REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministry of Higher Education and Scientific Research

University Mohamed khider – BISKRA

Faculty of Exact Sciences and Sciences of Nature and Life

Computer Science department

Ordre :.....

Serie :.....

Project

Presented for the diploma of

Master in Computer Science

Option: **Software Engineering and Distributed Systems**

Title of Project :

Application of the P-graph Methodology for Organization-Based Multi Agent System Designs , A graph Transformation approach

Presented on / /

By **NOUAR KHERKHACHI HOUSSAM**

Members of the jury :

	President	
Mr.	**Rapporteur**	
	Examinateur	BISKRA

Contents

List of Figures

List of Tables

General Introduction

Multiagent systems have become popular over the last few years for building complex, adaptive systems. The problem is that many multiagent systems are typically designed to work within a limited set of options. Even when the system possesses the resources and computational power to accomplish its goal, it may be constrained by its own structure and knowledge of its members capabilities. To overcome these problems, The framework OMACS is developed to allows the system, design its own organization at runtime [1].

OMACS defines the knowledge needed about a systems structure and capabilities to allow it to reorganize at runtime in the face of a changing environment and its agents capabilities [1] [2].

OMACS is a framework, that allow us to define a Multi-agent System with his component (Agents, Roles, Capabilities, Goals) and the relation between these components, by an oriented graph. Each component represent a node, and each relation represent an edge.

PNS is also an other framework basically developed to design a chemical reaction systems, represented by biparte graph, each node in this Graph represent a material and between two material there is a transition they called it operating unit, in our work we will use it to represent a multi agent system.

PNS come with three Algorithm to extract information from the PNS model. and we need these algorithm in our project, because of that we add this framework to our work.

The aim of our project is to find the best organization, optimum structure of a multi agent system by applying one of these algorithms, because of that we need to propose a new approach of model-to-model transformation, and this later allow us to transform OMACS model into PNS model. Our approach is based on graph transformation in AToM3 Tool.

Our document is organized as follows :

The first and the second chapter we presente and illustrate the OMACS, PNS frame-

works, and we use it to modeling multi-agent system. The third chapter cover some concept of model-to-model transformation, then we represente the tools AToM3, we use to create the formalism of these frameworks and the second PGraph-Studio which have the algorithm to apply on pns models. The fourth chapter explain our approach of transformation. Finaly this document end with general conclusion which is a collection of the main idea in this document.

CHAPTER 1

Multi-Agent System Framework

1.1 Introduction

Designing and implementing large, complex, and distributed systems by resorting to autonomous or semi-autonomous agents that can reorganize themselves by cooperating with one another represent the future of software systems [3].

This chapter introduces two framework for building complex, adaptive systems. It is based on the multi-agent systems paradigm and uses the Organization Model for Adaptive Computational Systems (OMACS). And presents a suite of Methodes including the Organization-based Multiagent Systems Engineering (O-MaSE) methodology [1].

1.2 Motivations to MaS Organization

Basically, a MAS is formed by the collection of autonomous agents situated in a certain environment, respond to their environment dynamic changes, interact with other agents, and persist to achieve their own goals or the global system goals. There are two viewpoints of MAS engineering, the first one is the agent-centered MAS (ACMAS) in which the focus is given to individual agents. With this viewpoint, the designer concerns the local behaviors of agents and also their interactions without concerning the global structure of the system [4].

The global required function of the system is supposed to emerge as a result of the lower level individual agents interactions in a bottom-up way Picard et al. stated that the agent-centered approach takes the agents as the engine for the system organization [4].

and agent organizations implicitly exist as observable emergent phenomena, which states a unified bottom-up and objective global view of the pattern of cooperation between agents. Further, Picard gives the ant colony as an example, where there is no

organizational behavior and constraints are explicitly and directly defined inside the ants. The main idea is that the organization is the result of the collective emergent behavior due to how agents act their individual behaviors and interact in a common shared and dynamic environment [4].

1.2.1 Organization

Each organization has an implicitly defined organization transition function that describes how the organization may transition from one organizational state to another over the lifetime of the organization [2].

Since agents in an organization as well as their individual capabilities may change over time, this function cannot be predefined, but must be computed based on the current state, the goal set, G, and the current policies. In our present research with purely autonomous systems, we have only considered reorganization that involves the state of the organization.

However, we have defined two distinct types of reorganization: state reorganization, which only allows the modification of the organization state, and structure reorganization, which allows modification of the organization structure (and may require state reorganization to keep the organization consistent).

We define the state of the organization as the set of agents, A, the possesses, capable, and potential functions, and the assignment set, φ. However, not all these components may actually be under the control of the organization. For our purposes, we assume that agents may enter or leave organizations or relationships, but that these actions are triggers that cause reorganizations and are not the result of reorganizations.

Likewise, possesses (and thus capable and potential as well) is an automatic calculation that determines the possible assignments of agents to roles and goals in the organization. The calculation of possesses is the only calculation totally controlled by the agent, the organization can only use this information in deciding how to make assignments. This leaves one element that can be modified via state reorganization φ [2].

1.2.2 Reorganization

Reorganization is the process of changing the assignments of agents to roles to goals as specified in φ [2].

The organizations oaf function is used to determine the best new φ however, total reorganization may not be necessary or efficient.

1.2.2.1 Goal Set Changes

Any change in G may cause reorganization. There are three basic types of events that can cause a change in G:

1. insertion of a new goal

2. goal achievement

3. goal failure

The first situation deals with new goals being added to G. However, [2] we cannot say with certainty that reorganization will occur based on a new goal in G.

It is possible that the organization will choose to forego reorganization for a number of reasons, the most likely being that it has simply chosen not to pursue any new goals added to G at the present time.

The second case deals with goal achievement. When a goal g is achieved, G is changed to reflect that event by

- removing g from G
- possibly adding new goals

which are enabled by the achievement of g, into G. Obviously, the agent assigned to achieve goal g is now free to pursue other goals.

The third instance involves goal failure, which really has two forms: agent-goal failure and goal failure.

When a specific agent cannot achieve goal g but g might still be achievable by some other agent, agent-goal failure occurs.

When agent-goal failure occurs, reorganization must occur to allow the organization to :

- choose another agent to achieve g
- not pursue g at the current time
- choose another goal to pursue instead of g

In any of these situations, g is not removed from G since it has not been achieved. In the case where the organization or the environment has changed such that a goal g can never be achieved, then goal failure occurs.

In this case, g is removed from G and the organization must attempt to assess whether it can still achieve the overall system goals. Reorganization may occur to see if the agent assigned to achieve g can be used elsewhere.

In all cases, the selection of the appropriate strategy is left to the organization [2].

1.2.2.2 Agent Changes

The second type of change that triggers reorganizations are changes to the set of agents, A, or their individual capabilities [2].

When an agent that is part of φ is removed from the organization, a reorganization must occur, even if only to remove the agent and its assignment(s) in φ.

In general, when changes occur in an agents capability, reorganization may or may not be necessary, based on the agents capable relation.

We have identified four specific types of changes in an agents capabilities that may indicate a need for reorganization:

1. when an agent gains the ability to play a new role
2. when an agent loses the ability to play a role
3. when an agent increases its ability to play a specific role
4. when an agent decreases its ability to play a specific role.

While case 2 requires reorganization if the agent is currently assigned to play the role for which it no longer has the capability to play, whether or not to reorganize is left up to theorganization when the other three cases (along with 2 when the agent is not currently assigned that role) occur [2].

1.2.2.3 Reorganization Triggers

There are a variety of events that may occur in the lifecycle of a multiagent team that may require it to reorganize. In general, reorganization is initiated when an event occurs such that the team [5]:

- has reached a goal or subgoal
- is no longer capable of reaching its overall goal
- realizes that it could reach its goal in a more efficient or effective manner

When the team is no longer capable of reaching its overall goal, we call this a goal failure. We have currently identified three role-related goal failure scenarios [5]:

1. When a required role has not been assigned
2. When an agent relinquishes some required role
3. When an agent suffers a failure that keeps it from accomplishing its role

1.3 Organization Multi-Agent System Engineering

This section introduce the definition of O-MASE framework, and the goal of this framework.

1.3.1 Overview of O-MaSE

The Organization-based Multiagent System Engineering (O-MaSE) is a framework that allows designers to create custom agent-oriented development processes.

This custom agent-oriented process is generated following a process metamodel and then instantiated from a set of method fragments and guidelines by using a method engineering approach.

O-MaSE defines a metamodel, a repository of method fragments and a set of guidelines. The O-MaSE metamodel defines general concepts used in multiagent systems along with their relationships and is based on an organizational approach [6].

The O-MaSE Process Framework is based on the OPEN Process Framework (OPF) and uses the OPF metamodel, as shown in Figure 1.1 [1].

1. level M2 : which defines processes in terms of Work Units (Activities, Tasks, and Techniques), Producers, and Work Products.
2. Level M1 : contains the definition of O-MaSE in the form of the O-MaSE metamodel, method fragments, and guidelines.
3. Level M0 : level for specific projects (a process instance).

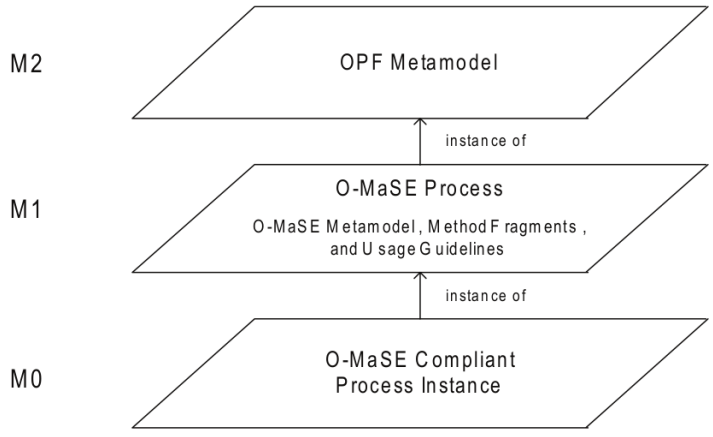


Figure 1.1: O-MaSE Process Framework [1]

1.3.2 The Goal of O-MaSE

The goal of the O-MaSE methodology framework is to allow method engineers to build custom agent-oriented methods using a set of method fragments, all of which are based on a common meta-model [7].

To achieve this, O-MaSE is defined in terms of a meta-model, a set of method fragments, and a set of method construction guidelines.

The O-MaSE meta-model defines a set of analysis, design, and implementation concepts and a set of constraints between them.

The method fragments define a set of work products, a set of activities that produce work products, and the performers of those activities.

And the method construction guidelines define how the method fragments may be combined to create O-MaSE compliant methods.

The O-MaSE methodology is supported by the agentTool III [1] development environment 2, which is designed as a set of Eclipse plug-ins. agentTool includes a plug-in for each O-MaSE model and the agentTool Process Editor (APE), which was developed to support the design and definition of O-MaSE compliant processes [7].

The APE plug-in is based on the Eclipse Process Framework and provides a process designer the ability to :

1. extend O-MaSE with new tasks, models, or usage guidelines
2. create new process instances by composing tasks, models, and producers from the O-MaSE method fragment library
3. verifying that they meet process guidelines

1.4 Organization Model for Adaptive Computational System

We represent in this section the OMACS Framwork, As an Adaptive computational System, Also we cite the main elements of this framework and the relation between these elements.

1.4.1 Overview of OMACS

The OMACS is Framework, this model grew from the MaSE metamodel, which was based on the original AGR[2] model [1].

Noting that agents could be assigned to roles based on the capabilities Required to play various roles and the capabilities possessed by the agents, we figured the agents could adapt their assignments based on the current

set of goals required to be achieved by the system. This basic idea led to the OMACS model as defined in DeLoach, Oyenan, and Matson (2007) and shown in Figure 1.2 [1].

OMACS defines an organization as a tuple O = $\langle$ G, R, A, C, Φ, P, $\sum$, oaf, achieves, requires, possesses $\rangle$ where :

[1] http://agenttool.cs.ksu.edu/
[2] Agent Group Roles

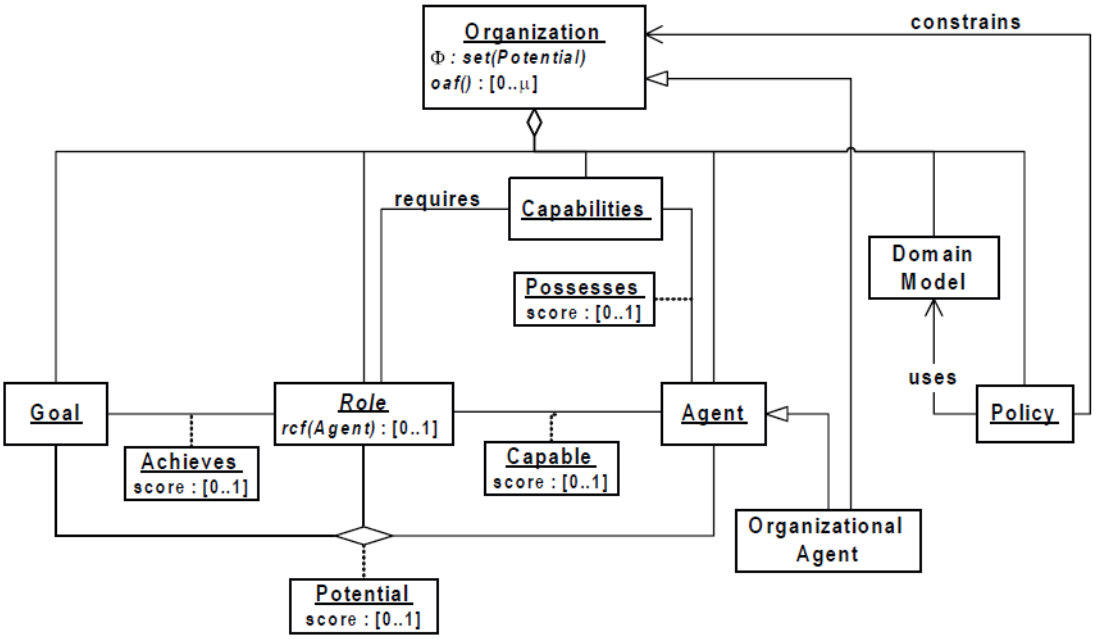


Figure 1.2: OMACS meta-models [3]

G : goals of the organization

R : set of roles defines a set of roles (i.e., positions within an organization whose behavior is expected to achieve a particular goal or set of goals)

A : is a set of agents, which can be either human or artificial (hardware or software) entities that perceive their environment

C : set of capabilities which define the percepts/actions at their disposal. Capabilities can be soft (i.e. algorithms or plans) or hard (i.e., hardware related actions)

Φ : relation over G - R - A defining the current set of agent/role/goal assignments

P : set of constraints on Φ formally specifies rules that describe how O OMACS may or may not behave in particular situations

$\sum$: domain model used to specify objects in the environment, their inter-relationships

achieves : function $G \times R \rightarrow [0..1]$ a function whose arguments are a goal in G as well as a role in R that generates an output which is a positive real number greater than or equal to 0 and less than or equal to 1, defining how effective the behavior defined by the role could be in the pursuit of the specified goal(the extent of achievement of a goal by a role)

requires : function $R \rightarrow P(C)$ a function that assumes a role in R, thereby yielding a set of capabilities required to play that role defining the set of capabilities required to play a role 1

possesses : function A × C → [0..1] a function with an agent in A and a capability in C as inputs yields a positive real number in the range of [0,1] defining the quality of an agents capability

OMACS also includes two additional derived functions to help compute potential assignment values: capable and potential. [3]

capable : function A × R → [0..1] function whose inputs are an agent in A OMACS and a role in R OMACS and generates an output, which is a positive real number greater than or equal to 0 and less than or equal to 1 defining how well an agent can play a role (computed based on requires and possesses)

$$\begin{cases} 0 & \text{if} \prod_{c \subset require(r)} possesses(a,c) = 0 \\ \frac{\sum_{c \subset require(r)} possesses(a,c)}{|require(r)|} & \text{else} \end{cases} \tag{1.1}$$

potential : function A × R × G → [0..1] a function with an agent in A OMACS, a role in R, and a goal in G as inputs yields a positive real number in the range of [0..1], thus yielding

$$potential(a,r,g) = achieves(r,g) \times capable(a,r) \tag{1.2}$$

defining how well an agent can play a role to achieve a goal (computed based on capable and achieves)

OAF : function (G × R × A) → [0.. ∞] the selection of φ from the set of potential assignments is defined by the organizations reorganization function, oaf, that assumes a set of assignments in φ, thereby yielding a positive real number in the range of [0.. ∞] defining quality of a proposed assignment set

$$OAF = \sum_{(a,r,g) \in \Phi} potential(a,r,g) \tag{1.3}$$

1.4.2 Main Element of OMACS

The first eight elements in the organization tuple defined above G, R, A, C, Φ, P, $\sum$ and oaf constitute the main elements of the OMACS model as depicted in Figure 1.2.

1.4.2.1 Goals

Artificial organizations are designed with a specific purpose, which defines the overall function of the organization [2].

Goals are defined as a desirable situation or the objective of a computational process. Within OMACS, each organization has a set of goals, G, that it seeks to achieve.

OMACS makes no assumptions about these goals except that they can be assigned to individual agents and individual agents have the ability to achieve them independently [2].

Figure 1.3: Goal Node

1.4.2.2 Roles

Within OMACS, each organization contains a set of roles (R) that it can use to achieve its goals. A role defines a position within an organization whose behavior is expected to achieve a particular goal or set of goals [2].

Thus, each role defines a set of responsibilities. Roles are analogous to roles played by actors in a play or by members of a typical corporate structure. A typical corporation has roles such as president, vice-president, and mail clerk.

Each role has specific responsibilities, rights and relationships defined in order to help the corporation perform various functions towards achieving its overall goal.

Specific people (agents) are assigned to fill those roles and carry out the roles responsibilities using the rights and relationships defined for that role [2].

OMACS roles consist of a name and a role capability function, rcf. Each role, r $\in$ R, is a tuple $\langle$ name, rcf[3] $\rangle$ where A $\times$ R $\rightarrow$ [0..1] .

The rcf is defined at design time for each role and computed in terms of the capabilities required to play that role. A default rcf (as shown in 1.1) would assume that all the capabilities required to play a role r are equally important, essentially taking the average of all the required possesses values (possesses(a,c)) (with the stipulation that none of those possesses scores are 0) [1].

Figure 1.4: Role Node

1.4.2.3 Agents

OMACS also includes a set of heterogeneous agents (A) in each organization. As described by Russell and Norvig, an agent is an entity that perceives and can perform actions upon its environment, which includes humans as well as artificial (hardware or software) entities [2].

For our purposes, we define agents as computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals. Thus, we assume that agents exhibit the attributes of autonomy,

[3]the same capable function

reactivity, pro-activity, and social ability. Autonomy is the ability of agents to control their actions and internal state.

Reactivity is an agents ability to perceive its environment and respond to changes in it, whereas pro-activeness ensures agents do not simply react to their environment, but that they are able to take the initiative in achieving their goals.

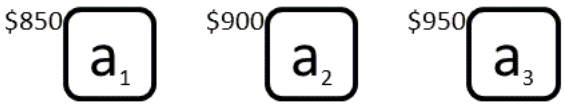


Figure 1.5: Agent Node

Finally, social ability allows agents to interact with other agents, and possibly humans, either directly via communication or indirectly through the environment.

Within the organization, agents must have the ability to communicate with each other, accept assignments to play roles that match their capabilities, and work to achieve their assigned goals [2].

1.4.2.4 Capabilities

Capabilities are the key to determining exactly which agents can be assigned to which roles within the organization. Capabilities are atomic entities used to define a skill or capacity of agents [2].

Capabilities can be used to capture soft abilities such as the access to control over specific resources, the ability to communicate with other agents, the ability to migrate to a new platform, or the ability to carry out plans to achieve specific goals.

Capabilities also capture the notion of hard capabilities that are often associated with hardware agents such as robots.

These hard capabilities are generally described as sensors, which allow the agent to perceive a real world environment, and effectors, which allow the agent to act upon a real world environment [2].

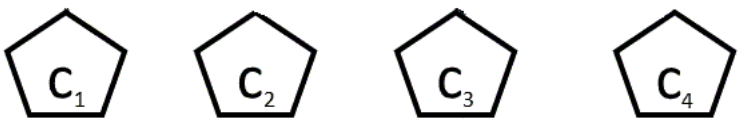


Figure 1.6: Capabilities Node

$$\forall\ c : C\ (\ a : A\ c\ \in \text{possese}(a,c)\ \geq 1 \vee\ r : R\ c\ \in \text{requires}(r)) \tag{1.4}$$

1.4.2.5 The Tuple φ

An assignment set φ is the set of agent-role-goal tuples $\langle$ a,r,g $\rangle$, that indicate that agent a has been as-signed to play role r in order to achieve goal g [1].

φ is a subset of all the potential assignments of agents to play roles to achieve goals. This set of potential assignments is captured by the potential function (see Equation 1.2), which maps each agent-role-goal tuple to a real value ranging from 0 to 1 representing the ability of an agent to play a role in order to achieve a specific goal.

If $\langle$a,r,g $\rangle \in \varphi$, then agent a has been assigned by the organization to play role r in order to achieve goal g [1].

The only inherent constraints on φ is that it must contain only assignments whose potential value is greater than zero (Equation 1.5) and that only one agent may be assigned to achieve a goal at a time (Equation 1.6)

$$\Phi \subseteq \{(a,r,g) \mid a \in A \land r \in R \land g \in G \land \text{potential}(a,r,g) \geq 0\} \tag{1.5}$$

$$\forall a1,a2{:}A\ r1,r2{:}R\ g1,g2{:}G\ (a1,r1,g1) \in \Phi \land (a2\ r2\ g2) \in \Phi \land a1{=}a2 \tag{1.6}$$

1.4.2.6 OAF

In order to select the best set of assignments to maximize an organizations ability to achieve its goals, OMACS defines an organizational assignment function, or oaf, which is a function over the current assignment set, oaf: $\varphi \rightarrow 0..\infty$. As with the rcf, the selection of assignments may be application specific [1].

Thus, each organization has its own application specific organization assignment function, oaf, which computes the goodness of the organization based on φ.

As with the rcf, we can define a default oaf, which is simply the sum of the potential scores in the current assignment set φ.

1.4.2.7 Domain Model

The domain model $\sum$, is used to define object types in the environment and the relations between those types [2].

The domain model is based on traditional object oriented class diagrams. They include object classes that each have a set of attribute types.

Relations between object classes include general purpose associations as well as generalization-specialization and aggregation.

Relations may also include multiplicities to constrain the number of object classes participating in any given relation [2].

1.4.3 Function of OMACS

There are three major relations/functions and two derived functions between the eight main elements that provide the power of the OAMCS model: achieves, requires, possesses, capable, and potential.

1.4.3.1 The Achieves

function (although somewhat confusingly named) actually defines how effective an agent could be while playing that role in the pursuit of a specific goal. For instance, if one role requires more resources or better capabilities, it can use a different approach and thus yield better results than a second role that requires fewer resources or capabilities [1].

Providing two different roles to achieve the same goal may provide the organization flexibility in deciding how to actually achieve a given goal.

The value of achieves can be predefined by the organization designer or learned before or during system operation (Odell, Nodine & Levy, 2005). Thus, the OMACS achieves function formally captures the effective-ness of a role in achieving a specific goal by defining a total function from the R $\times$ G to a real value in the range of 0 to 1, achieves: G $\times$ R $\rightarrow$ [0..1].

Thus, by definition, a role that cannot be used to achieve a particular goal must have an achieves value of 0, while a role that can achieve a goal would have an achieves value greater than zero [1].

Figure 1.7 represent the Relation between Role and Goal

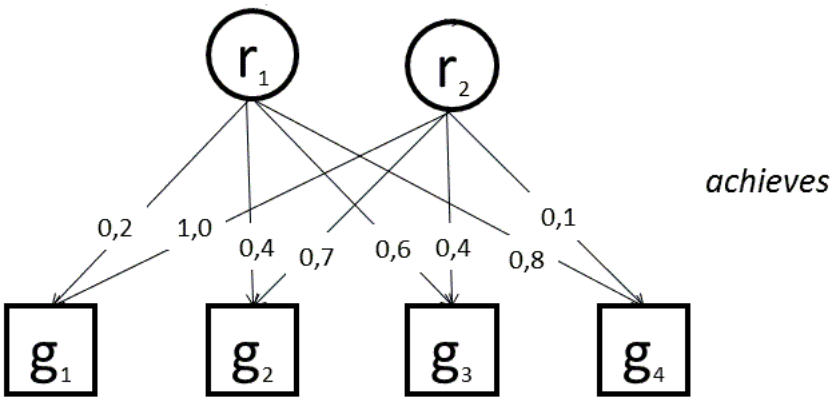


Figure 1.7: The Achivement Relation

1.4.3.2 Requires

In order to perform a particular role, agents must possess a sufficient set of capabilities that allow the agent to carry out the role and achieve its assigned goals [2].

For instance, to play the president role, a person would be expected to have knowledge of the corporations domain, experience in lower-level jobs in similar types of companies, and experience in managing people and resources an artificial organization is no different.

Roles require a certain set of capabilities while agents possess a set of capabilities [2]. and this Figure 1.8 represent the Relation between Role and Capabilities.

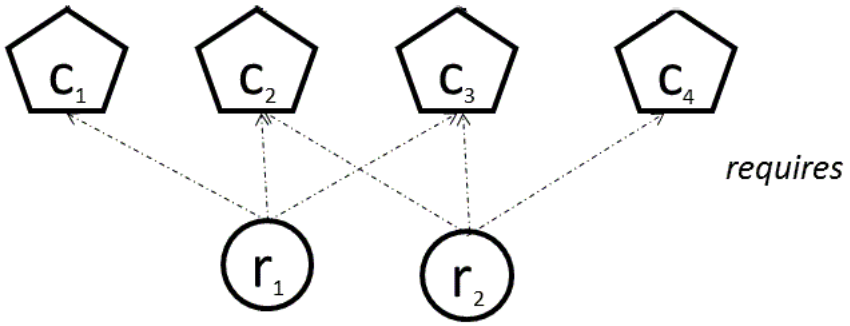


Figure 1.8: The Requirement Relation

1.4.3.3 Possesses

In order to determine if some agent has the appropriate set of capabilities to play a given role, OMACS defines a similar relation, the possesses relation, that captures the capabilities a specific agents actually possesses [1].

The possesses relation is formally captured as a function over agents and capabilities that returns a value in the range of 0 to 1,

possesses: A × C → [0..1]. The real value returned by the possesses function indicates the quality of each capability possessed by the agent 0 indicates no capability while a 1 indicates a high quality capability [1]. and this Relation represented in the next Figure 1.9

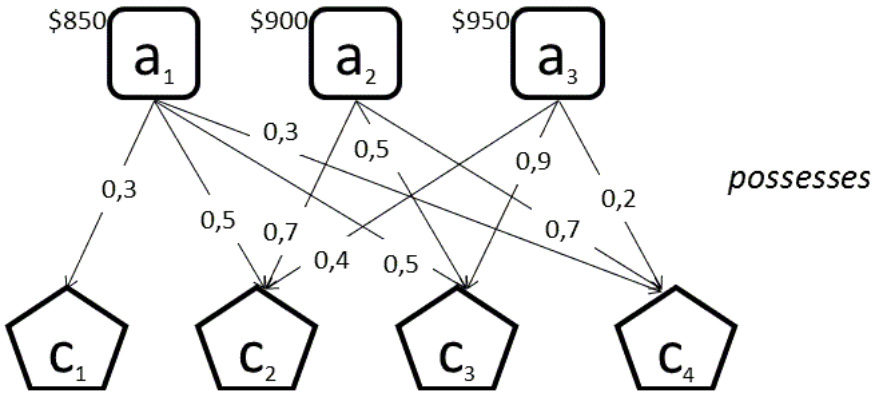


Figure 1.9: Possesses Relation

1.4.3.4 Capable

Using the capabilities required by a particular role and capabilities possessed by a given agent, we can compute the ability of an agent to play a given role, which we capture in the capable function. The capable function returns a value from 0 to 1 based on how well a given agent may play a specific role, capable: A × R → [0..1] [2].

Since the capability of an agent, a, to play a specific role r, application and role specific, OMACS provides the rcf defined in the previous section that controls how this value is computed. Thus, the capable score of an agent playing a particular role is defined via the designer defined rcf of each role.

$$\forall a : A \text{ r:R} capable(a, r) = r.rcf(a) \tag{1.7}$$

While the rcf is user defined, it must conform to one OMACS constraint. To be capable of playing a given role in the current organization, an agent must possess all the capabilities that are required by that role [2].

$$\forall a : A, r : R capable(a, r) > 0 \Leftrightarrow requires(r) \subseteq c | possesses(a, c) > 0 \tag{1.8}$$

The main goal of OMACS is to provide a mechanism to assign goals to agents [2], in such a way that agents cooperate toward achieving some top-level goal. Intuitively, this mechanism should provide a way to assign the best agents to play the best roles in order to achieve these goals.

Thus, OMACS has defined a potential function that captures the ability of an agent to play a role in order to achieve a specific goal [2].

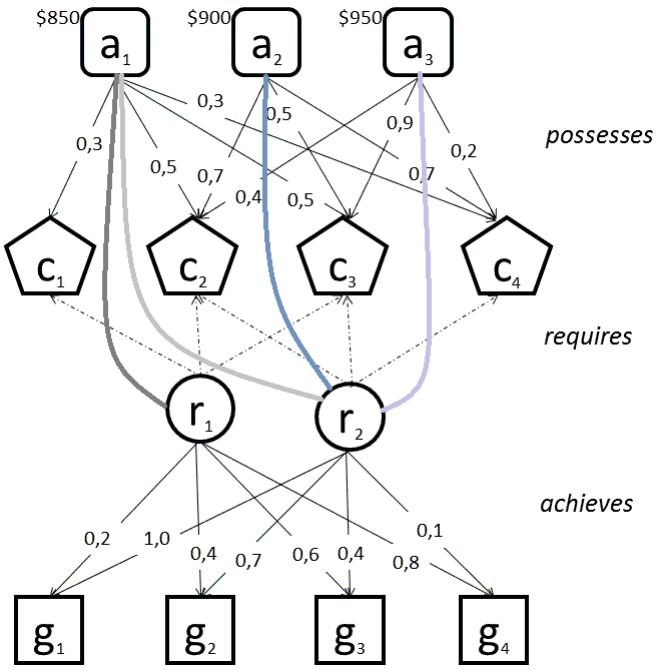


Figure 1.10: MaS with Capable of Playing Relation [3]

1.4.3.5 Potential

The potential function maps each agent-role-goal tuple to a real value ranging from 0 to 1, potential: A × R × G → [0..1] [1].

Here, a 0 indicates that the agent-role-goal tuple cannot be used to achieve the goal while a non-zero value indicates how well an agent can play a specific role in order to achieve a goal.

The potential of agent a to play role r to achieve goal g is defined by combining the capable and achieves functions [1].

$$\forall a : A \text{ r:R g:G} potential(a, r, g) = achieves(r, g) * capable(a, r) \tag{1.9}$$

and we depend on the common capabilities between the Agent and the Role Entities to Create a relation in this 1.10

1.5 Conclusion

This chapter describes how to design adaptive multi agent systems using an organizational model, which defines the entities and relationships of a typical organization. The major elements of the model consist of goals, roles, agents, capabilities, and the relationships between them. By designing a system using the model, and we focus on OMACS framework because it handling reorganization system more then other framework.

CHAPTER 2

PGraph and PNS

2.1 Introduction

In a process system, raw materials are consumed through various transformation to yield desired products. These transformations are carried out are termed operating units of the process, a given set of operating units with the plausible interconnection can be described by a network [3]. This chapter introduces a framework for designing a processes systems, it based on the transformation process between materials.

2.2 PGraph

The so called P graph (Process graph), which is a directed bigraph, has been used for modelling network structures for some time.

The vertices of the graph denote the operating units (O operating units) and the materials (M materials). The edges of the graph represent the material-flow between the materials and the operating units [8, 9].

2.2.1 Definition of P-Graph

The Pgraph is a bigraph, meaning that its vertices are in disjunctive sets and there are no edges between vertices in the same set.In case of P graphs the assignment of operating units and materials are strictly determined by the tasks given, i.e. an edge can point to an M material type vertex from an O operating unit type vertex, only if M is element of the output set of O, that is O produces M material namely, M $\in$ output O [8]. An edge can point from an M material type vertex to an O operating unit type vertex, only if M is element of the input set of O, that is O processes M material, namely M $\in$ input

O. Thus, the P-graph can be presented by the pairs of operating unit and the assigned material vertices set like the (M,O) P-graph [8].

The material type vertices can be put into several subsets. There are various subsets like the raw-material type one, which contains the input elements of the whole process, the product-material type subset, which gathers the results of the entire process, the intermediate-material type one, the elements of which emerge or are used between the processing phases [8], And finally the by-product-material type set, which contains the non desired results of the process. The applied operating unit and material element notations, in the P-graph notation are presented in figure 2.2. As an example let us consider a process network with 7 operating units, in which the operating units are 1,2....7 and the materials are A,B, ... L. A,B,C and D are the materials available for the production of L The possible structure is given in [8, 9].

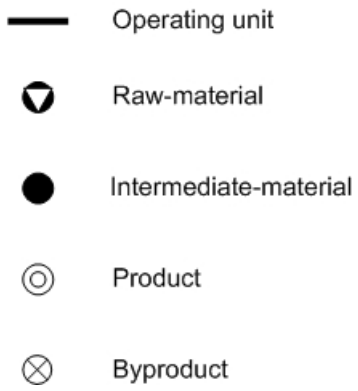


Figure 2.1: notation in pgraph [8]

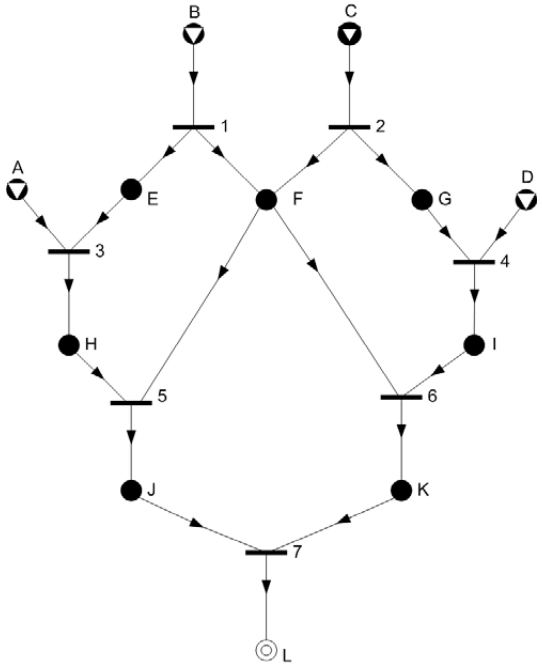


Figure 2.2: example for P-graph [8]

2.3 Process Network Synthesis

In the PNS problem a set of materials is given and also operating units which are transforming some subset of materials into some other subset. The subsets assigned to the operating unit are called its input and output materials [10, 11].

The PNS problem, two subsets of the materials are distinguished, one is the set of the raw materials and the other is the set of the desired products. Our goal is to find a minimal cost network of the operating units which can produce all desired products starting from the raw material. These systems can be modeled in the P-graph framework which is based on bipartite graphs [10, 12].

2.3.1 Basic Notation

The structural PNS problem can be modeled in the PGraph framework. In the PGraph (Process Graph) we have the set of the materials denoted by M, which contain two special subsets, the set of raw materials and the set of desired products denoted by R and P respectively [10].

The problem also contains a set of possible operating units which can transform some sets of materials.

The set of operating units is denoted by O. An operating unit u is given by two sets, in(u) denotes the set of the input materials out(u) denotes the set of output materials of the operating unit

This means that the operating unit can work in a solution structure if all of its input materials are produced and in this case it produces all of its output materials [10, 11].

The PGraph of the problem is defined by the sets M and O. It is a directed bipartite graph where the set of vertices is M ∪ O, and have the following two sets of edges:

1. Edges which connect the input materials to their operating unit
2. Edges which connect the operating units to their output materials

Then some of the subgraphs of this P-graph describe the feasible solutions which produce the required materials from the raw materials [10]. where m and o are the subsets of M and O, represent a feasible solution if and only if the following properties called axioms are valid:

1. m contain all element of P
2. a material from m is a raw material if and only if no edge goes into it in the P-graph (m, o)
3. For each operating unit u from o there exists a path in the P-graph (m,o) which goes into a desired product from u
4. m is the union of the input and output material sets of the operating units contained in set o

2.3.2 Mathematical definition

There is a finite set of material M (which contains the sets of P products and R raw-materials) and the finite set of O operating units [8]. Consequently, the set of P end-products and the set of R raw-materials must be subsets of M and the set of M materials and the set of O operating units are disjunctive. The basic relations between M,P,R and O are as follows :

$$P \subseteq M, R \subseteq M, M \cup O = \emptyset \tag{2.1}$$

As physical processes are defined, each operation unit produces output materials from input materials. Therefore two disjunctive sets can be assigned to each operating unit, i.e. the set of input and the set of output materials.

Let an arbitrary operating unit (α, β), then α is the set of input materials which are processed by the (α, β) unit and $\beta$is the set of output materials, which are produced by the given unit. Considering the process-network the output materials of each operating unit are the inputs of different operating units. In general, it can be proved that

$$O \subseteq \rho(M) \times \rho(M) \tag{2.2}$$

where O is the set of operating units, M is the set of materials and $\rho(M)$ is the power set, that is the set of subsets of M, and $\rho(M) \times \rho(M)$ represents the set of $\rho(M)$ and $\rho(M)$ pairs

Supposing that there is a finite set m, which is a subset of M, i.e. it is true that m $\subseteq$ M and there is an o finite set, which is a subset of O, i.e. it is true that o $\subseteq$ O and supposing that there is such a material which is an input for one or more operating units, and there is such material which is the output of one or more operating units, then :

$$o \subseteq \rho(m) \times \rho(m) \tag{2.3}$$

The PNS is defined as a bigraph, where the set of V vertices is made of the elements of the union of m and o that is

$$V = m \cup o \tag{2.4}$$

2.3.3 Algorithms MSG, SSG, and ABB

PNS representation of a process network and the set of axioms for solution structures, i.e., combinatorial feasible networks, render it possible to fashion the three mathematically rigorous algorithms: MSG, SSG, and ABB. [3] [13]

The algorithm MSG (Maximal-Structure Generation) generates the maximal structure (super-structure) of a process synthesis network. Also,

the algorithm SSG (Solution-Structure Generation) generates the set of feasible process structures from the maximal structure,

which leads to the algorithm ABB (Accelerated Branch and Bound) for computing the n-best optimal solution structure [3]

2.3.3.1 Maximal-Structure Generation

The maximal structure of the synthesis problem (P, R, O) contains all the combinatorially possible structures, which make the production of defined products possible from given raw-materials. [8] [13]

Therefore, it certainly contains the optimal structure as well.

The first phase is the input phase, in which the synthesis problem is defined (P, R, O) such a way, that the set of M all the plausible materials, the set of P end-products

The second phase is the elaboration of the input structure of the network, which is carried out by the linking of all the similar (same type) material type vertices.

The third phase is the elimination phase, where those materials and operating units are eliminated, which, taking the axioms into account, are not and cannot be linked to the maximal structure for sure

During the fourth phase the vertices are linked again from level to level, starting from the highest, the end-product level.

The maximal structure generated this way contains all the combinatorially possible structures and all of its elements fulfil the axioms [8, 13].

2.3.3.2 Solution Structure Generation

The maximal structure generated by the MSG algorithm contains all such combinatorically possible network structures that are able to produce the end-product from the given raw-materials [8, 13].

Consequently, it contains the optimal network as well. In most cases the optimalisation means to find the most cost effective solution.

The application of the SSG (Solution Structure Generation) algorithm enables the production of all the solution structures.

The SSG is a new mathematical tool which has been developed by Friedler et al [8,13].

2.3.3.3 Accelerated Branch and Bound

the branch-and-bound method has the advantages of being independent of an initial structure, ensuring the optimality provided that a bounding algorithm exists, and being capable of incorporating combinatorial algorithms [13]. it has been adopted for solving the routing and scheduling of evacuees, facing a life-threatening situation [9].

2.3.4 PNS and PetriNet :

The Table 2.1 represent the deference, and the common features, between these two mathematical tools, because the Petri is widely use more then PNS.

Petri Net	**Process Network Synthesis**
Source Place	Raw Material
Normal Place	Intermediate Material
Sink Place	Final Product
Token in Place	Requirement Flow
Transition	Operating Unit
Weight of in or out edges of the transition	producing rate of the operating unit
Modeling Parallel System	Basically use for Chemical Reaction

Table 2.1: Petri Net and PNS

2.4 PGraph Studio

PGraph Studio is a software that implements algorithms MSG, SSG, and ABB, and therefore, it is primarily used as a solver for process synthesis problems. Furthermore, it is also capable of constructing process synthesis models. As a modeling tool, it uses a tree-view that provides a clear overview of the actual problem under consideration and makes it possible to edit the properties of multiple materials and operating units in parallel. The handling of measurement units is aided with automated conversions [14, 15].

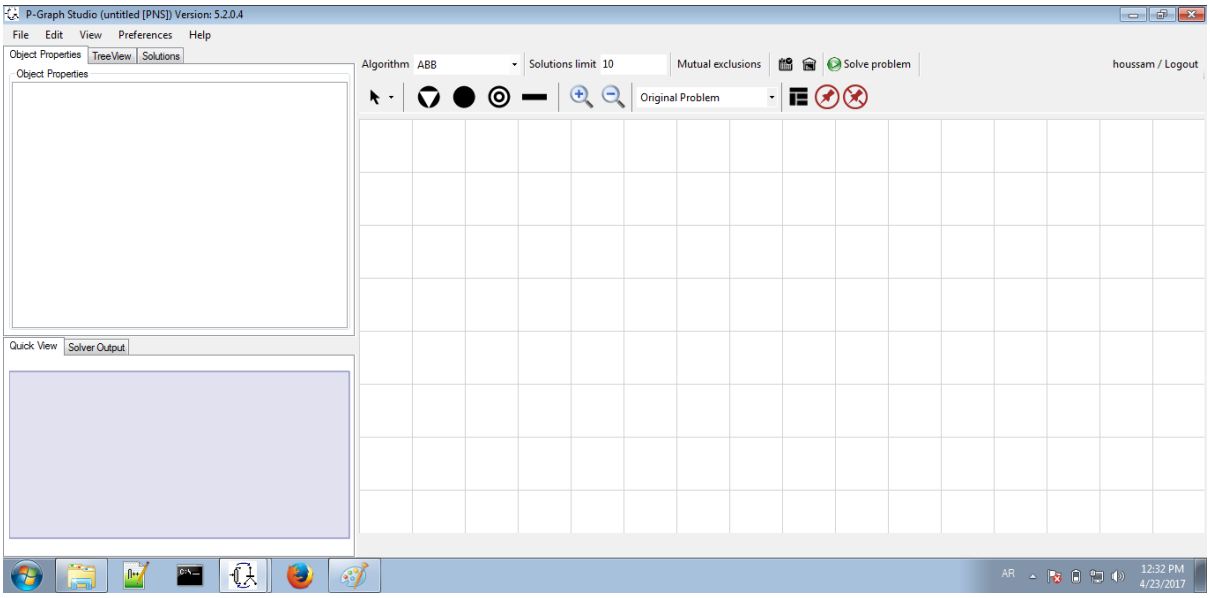


Figure 2.3: P-Graph Studio Window

As a solver, PNS Studio can generate the maximal structure, the combinatorial feasible structures, and the globally optimal and suboptimal solutions of the problem. In the latter case the objective can be either cost minimization or profit max- imization. PNS Studio provides a double pane view of solutions to compare alternatives. Models and initial structures created in PNS Drawn can be imported into PNS Studio where they can be further edited. It is also possible to export brief or more detailed reports from PNS Studio to Microsoft Excel [16].

2.5 Conclusion

We have seen in this chapter the definition of the second framework i use PGraph and the Process Network synthesis, with some Algorithm applicable on PNS. This framework originally is developed for chemical reaction, but you can use it for other system and modeling other system.

CHAPTER 3

Model to Model Transformation

3.1 Introduction

In this chapter, illustrate the Level of modelisation, and we cite the model-to-model transformation, specifically the definition of graph transformation type, and finally how to implement the graph transformation in the AToM3 tool.

3.2 Levels of Modilisation

What do we mean when we use the word model it has Several definitions among

1. A model is an abstraction of a system (real or language-based) allowing To draw predictions or conclusions [17].

2. The central idea of modeling is to produce a reduced version of the system To determine and evaluate its salient properties [18].

3. A model is a simplification of a system designed with a purpose in mind. The model should be able to answer questions in the system current. The responses provided by the model should be the same as those proposed by the system itself, provided that the questions are within the defined domain by the general objective of the system [19].

meta-model is a model of a modeling language. The term "meta" means above.

A meta-model a language of Modeling at a higher level of abstraction than the modeling language itself [20].

Meta-Modelling, which is the process of modelling formalisms. Formalisms are described as models described in meta-formalisms. The latter are nothing but expressive enough formalisms, such as Entity Relationship diagrams (ER) or UML class diagrams.

A model of a meta-formalism is called a meta-meta-model, a model of a formalism is called a meta-model [21].

Meta-model architecture allows a meta-model to be seen as a model, it is Itself described by another meta-model. This allows all meta-models to be Described by a single meta-model. This unique meta-model, known as a Meta-model, is the key to meta-modeling because it allows all languages Modeling to be described in a unified manner.

3.2.1 Architercteur of Meta-Modeling

The traditional meta-model architecture proposed by OMG is based on 4 Levels described in this Figure 3.1 [22, 23] .

1. **Model** is a simplified abstraction of a studied system, constructed in a Particular intent.

   It should be able to be used to answer questions about the system

   A system is a theoretical construct formed by the mind on a subject Example, an idea that is implemented to explain a physical phenomenon that can be represented by a mathematical model)

2. **Meta Model** is a language that expresses models. It defines Concepts as well as the relations between concepts necessary for the expression of models. A model is a possible construction of the metamodel in which it is defined.

   In the Literature, a model is said to conform to the metamodel in which it is defined

3. **Meta Meta Model** is a language used to express metamodels. For Ability to interpret a meta-model requires a description of the language in which It is written: a meta-model for meta-models. It is, of course, Meta-model by the term meta-meta-model.

   To limit the Number of levels of abstraction, the meta-meta-model must have the ability to describe itself, even.

MOF : (Meta-Object Facility) is set of Interfaces allow to define the syntax and semantic of modilisation language, is devloped by OMG [22, 22].

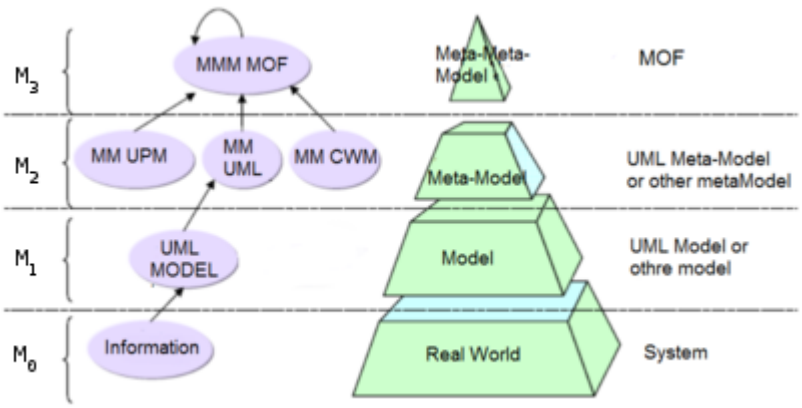


Figure 3.1: Pyramid of Meta-Level [22]

3.3 Model Transformation

Transformation is a fundamental theme in computer science and software engineering. After all, computation can be viewed as data transformation. Computing with basic data such as numeric values and with data structures such as lists and trees is at the heart of programming. Type systems in programming languages help ensure that operations are applied compatibly to the data [24].

3.3.1 Transformation Categories

In general, we can view transforming models to code as a special case of model-to- model transformations; we only need to provide a metamodel for the target programming language. However, for practical reasons of reusing existing compiler technology, code is often generated simply as text, which is then fed into a compiler. For this reason, we distinguish between model-to-code transformation (which would be better described as model-to-text since non-code artifacts such as XML may be generated) and model-to-model transformation. Several tools offer both model-to-model and model-to-code transformations [25].

3.3.1.1 Model To Code

In the model-to-code category, it is very basic code generation approach consists mechanism to traverse the internal representation of model and write code to a text stream, we distinguish between visitor-based and template-based approaches [25].

3.3.1.2 Model To Model

Model-to-model transformations translate between source and target models, which can be instances of the same or different metamodels, Most existing MDA tools provide only model-to-code transformations, we distinguish among direct-manipulation approaches, relational approaches, graph-transformation-based approaches [25].

3.3.2 Transformation Languages and Tools

3.3.2.1 ATL :

Transformation Language (Jouault et al., 2006) is a model transformation language and toolkit developed and maintained by OBEO and INRIA-AtlanMod (Czarnecki and Helsen, 2006) [26].

3.3.2.2 JTL :

Janus Transformation Language (JTL) is a bidirectional model transformation language specifically designed to support non-bijective transformations and change propagation (Cicchetti et al., 2011) [26].

3.3.2.3 ETL :

Epsilon family (Kolovos et al., 2006) is a model management platform that provides transformation languages for model-to-model, model-to-text, update-in-place, migration and model merging transformations [26].

3.3.2.4 Kermeta :

The Kermeta language was initiated by Franck Fleurey in 2005 , The Kermeta language borrows concepts from languages such as MOF, OCL and QVT [26].

3.3.2.5 QVT :

The OMG has defined a standard for expressing M2M transformations, called MOF/QVT or in short QVT (Eclipse, 2008). Eclipse has two extension for QVT called QVTd (Declarative) and QVTo (Operational/Procedural) [26].

3.3.2.6 MOFScript :

The MOFScript includes tools and frameworks for supporting model to text transformations, e.g., to support generation of implementation code or documentation from models [26].

3.4 Concept of Graph Transformation

The process of graph transformation consists in the iterative application of Rule to a graph. Each rule application replaces a part of the graph, As defined in the rule. The mechanics of the graph transformation works as follows :

Algorithm 1 application of graph transformation

Data: G : Graph Source , GG : Graph Grammar , R : one rule from GG , subG : sub Graph from G , i : integer represent the priority of Rule ; i=0;
Result: G graph contain the graph target

```
while ( R = ChooseFrom(GG,i) ) != null do
    while ( (subG = selectFrom(G)) != null do
        if not subG.isDoneWith(R) then
            R.ApplyOn(subG)
            subG.markAsDoneWith(R)
            print "We Applied Rule R on the subGraph subG"
        else
            print "This rule is applied before that on the same sub graph"
        end
    end
    i = i + 1
end
```

This operation is based on a A set of rules respecting a particular syntax, called the grammar model of Graph. Before starting apply rules we execute initial action, its prepare the envirenement to apply these rules and ending with Final Action, it about cleaning the after these rules [27, 28].

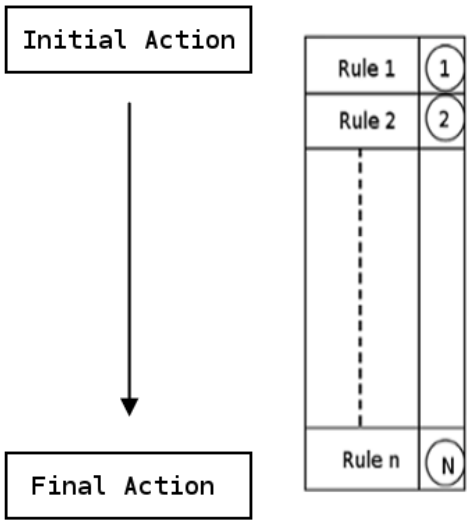


Figure 3.2: Cycle of Transformation [29]

3.5 Graph Grammar

Every Graph Grammer contain a set of rules to use in graph transformation and its define by [22, 28] : R = (LHS, RHS, K, glue, emb, cond).

- LHS graph of left part.
- RHS graph of right part.
- A subgraph K of LHS.
- A glue occurrence of K in RHS that connects the subgraph with the part graph right.
- An embedding relation emb which connects the vertices of the graph on the left-hand side And those of the graph on the right-hand side.
- A set cond which indicates the conditions of application of the rule

Applying a rule R = (LHS, RHS, K, glue, emb, cond) to a graph G produced in Result a graph H according to the following five steps

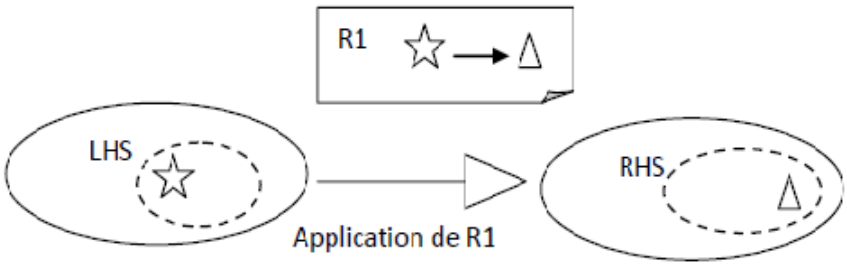


Figure 3.3: Rule Application [29]

1. Choose an instance of the left-hand LHS graph in G.
2. Check the conditions of application according to Cond.
3. Remove the occurrence of LHS (up to K) from G and the hanging arcs (all Arcs having lost their sources and / or their destinations). This provides the graph of Context D of LHS which left an occurrence of K
4. Paste the context graph D and the RHS right-hand graph according to the occurrence Of K in k = 1, .., ∞ D and in RHS, it is the construction of the union of disjunction Of D and RHS, and for each point in K, identify the corresponding point in D with the corresponding point in RHS
5. Press the right-hand graph in the LHS context graph following the Embedding relation emb: for each incident arc removed with a vertex v in D and with a vertex v1 in the occurrence of LHS in G, and for each vertex V2 in RHS, a new incident arc is established (same label) with the image of v And the vertex v2 provided that (v1, v2) belongs to emb.

3.6 Transformation system

We define a graph transformation system as a rewriting system Of graph that applies the rules of the graph grammar on its initial graph of Iteratively by the Engine, until no more rules are applicable [22, 30].

1. Define the source and target Meta-Model
2. Create the the source model according to the source Meta-Model
3. Define transformation rules to Transform from source model into target model

Finally the engine read source model and apply the transformation rules and write target model as the following Figure 3.4

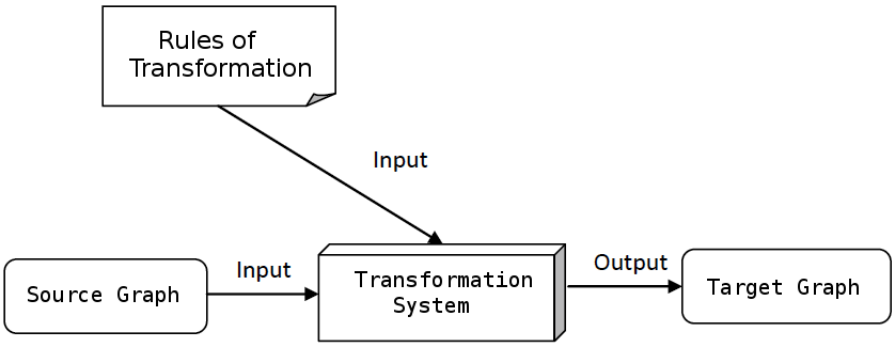


Figure 3.4: Transformation System [29]

3.7 AToM3

AToM3 (A Tool for Multi-formalism and Meta-Modeling) is a tool for model- Multi-paradigm model developed in the MSDL (Modeling, Simulation and Design Lab) of the Computer Science Institute at McGill University Montreal, Canada [31, 32].

It is Developed with the Python language in collaboration with Professor Juan de Lara de The Autonomous University of Madrid (UAM), Spain AToM3 is developed to satisfy two main features that are

1. Meta-modeling
2. The transformation of models

The formalisms and models in AToM3 are described graphically. From a Meta-specification (example: in the Entity-relation formalism) of a formalism, AToM3 Generates a tool to visually manipulate (create and modify) the models described in The specified formalism.

The transformations of the models are realized by the rewriting Graphs, which can be expressed in a declarative way as a model of Grammar of graphs [21].

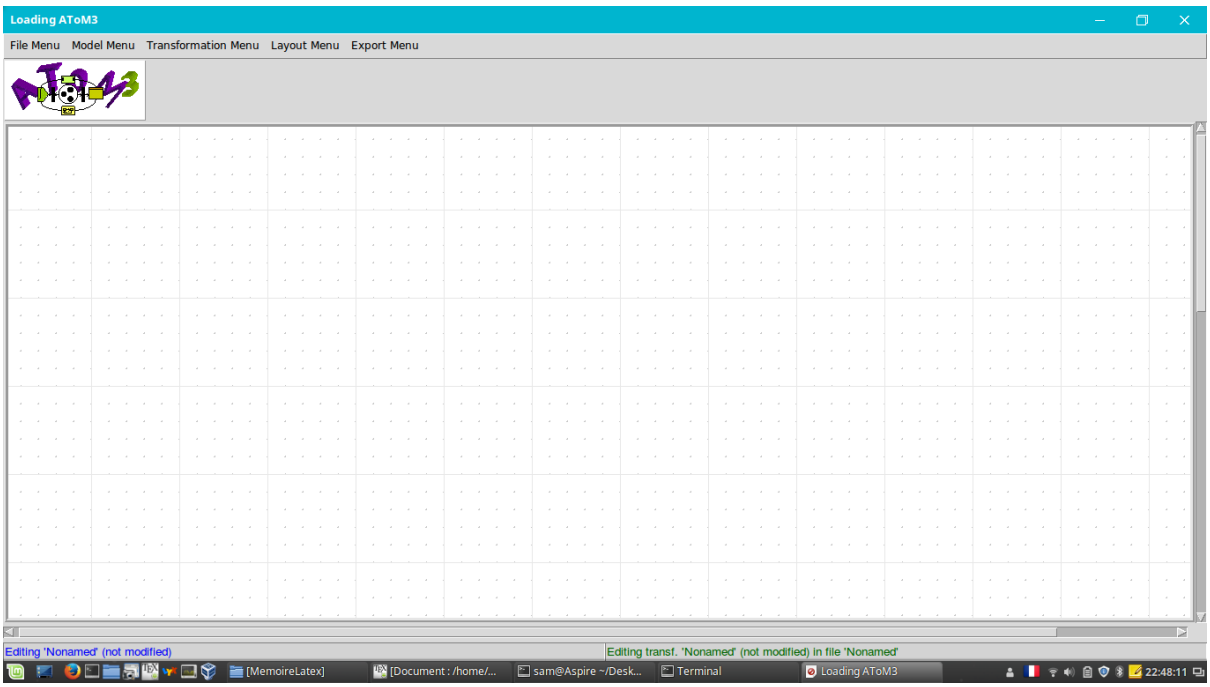


Figure 3.5: AToM3 Window

figure 3.5 illustrates the interface of AToM3

3.7.1 Classes in AToM3

In this work we use ClassDiagramm Formalism to create our Formalism or Meta-Model is built in the tool so we can load it and use it [21]. In AToM3 the meta-models can be constructed from Classes and Relationships.

The description of classes and association relations consists of

- Name
- Attributes
- Constraints
- Action
- Cardinalities
- Appearance

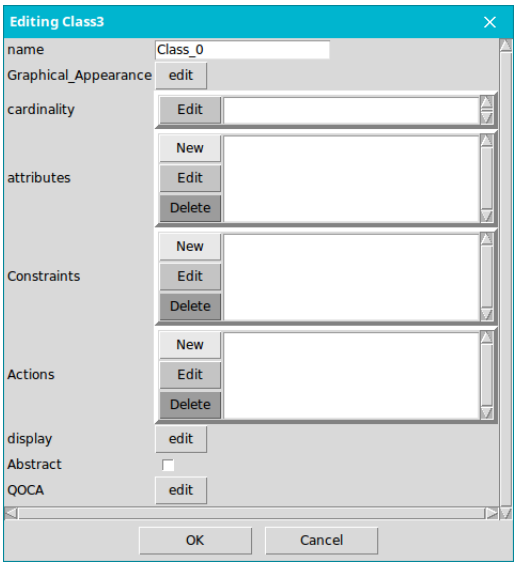


Figure 3.6: Class Editor

3.7.1.1 Constraint

Constraints can be specified as OCL (Constraint Object Language) or Python They have the following properties:

- constraint name
- triggering event like Drag, Move, Select .. and launch this event before (pre-condition) or after (post-condition)

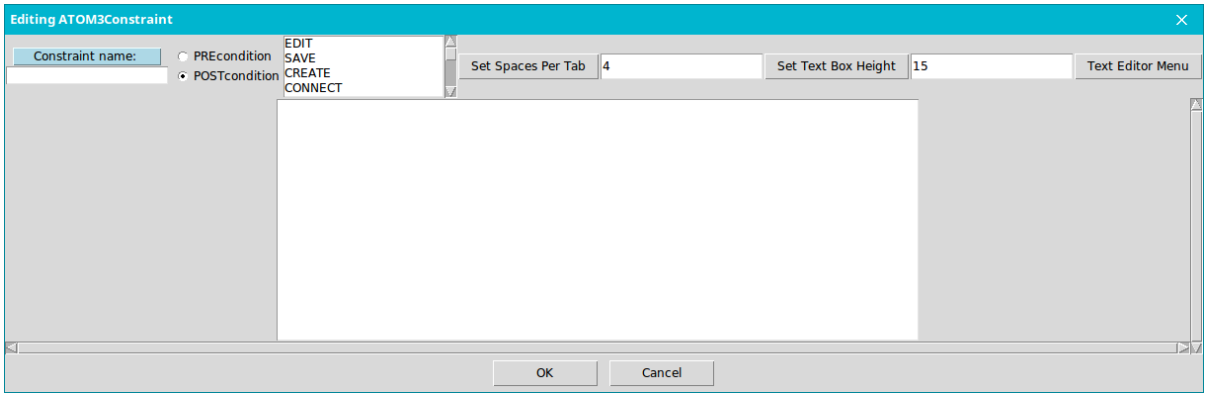


Figure 3.7: Constraint Editor

3.7.1.2 Action

An action is similar to a constraint except that it has other effects and is a Code in Pyton only, its have the same windows 3.7

They have the following properties :

- action name
- triggering event: It can be either
  1. Semantics such as saving a model
  2. Graphic or structural, such as moving or selecting an entity.
- The execution is either
  1. Before the event (precondition)
  2. After (pots-condition)

3.7.2 Graph Grammer in AToM3

In AToM3, grammar is characterized by

- An initial action.
- A final action.
- The set of rules.

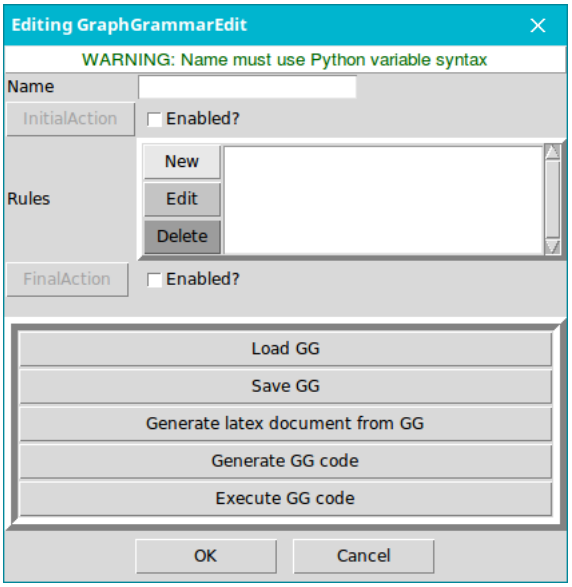


Figure 3.8: Graph Grammar Window

Each rule consists of: A specific name for the rule, and priority indicator the order in which the rule is applied, and LHS [1]:which is a graph, RHS[2]:that can be a graph and the condition that must be checked before the rule is Applied, finally An action (a Pyton code) that must be executed after the rule is Applied

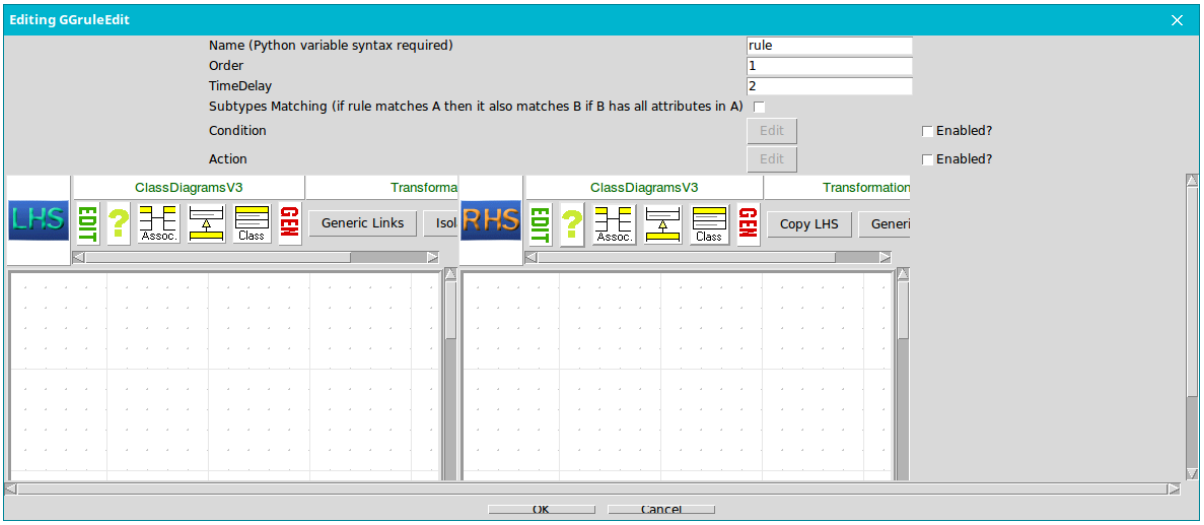


Figure 3.9: Rule Editor

3.8 Conclusion

we present in this portion of our document, the different type of transformation approach and we focus on graph transformation approach, then we cite the tool i will use in the next chapter to implement our approach of transformation.

[1]LHS:Left Hand Side
[2]RHS:Right Hand Side

CHAPTER 4

Transformation Approach

4.1 Introduction

We illustrate in this chapter the most important steps of our approach. Starting with the first approach of transformation from OMACS to PNS, then the second approach to transform PNS model into XML code, ending with import the XML code, to use it in the second tools.

4.2 Transformation Approach (OMACS To PNS)

To Transform OMACS Model into PNS Model, we start to :

1. Define OMACS Meta-Model
2. Define PNS Meta-Model
3. Define the rules of Transformation

4.2.1 OMACS

This section describe the work on the OMACS framework, you will see the Meta-Model of OMACS in the tools and example.

4.2.1.1 Meta Model of OMACS

To define OMACS meta model, we first load the Class Diagramm Formalism, to be able to create, manipulate our Meta-Model, The OMACS Meta-Model contain a set of classes:

- Agent : represent a node in the graph of MaS, and we assign the Role for this Agent (Node) to play
- Capabilities : its node in the MaS, this Entitie represent the skills Which owns by an agent
- Role : represented by a node in the graph, this is the work we assign for the agent
- Goal : it is also a node, we assign role for an agent depending common capabilites between an agent and role in order to achieve this goal

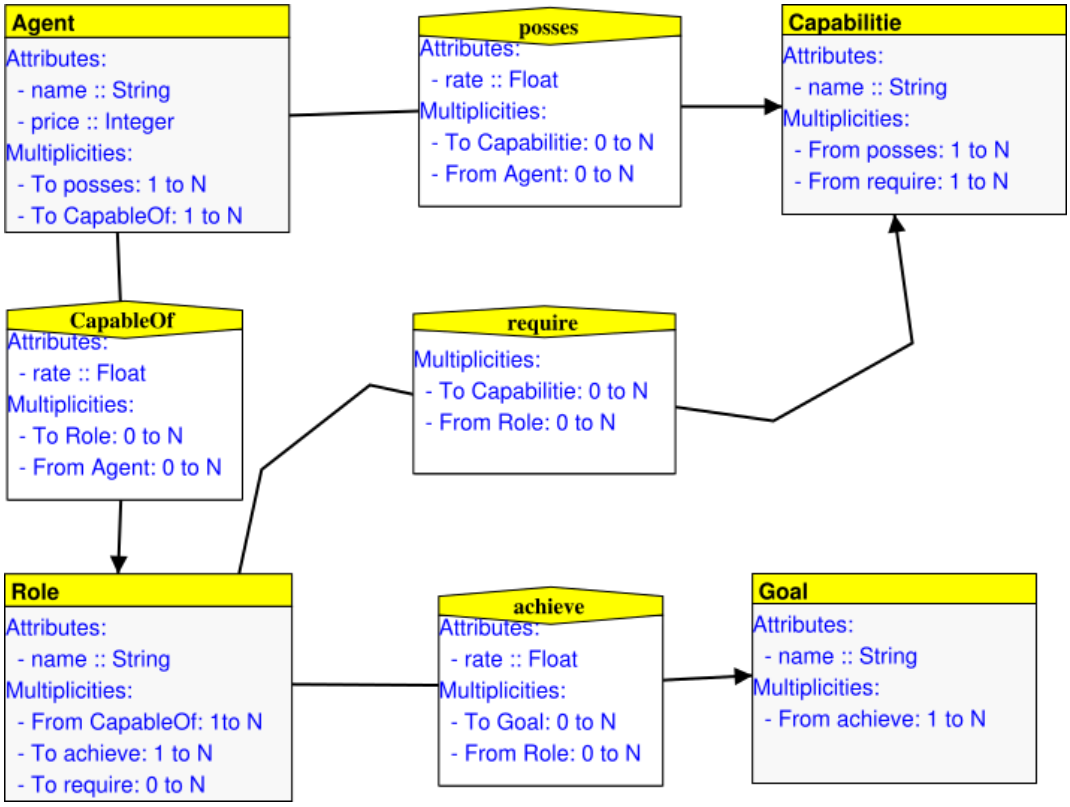


Figure 4.1: OMACS Meta-Model

figure 4.1 illustrate OMACS Meta-Model contain 4 classes (Agent, Capabilities, Role, Goal) and 4 relation (Possess, Require, CapableOf, Achieve). The Attribute Name in the classes represent the name of current node , and the rate attribute in the relation between classes represent the relation percentage between two node or entities.

For example between an Agent and Capabilities its means how mush this Agent possese this capibilites

4.2.1.2 Example of OMACS

Starting from Meta-Model of OMACS in figure 4.1, AToM3 generate a formalism of OMACS, this formalism allow us to create our OMACS Model (Multi Agent System).

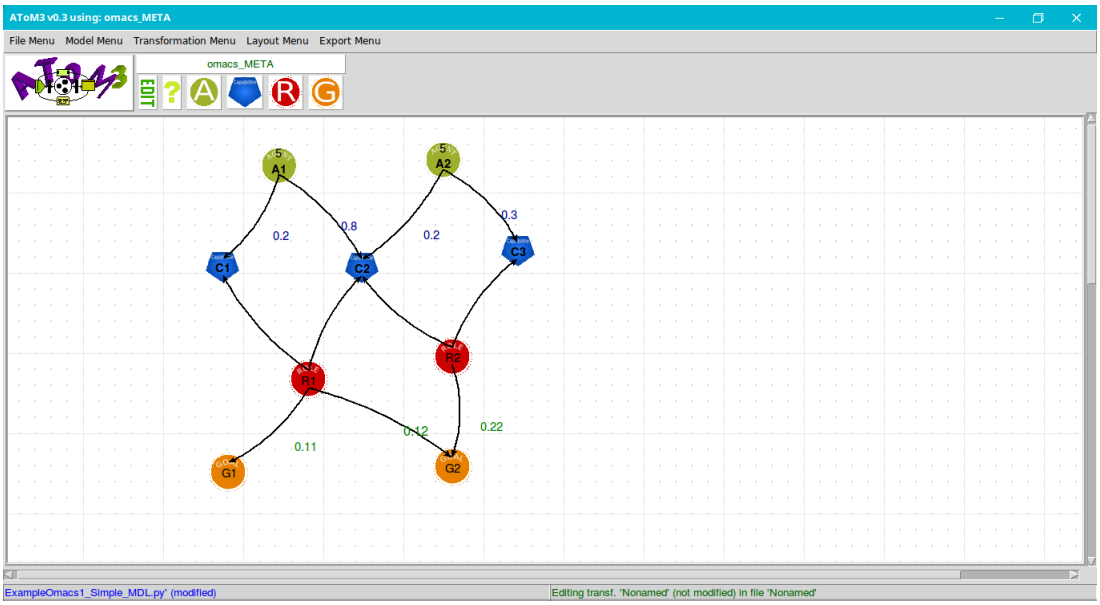


Figure 4.2: Formalism of OMACS

Example in this figure 4.2 represent a Multi Agent system which is a contain: 2 agent and 3 Capabilities, 2 Roles, 2 Goals. We assing 2 capabilites (C1, C2) for the Agent A1, and from other angle we se the role R1 require or demande the same capabilities the A1 has, so we can assign the R1 to A1 with the Capable of Relation, all that to achieve the Goals(G1 ,G2).

4.2.2 PNS

In this section we ilustrate and describe the work on the PNS framework, which is the Meta-Model of PNS in the tools and example.

4.2.2.1 Meta Model of PNS

To define PNS meta model, we first load the Class Diagramm Formalism, and it is contain 4 classes and 4 Relation :

- Raw Material : its node in the graph of PNS, represent the initial material, and this node does not have any input edges
- Intermediate Material : its node in the graph, represent a material in the system
- Final Product : it is also a node in the graph, represent the desire material
- Operating Unit : it is other node type in the graph ,represent the task or operating, it is consume the input material and produce the output material

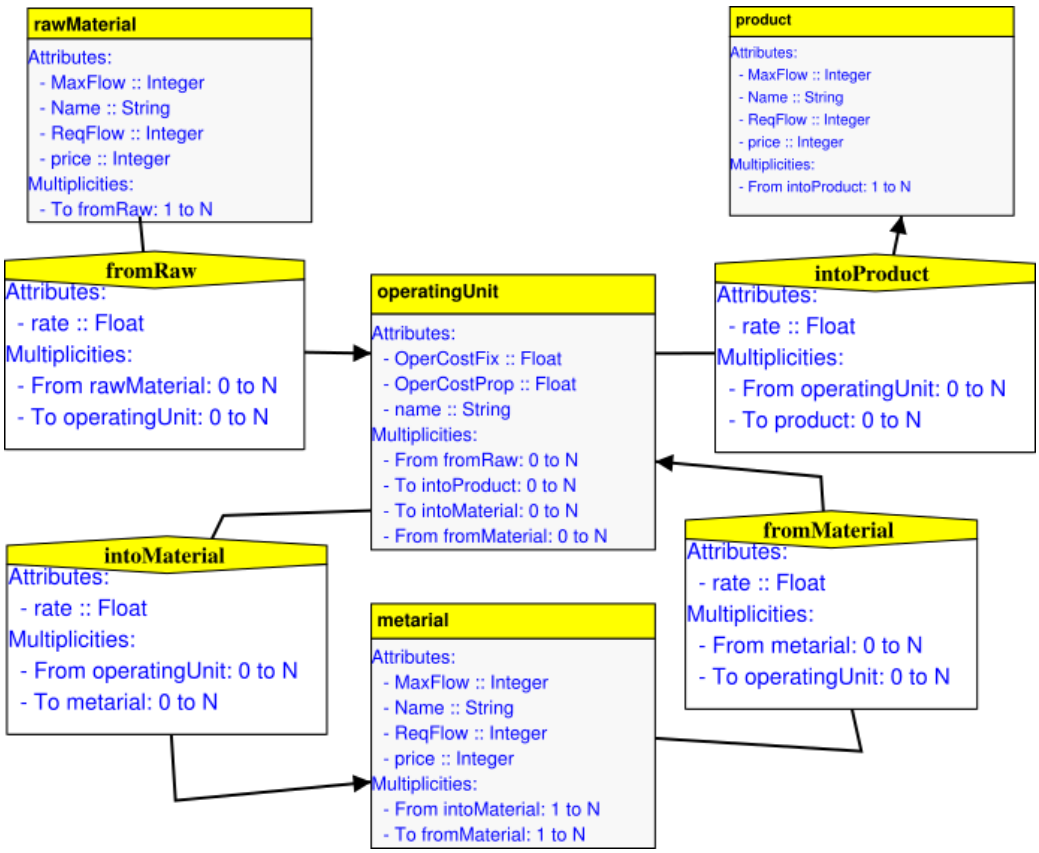


Figure 4.3: PNS Meta-Model

all material in figure 4.3 containt the same list of properties

- *Name* : name of Material
- *Price* : Price of Material
- *ReqFlow* : Requirment Flow means how much you have from this material in this system
- *MaxFlow* : Maximum Flow means how much your system can handle

and for the Operating Unit contain 3 Attribute :

- Name : name of this Task
- Operating Cost Fix : cost for entire period, Example : year
- Operating Cost Proportional : cost for every Operating from this unit

4.2.2.2 Example of PNS

The AToM3 Tools generate a formalism from the meta model we created before, load it and use it like figure 4.4

the figure 4.4 represent system de production contain 2 raw material and one intermediate material, final product and 3 operating unit.

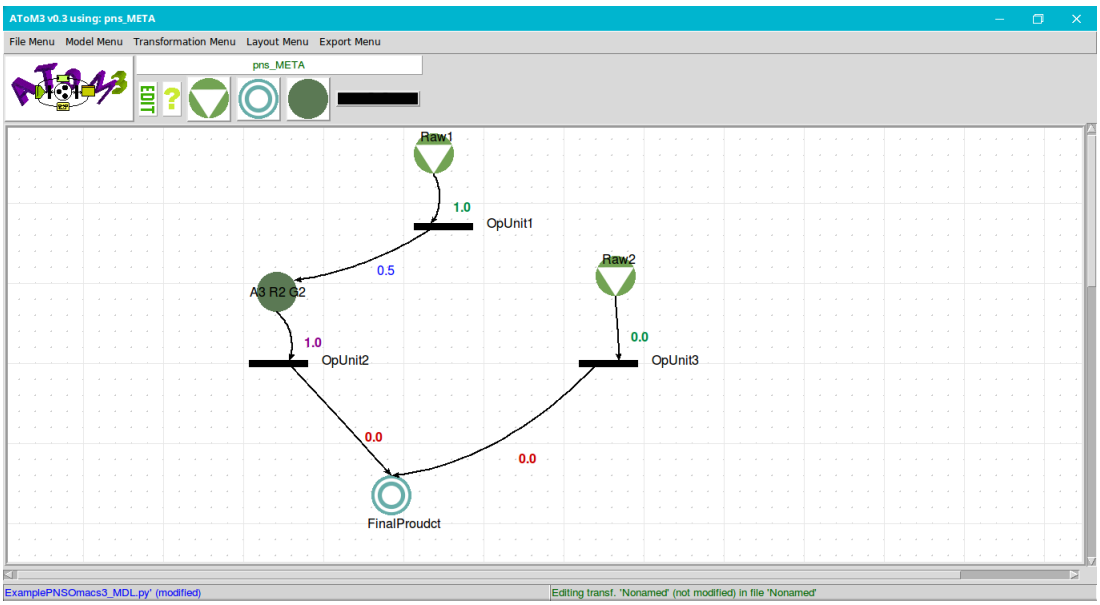


Figure 4.4: Formalism of PNS

4.2.3 Graph Grammar

A Graph grammar is a grammar consisting of a set of rules, allow to Transform a formalisms of the same nature or of a different nature.

Each Rule is composed of two parts, the left part (LHS) and the right part (RHS). Each part can be a subgraph of the formalisms considered in the transformation

in our work, the formalisms considered in the transformation are formalism OMACS as source graph to be transformed to PNS as target graph.

This grammar is defined bu using the tool AToM3 according to the following step

1. Load OMACS and PNS meta-models
2. Create the transformation grammar
3. Define the rules of grammar
4. Generate the executable file of the grammar

Our grammar is composed of :

Rules : each rule is characterized by a name and execution priority. They are classified in 04 categories :

1) Collect Categorie : Rules for Collecting and create relation between some entitie.

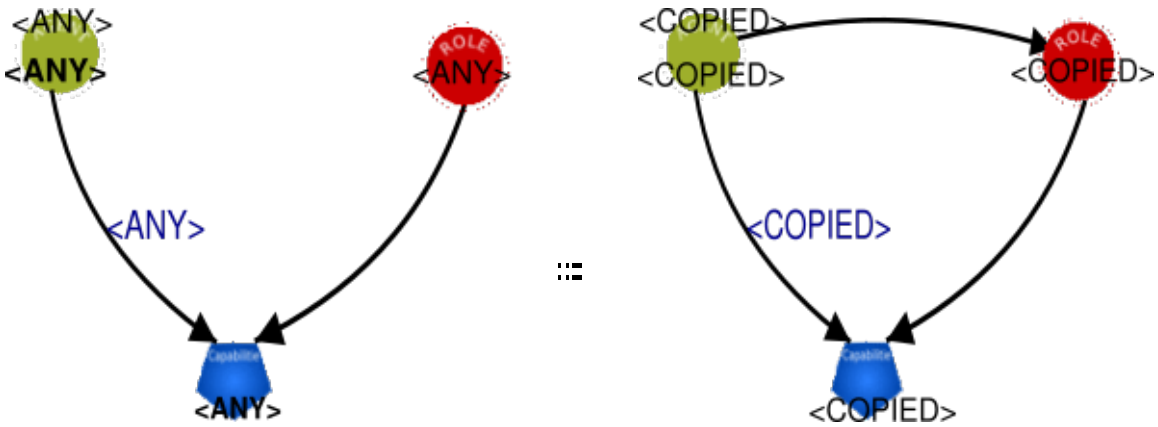
2) Transform Categorie: Rules for transforming nodes or links into materials or operating units.

3) Links Categorie: Rules to links generated materials and operating units.

4) Cleaning Categorie: Rules to cleaning unnecessary entities from the result .

Our approach contain 20 rules, we cite the most important transformation rules into the following steps :

1) Agent2RoleLink1 : Create the direct link between the Agent and the Role (order 1) : The Figure 4.5 illustrate how to create link between the agent and role depending on common capabilities, order 1 mean it is the first rule applied in the grammar.



(a) LHS (b) RHS

Figure 4.5: Assign Role to Agent

you will notice these words in LHS(ANY [1]) and RHS(COPIED[2] , SPECIFIED [3])

1.1) Condition for the rule Assign Role to Agent : Test if there is a node agent, role visited before in order not to fall in infinity loop, figure 4.5 represent the rule.

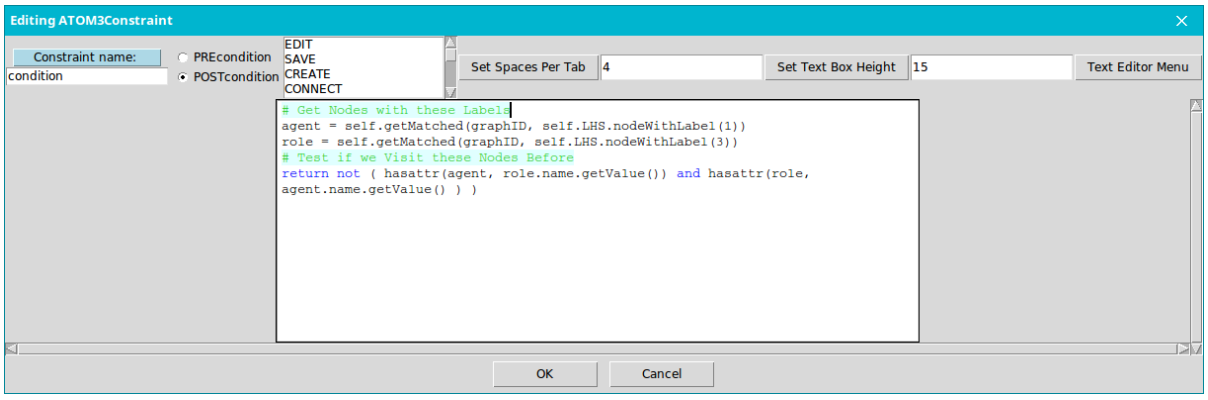
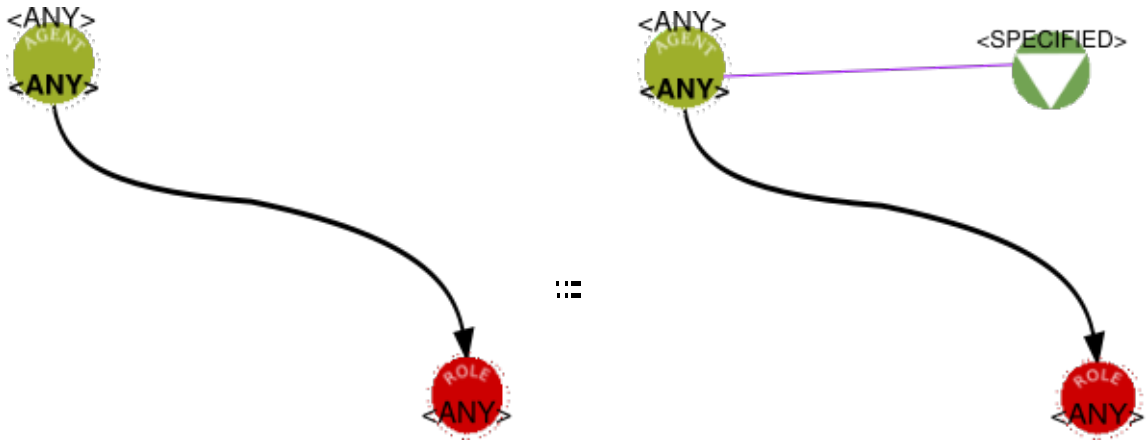


Figure 4.6: Condition link agent with role

[1]ANY:its attribue in the node and the engine will select every node with ANY attribute value
[2]COPIED:copie the attribue value from the node with same GGLabel in LHS
[3]SPECIFIED:specified the attribue value manualy or by python code

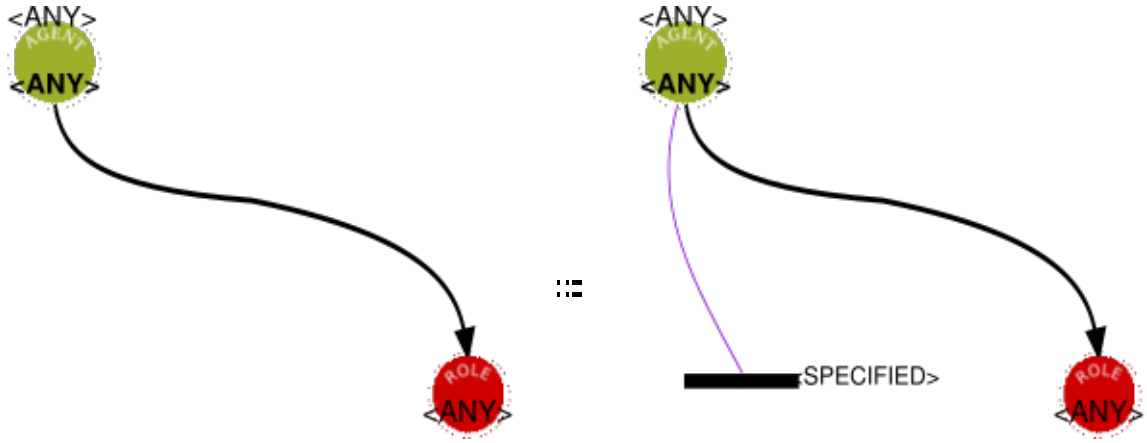
2) TransAgent2Raw : Transform Agent to Raw Material (order 9) : Application of this rule in (figure 4.7) transform every Agent in Multi Agent System into raw material, it has the same name and price.



(a) LHS (b) RHS

Figure 4.7: Transform Agent to Raw-material

3) TransLinkAR2OpUnit : Transform Link between Agent and Role into Operating Unit (order 10) : This rule in (Figure 4.8) allow to transform, the relation capable of playing between an agent and role into operating unit.



(a) LHS (b) RHS

Figure 4.8: Transform CapableOf relation into operating unit

4) TransGoal2Mat : Transform Goal to intermediate material (order11): Application of the rule illustrated in (Figure 4.9) Transform a goal in MaS into intermediate material in process system.

(a) LHS (b) RHS

Figure 4.9: Transform Goal to intermediate material

5) CreateFinalStat : Create the Product node (order 13) : Create the final material stat in the system. If you notice the LHS is empty because this rule applied for one time, and this rule represented in (Figure 4.10).

(a) LHS (b) RHS

Figure 4.10: Create the final material

5.1) Condition for Create Final Stat : This conditon test if this attribute Final stat equal zero , in other word we did not visit this rule before, the rule in figure 4.10.

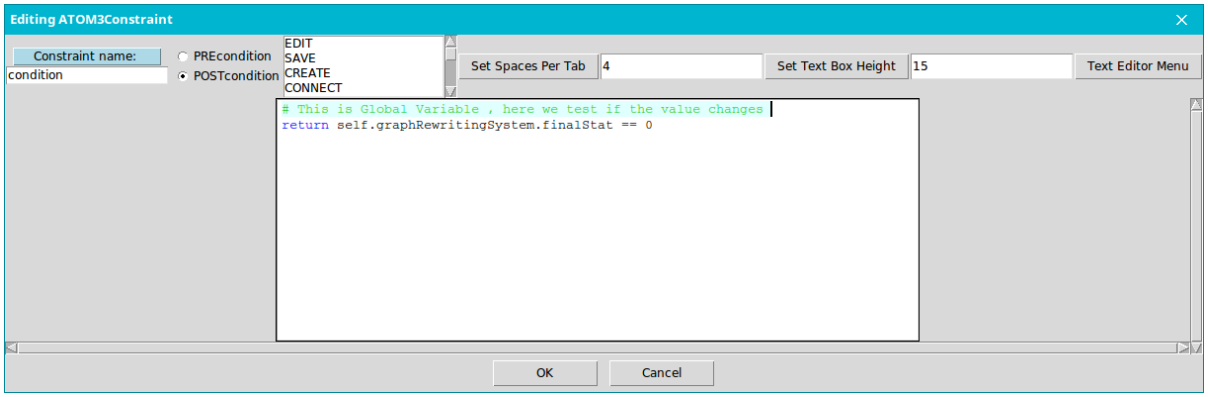
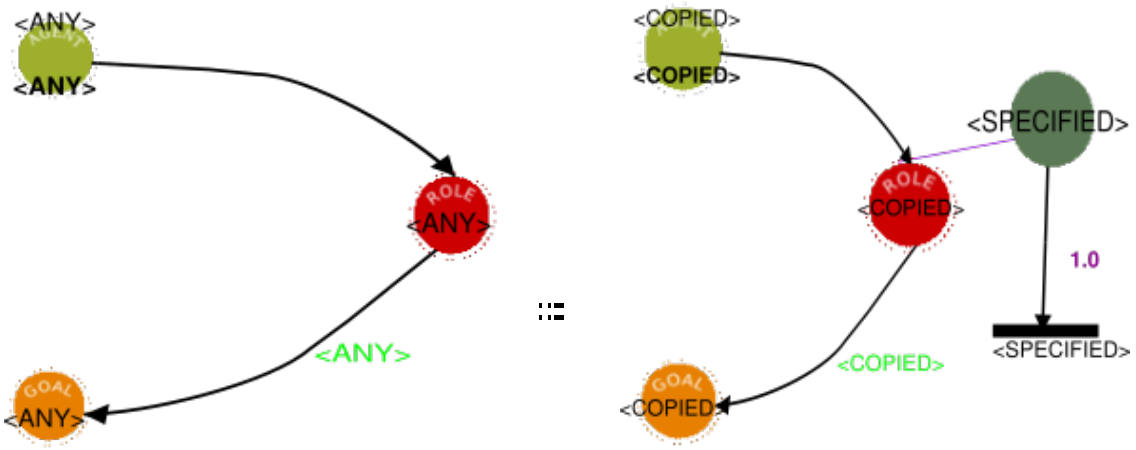


Figure 4.11: Condition of generate final stat

6) CreateMatARG : Generate auxiliary part (order 14) : The Application of this rule generate the auxiliary part which is intermediate material consumed by operating unit, and the auxiliary part represent an agent capable to play a role in order to achieve a specific goal (Figure 4.12).



(a) LHS (b) RHS

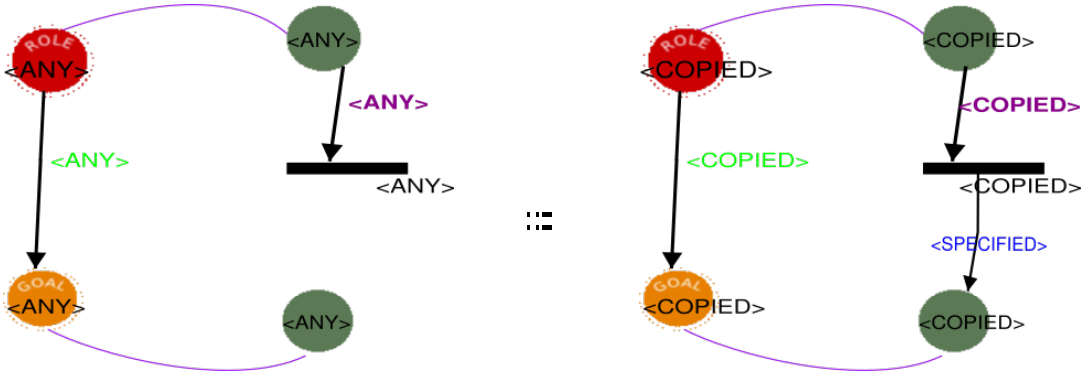
Figure 4.12: Generate auxiliary part

7) CreateLinkMatr2OAF : Create Operating unit between goal material and the product (order 15) : The rule (Figure 4.13) create an operating unit between the final material and the goals material.

(a) LHS (b) RHS

Figure 4.13: Create Operating unit between goal and OAF

8) CreateLinkMat-ARG2Goal : Create link between auxiliary part and goal material (order 19) : The rule in (Figure 4.14) describe how to link the auxiliary part with right goal.



(a) LHS (b) LHS

Figure 4.14: link between auxiliary part and goal

8.1) Condition for rule link between AUX part and Material Goal : Here we check if we did not visited before and the name of Auxiliary Part end with goal name to link it with the right goal material, the rule in figure 4.14.

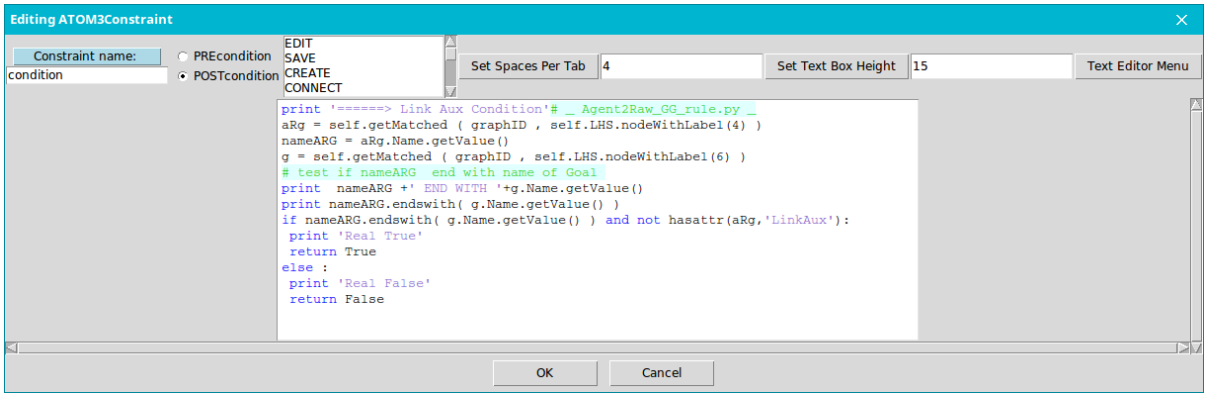
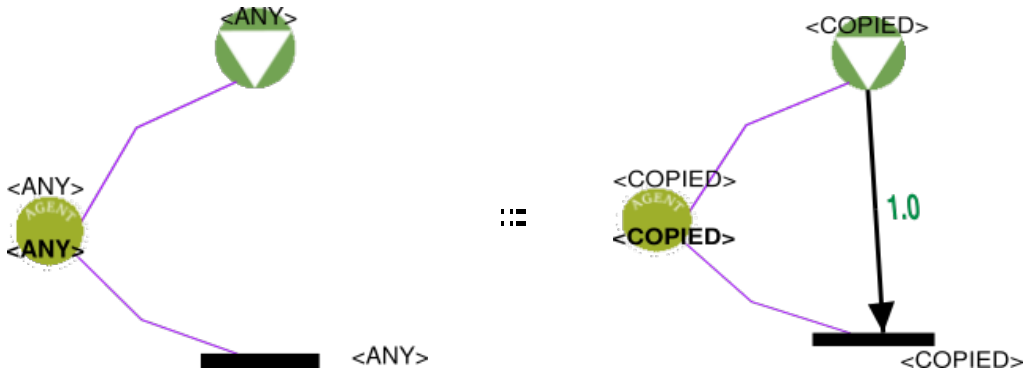


Figure 4.15: Condition of Auxiliary part

9) CreatLinkRaw2AR : Create link to consume the raw material by operating unit (order 20) : This Figure 4.16 illustrate this rule, and it about how to create an arc between the raw material was generated from an agent and the operating unit.



(a) LHS (b) LHS

Figure 4.16: consume the raw material by operating unit

4.2.4 Case study

we represent, in this section two examples, and examine the graph transformation process was developed in last section.

4.2.4.1 Simple multi Agent System

This example represents a very simple system composed of two agent and 3 capabilities, 2 role and 2 goals. Figure 4.17 illustrates this example :

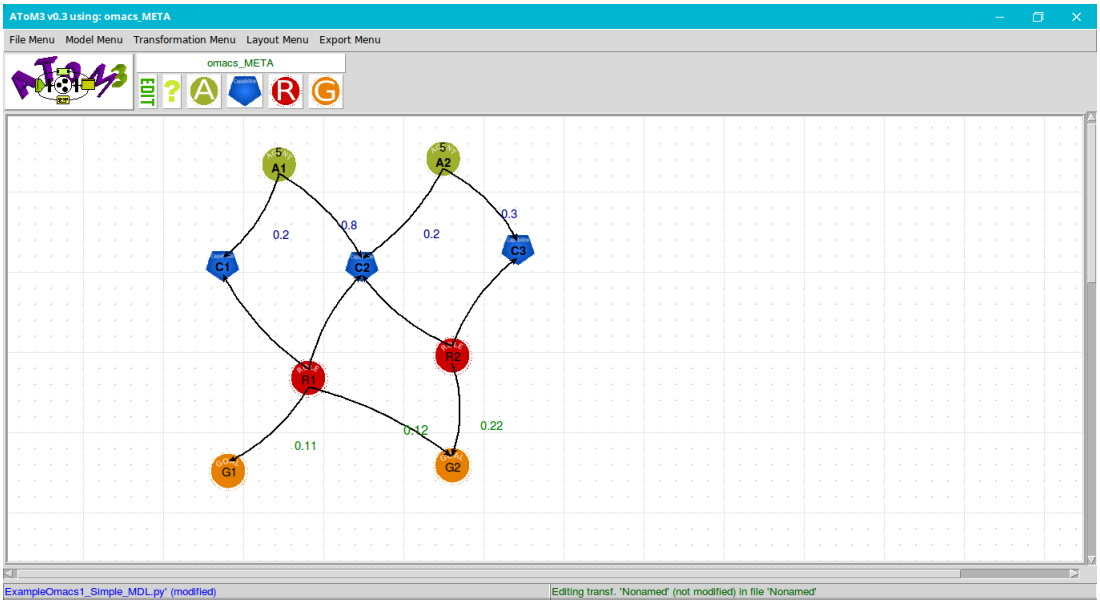


Figure 4.17: Example 1 Multi Agent System

The purpose of this example is to demonstrate the transformation of multi agent system represented in OMACS into pns model

Figure 4.18 shows the transformation result. It is clear that the resulting contain a set of material and operating unit

- The raw material A1, A2 represent the agent
- The operating unit A1 R1 represent the relation between an A1 and R1 in Mas before the transformation its the same for other entities like this one
- the operating unit and material named A2 R2 G2 represent agent A2 Playing R2 in order to achieve G2, it is the same for other entities
- the operating unit and material named A2 R2 G2 represent agent A2 Playing R2 in order to achieve G2

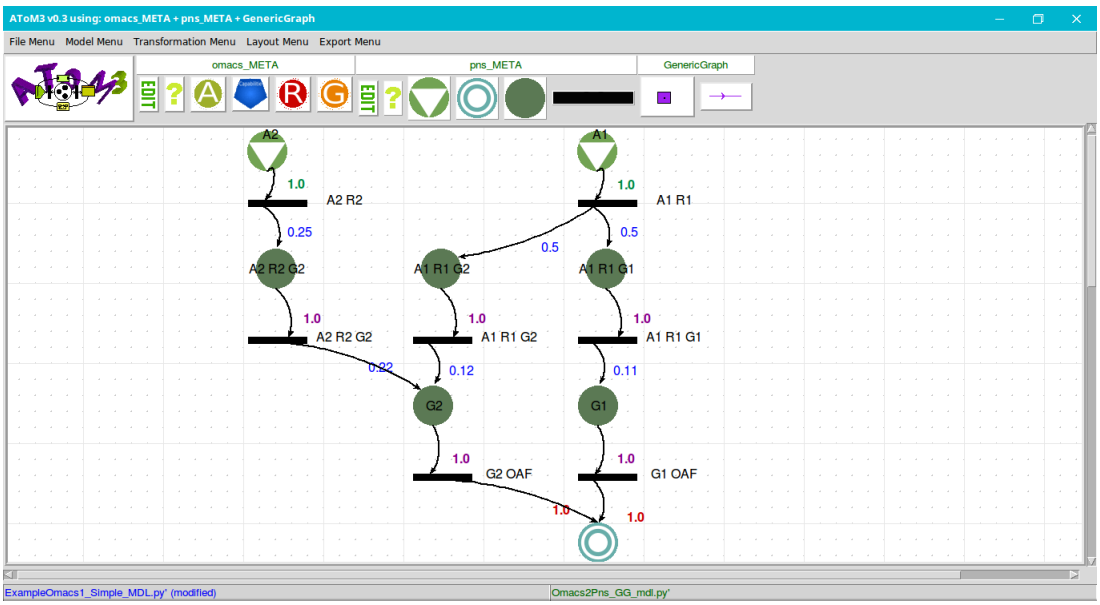


Figure 4.18: Multi Agent System After transformation

4.2.4.2 Complex multi Agent System

This example represents a complex system composed of 3 agent and 4 capabilities, 2 role and 4 goals. Figure 4.17 illustrates this example ,

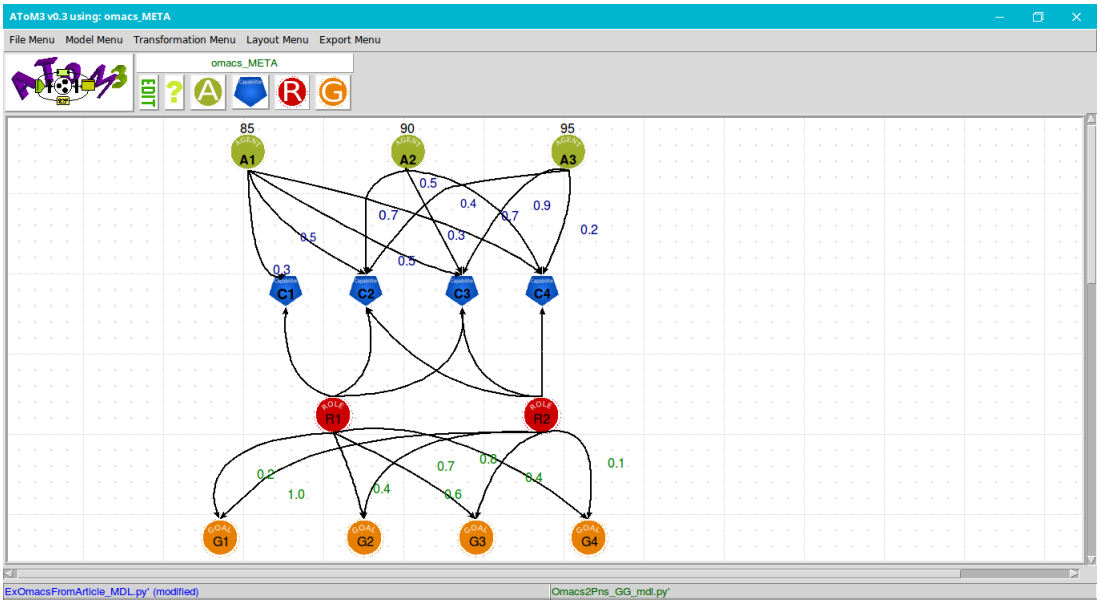


Figure 4.19: Complex Multi Agent System

- every agent possesses a set of capabilites. For example A1 possese all capabilies in the system
- Role require a list of capabilies every agent has the same capabilies or more, so he can play this role
- the Agent A1 possesses all capabilites in the system so we can say, A1 capable of playing R1 and R2

- Agent A1 capable of playing all roles in this system, so he can achieve all goal in the system

after the Transformation of Multi agent system from OMACS framework into PNS framework, we obtain the model is represented in figure 4.20 and next table represent the value of the edges in the last model

Operating Unit	**Input Material**	**Output Material**
G1OAF	G1(1)	OAF (1)
G2OAF	G2(1)	OAF(1)
G3OAF	G3(1)	OAF(1)
G4OAF	G4(1)	OAF(1)
A1R1G1	A1R1G1(0.433)	G1(0.2)
A1R1G2	A1R1G2(0.433)	G2(0.4)
A1R1G3	A1R1G3(0.433)	G3(0.6)
A1R1G4	A1R1G4(0.433)	G4(0.8)
A1R2G1	A1R2G1(0.433)	G1(1.0)
A1R2G2	A1R2G2(0.433)	G2(0.7)
A1R2G3	A1R2G3(0.433)	G3(0.4)
A1R2G4	A1R2G4(0.433)	G4(0.1)
A2R2G1	A2R2G1(0.633)	G1(1.0)
A2R2G2	A2R2G2(0.633)	G2(0.7)
A2R2G3	A2R2G3(0.633)	G3(0.4)
A2R2G4	A2R2G4(0.633)	G4(0.1)
A3R2G1	A3R2G1(0.5)	G1(1.0)
A3R2G2	A3R2G2(0.5)	G2(0.7)
A3R2G3	A3R2G3(0.5)	G3(0.4)
A3R2G4	A3R2G4(0.5)	G4(0.1)
A1R1	A1	A1R1G1(0.433) ,A1R1G2(0.433) ,A1R1G3(0.433) ,A1R1G4(0.433)
A1R2	A1	A1R2G1(0.433) ,A1R2G2(0.433) , A1R2G3(0.433) ,A1R2G4(0.433)
A2R2	A2	A2R2G1(0.633) ,A2R2G2(0.633) , A2R2G3(0.633) ,A2R2G4(0.633)
A3R2	A3	A3R2G1(0.5) ,A3R2G2(0.5) , A3R2G3(0.5) ,A3R2G4(0.5)

Table 4.1: Value of PNS Model

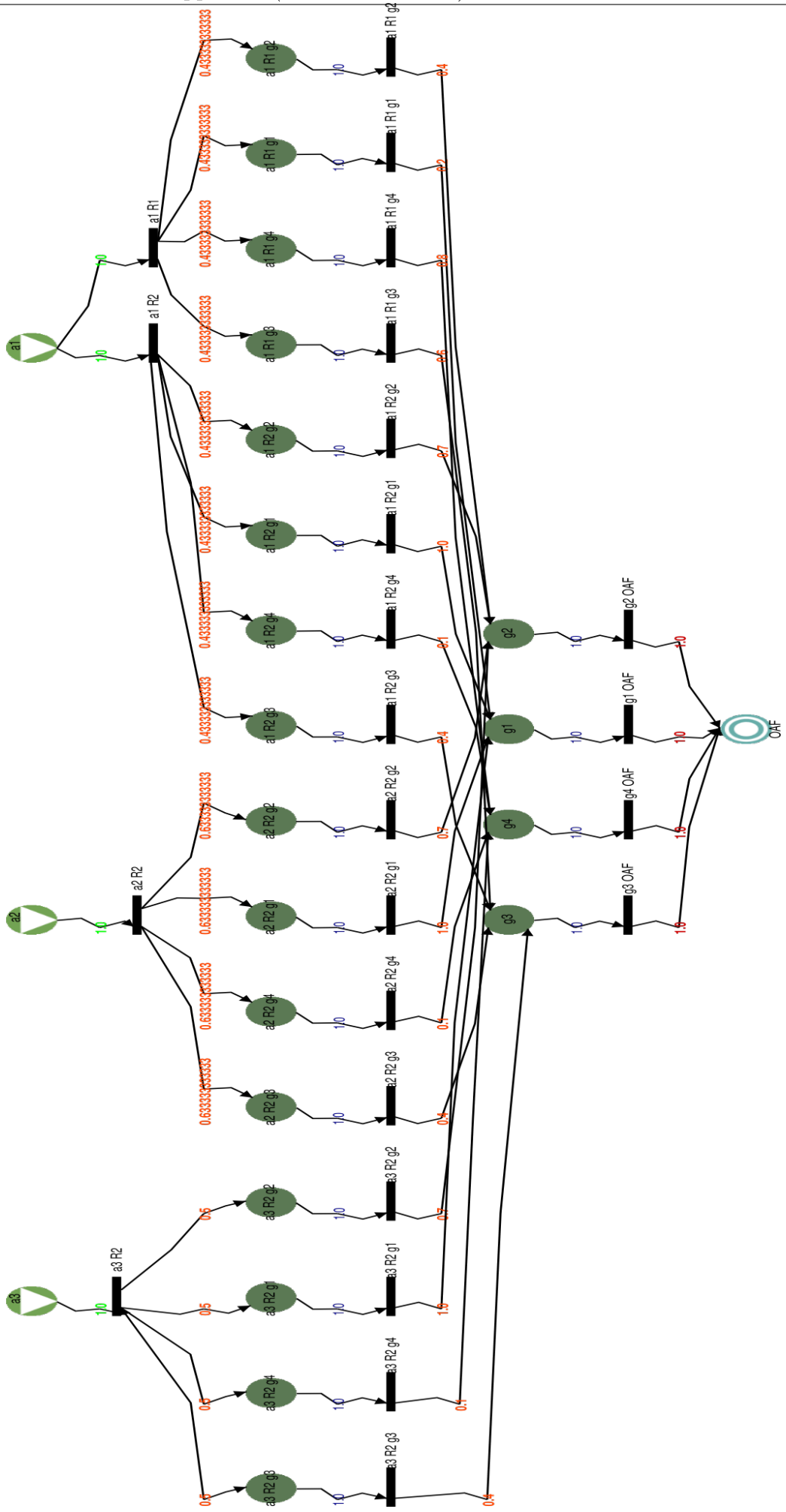


Figure 4.20: Complex Multi Agent System After transforamtion

4.3 Transformation Approach (PNS To XML file)

In Section 4.2 we presented our transformation approach That allows to transform the OMACS to PNS.

The purpose of this transformation is the Optimization and for this we Proposed Another processing approach That transforms PNS to XML files, to find the optimum structure.

4.3.1 Graph Grammar

this grammar is Composed of three parts: before we start, i want to description the most important tags in this xml format :

- PGraph : this is the root tag contain all the next tags
- Materials : This tag contain all the material tags from any kind (raw, intermediate, final)
- OperatingUnits : This tag contain all Operating units
- Edges : contain all edges between all nodes in the graph

❑ **Initial Action :** This portion of our grammar is used to initialize all Global variables used. They are used to meet various needs. in Figure 4.21

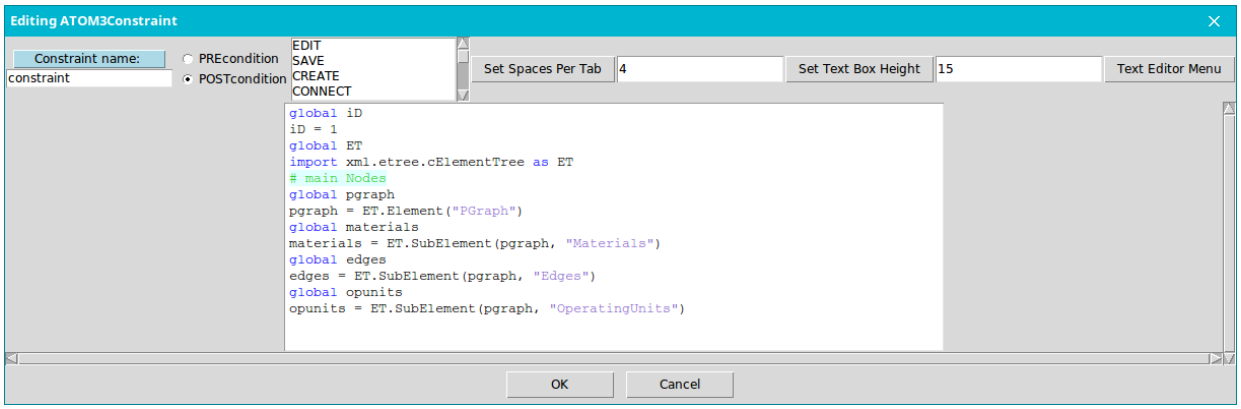


Figure 4.21: Code Initial Action

❑ **Set of Rules :**

Our grammar graphs Consists of ten rules. Each rule is Characterized by a name and an execution priority.

We cite here The Most important rules:

1) Transform RawMaterial into XML code : This rule (Figure 4.22) it is the first rule applied on the Model. To transforms the place to a XML tags.

(a) LHS (b) RHS

Figure 4.22: Raw Material into Xml

a) Condition for Raw To Xml For the application of this rule, it must be verified that the partition To be transformed is not treated before

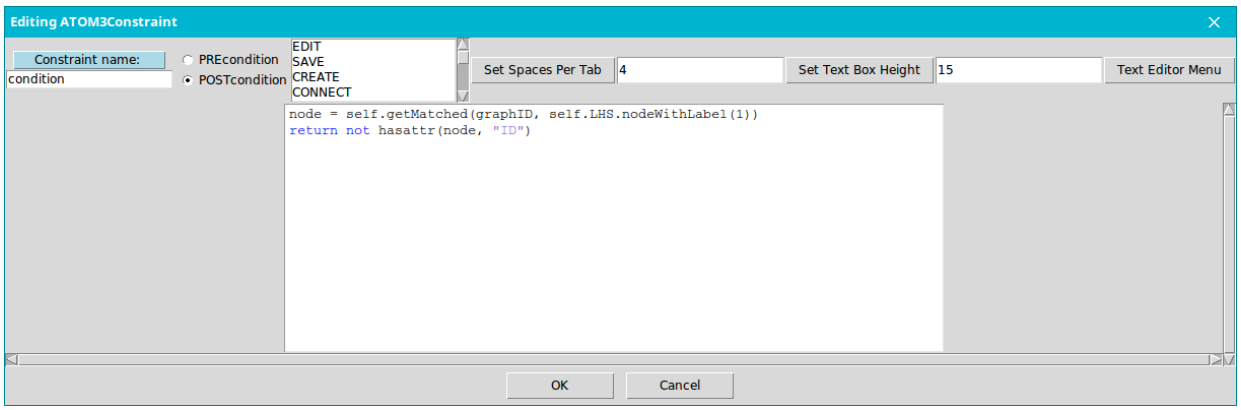


Figure 4.23: condition of Raw Material for Transfromation

b) Action for rule 4.22 Once The rule is applied, its Action Allows to create a XML code and mark the object from the left side to processed.

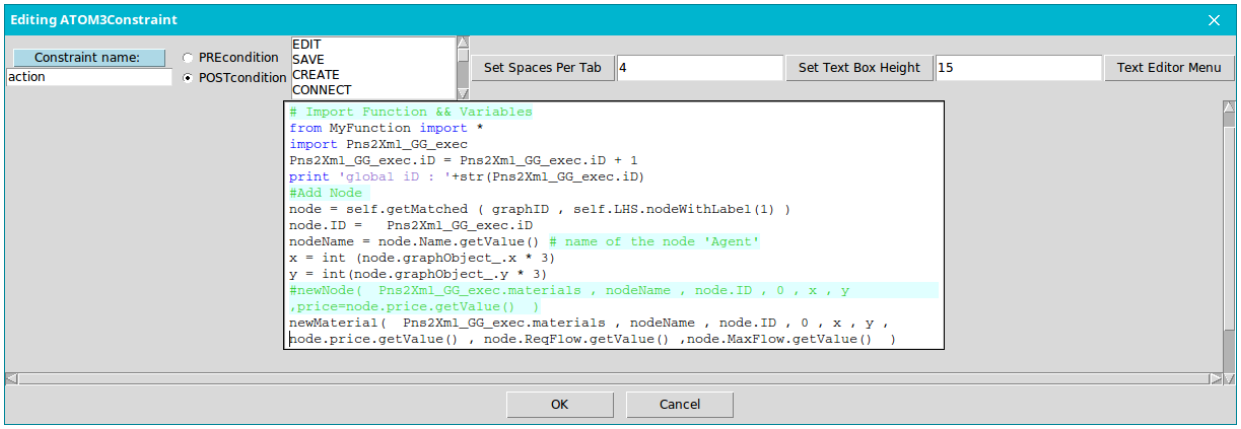


Figure 4.24: Action of Raw Material for Transfromation

First line we import MyFunction is external file, Python code to be easy to manipulate and this is the function to add new material (raw , intermadiate, final) and for the operating unit is the same function

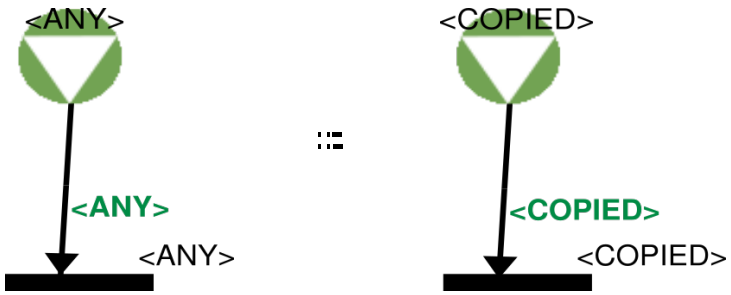
```
1  def newMaterial( parent=None, name= "None", ID="0", Type= "None", xN ='-1',
       yN ='-1' ,price='-1' ,reqF = 0 ,maxF =999999 ):
2    #Create new Node has the Material Name...and add it into the parent node
3    node = ET.SubElement(parent, "Material",ID=str(ID),Name=name,Type=str(
       Type))
4      print 'add Cords into node    -NewNode-'
5    #Create Corde for the node material
6    cords = ET.SubElement(node, "Coords")
7    ET.SubElement(cords, "X").text = str( xN )
8    ET.SubElement(cords, "Y").text = str( yN )
9    #Test if the patameter is not null or empty
10   if price != '-1' or  reqF != 0  or maxF !=999999 :
11     ParameterList = ET.SubElement(node, "ParameterList")
12     ET.SubElement(ParameterList, "Parameter",Name="price",Value=str( price)
         )
13     if reqF == 0 : reqF = '-1'
14     ET.SubElement(ParameterList, "Parameter",Name="reqflow",Value= str(reqF
       ) )
15     if maxF == 999999 : maxF = '-1'
16     ET.SubElement(ParameterList, "Parameter",Name="maxflow",Value=str(maxF)
         )
17   #Create Label for the MaterialNode
18   label = ET.SubElement(node, "Label",Text = str(name))
19   offset = ET.SubElement(label,  "Offset")
20   ET.SubElement(offset, "X").text =   str(5)
21   ET.SubElement(offset, "Y").text =   str(0)
22   return node
```

Listing 4.1: Python Function Material

2) Transform Edge between two entitie into XML code : in this transformation we want to create link between nodes Material and Operating unit and we have 4 type of Edges :

- Edges from raw material into Operating unit
- Edges from intermediate material into Operating unit
- Edges from Operating unit into final material
- Edges from Operating unit into intermadiate material

The figure 4.25 represent the first Edge type



(a) LHS (b) RHS

Figure 4.25: Edges from Raw Material into Operating unit to xml code

a) Condition for rule 4.25 it is the same with figure 4.23 but here we select the edges

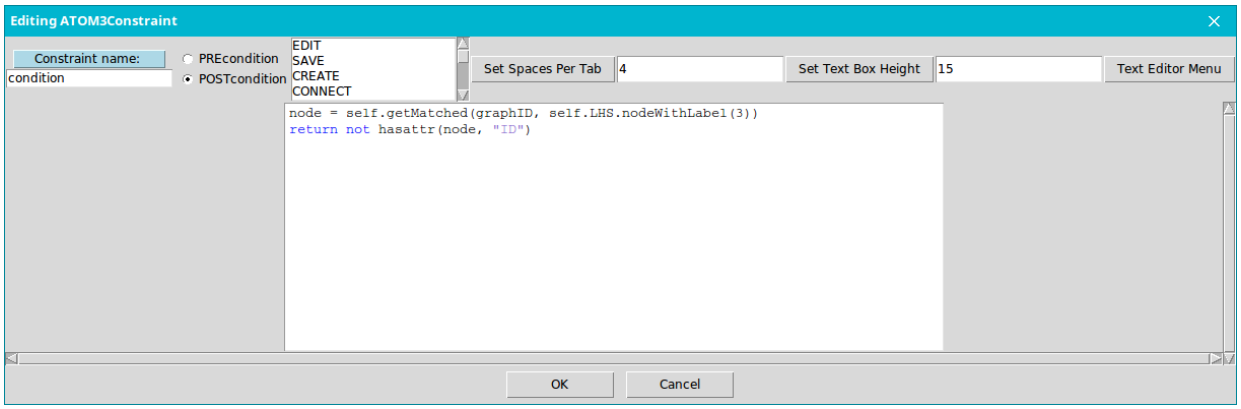


Figure 4.26: Condition of Edges (RawMaterial to Operating unit)

b) Action for rule 4.25 we add this action to Transform the Edges between Any Material and Any Operating unit into xml file, or to export the entitie from Graphical presentation into Xml Code

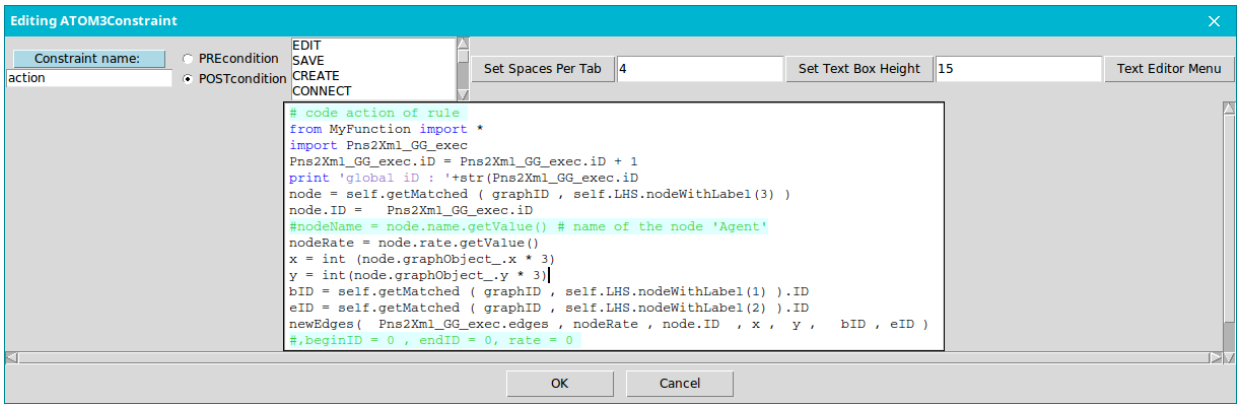


Figure 4.27: Action of Edges (RawMaterial to Operating unit)

and this code represent a Function to add new Edges into the Tree code to export the tree later

```
1  def newEdges( parent, name, ID, xN, yN  ,beginID, endID  ):
2    #Create new Node and add it into parent Node
3    node = ET.SubElement(parent, "Edge",ID=str(ID),BeginID=str(beginID),EndID
       =str(endID),Rate=str(name),Title=str(name), ArrowOnCenter="true",
       ArrowPosition="50")
4    #Create Corde for the node Edges
5    cords = ET.SubElement(node, "Coords")
6    ET.SubElement(cords, "X").text = str( xN )
7    ET.SubElement(cords, "Y").text = str( yN )
8    #Create Label for the Edges
9    label = ET.SubElement(node, "Label",Text = str(name))
10   offset = ET.SubElement(label,  "Offset")
11   ET.SubElement(offset, "X").text =   str(5)
12   ET.SubElement(offset, "Y").text =   str(0)
13
14   return node
```

Listing 4.2: Python Function new Edges

❑ **Final Action :** it is to create the xml file instead after we generate the graph variable from previous rules 4.28.

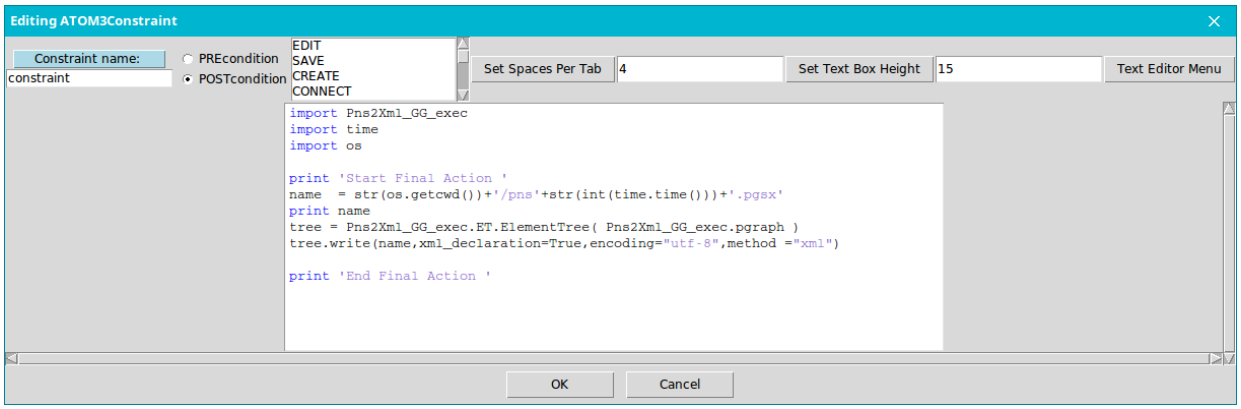


Figure 4.28: Code Final Action

4.4 Optimum structure of MaS

To optimize the multi-agent system in Process Process synthesis, we choose a tool called P-Graph Studio

figure 4.29 represent multi agent system, Previously mentioned in figure 4.20 After we

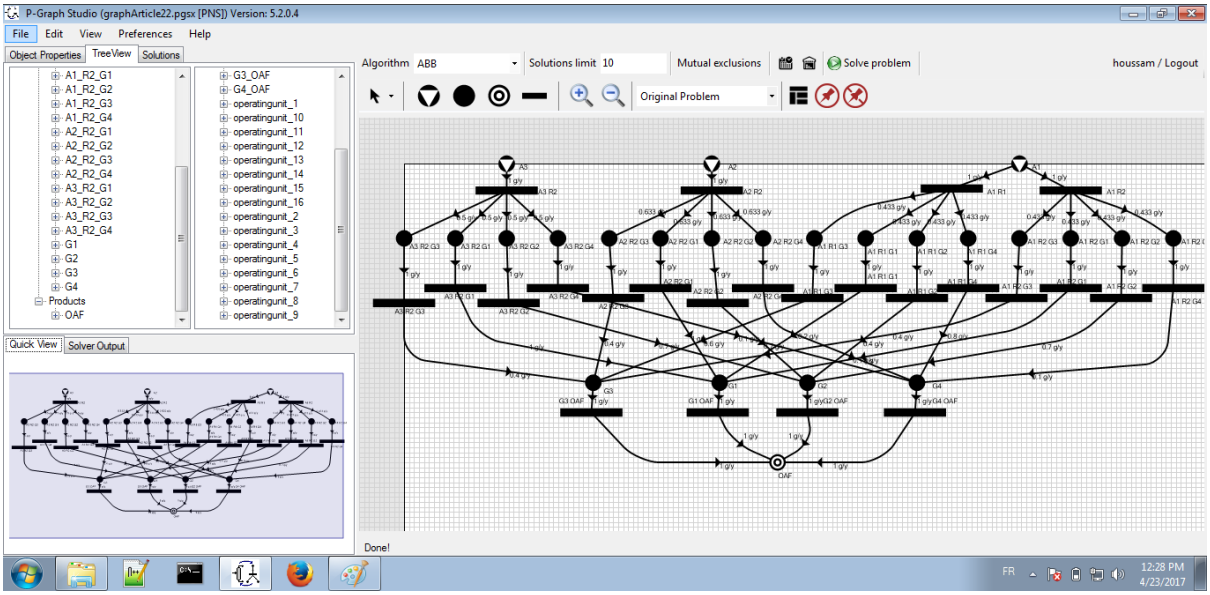


Figure 4.29: Multi Agent System in PGraph Studio

use this tool to optimise, we got this figure 4.30

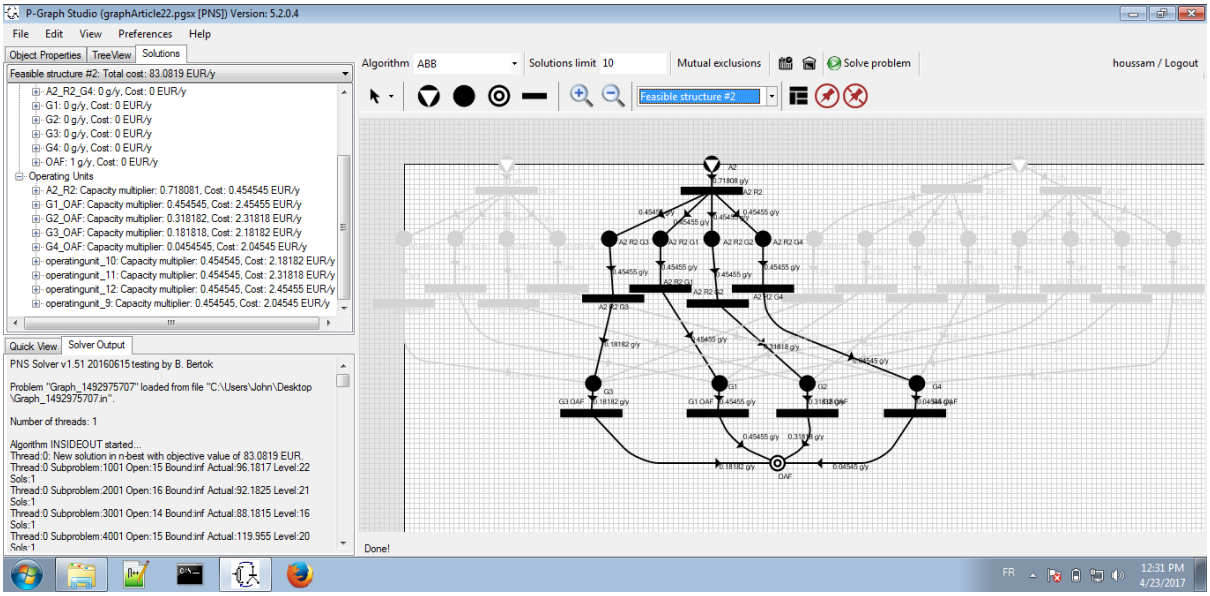


Figure 4.30: Multi Agent System in Optimum structure

PGraph Studio Generate 2 Files :

❑ Graph.in and contain :

1. information about Material, price, quanity
2. information about operating unit, income material and outcome material

```
material-to_operating_unit_flow_rates:
A1_R1: A1 => 0.433 A1_R1_G1 + 0.433 A1_R1_G4 +
        0.433 A1_R1_G3 + 0.433 A1_R1_G2
A1_R2: A1 => 0.433 A1_R2_G4 + 0.433 A1_R2_G3 +
        0.433 A1_R2_G2 + 0.433 A1_R2_G1
A2_R2: A2 => 0.633 A2_R2_G4 + 0.633 A2_R2_G3 +
        0.633 A2_R2_G2 + 0.633 A2_R2_G1
A3_R2: A3 => 0.5 A3_R2_G4 + 0.5 A3_R2_G3 +
        0.5 A3_R2_G2 + 0.5 A3_R2_G1
G1_OAF: G1 => OAF
G2_OAF: G2 => OAF
G3_OAF: G3 => OAF
G4_OAF: G4 => OAF
```

Listing 4.3: Part from Graph.in

4.5 Conclusion

We have presented in this chapter our approach of transformations of the MaS in OMACS framework to PNS, And we also presented our the second approach that transforms PNS Model into XML file after that we saw the tool PGraph Studio we use it to optimize multi agent system in pns frame work.

General Conclusion

Through our project, We explained the aim of our project wchich is to find the optimum organization of multi agent system, and this later represented according the OMACS framework.

We have introduced in the first chapter, Motivation and some concept of organization and reorganization in MaS and we present two framework (OMACS, PNS), that allow us to design a MaS.

The Second chapter was about the PGraph Framework and Process Network Synthesis, the basic concept and mathematical definition of this framework, and i mention three algorithm applicable on PNS

In the Third chapter, we explained the concept of model transformation approaches and we concentrated on the Graph transformation approach, followed by an overview about the Tool AToM3, it allow us to define the meta model and generate a formalism from these meta model.

In the fourth chapter, we presented our approach to transforamfor a MaS modeled in OMACS presentation into PNS Model, and we define the meta model for both of the framework (OMACS, PNS) so we can propose our approachs. we propose two approach, the (OMACS TO PNS) approach, its to transform OMACS model into PNS model, and the second approach its to transform the PNS model was generated from the last approach into XML file, to be able to apply ABB in PGraph Studio to find the best (optimum) structure.

Our work depend on tool called PGraph studio, to apply the (Branch Bound) algorithm, because of that we propose as a perspective to implement ABB algorithm using graph grammar to make our approaches autonomous.

Bibliography

[1] V. Dignum, *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models: Semantics and Dynamics of Organizational Models*. IGI Global, 2009.

[2] S. A. Deloach, W. H. Oyenan, and E. T. Matson, "A capabilities-based model for adaptive organizations," *Autonomous Agents and Multi-Agent Systems*, vol. 16, no. 1, pp. 13–56, 2008.

[3] J. C. García-Ojeda, B. Bertok, F. Friedler, A. Argoti, and L. Fan, "A preliminary study of the application of the p-graph methodology for organization-based multiagent system designs: Assessment," *Acta Polytechnica Hungarica*, vol. 12, no. 2, 2015.

[4] H. A. Abbas, S. I. Shaheen, and M. H. Amin, "Organization of multi-agent systems: an overview," *arXiv preprint arXiv:1506.09032*, 2015.

[5] S. A. DeLoach and E. Matson, "An organizational model for designing adaptive multiagent systems," in *The AAAI-04 Workshop on Agent Organizations: Theory and Practice (AOTP 2004)*, pp. 66–73, 2004.

[6] A. Lapouchnian, S. Liaskos, J. Mylopoulos, and Y. Yu, "Towards requirements-driven autonomic systems design," *SIGSOFT Softw. Eng. Notes*, vol. 30, pp. 1–7, May 2005.

[7] S. A. DeLoach and J. C. Garcia-Ojeda, "O-mase: a customisable approach to designing and building complex, adaptive multi-agent systems," *International Journal of Agent-Oriented Software Engineering*, vol. 4, no. 3, pp. 244–280, 2010.

[8] J. Tick, "P-graph-based workflow modelling," *Acta Polytechnica Hungarica*, vol. 4, no. 1, pp. 75–88, 2007.

[9] J. Garcia-Ojeda, B. Bertok, and F. Friedler, "Planning evacuation routes with the p-graph framework," *Chemical Engineering Transactions*, vol. 29, pp. 1531–1536, 2012.

[10] D. Almási, C. Imreh, T. Kovács, and J. Tick, "Heuristic algorithms for the robust pns problem," *Acta Polytechnica Hungarica*, vol. 11, no. 4, pp. 169–181, 2014.

[11] F. Friedler, L. Fan, and B. Imreh, "Process network synthesis: problem definition," *Networks*, vol. 31, no. 2, pp. 119–124, 1998.

[12] C. Holló, *Hálózati folyamatok szintézise*. PhD thesis, szte, 2004.

[13] F. Friedler, K. Tarjan, Y. Huang, and L. Fan, "Combinatorial algorithms for process synthesis," *Computers & chemical engineering*, vol. 16, pp. S313–S320, 1992.

[14] L. Vance, I. Heckl, B. Bertok, H. Cabezas, and F. Friedler, "Designing energy supply chains with the p-graph framework under cost constraints and sustainability considerations," *Computer Aided Chemical Engineering*, vol. 33, pp. 1009–1014, 2014.

[15] "`http://p-graph.com/`." Accessed: 2017-05.

[16] B. Bertok, M. Barany, and F. Friedler, "Generating and analyzing mathematical programming models of conceptual process design by p-graph software," *Industrial & Engineering Chemistry Research*, vol. 52, no. 1, pp. 166–171, 2012.

[17] T. Kühne, "Matters of (meta-) modeling," *Software and Systems Modeling*, vol. 5, no. 4, pp. 369–385, 2006.

[18] B. Selic, "Specification and modeling: an industrial perspective," in *Software Engineering, 2001. ICSE 2001. Proceedings of the 23rd International Conference on*, pp. 676–677, IEEE, 2001.

[19] J. Bézivin and O. Gerbé, "Towards a precise definition of the omg/mda framework," in *Automated Software Engineering, 2001.(ASE 2001). Proceedings. 16th Annual International Conference on*, pp. 273–280, IEEE, 2001.

[20] T. Clark, P. Sammut, and J. Willans, "Applied metamodelling: a foundation for language driven development.," 2008.

[21] J. De Lara, H. Vangheluwe, and M. Alfonseca, "Meta-modelling and graph grammars for multi-paradigm modelling in atom3," *Software & Systems Modeling*, vol. 3, no. 3, pp. 194–209, 2004.

[22] M. D. N. TISSILIA, *Un Cadre Formel pour La Modelisation et L.analyse Des Agents Mobiles*. PhD thesis, Universite Mentouri Constantine, 2013.

[23] J. Álvarez, A. Evans, and P. Sammut, "Mml and the metamodel architecture," in *WTUML: Workshop on Transformation in UML*, 2001.

[24] K. Czarnecki and S. Helsen, "Feature-based survey of model transformation approaches," *IBM Systems Journal*, vol. 45, no. 3, pp. 621–645, 2006.

[25] K. Czarnecki and S. Helsen, "Classification of model transformation approaches," in *Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture*, vol. 45, pp. 1–17, USA, 2003.

[26] F. E. M. C. G. Kardas, "*review of model-to-model transformation approaches and technologies," *ITEA3*, 2015.

[27] H. Ehrig, K. Ehrig, C. Ermel, F. Hermann, and G. Taentzer, "Information preserving bidirectional model transformations," in *International Conference on Fundamental Approaches to Software Engineering*, pp. 72–86, Springer, 2007.

[28] M. Andries, G. Engels, A. Habel, B. Hoffmann, H.-J. Kreowski, S. Kuske, D. Plump, A. Schürr, and G. Taentzer, "Graph transformation for specification and programming," *Science of Computer programming*, vol. 34, no. 1, pp. 1–54, 1999.

[29] I. Ammari, *Transforme Diagrammes d.etat transition vers GSPN*. 2016.

[30] H. HACHICHI, *Test formel des systemes temps reel : Approche de transformation de graphes*. PhD thesis, Universite Mentouri de Constantine, 2013.

[31] "`http://atom3.cs.mcgill.ca/`." Accessed: 2017-05.

[32] J. De Lara and H. Vangheluwe, "Atom3: A tool for multi-formalism and metamodelling," in *International Conference on Fundamental Approaches to Software Engineering*, pp. 174–188, Springer, 2002.