms, and not only the logic ones;
- A model capable to manipulate complex systems formed by KBS through modularize process that consider hierarchic mechanisms.

Currently, it is under development a computational tool based on this approach. In its first version, the software generate a knowledge-base in XML format. The reason is to use this format as a middle language between the tool and the system which it will use it, when it is not possible to use a direct representation.

References

[1] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Pub Co., 1999, iSBN:0201360489.

[2] E. H. Durfee and J. S. Rosenschein, "Distributed problem solving and multi-agent systems: Comparisons and examples," in *International Workshop on Distributed Artificial Intelligence*, May 1994.

[3] S. Russel and P. Norvig, *Artificial Intelligence, A Modern Approach*. Alan Apt, 1995.

[4] J. S. Sichman, Y. Demazeau, and O. Boisser, "When can knowledge-based systems be called agents?" in *Anais do IX Seminário Brasileiro de Inteligência Artificial*, October 1992, pp. 172–185.

[5] D. Xu, R. A. Volz, and J. Yen, "Modeling and analysing multi-agent behaviors using predicate/transitions nets," *International Journal of Software Engineering and Knowledge Engineering*, vol. 13, no. 1, pp. 103–124, 2003.

[6] D. Xu, R. A. Volz, and T. R. Ioerger, "Generating parallel based on planning graph analysis of predicate/transition nets," in *Proceedings of the 2002 International Conference on Artificial Intelligence (IC-AI'02)*, June 2002, pp. 440–446.

[7] A. L. Blum and M. L. Furst, "Fast planning through planning graph analysis," in *IJCAI-95*. Morgan Kaufmann, 1995, pp. 1636–1642.

[8] T. Holvoet and P. Verbaeten, "Synchronization specifications for agents with net-based behavior description," in *Proceedings of Conference, Symposium on Discrete Events and Manufacturing Systems*, Lille, France, July 1996, pp. 613–619.

[9] S. Jindian, G. Heqing, and Y. Shanshan, "A coloured petri net model for composite behaviors in multi-agent system," in *Proceedings of IEEE Conference on Cybernetics and Intelligent Systems*, 2008, pp. 677 – 680.

[10] M. V. C. Miranda and A. Perkusich, "Modeling and analysis of a multi-agent system using colored petri nets," in *Proceedings of the Workshop on Application of Petri Nets to Intelligent System Development*, L. Portinale, R. Valette, and D. Zhang, Eds., 1999, pp. 59–70.

[11] E. de Barros Costa and A. Perkusich, "Modeling the cooperative interactions in a teaching/learning situation," in *Proceedings of The Intelligent Tutoring Systems (ITS-96)*, Montreal, Canada, June 1996.

[12] H. O. de Almeida, L. D. da Silva, A. Perkusich, and E. de Barros Costa, "A formal approach for the modelling and verification of multiagent plans based on model checking and petri nets," in *Software Engineering for Multi-Agent Systems III: Research Issues and Pratical Applications*, R. Choren, A. Garcia, C. Lucena, and A. Romanovsky, Eds. Springer-Verlag GmbH, February 2005, vol. 3390/2005, iSSN: 0302-9743.

[13] D. L. Nazareth, "Investigating the applicability of petri nets for rule-based system verification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 4, no. 3, pp. 402–415, June 1993.

[14] D. Zhang and D. Nguyen, "Prepare: A tool for knowledge base verification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 6, no. 6, December 1994.

[15] C.-H. Wu and S.-J. Lee, "Enhanced high-level petri nets with multiple colors for knowledge verification/validation of rule-based expert systems," *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics*, vol. 27, no. 5, pp. 760–773, October 1997.

[16] X. Li and W. Yu, "Object oriented fuzzy petri net for complex knowledge system modeling," in *Proceedings of the 2001 IEEE International Conference on Control Applications*, Mexico City, Mexico, September, 5-7 2001, pp. 476–481.

[17] C. G. Looney, "Fuzzy petri nets for rule-based decisionmaking," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, January/February 1988.
[18] S. M. Koriem, "A fuzzy petri net tool for modeling and verification of knowledge-based systems," *The Computer Jornal*, vol. 43, no. 3, 2000.
[19] N. K. Liu, "A formal description technique for the verification of fuzzy knowledge base redundancy and subsumption," in *Proceedings of First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, Dunedin, New Zealand, November, 24-26 1993, pp. 142–145.
[20] X. Li, W. Yu, and F. Lara-Rosano, "Dynamic knowledge inference and learning under adaptive fuzzy petri net framework," *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, vol. 30, no. 4, pp. 442–450, November 2000.
[21] S.-M. Chen, J.-S. Ke, and J.-F. Chang, "Knowledge representation using fuzzy petri nets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 2, no. 3, pp. 311–319, September 1990.
[22] J. Lee and L. F. Lai, "A high-level petri nets-based approach to verifying task structures," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 2, pp. 316–335, March/April 2002.
[23] J. Lee, "Task structures as a basis for modeling knowledge-based systems," *International Journal of Intelligent Systems*, vol. 12, pp. 167–190, March 1997.
[24] M. L. Goc, C. Frydman, and L. Torres, "Verification and validation of the sachem conceptual model," *International Journal Human-Computer Studies*, vol. 56, pp. 199–223, 2002.
[25] A. Newell, "The knowledge level," *Artificial Intelligence*, vol. 18, no. 1, pp. 87–127, 1982.
[26] J. F. Hubner, J. S. Sichman, and O. Boissier, "Moise$^+$: Towards a structural, functional, and deontic model for mas organization," in *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*. ACM Press, 2002, pp. 501–502.
[27] J. F. Hübner, R. H. Bordini, and R. Vieira, "Introdução ao desenvolvimento de sistemas multiagentes com jason," in *Anais a XII Escola Regional de Informática - SBC*, 2004.
[28] C. Sibertin-Blan, "High-level Petri nets with data structures," in *European Workshop on Application and Theory of Petri net*, Helsinki, Finland, jun 1985, pp. 141–170.
[29] K. Jensen, "Coloured petri nets: A high level language for system design and analysis," in *Advances in Petri Nets 1990*, G. Rozemberg, Ed. Springer-Verlag, November 1990, iSSN 0105-8517.
[30] E. M. N. Gonçalves, "An approach to specify knowledge in multi-agent systems using petri nets," in *Network and System Security (NSS), 2010 4th International Conference on*, sept. 2010, pp. 456–461.
[31] M. Chen, E. Foroughi, F. Heintz, Z. X. Huang, S. Kapetanakis, K. Kostiadis, I. N. Johan Kummeneje, O. Obst, P. Riley, Y. W. Timo Steffens, and X. Yin, *RoboCup Soccer Server: for Soccer Server Version 7.07 and later*, May 2001, www.robocup.org.
[32] A. L. da Costa and G. Bittencourt, "From a concurrent architecture to a concurrent autonomous agents architecture," in *International Joint Conference on Artificial Intelligence (IJCAI'99)*, 1999.

Eder Mateus Gonçalves was born in Rio Grande - RS, Brazil. He received his PhD degree in electrical engineering from the Federal University of Santa Catarina (UFSC), Florianpolis, Brazil, in 2006, his MS degree in electrical engineering also from the Federal University of Santa Catarina (UFSC) in 2001, and his BS degree in electrical engineering from the Catholic University of Pelotas, Pelotas-RS, in 1998.

He is currently an Adjunct Professor of Center for Computational Sciences - C3, in Federal University of Rio Grande (FURG), Rio Grande-RS, Brazil. His current research interests include agents, multi-agent systems, discrete event-driven systems, petri nets and robotics.