Contents

List of Figures

List of Tables

General Introduction

Systems are becoming more complex, in part due to increased customer requirements and the expectation that applications should be seamlessly integrated with other existing, often distributed applications and systems. In addition, there is an increasing demand for these complex systems to exhibit some type of intelligence as well.

Multi-agent systems tend to be more robust and, in many cases, more efficient (due to their ability to perform parallel actions) than single monolithic applications. In addition, the individual agents tend to be simpler to build, as they are built from a single agents perspective.

Graphs represent a graphical and direct tool for visualizing the complex structure of a system. It is a practical and intuitive tool for modeling. And when we talk about modeling multi-agent system we have serval framawork to use OMACS , PNS Are very practical examples of modeling graphs

OMACS is a framework to allow us to define a graph represent a Multi-agent System with his component (Agents , Roles , Capabilities , Goals) and the relation between these components .

PNS is also an other framework basically is developed for chemical reaction systems , each node in this framework represent a material and between two material there is a transition they called it operating unit , but we will use it to represent a multi agent system .

And it have some algorithms we need in our research , the goal of our research is to optimize a multi agent system by applying one of these algorithms , because of that we need to propose a new approach allow to transform a OMACS graph to PNS graph .

Our approach is base on Graph Transformation in AToM3 Tools , and this document will show you our work and the framework I used in three chapiter :

1. Chapter 1 : First Framework OMACS
   (a) Introduction
   (b) Definition
   (c) Main Element
   (d) Relation between Elements
   (e) Conclusion

2. Chapter 2 : Second Framework PNS
   (a) Introduction
   (b) Definition of P-Graph
   (c) Definition of PNS
   (d) Mathematical Definition
   (e) Algorithm
   (f) Conclusion
3. Chapter 3 : Our Approach of transformation and the tool i will use for transformation $AToM^3$ and the steps of transformation , some tips how to use this tool and finally we use other tools to apply the algorithm of optimization which called P-Graph Sudio.

Chapter 1

Organization Multi-Agent System

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 A set of methodologies a selection of design processes , and a collection of frameworks are available in the literature to provide the basis for constructing sophisticated autonomous multi-agent organizations . [1]

Moreover, a set of metrics and methods have been suggested with the intention of providing useful information about key properties (e.g., complexity, flexibility, self-organized, performance, scalability, and cost) of these multi-agent organizations . [1]

This chapter introduces a suite of technologies for building complex, adaptive systems. It is based in the multi-agent systems paradigm and uses the Organization Model for Adaptive Computational Systems (OMACS). And presents a suite of technologies including the Organization-based Multiagent Systems Engineering (O-MaSE) methodology [2]

Framework of Modilization MaS

1.1 Organization Multi-Agent System Engineering

1.1.1 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. [3]

The O-MaSE Process Framework is based on the OPEN Process Framework (OPF) and uses the OPF metamodel , as shown in Figure 1.1 . [2]

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).

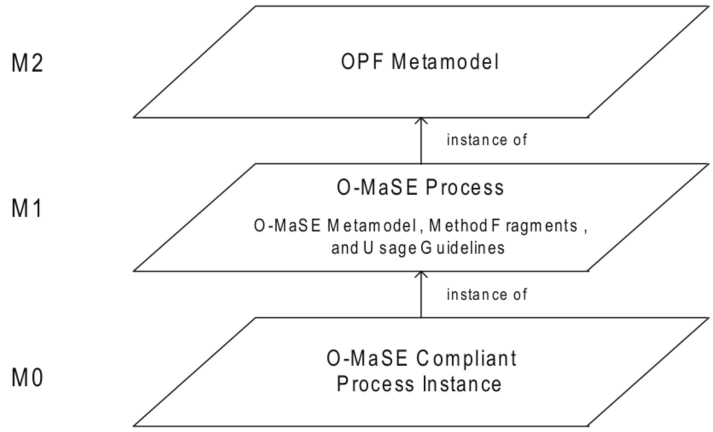


Figure 1.1: O-MaSE Process Framework [2]

1.1.2 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. [4]

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. [4]

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

[1]http://agenttool.cs.ksu.edu/

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

1.2 Organization Model for Adaptive Computational System

1.2.1 Overview of OMACS

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

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 . [2]

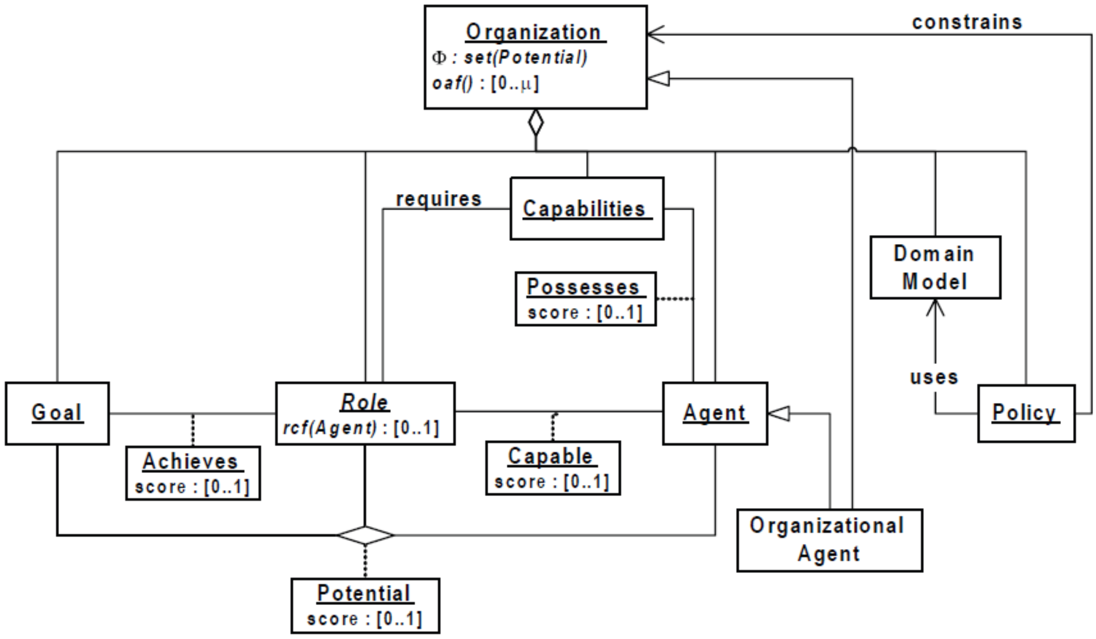


Figure 1.2: OMACS meta-models [1]

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

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)

[2]Agent Group Roles

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 $\times$ C $\rightarrow$ [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 agent?s capability

OMACS also includes two additional derived functions to help compute potential assignment values: capable and potential. [1]

capable : function A $\times$ R $\rightarrow$ [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)

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

potential : function A $\times$ R $\times$ G $\rightarrow$ [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.. ∞] he selection of φ from the set of potential assignments is defined by the organization?s 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

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

1.2.2 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.

1.2.2.1 Goals

Artificial organizations are designed with a specific purpose, which defines the overall function of the organization. [5]

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 . [5]

Figure 1.3: Goal Node

1.2.2.2 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. [5]

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. [5]

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). [2]

Figure 1.4: Role Node

1.2.2.3 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. [5]

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, 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.

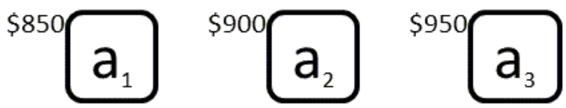


Figure 1.5: Agent Node

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. [5]

[3] the same capable function

1.2.2.4 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. [5]

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 [5]

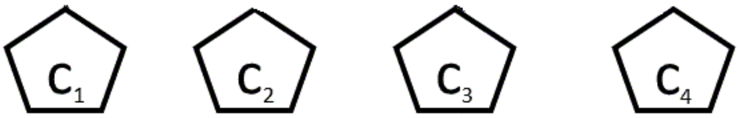


Figure 1.6: Capabilities Node

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

1.2.2.5 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 .[4-OAMCS]

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 . [4-OAMCS]

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)

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

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

1.2.2.6 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. [2]

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.

1.2.2.7 Domain Model

The domain model, $\sum$, is used to define object types in the environment and the relations between those types. [5]

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. [5]

1.2.3 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.

1.2.3.1 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. [2]

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. [2]

Figure 1.7 represent the Relation between Role and Goal

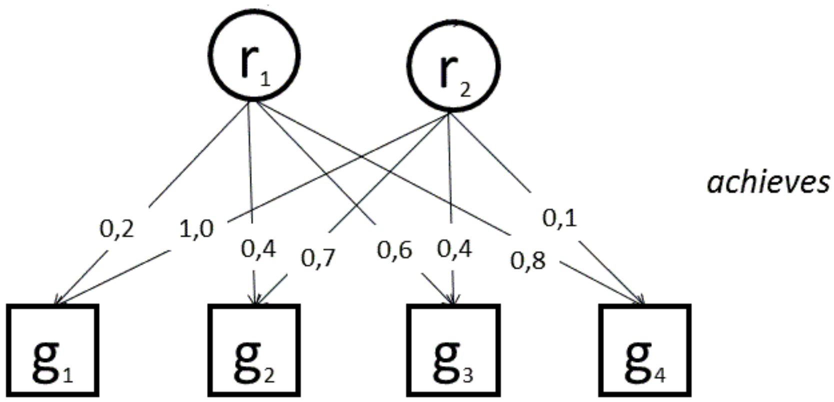


Figure 1.7: The Achivement Relation

1.2.3.2 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. [5]

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 [5] and this Figure 1.8 represent the Relation between Role and Capabilities

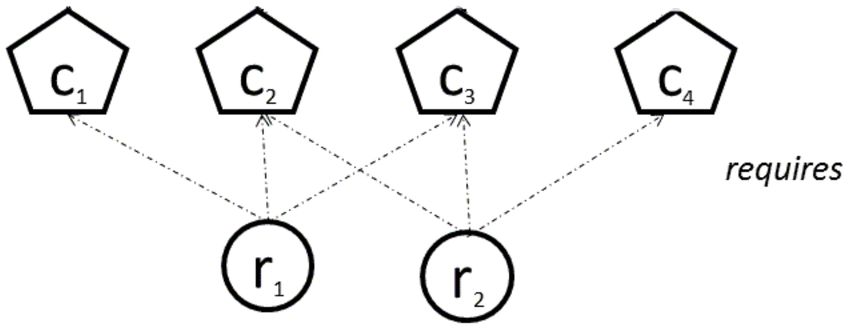


Figure 1.8: The Requirement Relation

1.2.3.3 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. [2]

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. [2] and this Relation represented in the next Figure 1.9

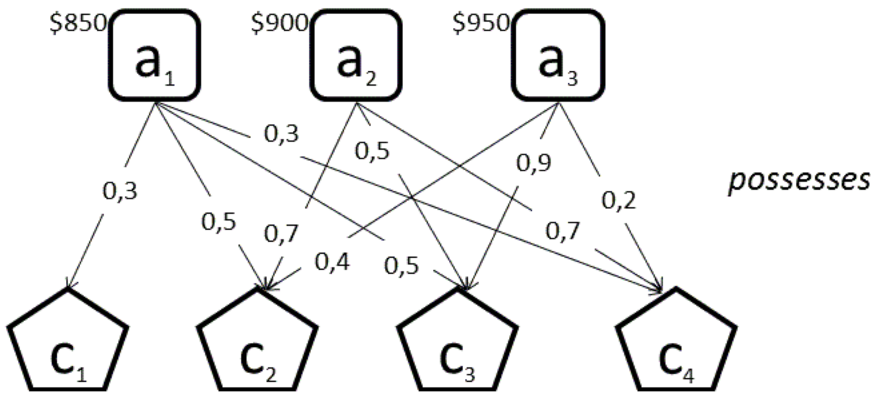


Figure 1.9: Possesses Relation

1.2.3.4 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] . [5]

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 . [5]

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

The main goal of OMACS is to provide a mechanism to assign goals to agents [5]

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. [5]

1.2.3.5 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]. [2]

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. [2]

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

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

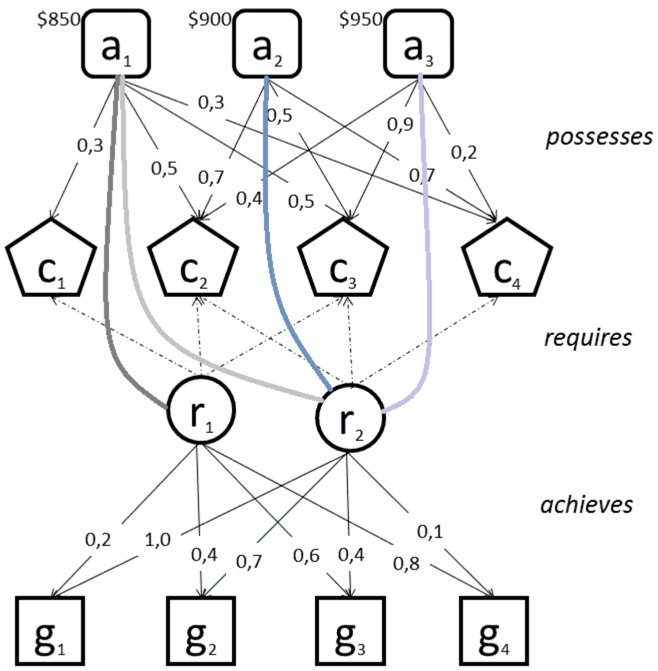


Figure 1.10: MaS with Capable of Playing Relation [1]

1.3 Organization & Reorganization Dynamic System

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. [6]

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 [6]

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 [6]

1.3.1 Organization and Reorganization

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. [5]

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. [5]

1.3.1.1 Goal Set Changes

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

1. insertion of a new goal

2. goal achievement

3. goal failure

is discussed below.

The first situation deals with new goals being added to G. However, [5] 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. [5]

1.3.1.2 Agent Changes

The second type of change that triggers reorganizations are changes to the set of agents, A, or their individual capabilities. [5]

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. Likewise,when an agent that is part of φ loses a capability that negates its ability to play a role that it is assigned, reorganization must occur as well.

In general, when changes occur in an agent?s 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. [5]

1.3.2 Reorganization

Reorganization is the process of changing the assignments of agents to roles to goals as specified in φ. [5]

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

1.3.2.1 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 [7]

- ❑ 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: [7]

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

1.3.2.2 General Purpose Reorganization Examples

For general-purpose reorganization, the Developers have developed several reorganization algorithms that give us a default reorganization capability. When a reorganization trigger occurs, general-purpose reorganization algorithms can be used to find appropriate assignments to achieve the organizations goals, if possible. [5]

To compute the best reorganization, an algorithm that simply optimizes the organizations oaf might seem appropriate; however, this approach is short sighted. First, it does not deal with the cost associated with reorganizing and, second, it does not consider the reason reorganizing was initially undertaken. Exploiting reorganizing costs requires a distributed solution since the cost for robots to change roles is not globally known.

For instance, if an agent is required to perform a complex computation, any effort toward that computation would be lost if the agent was reassigned to another role/goal Considering the reason for reorganization may enable less extensive (and less costly) reorganization. If the reason for reorganizing is to fill a single role, then a total reorganization may be a waste of time and resources. [5]

1.4 Conclusion

This chapter describes how to design adaptive multiagent 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 handeling reorganization system more then other framework

In the following chapter will discuss other model we can use it to modeling the multi-Agent System

Chapter 2

PGraph and PNS

In a process system , raw materials are consumed through various transformation to yield desired products . This is usually accompanied bu the generation of wastes .

Vassels in which these transformations are carried out are termed operating units of the process , a given set of operating units with the plausible interconnections can be described bu a network

The desired products can often be manufactured using some sub networks of this network , thus a given network may give rise to a variety of processes producing the desired products , and each of such process corresponds to sub network which can be considered to be its structure [1]

2.1 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]

2.1.1 General 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 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]

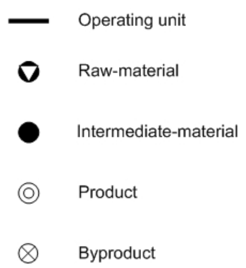


Figure 2.1: notation in pgraph [8]

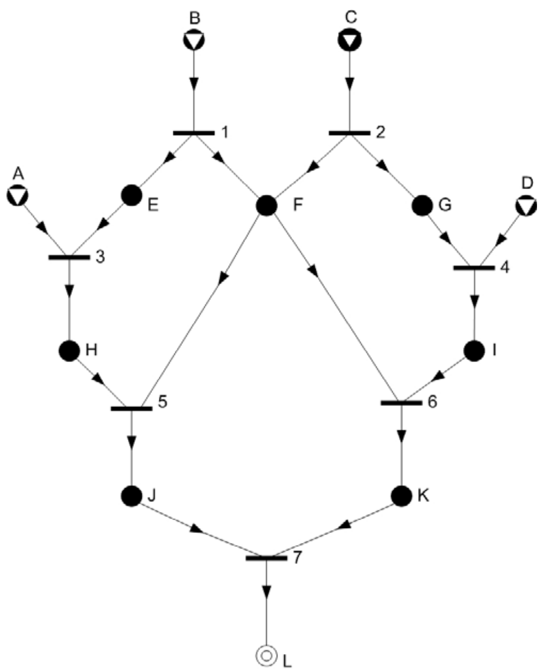


Figure 2.2: example for P-graph [8]

2.2 Process Network Synthesis

In the Process Network Synthesis (PNS for short) 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]

In the 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]

In these P-graphs we have two sets of vertices, one of them contains the possible materials, the other the operating units. The edges lead to an operating unit from its input materials and from an operating unit to its output materials. [12] [10]

Then the subgraphs satisfying some properties describe the feasible processes which produce the desired products from the raw materials. Thus our goal is to find the least expensive such subgraph. In the structural model the amounts of the material flows are not taken into account thus the cost of an operating unit is a constant, and the cost of a subgraph is the sum of the costs of the operating units contained in it. [10] [11]

2.2.1 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 $M \cup O$, 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

2.2.2 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

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

2.2.3 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. [1] [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 [1]

2.2.3.1 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]

2.2.3.2 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]

2.2.3.3 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]

2.2.4 Comparaison PNS and PetriNet :

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

2.3 Conclusion

We have seen in this chapter the definition of the second framework i use PGraph and the Process Network synthesis , with some Algorithme 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

Graph Transformation Approach

3.1 Introduction

i will show in this chapter my work which is Optimise Multi Agent system represented in OMACS Framework , but the problem is OMACS does not have any Any algorithm to do that , Because of That i represent the second Frameword PNS which contain an Algorithm to find the optimum structure . so we need to work on th Transformation from OMACS MODEL into PNS MODEL [14].

3.2 Model and Meta-Model

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. [15][27matters]
2. The central idea of modeling is to produce a reduced version of the system To determine and evaluate its salient properties. [16][36industrial perspective]
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. [17] [9TowardsAprec]

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

[18] [15 Applied]

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 Re- lationship 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.

[19][18Multi paradigm]

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 . [20] [21]

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 [20, 20]

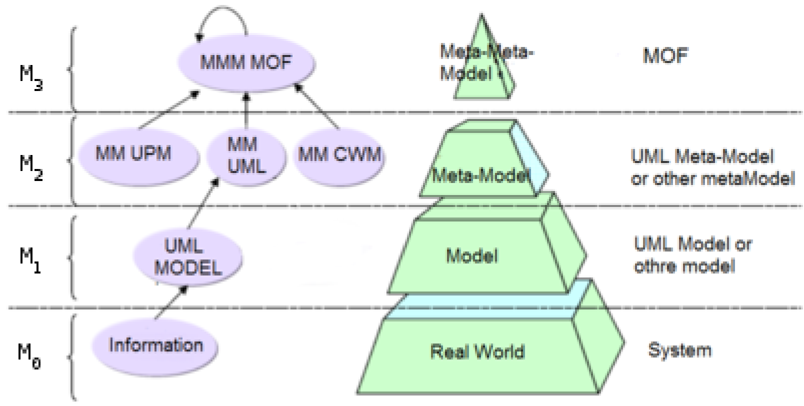


Figure 3.1: Pyramid of Meta-Level [20]

3.3 Transformation Model

3.3.1 Definition

A transformation is the automatic generation of a target model from a Source model, according to a transformation definition.

A definition of transformation Is a set of transformation rules that describes how a source model Can be transformed into a target model [22]

the transformation of the model is For mapping models across domains to analyze or The automatic generation of code from themselves [23]

3.4 Norm of 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:

1. select an applicable rule from the set of rules;
2. Apply this rule to the input graph;
3. Search for another applicable rule
4. do these steps Until no rule can be applied

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 [23, 24]

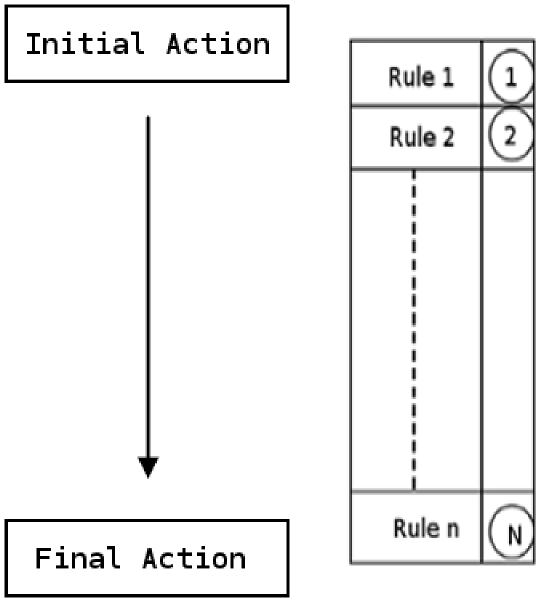


Figure 3.2: Cycle of Transformation [25]

3.5 Graph Grammar

Every Graph Grammer contain a set of rules to use in graph transformation and its define by : [20, 24] R = (LHS, RHS, K, glue, emb, cond).

- ❑ LHS graph of left part.
- ❑ RHS gra5ph 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

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

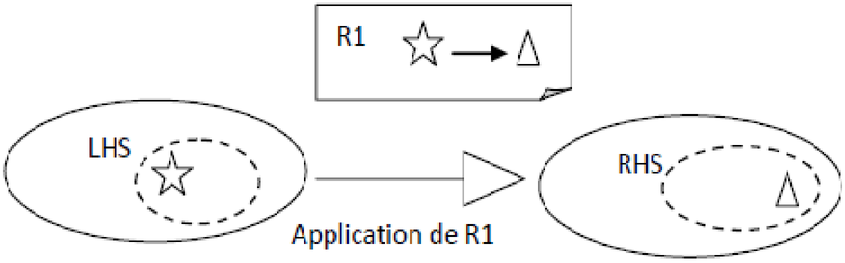


Figure 3.3: Rule Application [25]

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 [20, 26]

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

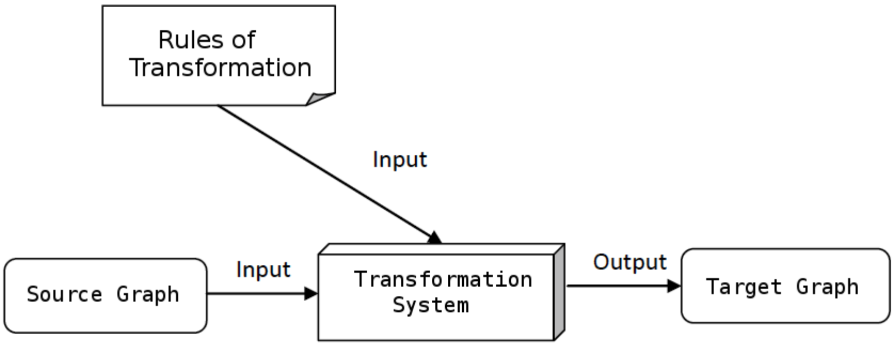


Figure 3.4: Transformation System [25]

3.7 AToM ^{3}

AToM^3 (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. [27, 28]

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 [19][18Multi paradigm]

figure 3.5 illustrates the interface of AToM3

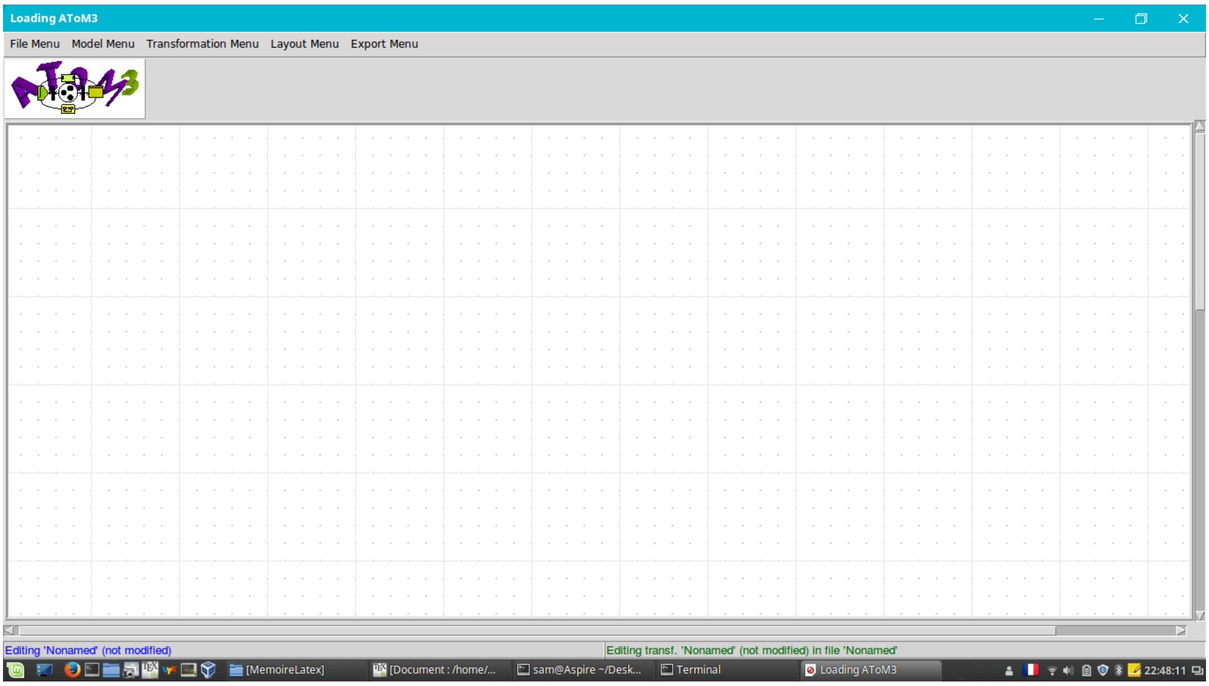


Figure 3.5: AToM ^{3} Window

3.7.1 Classes in AToM^3

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 [19][18Multi paradigm] In AToM^3 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

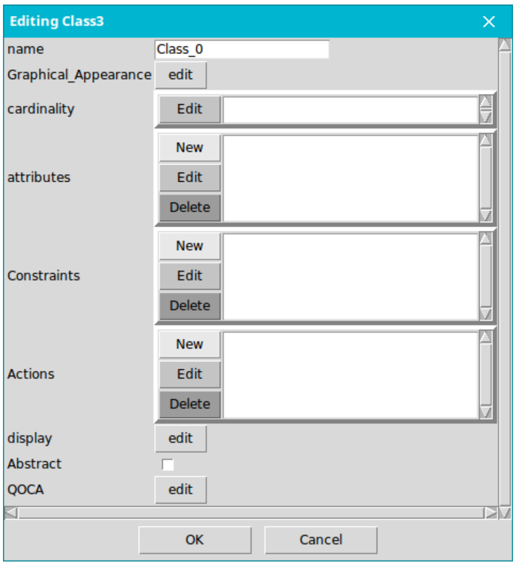


Figure 3.6: Class Editor

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)

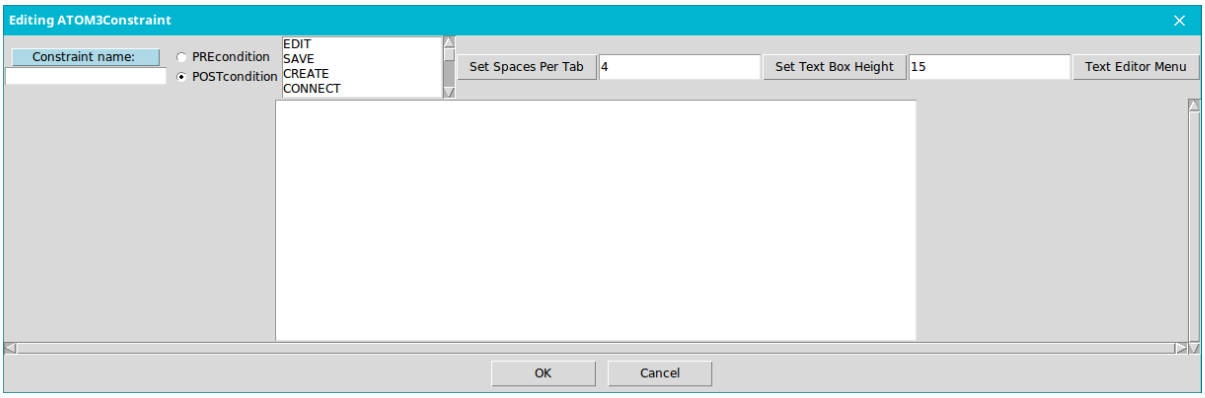


Figure 3.7: Constraint Editor

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)

3.8 Graph Grammer in AToM 3

In AToM3, grammar is a model characterized by

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

and this figure 3.8 Graph Grammar Editor

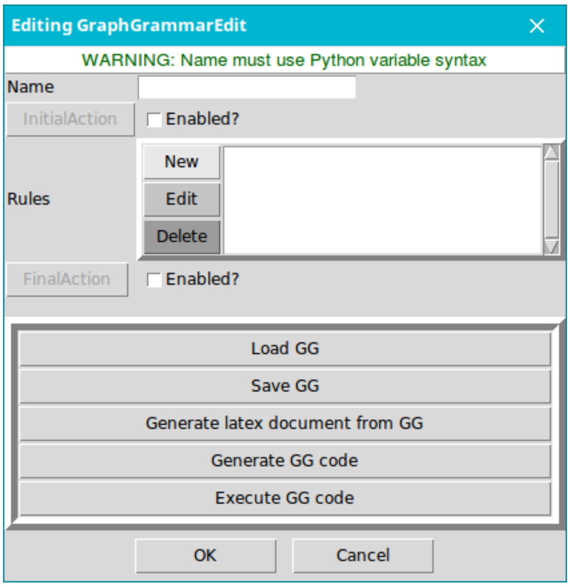


Figure 3.8: Graph Grammar Window

Each rule consists of:

- ❑ A specific name for the rule.
- ❑ A priority indicating the order in which the rule is applied.
- ❑ A Left Hand Side (LHS) which is a graph.
- ❑ A right hand side (RHS) that can be a graph.
- ❑ A condition (Pyton code) that must be checked before the rule is Applied.
- ❑ An action (a Pyton code) that must be executed after the rule is Applied

The rule editor figure 3.9 allows the editing of the different parts of the rule as well as The condition and action of each rule

The condition editor and the action editor of a rule are similar to the editor of Constraints presented in Figure 3.7

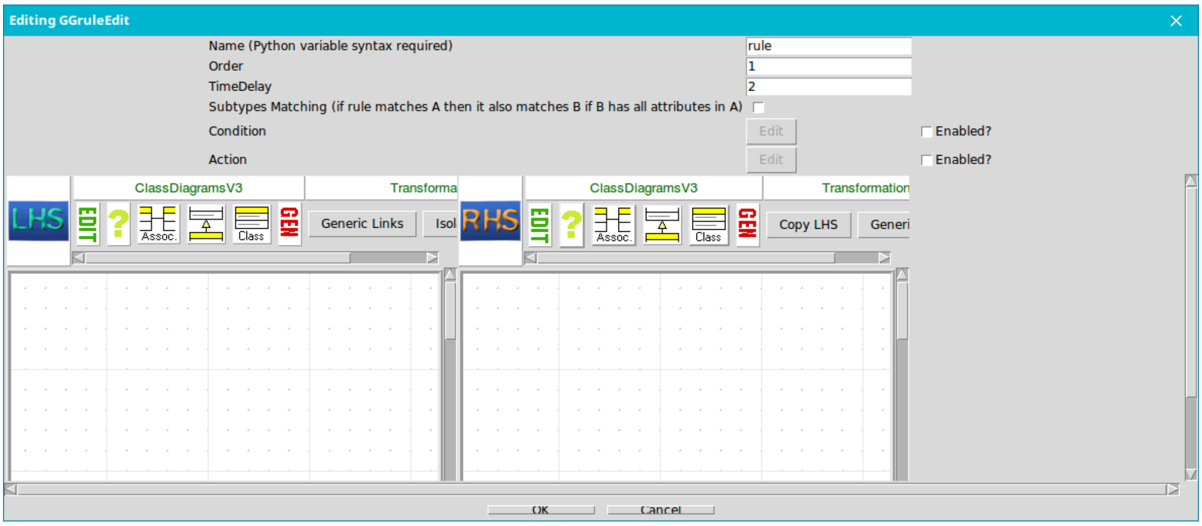


Figure 3.9: Rule Editor

3.9 Implementation of the Transformation (OMACS into PNS)

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

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

3.9.1 Meta-Model of OMACS and PNS

Before starting define our Meta-Model we load the Class Diagramm Formalism , to be able to create , manipulate our Meta-Model in OMACS Meta-Model contain set of classes:

- ❑ Agent
- ❑ Capabilities
- ❑ Role
- ❑ Goal

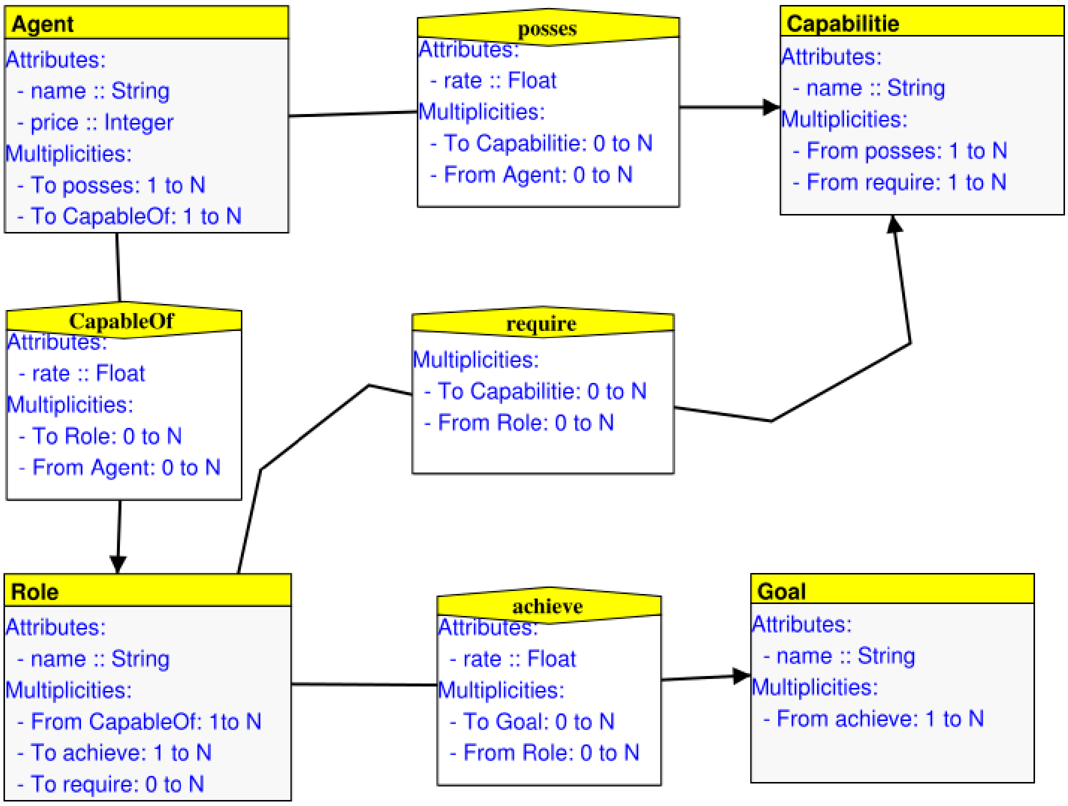


Figure 3.10: OMACS Meta-Model

figure 3.10 illustrate OMACS Meta-Model contain 4 classes and 4 relation 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

the Same Steps for the PNS Meta-Model should load the Class Diagramm first to create PNS Meta-Model , and its contain 4 classes and Relation represent :

- ❑ Raw Material
- ❑ Intermediare Material

- ❏ Final Product
- ❏ Operating Unit

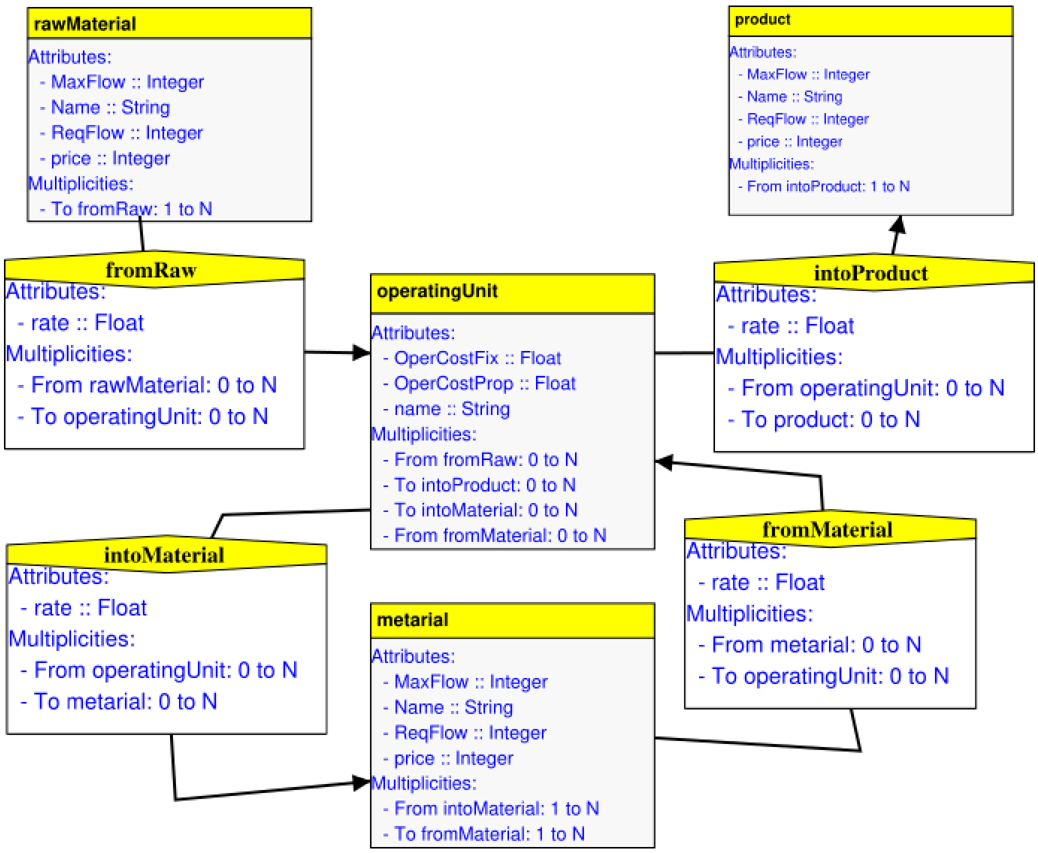


Figure 3.11: PNS Meta-Model

all material in figure 3.11 containt the same Attribute

- ❏ *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

3.9.2 Model in OMACS and PNS

Starting from Meta-Model of OMACS in figure 3.10 , $AToM^3$ generate a formalism of OMACS , this formalism allow us to create our OMACS Model (Multi Agent System) .

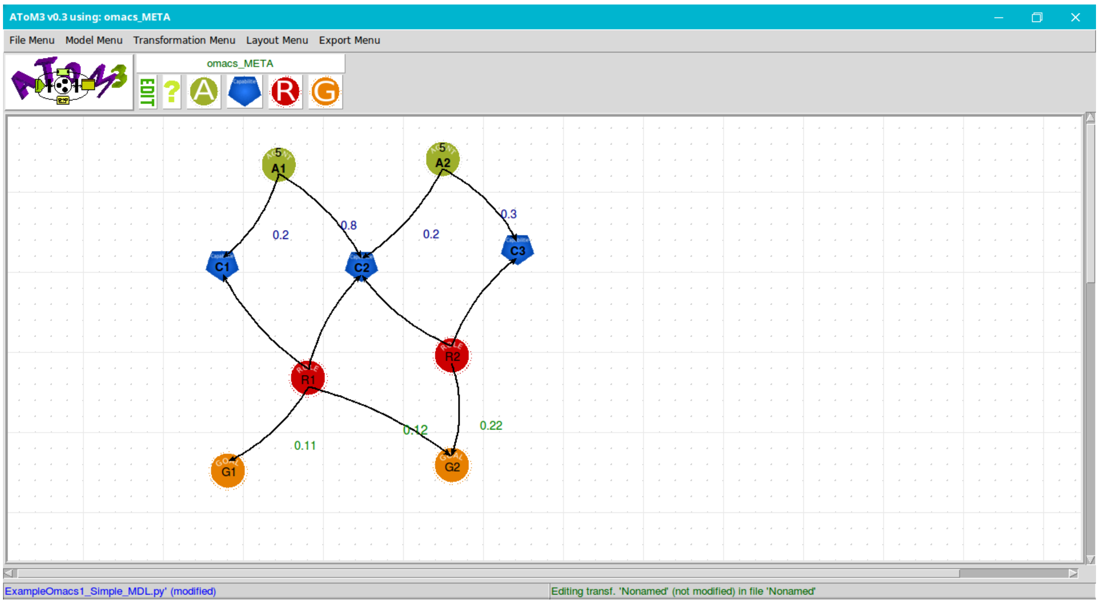


Figure 3.12: Formalism for OMACS Generated by $AToM^3$

Example in this figure 3.12 represent a Multi Agent system which is a component of 2 agent and 3 Capabilities , 2 Roles , 2 Goals

its Also the same steps to be able to create model according the PNS , $AToM^3$ generate a formalism of PNS , load it and use it like figure 3.13

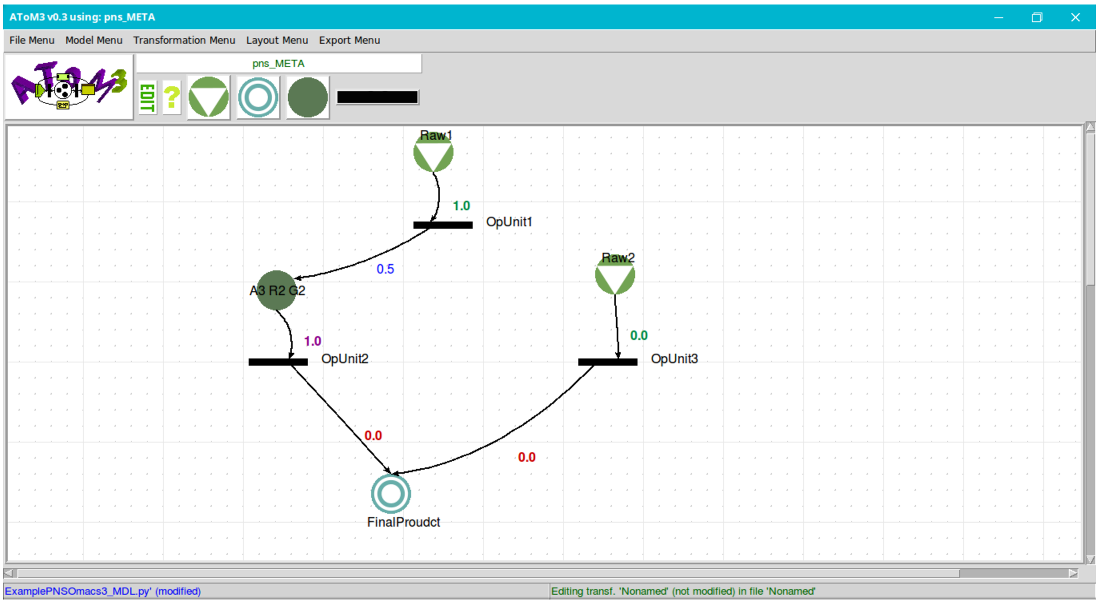


Figure 3.13: ToolBar for PNS Generated by $AToM^3$

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

3.9.3 Rule of Transformation

A Graph grammar is a grammar consisting of a set of rules, allow to Transforming 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 into PNS as target graph.

This grammar is defined using the AToM^3 tool 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

3.9.3.1 Graph Grammar

Our grammar is composed of :

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

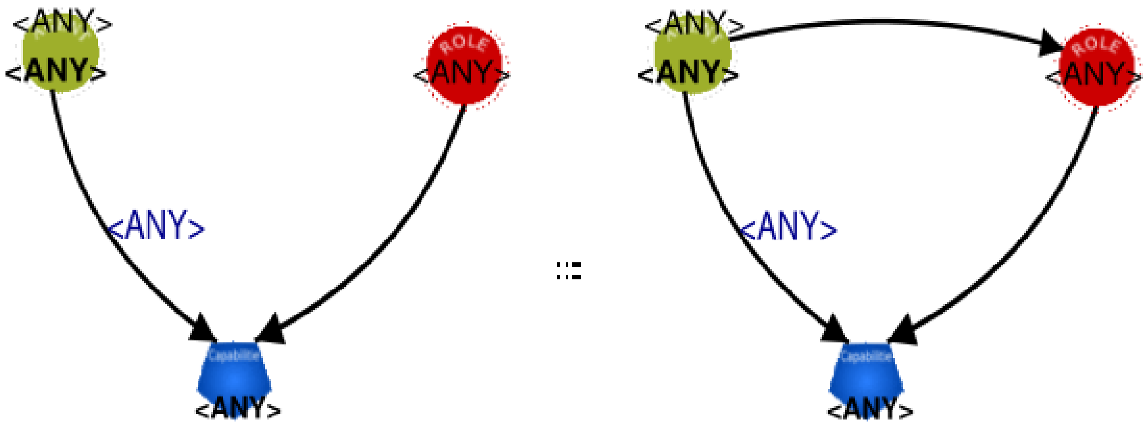
1. Rules for Collect and create relation between some entitie
2. Rules for transforming nodes or links into materials or operating units .
3. Rules to links generated materials and operating units
4. Rules to cleaning unnecessary entities from the result

I Summarize the transformation into the following steps :

- ❑ Create link between the agent and role depending on the common capabilites
- ❑ Generate for every agent in OMACS Model into raw material in PNS Model and the generated material has the same name and price
- ❑ Generate for every goal in OMACS Model into intermediate material in PNS Model and the generated material has the same name
- ❑ The product in the target graph which is PNS Model represent the organization of the OMACS Model
- ❑ Create Operating unit for each direct link between the role and an agent
- ❑ Link the material was generated from goal with final product by operating unit

Here are the most important rules :

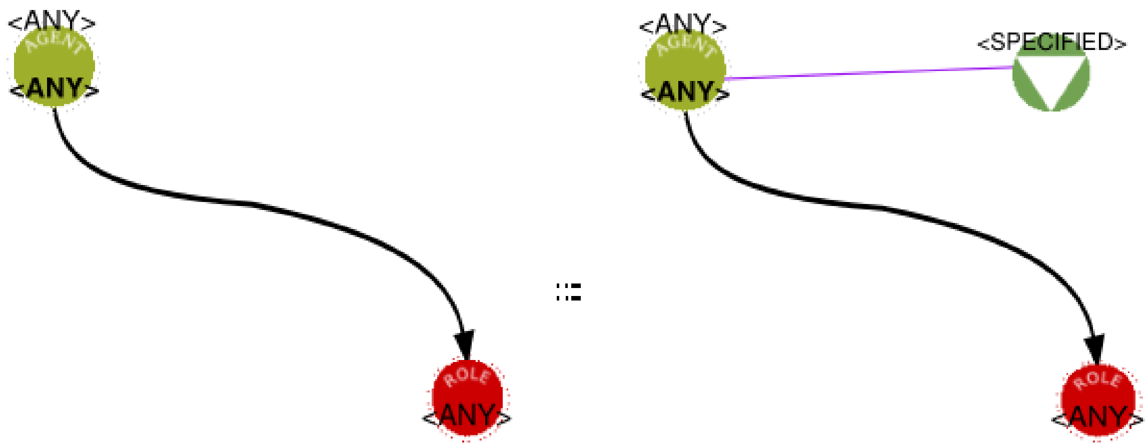
1) Create the direct link between the Agent and the Role (order 1) : The Figure 3.14 illustrate how to create link between the agent and role depending on commun capabilities , order 1 mean it is the first rule applied in the grammar .



(a) LHS (b) RHS

Figure 3.14: Assign Role to Agent

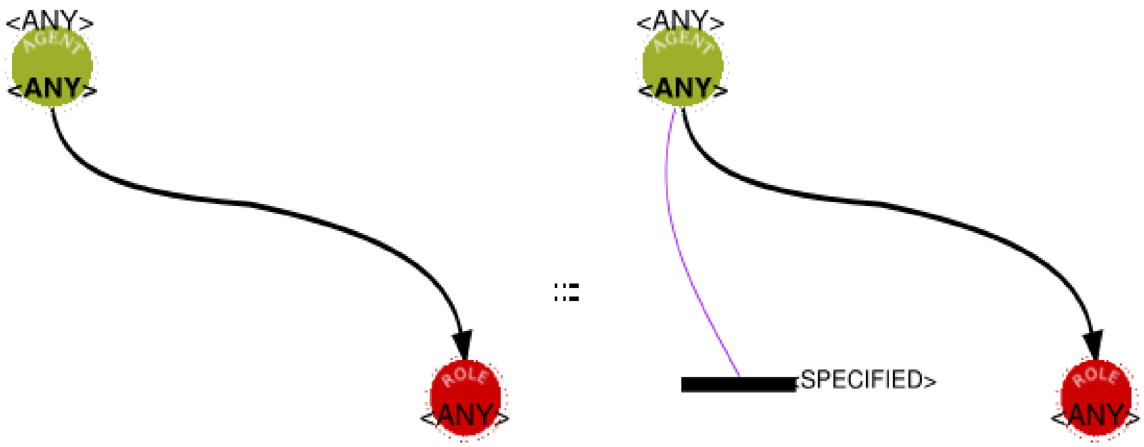
2) Transform Agent to Raw Material (order 5) : Application Of this rule in (figure 3.15) makes it possible to transform every Agent in Multi Agent System into raw material , it has the same name and price .



(a) LHS (b) RHS

Figure 3.15: Transform Agent to Raw-material

3) Transform Link between Agent and Role into Operating Unit (order 7) : This rule (Figure 3.16) allow to transform , the relation capable of playing between an agent and role into operating unit .



(a) LHS (b) RHS

Figure 3.16: Transform CapableOf relation into operating unit

4) Transform Goal to intermediate material (order9) : Application of the rule illustrated in (Figure 3.17) Transform a goal in MaS into intermediate material in process system .

(a) LHS (b) RHS

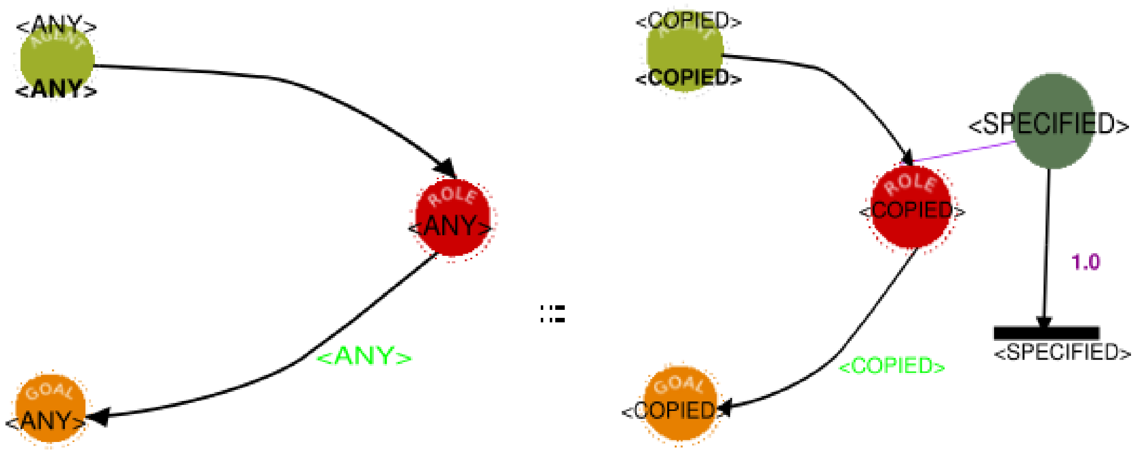
Figure 3.17: Transform Goal to intermediate material

5) Create the Product node (order10) : Create the final material stat in the system like you notice the LHS is empty because this rule applied for one time , and this rule represented in (Figure 3.18) .

(a) LHS (b) RHS

Figure 3.18: Create the final material

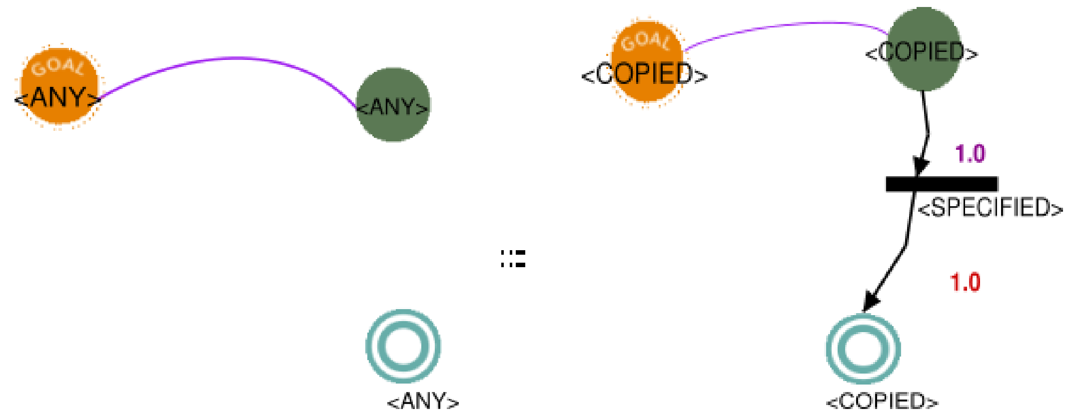
6) Generate auxiliary part (order 13) : The Application of this rule generate the auxiliary part which is intermediate material consumed by operating unit , and the auxiliary part represent an agent capable to playing role in order to achieve a specific goal , you can see that from (Figure 3.19) .



(a) LHS (b) RHS

Figure 3.19: Generate auxiliary part

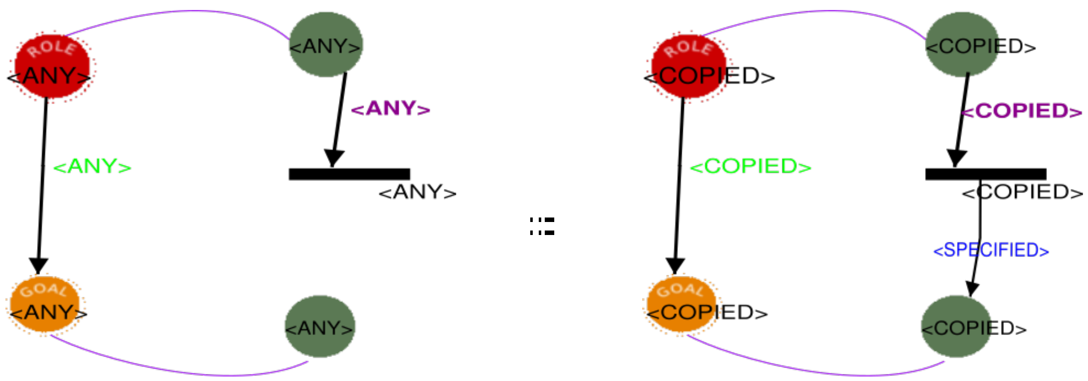
7) Create Operating unit between goal material and the product (order 15) : The rule in (Figure 3.20) to Create an operating unit between the final material and the goals material .



(a) LHS (b) RHS

Figure 3.20: Create Operating unit between goal and OAF

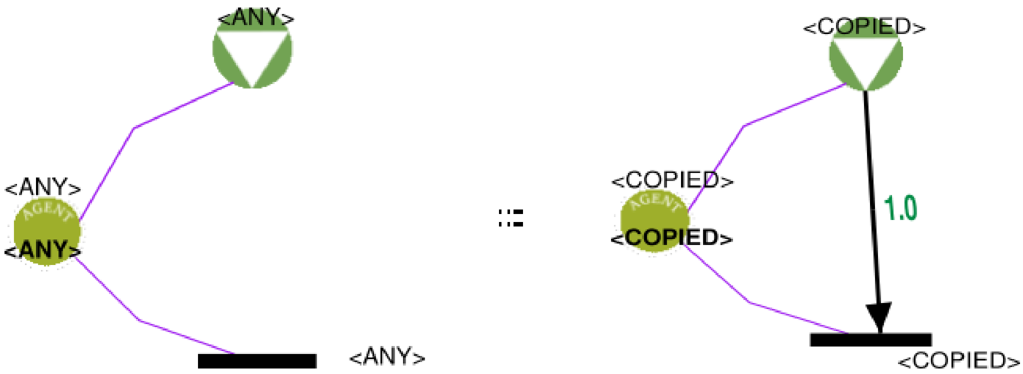
8) Create link between auxiliary part and goal material (order 19) : The rule in (Figure 3.21) describe how to link the auxiliary part with right goal .



(a) LHS (b) LHS

Figure 3.21: link between auxiliary part and goal

9) Create link to consume the raw material by operating unit (order 20) : This Figure 3.22 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 3.22: consume the raw material by operating unit

After that you notice some word in the rules like :

- Left hand side

  1. ANY : its attribue in the node and the engine will select every node with ANY attribute value

- Right hand side

  1. COPIED : copie the attribue value from the node with same GGLabel in LHS
  2. SPECIFIED : specified the attribue value manualy or by code

Now it's supposed to be aware every rule has a condition must be true before the engine run this rule , i mention some these condition in the next part .

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 3.14 represent the rule .

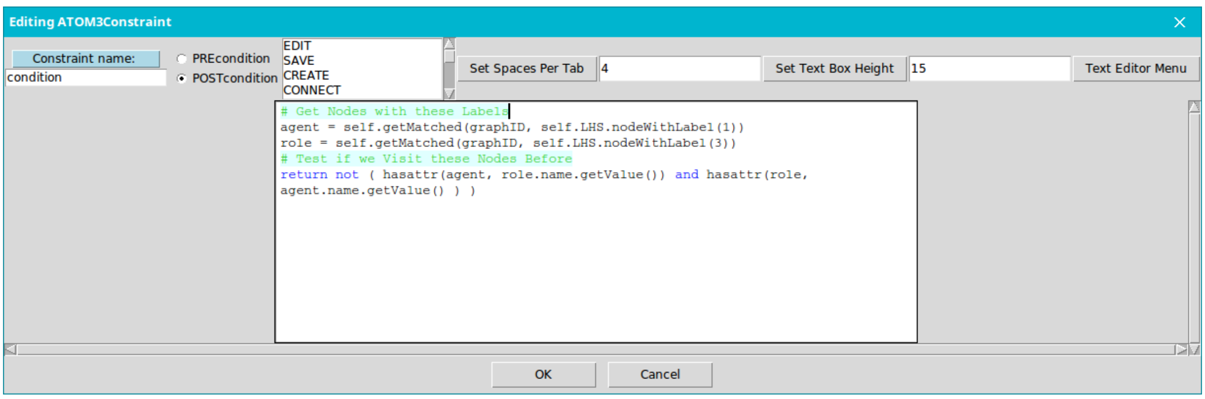


Figure 3.23: Condition link agent with role

2) Condition for Create Final Stat : This conditon test if this attribute Final stat equal zero , in other word we did not visite this rule before , the rule in figure3.18 .

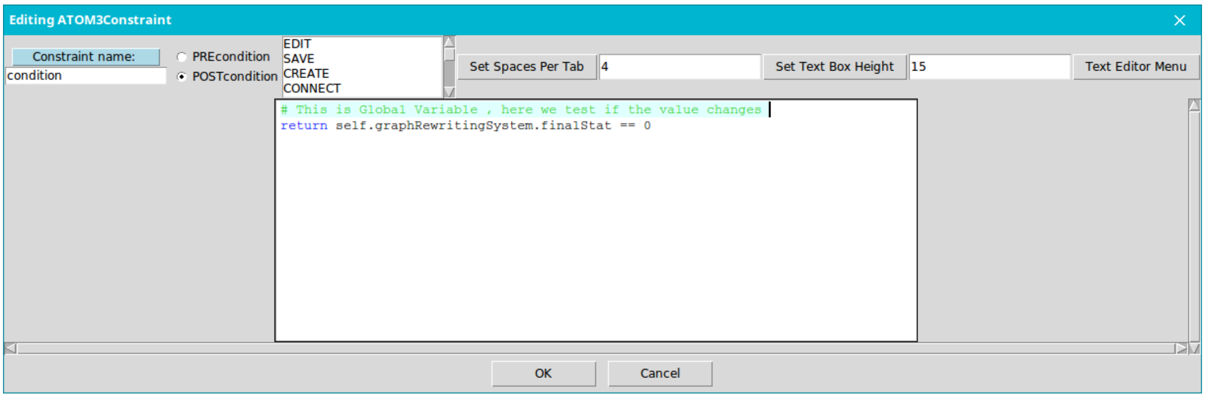


Figure 3.24: Condition of generate final stat

3) Condition of 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 3.21 .

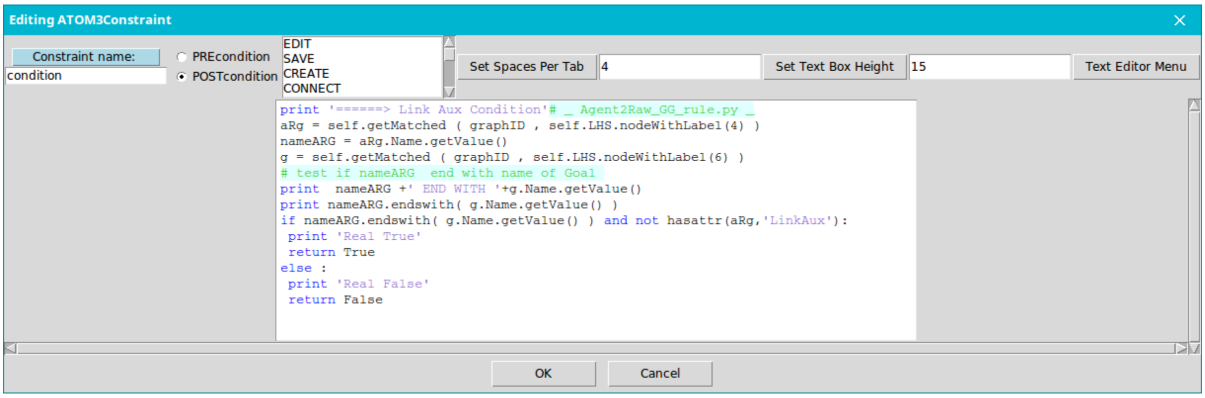


Figure 3.25: Condition of Auxiliary part

3.9.4 Examples

3.9.4.1 Simple multi Agent System

This example represents a very simple system composed of two agent and 3 capabilities , 2 role and 2 goals . Figure 3.26 illustrates this example : The purpose of this example is

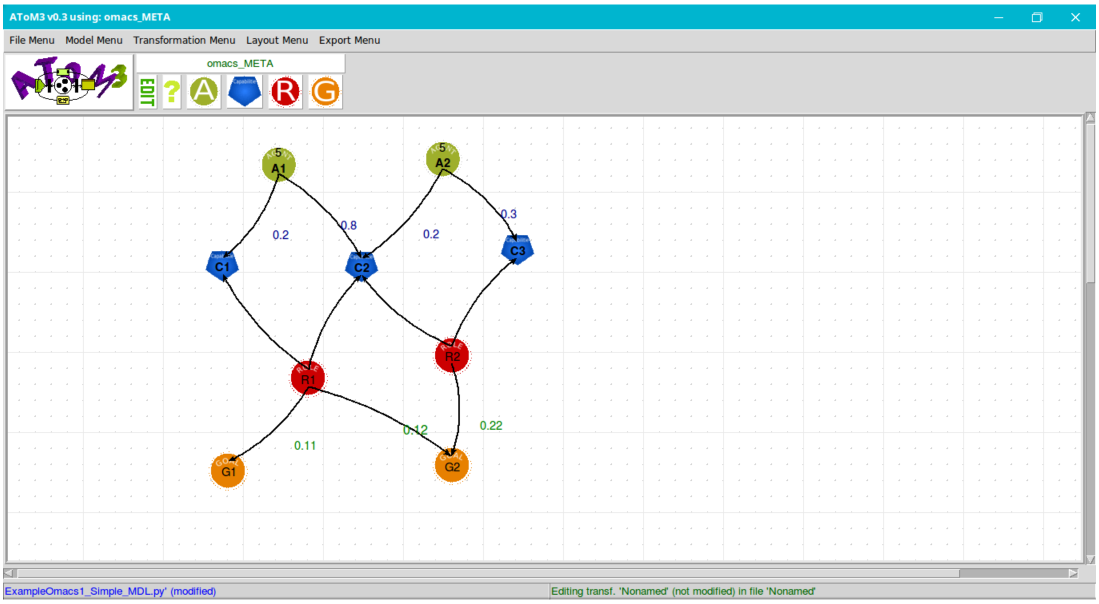


Figure 3.26: Example 1 Multi Agent System

to demonstrate the transformation of multi agent system in OMACS into pns model

Figure 3.27 shows the transformation result. It is clear that the resulting contain 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
- the operating unit and material named A1 R1 G2 represent agent A1 Playing R1 in order to achieve G2 and G2 , any agent in this is capable to play R1 can reach G1 and G2

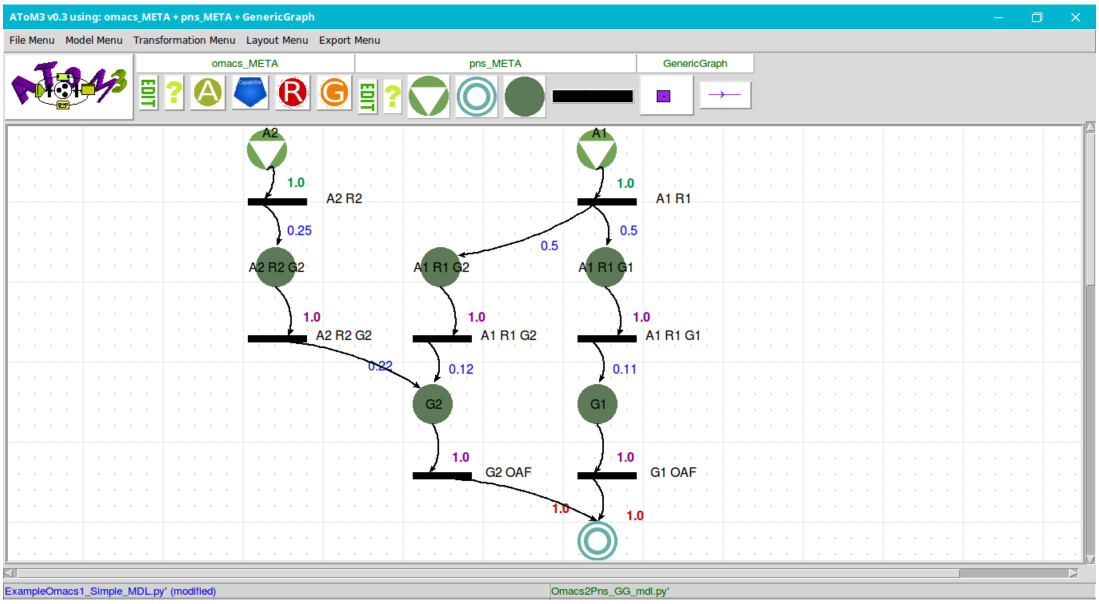


Figure 3.27: Multi Agent System After transformation

3.9.4.2 Complex multi Agent System

This example represents a complex system composed of 3 agent and 4 capabilities , 2 role and 4 goals . Figure 3.26 illustrates this example ,

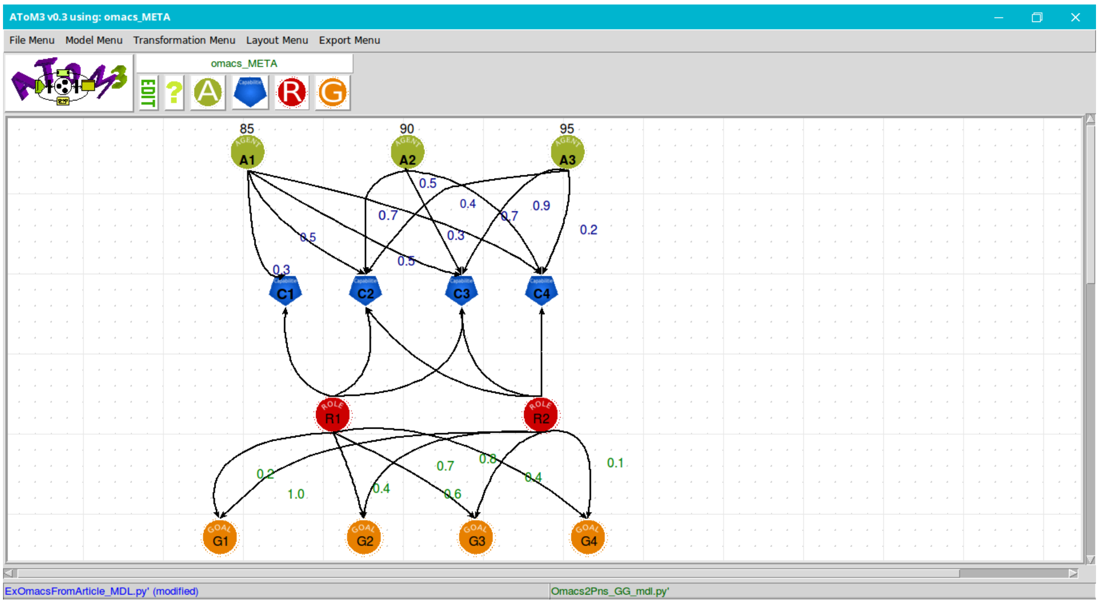


Figure 3.28: Complex Multi Agent System

- every agent possesses set of capabilites for example A1 possese all capabilies in the system
- Role require list of capabilies every agent has the same capabilies or more , so he can play this role
- Role R1 , R2 achieve all goal in the system

- the Agent A1 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 frame into PNS framework we get this model in figure 3.29 and next table represent the value of this 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 3.1: Value of PNS Model

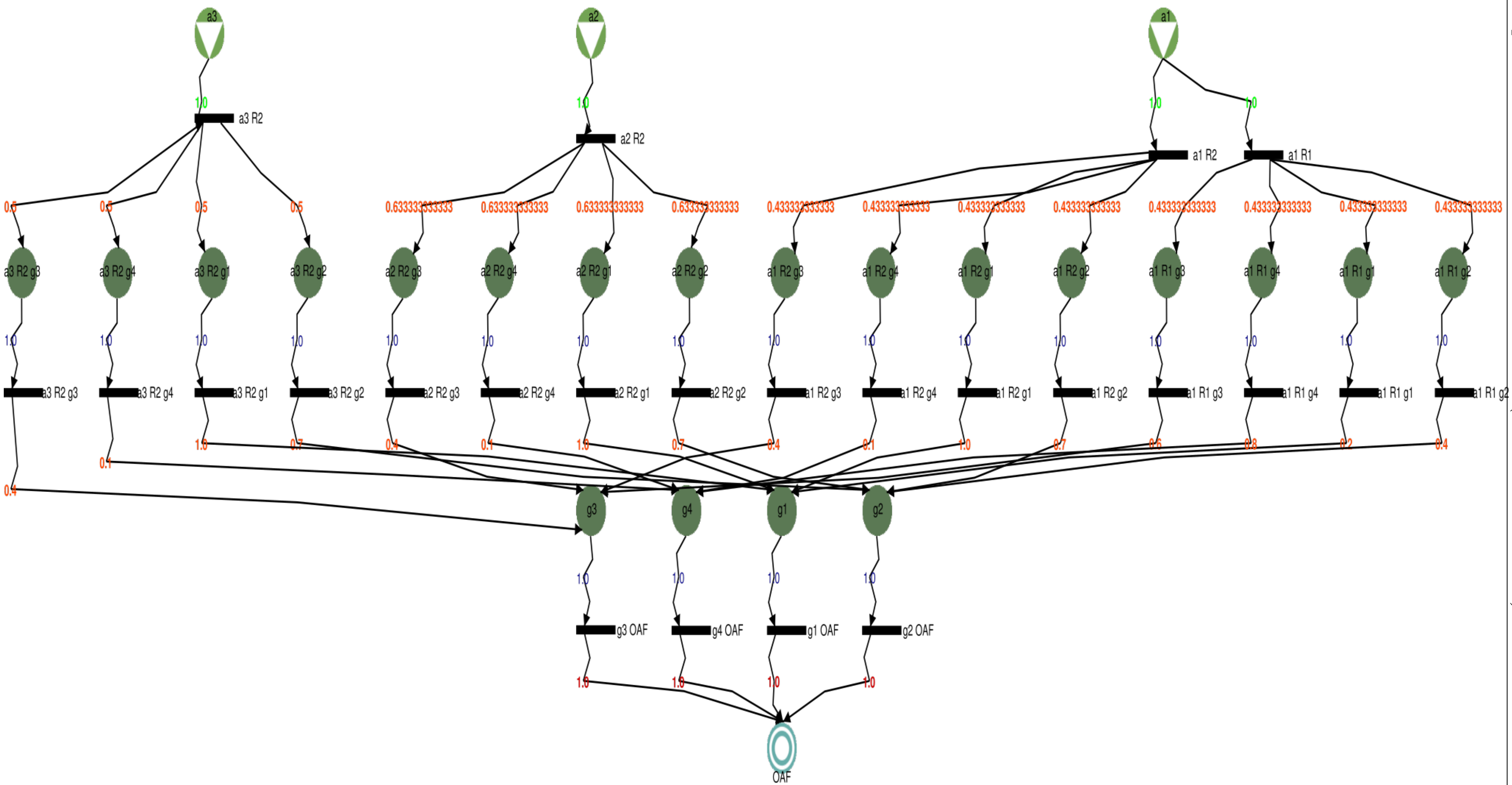


Figure 3.29: Complex Multi Agent System After transforamtion

3.10 Transformation Approach (PNS into XML file)

In Section 3.9 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 analysis for a optimization tool

3.10.1 Graph Grammar

this grammar is Composed of three parts:

- **Initial Action :** This portion of our grammar is used to initialize all Global variables used. They are used to meet various needs. in Figure 3.30

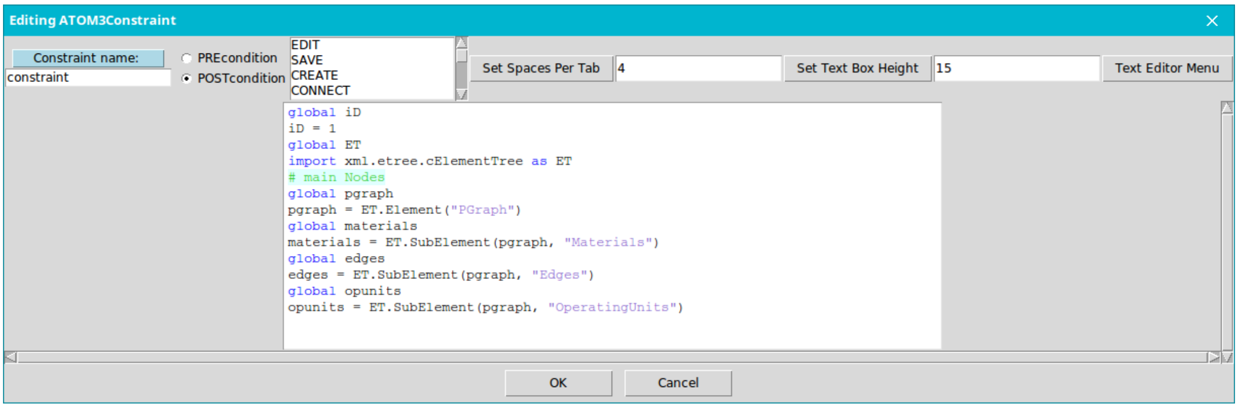


Figure 3.30: Code Initial Action

- **Final Action :** Final Action Allows Completing the .xml file by the tags XML Completing the after-processing of all PNS nodes and save the page name. in figure 3.31

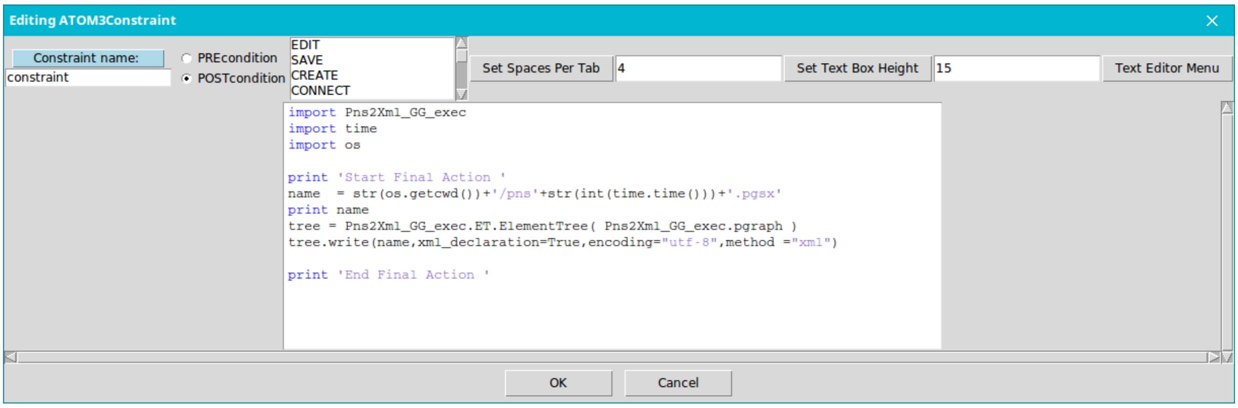


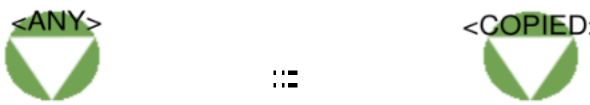
Figure 3.31: Code Final Action

❑ Set of Rules :

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

We quote here The Most major rules:

1) Transform RawMaterial into XML code : This rule (Figure 3.32) to priority equal to "1", ie, it is the first rule applied on the Model . She transforms the place into a XML



(a) LHS (b) RHS

Figure 3.32: Raw Material into Xml

a) Condition for rule 3.32 For the application of this rule, it must be verified that the partition To be transformed is not treated before

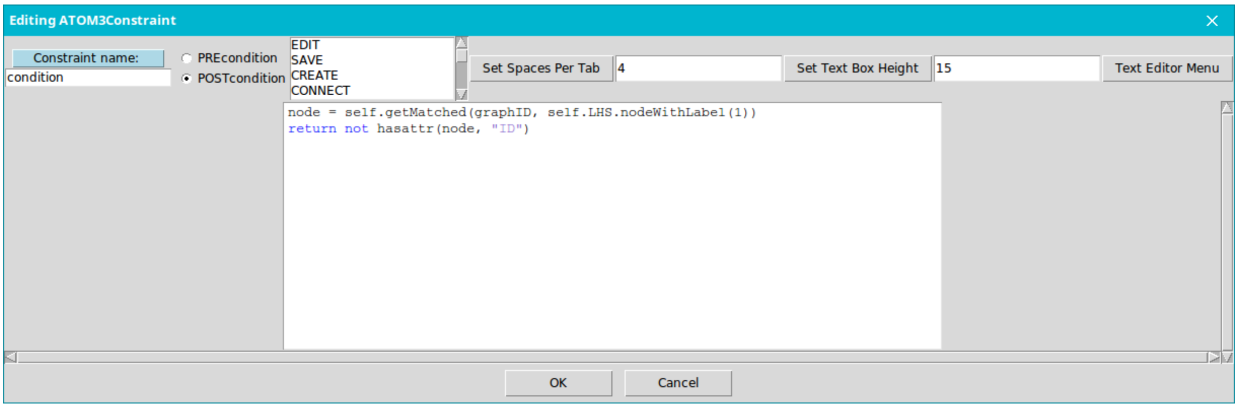


Figure 3.33: condition of Raw Material for Transfromation

b) Action for rule 3.32 Once the rule is applied, it Action Allows to create a code INSTEAD XML and mark the object to the left side to processed.

like you see in the first line i 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 similair function

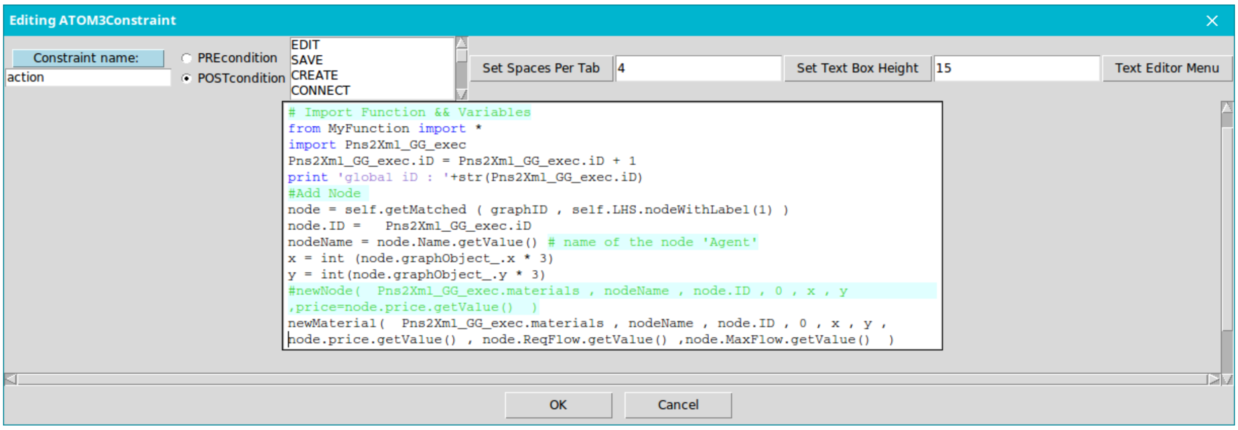


Figure 3.34: Action of Raw Material for Transfromation

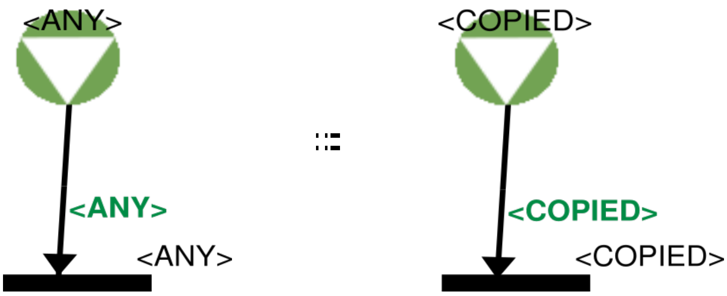
```python
def newMaterial(  parent=None , name= "None" , ID="0" , Type= "None" , xN =
    '-1', yN ='-1' ,price='-1' ,reqF = 0  ,maxF =999999 ):
  #Create new Node has the Material Name ...and add it into the parent node
  node = ET.SubElement(parent, "Material",ID=str(ID),Name=name,Type=str(
    Type))
     print 'add Cords into node    -NewNode-'
  #Create Corde for the node material
  cords = ET.SubElement(node, "Coords")
  ET.SubElement(cords, "X").text = str( xN )
  ET.SubElement(cords, "Y").text = str( yN )
  #Test if the patameter is not null or empty
  if price != '-1' or  reqF != 0  or maxF !=999999 :
     ParameterList = ET.SubElement(node, "ParameterList")
     ET.SubElement(ParameterList, "Parameter",Name="price",Value=str( price)
     )
     if reqF == 0 : reqF = '-1'
     ET.SubElement(ParameterList, "Parameter",Name="reqflow",Value= str(reqF
    ) )
     if maxF == 999999 : maxF = '-1'
     ET.SubElement(ParameterList, "Parameter",Name="maxflow",Value=str(maxF)
     )

  #Create Label for the MaterialNode
  label = ET.SubElement(node, "Label",Text = str(name))
  offset  = ET.SubElement(label, "Offset")
  ET.SubElement(offset, "X").text =   str(5)
  ET.SubElement(offset, "Y").text =   str(0)

  return node
```

Listing 3.1: 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

and the figure 3.35 represent the first Edge type



(a) LHS (b) RHS

Figure 3.35: Edges from Raw Material into Operating unit to xml code

a) Condition for rule 3.35 it is the same with figure 3.33 but here we select the edges

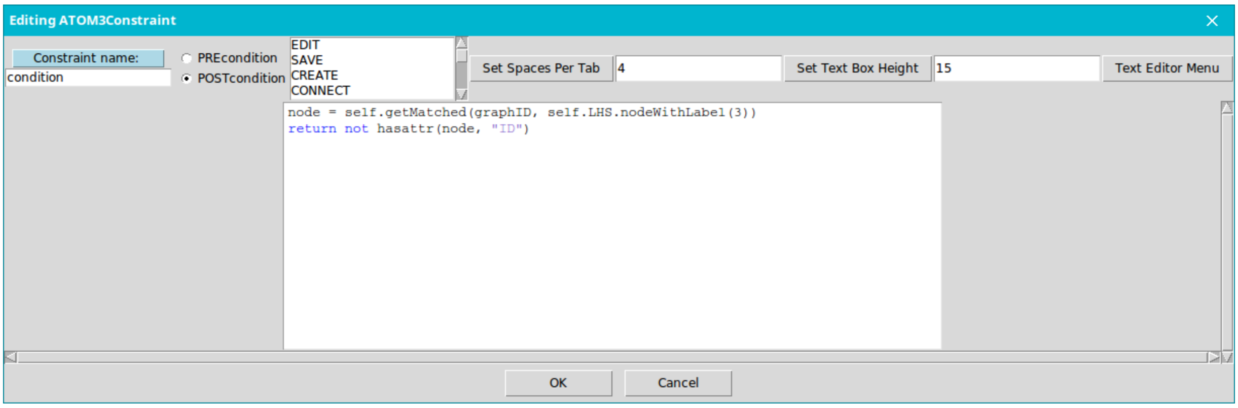


Figure 3.36: Condition of Edges (RawMaterial to Operating unit)

b) Action for rule 3.35 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

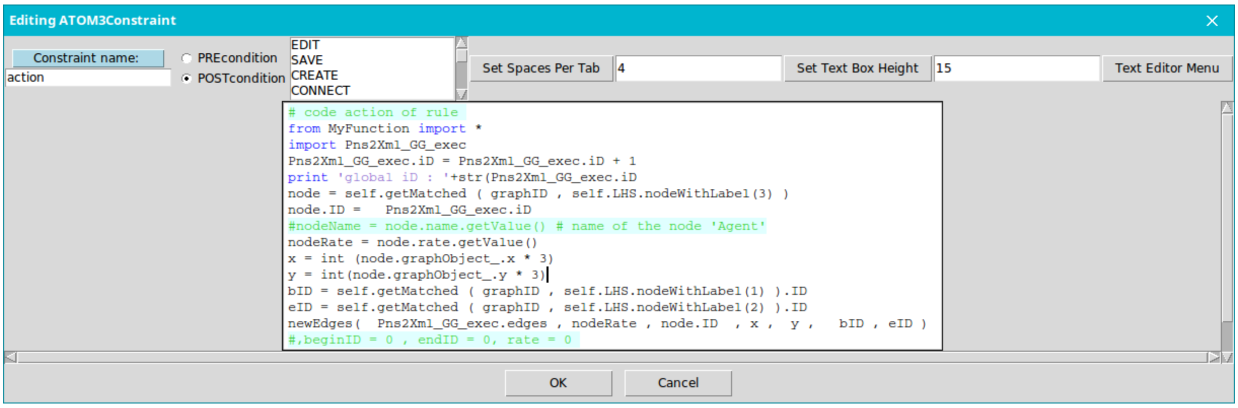


Figure 3.37: Action of Edges (RawMaterial to Operating unit)

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

```
def newEdges(   parent    , name    , ID      , xN    , yN      ,beginID     , endID
    ):
  #Create new Node and add it into parent Node
  node  = ET.SubElement(parent ,  "Edge",ID=str(ID),BeginID=str(beginID),EndID
    =str(endID),Rate=str(name),Title=str(name)  ,  ArrowOnCenter="true"  ,
    ArrowPosition="50")

  #Create Corde for the node Edges
  cords  = ET.SubElement(node,  "Coords")
  ET.SubElement(cords ,  "X").text  =  str(  xN  )
  ET.SubElement(cords ,  "Y").text  =  str(  yN  )

  #Create Label for the Edges
  label  = ET.SubElement(node,  "Label",Text  =  str(name))
  offset  = ET.SubElement(label ,   "Offset")
  ET.SubElement(offset ,  "X").text  =    str(5)
  ET.SubElement(offset ,  "Y").text  =    str(0)

  return  node
```

Listing 3.2: Python Function new Edges

3.11 Optimisation Multi Agent System

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

P-Graph 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. [29,30]

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. [31]

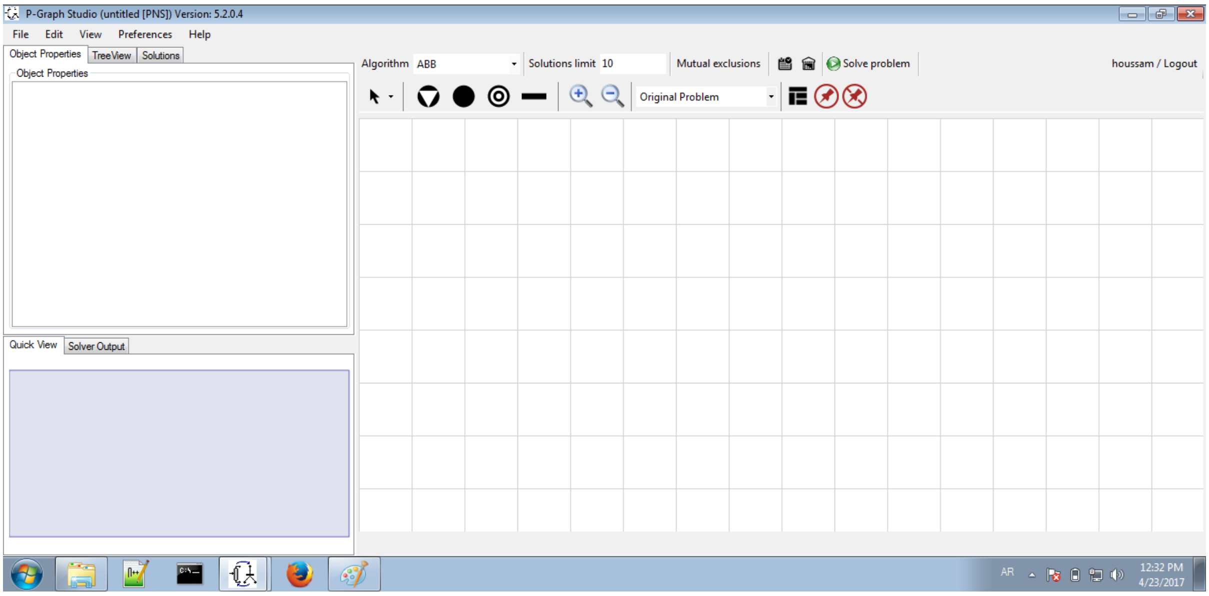


Figure 3.38: P-Graph Studio Window

figure 3.39 represent multi agent system , Previously mentioned in figure 3.29 After

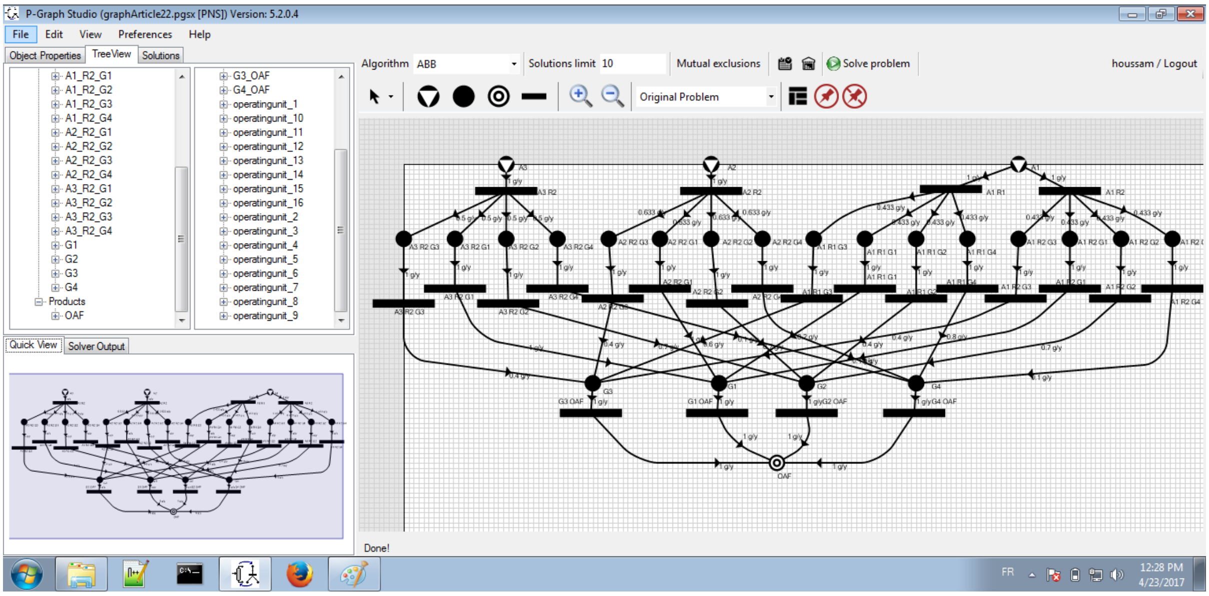


Figure 3.39: Multi Agent System in PGraph Studio

we use this tool to optimise , we got this figure 3.40

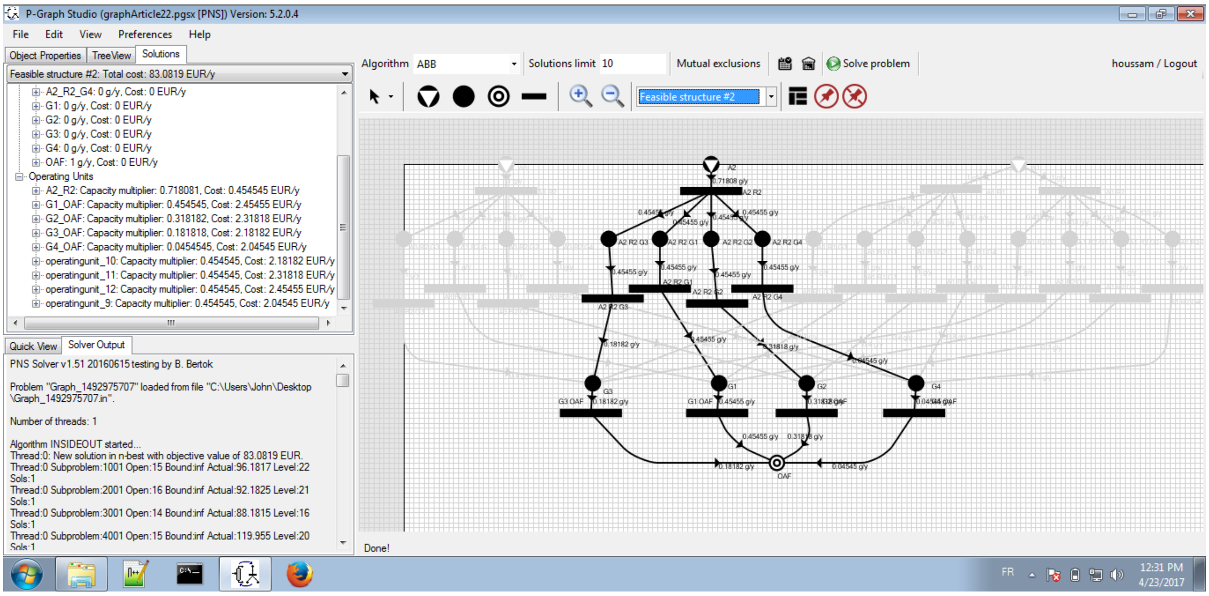


Figure 3.40: Multi Agent System in Optimum structure

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 3.3: Part from Graph.in

- Graph.out and contain :
  1. All possible solution , starting with the optimum solution
  2. every solution presented by set of operating unit , material and consumtion rate

```
Feasible structure #2:
Materials:

A2 (64.6273 EUR/y): −0.718081 g/y
A2_R2_G1: balanced
A2_R2_G2: balanced
A2_R2_G3: balanced
A2_R2_G4: balanced
G1: balanced
G2: balanced
G3: balanced
G4: balanced
OAF (−0 EUR/y): 1 g/y

Operating units:

0.718081*A2_R2 (0.454545 EUR/y):
    A2 (−0.718081 g/y) => A2_R2_G1 (0.454545 g/y) A2_R2_G2 (0.454545 g/y)
                A2_R2_G3 (0.454545 g/y) A2_R2_G4 (0.454545 g/y)

0.454545*G1_OAF (2.45455 EUR/y): G1 (−0.454545 g/y) => OAF (0.454545 g/y)
0.318182*G2_OAF (2.31818 EUR/y): G2 (−0.318182 g/y) => OAF (0.318182 g/y)
0.181818*G3_OAF (2.18182 EUR/y): G3 (−0.181818 g/y) => OAF (0.181818 g/y)
0.045454*G4_OAF (2.04545 EUR/y): G4 (−0.045454 g/y) => OAF (0.045454 g/y)

0.454545*A2_R2_G3 (2.18182 EUR/y):
        A2_R2_G3 (−0.454545 g/y) =>  G3 (0.181818 g/y)

0.454545*A2_R2_G2 (2.31818 EUR/y):
```

```
        A2_R2_G2 (-0.454545 g/y) =>   G2 (0.318182 g/y)

0.454545*A2_R2_G1 (2.45455 EUR/y):
        A2_R2_G1 (-0.454545 g/y) =>  G1 (0.454545 g/y)

0.454545*A2_R2_G4 (2.04545 EUR/y):
        A2_R2_G4(-0.454545 g/y) => G4 (0.0454545 g/y)
Total annual cost= 83.0819 EUR/y
```

Listing 3.4: Part from Graph.out

3.12 Conclusion

We have presented in this chapter the principles of graph transformations and then we have presented our approach of transformations of the multi agent system in OMACS frame work to PNS , And we also presented our 2nd 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 this paper, we saw out objective was to optimise a multi agent system and this MaS represented according the OMACS framework .

We have introduce in the first chapter the definition of elements and the relation between these element in OMACS .

The Second chapter was 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 presented our approach for transforam MaS in OMACS presentation into PNS Model , we start with defining the meta model for both framework i use (OMACS and PNS) followed by an overview about the Tool $AToM^3$ that tool allow us to define the meta model and generate a formalism from these meta model , and use formalism to create ot modelise you own model .

the next step it is to define a graph grammar that allows to transform from source graph in this case is OMACS model into target graph (PNS model) , after execute this transforamtion we can not be used directly for optimisation with the existing tools , for thsi we have to propose another approach to transforam the model in PNS into xml file .

Now we can import the xml file into the PGraph-Studio and apply the Algorithm of optimisation .

Bibliography

[1] B. B. . F. F. . A. A. . Juan C. GarcÃa-Ojeda 1, 2 and L. T. Fan, "A preliminary study of the application of the p- graph methodology for organization-based multiagent system designs: Assessment," vol. 12, no. 02, 2015.

[2] K. K. Carole Coulson, Jamie Snavely, ed., *Handbook of Research on Multi-agent Systems : Semantics and Dynamics of Organizational Models*. Information Science Reference, 2009.

[3] W. H. Oyenan and S. A. DeLoach, "Towards a systematic approach for designing autonomic systems," vol. 8, no. 1, 2010.

[4] sdeloach@k state.edu, "O-mase: a customisable approach to designing and building complex, adaptive multi-agent systems," vol. 4, no. 3, 2010.

[5] E. T. M. Scott A. DeLoach, Walamitien H. Oyenan, "A capabilities-based model for adaptive organizations," vol. 16, no. 1, pp. 13–56, 2008.

[6] S. I. S. . M. H. A. Hosny Ahmed Abbas, "Organization of multi-agent systems: An overview," *An Overview. International Journal of Intelligent Information Systems*, vol. 4, no. 3, pp. 46–57, 2015.

[7] S. A. D. E. Matson, "An organizational model for designing adaptive multiagent systems,"

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

[9] F. F. Juan C. Garcia-Ojeda, Botond Bertok, "Planning evacuation routes with the p-graph framework," *Chemical Engineering Transactions*, vol. 29, 2012.

[10] T. K. . J. T. Denes Almasi, Csanad Imreh, "Heuristic algorithms for the robust pns problem," *Acta Polytechnica Hungarica*, vol. 11, no. 4, 2014.

[11] B. F.Friedler, L.T.Fan, "Process network synthesis : Problem definition," 1993.

[12] H. Csaba, *Halozati folyamatok szintezise ,Process network synthesis*. Thesis (PhD), 2017. Mar. 18. 15:35 2010.

[13] Y. H. F. Friedler ', K. Tarjan and L. Fan:2, "Combinatorial algorithms for process synthesis,"

[14] G. FAYÇAL, *Une Approche de Transformation des Diagrammes d.Activites d.UML Mobile 2.0 vers les Reseaux de Petri.* 2011.

[15] T. KÃŒhne, "Matters of (meta-) modeling," *Software and Systems Modeling*, vol. 5, no. 4, pp. 396–385, 2006.

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

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

[18] J. W. . Tony Clark, Paul Sammut, *Applied MetaModelling A Foundation For Language Drivern Development.* 2 ed., 2008.

[19] M. A. . Juan de Lara 1, Hans Vangheluwe 2, "Meta-modelling and graph grammars for multi-paradigm modelling in atom 3," pp. 194–209, 2003.

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

[21] J. alvarez, "Mml and the metamodel architecture," in *WTUML: Workshop on Transformation in UML 2001*, 2001.

[22] J. Bezivin, R. Heckel, J. Bezivin, R. Heckel, T. Mens, K. Czarnecki, and P. V. Gorp, "04101 discussion a taxonomy of model transformations."

[23] C. E. F. H. Hartmut Ehrig, Karsten Ehrig and G. Taentzer, "Information preserving bidirectional model transformations," *Springer*, pp. 79–86, 2007.

[24] A. H. B. H. H.-J. k. s. j. d. P. A. S. G. T. Marc Andries, Gregor Engels, "Graph transformation for specification and programming," *Science of Computer programming*, pp. 1–54, 1999.

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

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

[27] "`http://atom3.cs.mcgill.ca/`."

[28] . Juan de Lara 1 and H. V. 2, "Atom 3 : A tool for multi-formalism and meta-modelling," *Springer*, pp. 172–188.

[29] 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," in *24th European Symposium on Computer Aided Process Engineering* (P. S. V. JiÅÃ JaromÃr KlemeÅ¡ and P. Y. Liew, eds.), vol. 33 of *Computer Aided Chemical Engineering*, pp. 1009 – 1014, Elsevier, 2014.

[30] "`http://p-graph.com/`."

[31] M. B. Botond Bertok and F. Friedler, "Generating and analyzing mathematical programming models of conceptual process design by pâgraph software," *Industrial & Engineering Chemistry Research*, pp. 166–171, 2013.