TPO and SML

A Societal Type System and a Society Modeling Language for Agent Societies

▷ The Synchronic Perspective of Agent Societies ◁

WORK IN PROGRESS
(*Incomplete, Non-Revised*)

Antônio Carlos da Rocha Costa

Programa de Pós-Graduação em Computação
Universidade Federal do Rio Grande - FURG
96.203-9000 Rio Grande, RS, Brazil.
Programa de Pós-Graduação em Informática na Educação
Universidade Federal do Rio Grande do Sul - UFRGS
90.040-060 Porto Alegre, RS, Brazil.

Email: ac.rocha.costa@gmail.com

April 2017

Overview

In this report we introduce *TPO*, a societal type system, and *SML*, a TPO-based society modeling language, to support the *type-based modeling* of full-fledged *agent societies* and *inter-societal agent systems.*

The *TPO* type system is proposed as a categorial limit of the type systems of agent societies, that is, as a type system that encompasses the essential type structures of every full-fledge society modeling language. We do not prove this proposition, neither in general nor regarding the existent type systems for agent societies, since most of the latter are only informally stated. But we give enough empirical evidence for that proposal by re-typing with *TPO* various *classical organizational models* of agent societies, which in general were implicitly typed with their own ad-hoc type systems.

The *SML* modeling language is proposed as a concrete linguistic realization of *TPO.* It is introduced, thus, as a *universal* modeling language for agent societies and inter-societal agent systems, capable of being the target language for the translation of any model of agent society or inter-societal agent system. It is introduced, thus, as a modeling language capable of being compiled to any agent platform or framework that is strong enough to support full-fledged organizational models of agent societies and inter-societal agent system.

In this respect we again do not prove such proposition, neither in general nor for any particular case, but we give enough evidence for it by translating to *SML* different *examples of agent societies*, defined on the bases of the classical organizational models considered previously.

To achieve the flexibility required by the latter task, *SML* makes extensive use of the *external type* feature of TPO, which allows for syntactical peculiarities of any given society model, or of any agent society platform or framework, to be incorporated into *SML* models through a <*quote*> mechanism (e.g., expressions for norms, processes, etc.).

Also, we detail the way *TPO* and *SML* relate to the earlier versions of the *PopOrg* model of agent societies, which originated them.

Finally, we remark that this draft is incomplete and not revised. In particular, the treatment of inter-societal agent systems is not included in it.

Comments, criticisms and suggestions are certainly welcome.

Contents

Chapter 1

Introduction

The concept of *type* is an optional feature in the definition of computational languages. But its introduction in the theory of programming languages represented a crucial step in the development and consolidation of methods and techniques of Software Engineering, independently of it appearing explicitly or implicitly in the languages, for types are the formal bases on which the essential engineering concepts of *object* and *module* are founded.

We think that the development of type systems for agents and agent systems is a crucial step that is still lacking in the Multiagent Systems area, possibly being the main reason for the lack of a sound notion of *modularity* in those systems. And the latter seems to be the main reason for the conceptual difficulties that arise in attempts to integrate, on principled bases, *conventional* and *agent-based* system models[1].

In this report, we introduce a *type-based approach* to the *formal characterization* of agent societies and multi-societal agent systems.

By *formal characterization* we understand a special kind of *formal specification* that is more appropriate for agent societies and multi-societal agent systems than conventional concepts of formal specification. We analyze below, in some detail, such concept of *formal characterization.*

To leverage the proposed approach, we define *TPO*, an *observational type system*, and *SML*, a *society modeling language* designed to provide a concrete syntax for the objects of *TPO*. Both *TPO* and *SML* build on the various previous versions of the *PopOrg* model of agent societies that we have been developing for some time, and of which they represent together the first consolidated formulation (see [4, 5, 6, 7] for a general view of the evolution of the *PopOrg* model).

TPO is a *set-based* type system, and the types of *TPO* are said to be *observational*: they type just the structural and operational features that observers of agent societies and multi-societal agent systems can take into account when they adopt an *externalist* point of view, that is, when they do not examine the minds of the agents.

The syntax and semantics of *SML* are made very simple, to emphasize the types of the objects it models, and to which it gives concrete syntax.

Even though several of the objects typed by *TPO* are *time-indexed structures*, the report concentrates on the *synchronic* characterization of agent societies and multi-societal agent systems, that is, characterizations that do not contemplate the issue of the *structural evolution* of those systems. The treatment of such *diachronic* characterizations is left for future work.

The purpose of the case studies presented in the report is double. First, to show that the core of TPO constitutes a strong enough *basic* type system for the modeling and specification of agent societies, which can be *systematically extended*, by the composition of the already defined types, whenever needed. Second, to evince that *SML* is a low-level language, which can serve as a *universal* target for the translation of the specification languages of the models that are studied.

As such, it is hoped that *SML* can serve as a basis for the definition of a higher-level agent society specification language, of *wider scope*, capable to reach even the architectural level of the *inter-societal agent systems* (not included in the present report), and that can run on any *strong enough* agent society platform.

[1]See [1, 2, 3] for the development of an argument in that direction, and for the proposal of a notion of modularity for multiagent systems.

Part I

The Conceptual Framework

Chapter 2

Core Concepts

2.1 Agent Societies and Inter-Societal Agent Systems

We show in Fig. 2.1 the basic form of the architecture of agent societies that we take into account in the present work, with its five main *components*: the *populational structure* (*Pop*), the *sociability structure* (*Soc*), the *organizational structure* (*Org*), the *material environment* (*MEnv*) and the *symbolic environment* (*SEnv*). Table 2.1 indicates the main elements of each of those components.

Component		**Main Elements**
Pop		agents, agent networks
Soc	Soc_σ	sociability roles
	Soc_Σ	sociability networks
Org	Org_ω	organizational roles
	Org_μ	organization units
	Org_Ω	organization unit networks
MEnv		material objects
SEnv		symbolic objects

Table 2.1: The main elements of the architecture of agent societies.

Figure 2.2 illustrates the *import-export structure* of inter-societal agent systems, proposed in [8], which allows agent societies to *interoperate* by means of *import* and *export channels.* Figure 2.3 illustrates an *inter-societal agent system*, showing the interconnection of the multiple agent societies that constitute it. Notice that not every connection has to involve the two types of import and export channels. Notice also that we call *import-export agent societies* (*ieAgSoc*) the agent societies that are endowed with import and export channels.

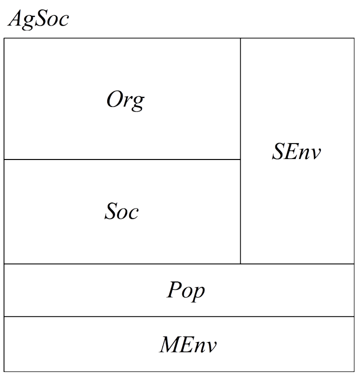

Figure 2.1: The main components of the architecture of agent societies focused by *TPO*.

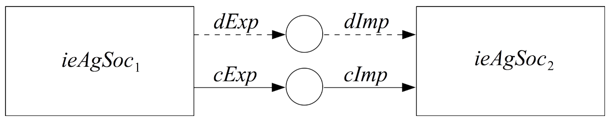

Figure 2.2: Agent societies interconnected through discrete and continuous import and export channels.

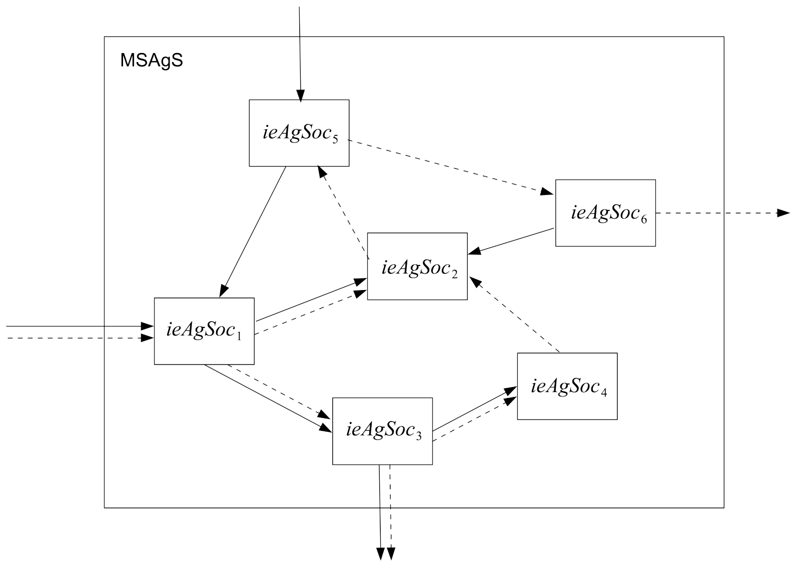

Figure 2.3: An inter-societal agent system, with its agent societies connected through import and export channels.

TPO aims to formally characterize the essential structural and operational details of agent societies and inter-societal agent systems, as we have just informally described them.

2.2 The Concept of Formal Characterization of Agent Societies and Inter-Societal Agent Systems

Agent societies and *inter-societal agent systems* are distinguished from *conventional software systems* on the basis of several crucial differences. One of the most important of these, for our present concern, is the *dynamicity* of their structures and modes of operation. For, agent societies and inter-societal agent systems are *open systems* that may be *dynamically reconfigured* through *organizational*, *cultural* and *environmental* changes. In addition, such systems are *history-dependent*, in the sense that their current structures and ways of operation may depend on the details of the *history of the changes* that they have gone through.

Thus, while conventional software systems often require that *full formal specifications* be given, previously to the systematic construction of their usually *static* structure, agent societies and inter-societal agent systems are better seen as requiring a more flexible type of formal specification, that we call *formal characterization.*

We take formal characterizations to be capable of characterizing large phases of the temporal evolution of agent societies and inter-societal agent systems, but in evolutive ways, that is, so that such formal characterizations may historically evolve, in accordance with the historical evolution of the systems that they characterize.

So, the *initial formal characterization* of an agent society or inter-societal agent system should be taken as *provisional* and prone to be dynamically overtaken by another formal characterization, as soon as a *societal transition* (evolution, reform, revolution, etc.) occurs in such system[1].

[1]On the issue of societal transitions see, e.g., [9].

We call *synchronic* any formal characterization of agent society or inter-societal agent system that characterizes a period where no *societal transition* occurs. A formal characterization is said to be *diachronic* if it is capable of characterizing not only different *synchronic periods*, but also the different *societal transitions* that may occur between those synchronic periods.

The two parts of this report (this first one about TPO, the second one about SML) concern only the *synchronic* characterization of agent societies. The *diachronic* characterization of agent societies and the *synchronic* and *diachronic* characterization of inter-societal agent systems are left for further work.

2.3 Animate and Inanimate Objects

A fundamental feature of *TPO* is that, from its observational point of view, objects may be characterized either as *animate* or as *inanimate*:

- An object is said to be *animate* when one can assign a *causal power* to it, in the sense that at least one *sequence of events* can be observed and considered to be caused by that object.

- Otherwise, the object is said to be *inanimate*.

Clearly, being of an observational character, the categorization of objects either as animate or as inanimate is conjunctural and relativistic.

2.4 Material and Symbolic Objects

Another important feature of *TPO* is the distinction between *material* and *symbolic* objects, and the treatment of the latter as first-class objects.

Material objects are the objects of the *material environment* of agent societies and inter-societal agent systems. *Symbolic objects* are the objects of their *symbolic environment*.

The treatment of symbolic objects as first-class objects allows the treatment of *symbolic environments* as effective environments in which agent societies and inter-societal agent systems may operate.

Symbolic environments model the *cultural aspects* of agent societies and inter-societal agent systems [7, 10]. In particular, they capture the *ideological notions* that regulate the way agents, organization units and societies behave and interact with each other [11].

Clearly, of the symbolic objects, *norms* are of particular importance, since norms are the main means through which the *regulation of behaviors and interactions* is carried on.

2.5 Norms

Due to their importance for the constitution of *organizational roles*, *organization units*, *agent societies* and *inter-societal agent systems*, and their importance for the way such entities operate, *norms* deserve a special treatment in symbolic environments.

This is achieved in *TPO* through the definition of the type *norm*, considered as a sub-type of the type *symbolic object*, and built on the basis of the external type *norm expression*.

2.6 Externalism, Internalism, Intentionality, and the Fundamental Types

Since, in *TPO*, all of agents, organization units, agent societies and inter-societal agent systems are considered *animate* objects, types for *actions*, *behaviors* and *interactions* should be at the core of that type system. Formally, that implies that the notions of *time*, *event* and *process* should be taken as the *fundamental types* of *TPO*.

On the other hand, agents, organization units, agent societies and inter-societal agent systems are complex entities, and means should be provided for typing their internal structures.

A crucial issue appears here, however: the internal structures of *agents* are very distinct from those of organization units, agent societies and inter-societal agent systems. A similarity between the former and latter three types of structures can be noticed, but important differences prevent the *a priori* adoption of a *uniform* type system for the four types of objects.

We categorize that difference by saying that the typing of the structure of *agents* often requires an *internalist* point of view, where *mental concepts* play central roles (as components of the agents' *intentionality*), while the typing of *organization units*, *agent societies* and *inter-societal agent systems* it is usually enough to adopt an *externalist* point of view, where such *mental concepts* play only a complementary role, in the form of the *cultural* features of those components[2].

On the other hand, *externalism* allows for the *ascription of intentionality* to entities, an intentionality that those entities may not effectively have, or which is not necessarily coincident with the intentionality they effectively have (see, e.g., [12] or [13]).

Thus, we consider satisfactory that *TPO* adopts an *externalist* perspective, in which mental concepts like *beliefs*, *desires* and *intentions* are taken as *properties* that entities (agents, organization units, societies, inter-societal agent systems) may be *considered* to have (on the basis of observations), not features that they are *certain* to have[3].

2.7 The Set-Theoretic Semantics of *TPO*

When type systems are explicitly adopted in a given language, a variety of possibilities arise for the formal semantical definition of the types of the language: sets, ordered sets, categories, etc. The *TPO* type system has *sets* as the semantics of its types, which is a simple solution, convenient for this exploratory work.

Chapter 5 gives, in the form of tables of a synthetic character, a complete list of the types of *TPO*, together with the sets that constitute the domains of their respective objects. The details of *TPO* (types, typing rules, etc.) are given in Chap. 6, which develops the system in a bottom-up, step-by-step way.

2.8 Typing, Re-typing and Translating

In Chap. 7, we analyze several MAS organization models, all of them classics of the literature of agent system[4].

For each model, we first give an informal presentation and then we represent the main concepts of the model as types of the *TPO* system. Since *TPO* is a typing system, we call this representation "re-typing", which serves the purpose of emphasizing that we see those concepts as akin to *types*.

Given the re-typing of a certain organizational model into *TPO*, we use that re-typing, in the second part of the report, as a guide to the *translation* of example organization specifications, as given in the original specification language of that model, into system characterizations in *SML*.

Figure 2.4 illustrates the relationship between re-typing and translation.

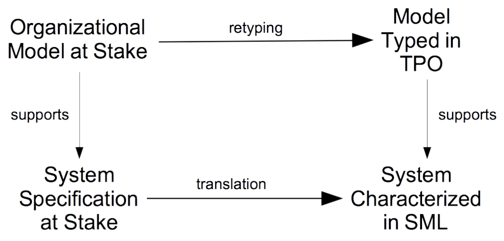
Organizational Model at Stake
retyping
Model Typed in TPO
supports
supports
System Specification at Stake
translation
System Characterized in SML

Figure 2.4: The relationship between *re-typing* and *translation*.

[2]In fact, this idea of the *bounds of internalism* in agent societies was the trigger for the evolution of the *PopOrg* model from its initial conception [4, 5] (see Chap. 4).

[3]An externalist perspective like ours, but set in much more strict terms, was proposed by Jacques Ferber and colleagues, in their seminal works [14, 15, 16] (see also Chap. 7.1).

[4]We made our best to take the examples from their first publications. When that was not possible, we took them as they appeared in their most widely known sources.

2.9 The Issue of Universality vs. Interoperability

TPO is proposed as one example of a *universal type system* for agent societies and inter-societal agent systems. *SML* is proposed as a *universal modeling language* for such systems[5].

The implications of such proposals are illustrated in Fig. 2.5 (*SEAP* stands for *Strong Enough Agent Platform*). One sees that one can understand *TPO* as a compatibility requirement between *SML* and any *SEAP*.

The figure pictures *SML* as an intermediary between *SEAP* and the organizational models taken as case studies in this report, because we acknowledge that often the *translation* into *SEAP code* cannot be done automatically, given the informal character of the definitions of most of those models.

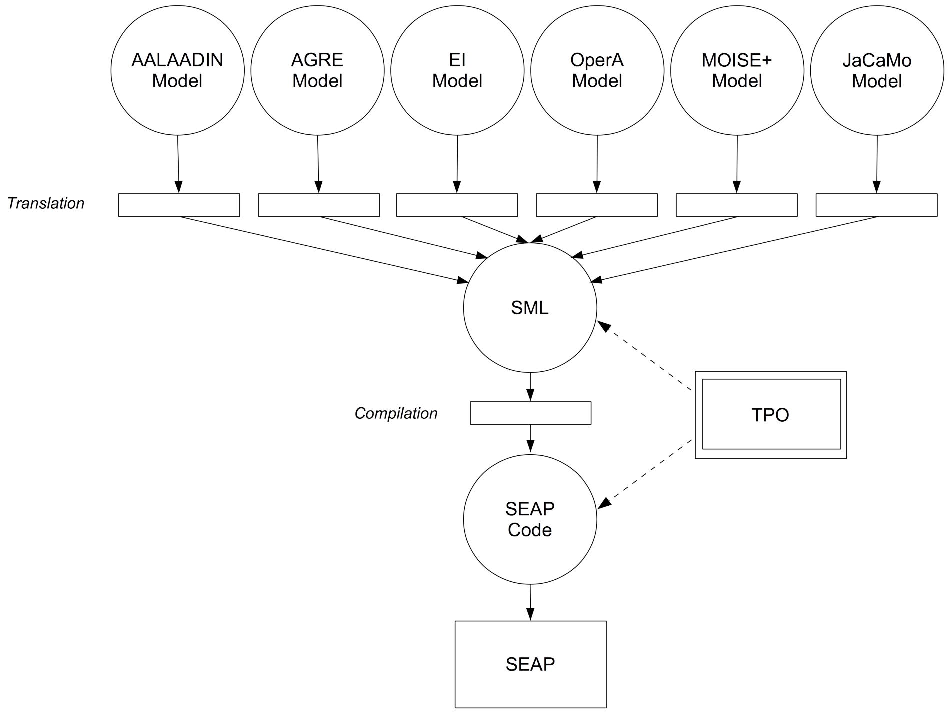

Figure 2.5: Illustration of the idea of *TPO* and *SML* as a *universal type system* and a *universal modeling language* for agent societies and inter-societal agent systems.

On the other hand, a notion that seems to be related to that of *universality* is the notion of *interoperability*. Conceptual models *do not interoperate*, since they *do not operate*, at first place. What can interoperate are the *systems* that are build according to those conceptual models.

That is, the issue of *system interoperability* is an issue that happens at the *interface* between the systems that are to interoperate, and has nothing to do with the relations that may exist between the *internal structures* of those systems (even less with the relations that may exist between the internal structures of their conceptual models). In fact, the very concept of *interoperability* was created because it frees the developers from dealing with the internals of the *systems* that are to interact with each other (see the *W3C SOAP* protocol[6], for instance).

Conceptual models relate to each other in a way that is different from *interoperability*: they relate to each other in terms of a *subsumption relation*, by which they may also relate to a universal model that subsumes all of them[7].

[5]Universality in the *category-theoretic* sense, not necessarily in the *Turing-computability* sense.

[6]`https://www.w3.org/TR/soap12`

[7]But see [17] for the exactly opposite conception, promoting the concept of interoperability of organizational models and the importance of relating, and even translating, the internal structures of organizational models to each other.

That is why *TPO* is proposed in terms of *universality*, not *interoperability*. And that is why *TPO* provides the conceptual means to leverage the organizational and societal interoperability of only *effectively realized* agent societies and inter-societal agent systems (by means of *input* and *output ports*, for *organization units*, and *import* and *export channels*, for agent societies) [1, 2, 3, 8].

2.10 Extensibility

TPO operates as a meta-language for *architectural models* of agent societies. It provides a constructive means to extend the basic architectural model that it consolidates, namely, the *PopOrg* model.

That is, using the type constructors provided by TPO, and others that can be systematically incorporated into it, the *TPO* basic architectural model can be extended and refined as demanded by particular applications.

Chapter 3

The Quest for Modularity in Agent Systems

It has been asked numberless times why agent technology has not made its way into conventional Software Engineering. Many different answers have been given to such question, all more or less based on the (effective) incompatibility between the concepts underlying conventional software (algorithms, deterministic objects, rigidity of systems' structures, etc.) and the concepts underlying agent systems (heuristics, autonomous agents, organizational flexibility, etc.).

But incompatibilities as strong as those exist among concepts underlying different programming language paradigms: procedural, logical, functional, object-oriented, etc., without anyone questioning the possibility of modules written according to each of those paradigms to interoperate seamlessly inside an integrated system.

The reason is, as mentioned above, that interoperability is not a feature related to the *internals* of the components of a system, but a feature related to the proper definition of *interfaces* between such components, which all those language paradigms support.

Another way to say that is to say that interoperatiblity is a feature related to the modularity of a system's composition. And that seems to be the deep reason for agent technology having not found its way toward conventional Software Engineering: agent systems lack *a proper notion of modularity*, there is no clear concept of what a *module* of an agent system should be.

At the beginning, the concept of *agent* was thought to be itself a concept of module, and the term *agentification* was coined [18] to express this idea of a system component acquiring the constitution of an agent, to connect to other *agentified* components.

However, the characteristic of *autonomy* of agents soon undermined that idea, for it required not only flexibility in the agents' internal decision making process, but also flexibility in their way of interacting with their environments. Modularity requires a minimum of stability and predictability.

Much work has been done, on the other hand, on the modularization of agents themselves, but that is another scope of modularity, not the scope related to the integration of agent systems with conventional software systems.

A successful tendency that soon appeared was that of treating conventional software systems as components of agents' environments, namely, the *agent-and-artifact* approach [19]. By giving the agents systematic access to operations on the components of those software systems, the agents could treat those components as environment *artifacts.*

In retrospect, the success of the *artifacts* approach should have been expected, due to two facts. First, the stability and predictability of the conventional software systems, that allow agents to deal with stable and predictable environment components. Second, the fact that agents run on platforms that are conventional software systems and such type of integration is, essentially, an integration of the conventional software systems with the agents' platforms.

But, clearly, that is a one-way solution, allowing just the integration of conventional software systems into agent systems, not the other way round. The real difficulty is in having agent systems integrated as components of conventional software systems. And that is the integration that really matters, for with it agent systems would have, in principle, all the enormous base of existent conventional software systems to make their way in.

We submit that the proper concept of *modularity of agent systems* is to be located, at two different levels, in *organization units* and in *agent societies*. Organization units, structured as suggested by the *TPO* type system, with an explicitly declared *interface* and *input* and *output ports* to regulate their exchanges with the outside, may well operate as *organizational modules* for the internal constitution of agent societies.

Libraries of *organization units* could be created, similarly to the libraries of modules for conventional software systems. Such organization units could be seamlessly integrated to any agent society where at least another organization unit is able to interact with any other organization unit externally integrated to the society. And that, independently of the agent programming language and organizational model adopted for the organization units being integrated into the agent society. Much in the same way that, in human societies, any organization can be integrated into any society (with a minimum of adjustments of its internal organization and interface to that society).

Agent societies themselves, on the other hand, seem to be the appropriate level of modularity to be taken when integrating agent systems with conventional software systems. Their *import* and *export channels* serve well the same purpose as the inport and export ports of organization units, in giving stability and predictability to their exchanges with conventional software systems.

Of course, the question that immediately arises is: Why agent societies? Why not organization units, as the one single level of modularity allowing for the integration of agent systems into conventional software systems? The answer is double, and has to do with issues of legal and moral responsibilities of effectively operating agent systems

In one sense, that possibility is perfectly sensible, whenever the organization unit can be submitted to the same type of verification and/or validation that conventional software systems go through [1]. In such situation, the developer of the conventional software system would have no doubt in assuming any moral and legal responsibility for integrating that organization into its conventional software system (in particular, moral and legal responsibility for the consequences that the decisions taken inside that organization could bring for the users of his conventional software system).

Wherever such verification and/or validation process cannot be done, however, the agent module to be integrated into the conventional software system has to be capable of legally and morally accounting *by itself* for the impacts of the decisions and actions the agents of that module take, after the integration. This requires that the agent module be endowed with legal and/or moral systems of its own, that could be checked for their reasonableness at any time, and that could morally and legally account for decisions and actions taken by the module's agents, and validly report such moral and legal accounts to the external world, through the conventional software system.

As we attempted to show elsewhere [20, 21, 11, 22], the level of *agent societies* is the right architectural level for the endowment of agent systems with *legal* and *moral systems* of their own, and capable of verified legal and moral accountability, as demanded by legal and moral requirements that may have to be eventually satisfied, before allowing their integration into conventional software systems.

Chapter 4

Relation to the Pop-Org Model

4.1 Semantical and Syntactical Issues

The *PopOrg* model is a *semantical* model for agent societies that developed through a series of papers, beginning with [4] and [5].

Unfortunately, no complete presentation of the model was ever written (for the simple reason that the model was never completed...). In each paper, a new feature was added, or some existent feature was revised, and no line of systematic development was ever drawn and followed.

In the present report, we summarize in a reasonably systematic way the main ideas that arose during the (somewhat erratic) development of the *PopOrg* model.

As shown below, the *TPO/SML* model presented here outstrips the original scope of the *PopOrg* model, aiming to account for full-fledged *agent societies* and, more recently, *inter-societal agent systems.*

Yet, the *TPO/SML* model still sticks to the most important original constraint, namely, that of being a *minimal semantical model*, in the sense of being restricted to deal only with *observational* features of agent systems, that is, only with the operational and structural features that are reachable by *external* observers, without undertaking the examination of the internals of the minds of the agents. Such possibility is leveraged by the conceptual separation of both the *organizational* and the *sociabiity* structures from the *populational structure*, the feature that originally motivated the name "PopOrg", and that pairs the model with certain classical sociological models, see e.g. [23] (see also [24]).

In relation to that, it should be emphasized that the *PopOrg* model was never meant to be a *syntactical formalism* for the modeling and specification of agent societies and inter-societal agent systems, since it aimed to be in fact a *semantical model* (in the logical sense of "semantical model", see e.g. [25]), on which appropriate syntactical formalisms could be formally interpreted.

In fact, the decision of presenting the ideas that constitute the *PopOrg* model in terms of this *TPO* type system was taken to set the stage for a possible future development of such a syntactical formalism, i.e., an *observational modeling and specification language* based on *TPO*, of which the *SML* language introduced here is a first, tentative version.

4.2 The *Sociability* and the *Organizational* Structures

Since its beginning, the *PopOrg* model conflated *sociability elements* and *organizational elements* (e.g., sociability and organizational roles, and their properties and relations) into one single architectural level, constituted by the organizational structure *Org*.

The reading of Piaget [26] and [27], with his distinction between *substitutable* and *non-substitutable* social roles, lead to the detachment of the *sociability structure* from the *organizational structure.* Although the meanings of those names are not self-revealing, their adoption is enough for distinguishing between the following:

- *sociability structure*: a structure that attempts to formally capture relations and interactions where the *singularity* of each agent cannot be abstracted way, that is, where the agents are *non-substitutable* by other agents;

- *organizational structure*: a structure that attempts to formally capture relations and interactions realized by agents that are *substitutable* by other agents.

For instance, that John loves Valentine is a relation that is radically changed if John is substituted by Peter, or Valentine by Cosette. On the other hand, the hierarchical relation between the head of a computer science department and the lecturers of that department is, in principle, supposed to be immune to changes in the person that enacts the role of head of the department, or to changes in the composition of the department's faculty.

With that understanding, we have separated the two architectural levels of the *sociability structure* and the *organizational structure.*

In such situation, it became clear that the *populational structure* of an agent society constitutes the architectural level where are present what can be called (adapting in a relativistic way John Searle's terminology [28]) the *brute* social facts of that society.

Brute social facts can, then, be construed as *institutional* facts in either of two architectural levels. If they are characterized as being of the *non-substitutable* type, they are to be construed as *"personified facts"*, inside the *sociability structure.* If they are characterized as being of the *substitutable* type, they are to be construed as *"unpersonified facts"*[1], inside the *organizational structure.*

In terms of the architecture of agent societies, the main advantage of having the *sociability level* explicitly constituted is that of allowing the agents to constitute socially meaningful relations, and interactions and networks, on the basis of structural elements (sociability roles, sociability networks, etc.) that do not require, for their constitution, the realization of *official* interaction channels (e.g., the *input* and *output ports* defined in the *TPO* type system), as the unpersonified *organization units* of the organizational structure require. That is, in the sociability structure, interactions occur directly on a personified agent-to-agent basis, while in the organizational structure, interactions should occur through official channels between agents that operate on an unpersonified basis, as members of organizations and organizational units.

Even though the distinction between unpersonifed and personified situations may give rise to practical characterization problems, when considered in the context of empirical researches based on this conceptual model, we feel that - at the conceptual level at least - the distinction is important, as emphasized by Piaget [26, 27].

In fact, it seems clear that the architectural level of the *sociability structure* has also a theoretical importance of its own, as it is the level that can formally account for the social phenomena studied in classical works by, e.g., Georg Simmel [30], Marcel Mauss [31], Erving Goffman [32], Fritz Herder [33] and George Homans [34], so that the *sociability structure* could be taken as the structure for the "presentation-of-selves", in Goffman's terminology.

Notwithstanding that, we would expect that in many applications the *Soc* structure would be made transparent, only the *Pop* and *Org* structures being effectively used, with *Pop* directly implementing *Org*, as in the original formulation of the *PopOrg* model. On the other hand, we would also expect that in many applications, *Soc* would be necessary for the implementation of *Org* (e.g., for a certain "psychological" characterization of the organizational roles), while for some other applications it is the *Pop* structure that would be of a secondary importance, or even irrelevant.

4.3 Directions of Fit of the Organizational Models

The introduction of the sociability structure (*Soc*) reinforces the applicability of the agent society model in another direction, which belonged to it since the beginning: that of serving as a *formal semantical model* for social theories, a feature that we have explored sometimes, in a non-systematic way [11, 21, 35].

We may then say that the organizational models *agent society* and *inter-societal agent system* may be used in two different ways.

In analogy to what Searle calls the *world-mind direction of fit* [28], agent societies and inter-societal agent systems can be used as semantical models for *formal specification languages*, that is,

[1]We took the term "unperson" from George Orwell's book "1984" [29].

formal languages on the bases of which agent systems architects and engineers specify *concrete agent societies and inter-societal systems* that are to be computationally realized on agent platforms. This is the use that we have been mentioning in this report.

On the other hand, in analogy to what Searle calls the *mind-world direction of fit*, agent societies and inter-societal agent systems can be used as formal semantical models for *sociological descriptive languages*, that is, formal languages with which sociologists, political scientists, cultural historians, and other students of social systems (human or not), may describe the *concrete social and cultural systems* they find in their empirical researches.

The aim of the extended *PopOrg* model, as it is consolidated and presented by *TPO* and *SML*, is to serve both purposes.

4.4 Main Advances Regarding Previous Versions of *PopOrg*

The main differences between the concepts underlying *TPO* and *SML*, and the concepts underlying the earlier versions of the *PopOrg* model, are the following:

1. the *populational structure* (*Pop*), constituted by the agents that inhabit the society, was put to serve as an *operational hub* between the *sociability structure* (*Soc*), the *organizational structure* (*Org*), the *material environment* (*MEnv*) and the *symbolic environment* (*SEnv*) of the agent societies;

2. the sociability structure *Soc*, introduced here, comprises the personified structures that agents may form between them, internally structured in terms of a *sociability micro-level* (Soc_σ) and a *sociability macro-level* (Soc_Σ);

3. the terminology of *micro-organizational level*, *meso-organizational level* and *macro-organizational level* was changed to *organizational micro-level*, *organizational meso-level* and *organizational macro-level*, respectively

4. the denotation of the *organizational meso-level*, often written as $Org_{\omega\Omega}$, was changed to Org_μ;

5. *interfaces* were introduced, so that *organization units* (in the organizational meso-level Org_μ) could have definite members responsible for their interactions with other architectural elements;

6. *input* and *output ports* were introduced in the organization units of the organizational meso-level (Org_μ), so that organization units could have definite means for the exchange of objects, in their interactions with other architectural elements, enhancing and making clear the possibility of their interoperability (the *organizational interoperability)*;

7. an *inter-societal level* was introduced, overarching a set of agent societies, so that agent societies could inter-operate with each other, constituting *inter-societal agent systems*;

8. *import* and *export channels* were introduced in agent societies, constituting *import-export agent societies* (*ieAgSoc*), so that agent societies could interact with each other, in the inter-societal level, allowing for their interoperability (the *societal interoperability*);

9. *social sub-systems* (often denoted by *SSS* or by *SS*), which were initially conceived as being *structural* elements of the organizational macro-level (Org_Ω), were expunged from the set of *structural concepts* that account for agent societies and promoted to *functional concepts*, serving the functional constitution of agent societies and inter-societal agent systems, in the form of *functional sub-systems*, a notion that we reserve to examine in detail in a future work;

10. the *architectural location* previously occupied by social sub-systems in the organizational macro-level is now occupied by the concept of *network of organizations*, introduced in this report;

11. the main components of the *PopOrg* model, and the elements that compose them, were submitted to a *typing process*, from which originated in the *TPO* type system.

Part II

The Type System *TPO*

Chapter 5

The Meta-Language Used to Present *TPO*

In this chapter, we present the main elements of the the *meta-language* (that is, the *notation*) that we use to present *TPO*. The *TPO* system itself is presented in Chap. 6.

5.1 Bases

5.1.1 The Concept of Observational Typing

The work reported here belongs to a kind of work that can be labeled *observational typing of system architectures*. Such kind of work is based on the idea that architectures and architectural elements of general systems have *observational types*, that is, types that can be grasped on the bases of external observations of those systems[1].

On such basis, we introduce here the type system called *TPO*, specially conceived for the observational typing of the architectures and architectural elements of *agent societies* and *inter-societal agent systems* that are constructed along the lines indicated by the extended *PopOrg* model.

The main architectural elements of *general* systems are shown in Table 5.1. The architectural elements of *agent societies* are a specialization of such general architectural elements. The types of the *TPO* type system, introduced in Chap. 6, are types for such specialized architectural elements of agent societies.

Main Architectural Elements of General Systems
structures
sub-structures of a structure
components of a structure
sub-components of a component
behaviors of components
interactions between components
relations between components

Table 5.1: The main architectural elements of general systems.

[1]The idea that architectures and architectural elements of hardware and software systems can be observationally typed was well established already in the early 1980's, after the impulse of the *object-orientation* movement. That the architecture of *multiagent systems* can also be observationally typed results from the mere application of that idea to such systems.

5.1.2 The Concepts of Animate and Inanimate Objects

TPO categorizes objects as either of the *animate* or of the *inanimate* kind. For each kind, *TPO* defines a set of observational features compatible with that kind:

- *Animate objects* may be typed with the following observational features:
  - *Properties*: the qualities that the animate objects may present;
  - *Behaviors*: the processes that they may cause;
  - *Interactions*: the interactions that they may have with other animate objects;
  - *Relations:* the non-interactive relationships they may have with other animate objects;
- *Inanimate objects*, on the other hand, may be typed with the following observational features:
  - *Properties*: the qualities that inanimate objects may present;
  - *Relations:* the non-interactive relationships they may have have with other inanimate objects.
- In addition, both animate and inanimate objects may be endowed with an *internal structure*, whose observational features also characterize them.

The following table indicates the main animate and inanimate types of objects of *TPO*:

Animate Objects	**Inanimate Objects**
Material Objects	Time Instants
Agents	Properties
Sociability Roles	Relations
Sociability Role Networks	Interactions
Organizational Roles	Symbolic Objects
Organizational Role Networks	Processes
Organization Units	Exchange Processes
Organization Unit Networks	
Agent Societies	
Inter-Societal Agent Systems	

5.1.3 The Hierarchy of Types

We adopt the main type-theoretic conceptions presented in [36]. Thus, we say that *TPO* is a type system with one meta-kind (*Set*), two *kinds* (*Animate* and *Inanimate*), and two type constructors, *product* and *power-set*.

Every *type* is of a certain kind (*Animate* or *Inanimate*), and of the meta-kind *Set*. Every type is either *basic* (defined internally to *TPO*, or externally to it) or *constructed* by means of one of the *type constructors*.

Types constructed with the *power-set constructor* are denoted by $\wp(T)$, for any base type T, of any kind. Product types are often introduced by a *new type name*, *not* by a product notation like $(T_1 \times ... \times T_n)$, leaving implicit the fact that they are a product type.

We denote by $K :: MK$ the fact that kind K is a kind of the meta-kind MK, and by $T :: K$ the fact that type T is of the kind K. Also, we denote by $T_2 :: T_1$ the fact that type T_2 is a subtype of type T_1. As usual, we denote by $O : T$ the assertion that object O is of the type T.

The following is the type-hierarchy of *TPO* (all types are required to be non-empty):

- meta-kind:
  - *Set*
- kinds:
  - *Animate* :: *Set*
  - *Inanimate* :: *Set*
- types:

- basic types (predefined in *TPO*)
  - internal basic types
    (e.g., T :: *Inanimate*, the type of the instants of time)
  - external basic types
    (e.g., *Agent* :: *Animate*, the type of the agents)
- constructed types (predefined in *TPO* or user defined)
  (e.g., *OrgUn* :: *Animate*, the type of the organization units)

The following is the set of type constructors:

- *power-set* type constructor:
  - $\wp(T)$, for any type T of any kind;
- *product* type constructor:
  - $T_1 \times ... \times T_n$, for the product of types T_1,...,T_n of any kind;
- *new type name*, for a *product* type;
- *function space* type constructor:
  - $T_1 \rightarrow T_2$, for the set of functions between the types T_1 and T_2 of any kind.

Table summarizes the main features of the *predefined* types of *TPO*.

	Basic	**Constructed**
Internal	Inanimate	Inanimate, Animate
External	Animate, Inanimate	–

Table 5.2: Main features of the predefined types of *TPO*

5.1.4 Meta-Rules

The *rules* of *TPO* serve the purpose of typing the objects to which *TPO* refers. *Types* are objects themselves, so - as mentioned above - types may also be typed, by means of meta-types. *Meta-rules* are rules that type *types* by means of *meta-types*. The general form of a meta-rule is:

$$\frac{}{\mathit{Type} :: \mathit{MetaType}}\ MR_{RuleName}$$

TPO admits sub-typing, that is, given any type T_1, we may define a sub-type T_2 of T_1 by means of the meta-rule:

$$\frac{}{T_2 :: T_1}\ MR_{RuleName}$$

Specific typing-rules may be added to T_2, which do not affect the objects of T_1 that are not of the subtype T_2. But, all the typing-rules defined for the objects of the type T_1 apply to the objects of the sub-type T_2.

5.1.5 Typing Rules, Type Constructors and Type Deconstructors

An *unconditional* object typing and the corresponding *unconditional* type construction are indicated by the usual unconditional form of typing rule:

$$\frac{}{\mathit{obj} : \mathit{Type}}\ R_{RuleName}$$

A *conditional* object typing and the corresponding *conditional* type construction are indicated by the usual conditional form of typing rule:

$$\frac{obj_1 : Type_1 \quad obj_2 : Type_2 \quad ... \quad obj_n : Type_n}{obj_{n+1} : Type_{n+1}}\, R_{RuleName}$$

where $n \geq 1$ and, in general:

- obj_{n+1} is a composition of the objects that appear in the pre-condition of the typing rule, in the form $obj_{n+1} = ObjComp(obj_1, ..., obj_n)$, for some *object composition* operator $ObjComp$;
- $Type_{n+1}$ is a composition of the types that appear in the pre-condition of the typing rule, in the form $Type_{n+1} = TypeComp(Type_1, ..., Type_n)$, for some *type composition* operator $TypeComp$;
- $RuleName$ is the name of the typing rule.

When multiple pre-conditions appear in a typing rule, we may pile them on top of each other, above the *consequence bar*:

$$\frac{\begin{array}{c} obj_1 : Type_1 \\ obj_2 : Type_2 \\ ... \\ obj_n : Type_n \end{array}}{obj_{n+1} : Type_{n+1}}\, R_{RuleName}$$

5.1.6 Internal and External Types

As the scope of application of the type system cannot be predetermined, *TPO* allows for the introduction of *external types*, that is, types defined externally to it. So, not all types of *TPO* are *internal* types, that is, types defined in *TPO* itself.

The two main sources of external types are the set of types that are predefined in particular *modeling and specification languages* into which the type system is incorporated, and the set of *programmer defined types* in models or specifications developed with those languages.

We denote the names of *internal* typing rules with expressions the form $R_{RuleName}$, and names of *external* typing rules with expressions the form ER_{name}. Meta-rules are a particular form of internal rules. They are named in the form: $MR_{RuleName}$.

When a typing rule types *external objects*, we denote by $[E : ...]_{RuleName}$ the *external conditions* of rule *RuleName*, which should be externally satisfied for the typing of the external object to be validated. E is the term that denotes the *external object.*

A typing rule involving a condition that refers to an object of an external type is of the following general form:

$$\frac{Cond_1 \quad Cond_2 \quad ... \quad [E : ...]_{RuleName}}{E : T}\, RuleName$$

Notice that the type of *agent societies* involves external types, as *agents* and *environmental objects* are of external types.

Notice also that *agents*, *environmental objects* and *agent societies*, among other types of objects, are *time-indexed entities*, that is, their constitution vary with time. However, the *TPO* internal types, and the typing rules themselves, are time-independent.

5.1.7 Quantification, and Quantification over Sets of Structures

We use the notation $\forall x : T\ [P(x)]$ to denote the statement $\forall x[x : T \Rightarrow P(x)]$, i.e., the statement that every x of type T has the property P. We use the notation $\exists x : T\ [P(x)]$ to denote the statement $\exists x[x : T \wedge P(x)]$, i.e., the statement that there is an x that is of type T and has the property P.

We use the notation $\forall \langle x, ..., z\rangle : T[P(\langle x, ..., z\rangle)]$ to denote the statement:

$$\forall x ... \forall z[\langle x, ..., z\rangle : T \Rightarrow P(\langle x, ..., z\rangle)]$$

and we use $\exists\langle x, ..., z\rangle : T[P(\langle x, ..., z\rangle)]$ to denote the statement:

$$\exists x ... \exists z[\langle x, ..., z\rangle : T \wedge P(\langle x, ..., z\rangle)]$$

In all such cases, the quantifiers range over the set of elements of the agent society (or inter-societal agent system) to which the quantified expression refers.

5.1.8 Rule Constraints and the Complete Presentation of Typing Rules

Certain typing rules may require, for their complete presentation, that the objects they type satisfy certain *constraints*, which are not expressed in the pre-conditions of the rules. Such constraints are presented together with the typing rules themselves.

Thus, the complete presentation of a typing rule has the general form (the constraint part is optional):

———————————— . ————————————

• Rule *RuleName*:

$$rule$$

• Constraints *RuleName*:

$$constraints$$

———————————— . ————————————

Rules (internal or external) that require constraints are marked with an asterisk: $R^*_{RuleName}$, $ER^*_{RuleName}$, and $MR^*_{RuleName}$.

Sometimes, the presentation of a typing rule is preceded by a *comment* that aims to clarify issues related to the intuitive understanding of the type being defined by that rule.

5.2 The Meta-Language

5.2.1 The Kinds *Animate* and *Inanimate*

All types of *TPO* are either of the kind *Animate* or of the kind *Inanimate*. Thus, for each type X of *TPO* one of the following two meta-rules applies:

———————————— . ————————————

• Meta-Rule MR^*_X:

$$\frac{}{X :: Animate}\; MR^*_X$$

• Constraints MR^*_X:

$$X \neq \emptyset$$

———————————— . ————————————

• Meta-Rule MR_X^*:

$$\frac{}{X :: \mathit{Inanimate}}\ MR_X^*$$

• Constraints MR_X^*:

$$X \neq \emptyset$$

In what follows, the constraint that types should be non-empty is always assumed to hold. That constraint is, thus, omitted in the presentation of meta-rules that introduce new types. Then, meta-rules are denoted simply in the form MR_X (without the superscript "*"), unless it should be subject to some additional particular constraint.

5.2.2 The Basic Types

The five main basic types of *TPO* are *Time*, *Prop*, *Rel*, *NrmExp* and *Event*. As already mentioned, all basic types are of the kind *Inanimate*.

- *Time* is an essential type in *TPO*, which we denote simply by T. Type T is essential because every *active* object in *TPO* is taken to be a *time-indexed structure*, with general form $\mathit{Strct}^t = (\mathit{Comp}_1^t, \mathit{Comp}_2^t, ..., \mathit{Comp}_n^t)$, where $t : \mathrm{T}$. Each of the structural components Comp_i^t is itself, in principle, a time-indexed structure embedded in *Strct*.

- *Prop* is the type of *properties* that can be (rightly or wrongly) assigned to the objects that can be typed by *TPO*. *Properties* (i.e., objects of the type *Prop*) should be introduced as external objects, so that all the variety of properties (for qualifying agents, sociability roles, organizational roles, organizational units, etc.) can be introduced in *TPO* as required by any concrete application.

- *Rel* is the type of *relations* that can be (rightly or wrongly) assigned to pairs of objects that can be typed by *TPO*. *Relations* (i.e., objects of the type *Rel*) should be introduced as external objects, so that all the variety of relations (for relating agents, sociability roles, organizational roles, organizational units, etc.) can be introduced in *TPO* as required by any concrete application.

- *NrmExp* is the type of *norm expressions* that can be used to express *norms*. *Norm expressions* (i.e., objects of the type *NrmExp*) should be introduced as external objects, so that all the variety of norm expressions (for the expression of *agent norms*, *sociability role norms*, *organizational role norm*s, *organization unit norms*, etc.), and the norms that can be defined on their bases, can be introduced in *TPO* as required by any concrete application.

- *Event* is the type of events. *Events* are assumed to be elements that *occur* at certain *time instants*, and whose causes are allowed to be *abstracted away*. That is, the type system is such that an event may be presented without the need of indicating the element or elements responsible for its occurrence.

5.2.3 The Product Type Constructor

The meta-rules for introducing types constructed with the *product* type constructor have the following forms:

- Meta-Rule $MR_{\times_1}$:

$$\frac{T_1 : \mathit{Inanimate} \qquad T_2 : \mathit{Inanimate}}{T_1 \times T_2 : \mathit{Inanimate}}\ MR_{\times_1}$$

or:

- Meta-Rule $MR_{\times_2}$:

$$\frac{T_1 :: \mathit{Inanimate} \qquad T_2 :: \mathit{Animate}}{T_1 \times T_2 :: \mathit{Inanimate}}\ MR_{\times_2}$$

- Meta-Rule $MR_{\times_3}$:

$$\frac{T_1 :: \mathit{Animate} \qquad T_2 :: \mathit{Inanimate}}{T_1 \times T_2 :: \mathit{Inanimate}}\ MR_{\times_3}$$

- Meta-Rule $MR_{\times_4}$:

$$\frac{T_1 :: \mathit{Animate} \qquad T_2 :: \mathit{Animate}}{T_1 \times T_2 :: \mathit{Inanimate}}\ MR_{\times_4}$$

5.2.4 The Function Space Type Constructor

The meta-rules for introducing types constructed with the *function space* type constructor have the following forms:

- Meta-Rule $MR_{\rightarrow_1}$:

$$\frac{T_1 : \mathit{Inanimate} \qquad T_2 : \mathit{Inanimate}}{T_1 \rightarrow T_2 : \mathit{Inanimate}}\ MR_{\rightarrow_1}$$

• Meta-Rule $MR_{\rightarrow_2}$:

$$\frac{T_1 :: \mathit{Inanimate} \qquad T_2 :: \mathit{Animate}}{T_1 \rightarrow T_2 :: \mathit{Inanimate}}\ MR_{\rightarrow_2}$$

• Meta-Rule $MR_{\rightarrow_3}$:

$$\frac{T_1 :: \mathit{Animate} \qquad T_2 :: \mathit{Inanimate}}{T_1 \rightarrow T_2 :: \mathit{Inanimate}}\ MR_{\rightarrow_3}$$

• Meta-Rule $MR_{\rightarrow_4}$:

$$\frac{T_1 :: \mathit{Animate} \qquad T_2 :: \mathit{Animate}}{T_1 \rightarrow T_2 :: \mathit{Inanimate}}\ MR_{\rightarrow_4}$$

5.2.5 The Power-Set Type Constructor

The meta-rules for introducing types constructed with the *power-set* type constructor have one of the following forms:

• Meta-Rule $MR_{\wp_1}$:

$$\frac{T :: \mathit{Animate}}{\wp(T) :: \mathit{Animate}}\ MR_{\wp_1}$$

or:

• Meta-Rule $MR_{\wp_2}$:

$$\frac{T :: \mathit{Inanimate}}{\wp(T) :: \mathit{Inanimate}}\ MR_{\wp_2}$$

We treat the two power-set operators polymorphically [36] and write both $MR_{\wp_1(T)}$ and $MR_{\wp_2(T)}$ simply as $MR_{\wp(T)}$.

5.2.6 Operators on Objects of Product Types

We use an expression of the form $\langle obj_1, obj_2, ..., obj_n \rangle$ to denote the n-tuple formed by $obj_1, obj_2, ..., obj_{n-1}$ and obj_n in the consequence of a conditional typing rule for an object of a *product* type.

If $obj_i : Type_i$ is a precondition for the formation of n-tuple object $\langle obj_1, obj_2, ..., obj_n \rangle : Type$, we take that $\mathrm{Type}_i[\langle obj_1, obj_2, ..., obj_n \rangle] = obj_i$, so that Type_i operates as a *deconstructor* of the type *Type*.

If $R_{RuleName}$ is a typing rule of the form:

$$\frac{obj_1 : Type_1 \quad \dots \quad obj_n : Type_n}{obj : Type} \; R_{RuleName}$$

we call $DR[Type_i]_{RuleName}$ the rule that *deconstructs* objects of the type *Type*, producing objects of the type $Type_i$:

$$\frac{obj : Type}{\mathrm{Type}_i[obj] : Type_i} \; DR[Type_i]_{RuleName}$$

5.2.7 Operators on Objects of Function Space Types

[in construction]

5.2.8 Operators on Objects of Power-Set Types

The typing rules for introducing objects into *power-set types* are:

———————————— . ————————————

• Rule $R_\in$:

$$\frac{x : \mathrm{T}}{\{x\} : \wp(T)} \; R_\in$$

———————————— . ————————————

———————————— . ————————————

• Rule $R_\subseteq$:

$$\frac{X : \wp(T) \qquad Y \subseteq X}{Y : \wp(T)} \; R_\subseteq$$

———————————— . ————————————

Objects of *power-set types*, on the other hand, are endowed with typing rules that are correlated to the usual *set operations* and *relations*. The following are examples of such rules:

———————————— . ————————————

• Rule $R_\cup$:

$$\frac{X : \wp(T) \qquad Y : \wp(T)}{X \cup Y : \wp(T)} \; R_\cup$$

———————————— . ————————————

• Rule $R_\cap$:

$$\frac{X : \wp(T) \quad Y : \wp(T)}{X \cap Y : \wp(T)}\, R_\cap$$

• Rule R_{SetDiff}:

$$\frac{X : \wp(T) \quad Y : \wp(T) \quad Y \subset X}{X - Y : \wp(T)}\, {}_{\mathit{SetDiff}}$$

In the rule R_{SetDiff}, the *strict inclusion* precondition ($Y \subset X$) is necessary to guarantee that the difference $X - Y$ is non-empty.

5.3 Type, Object and Variable Names

Due to the variety of names of types, objects and variables that have to be used in the definition of *TPO*, a systematic procedure has to be adopted for the definition and use of those names. We follow the convention that is usual in programming methodology, of naming objects and types with *literal abbreviations* of their meanings.

Abbreviations starting with a *lower-case* letter are used either as names for *individual objects* or as *individual object variables.* Abbreviations starting with an *upper-case* letter are used either as names of *individual sets* or as *set variables* (when appearing to the left of the typing sign (":")), or else as names for *types*, *kinds* or *meta-kinds* (when appearing to the right of typing sign). For instance, *agbeh* denotes a particular agent behavior or a variable ranging over agent behaviors, while the type *Agent Behavior* is denoted by *AgBeh*.

Also, when re-typing organizational models into *TPO*, in the case studies, we make use of the following convention. We let the name of an original construct of the model in question (an object, a concept, a type, etc.) to be written in extenso, while any reference to its re-typing be written in the above convention for type names. For instance, a reference to the re-typed *MOISE+* concept of *Functional Scheme* is written as *FuncSch*.

5.4 A Bird's Eye View of *TPO*

In the tables shown in the following pages, we summarize all the *predefined* types of the *TPO* type system, indicating the domains of their respective objects.

The details of each of the types listed in the tables are given in Chap. 6. It should be clear that *TPO* is assumed to be extensible in its set of types, set of type constructors, and set of typing rules.

(*Caveat: the following tables are subject to revision. Trust better the rules in Chap. 6.*)

Type Group	**Type**	**Formal Name**	**Domain**
Basic Type	Meta-type Set	*Set*	External
	Power-set Constructor	$\wp$	External
	Time	T	$\mathbb{N} = 0, 1, 2, \ldots$
	Property	*Prop*	External
	Relation	*Rel*	External
	Event	*Event*	External
	Process	*Proc*	$T \rightarrow \wp(Event)$
	Exchange Process	*ExchProc*	$T \rightarrow \wp(Event) \times \wp(Event)$
	Input Port	*InpPort*	External
	Output Port	*OutPort*	External
	Import Channel	*ImpChnl*	External
	Export Channel	*ExpChnl*	External
Populational Structure	Agent	*Agent*	External
	Agent Property	*AgProp*	$Agent \times Prop$
	Agent Behavior	*AgBeh*	$Agent \times Proc$
	Agent Interaction	*AgInter*	$Agent \times ExchProc$
	Agent Relation	*AgRel*	$Agent \times Agent \times Rel$
	Populational Structure	*Pop*	$\wp(Agent) \times \wp(AgBeh) \times \wp(AgLnk)$
Sociability Structure	Sociability Role	*SocRo*	External
	Sociability Role Property	*SocRoProp*	$SocRo \times Prop$
	Sociability Role Behavior	*SocRoBeh*	$SocRo \times Proc$
	Sociability Role Interaction	*SocRoInter*	$SocRo \times SocRo \times ExchProc$
	Sociability Role Relation	*SocRoRel*	$SocRo \times SocRo \times Rel$
	Sociability Role Link	*SocRolLnk*	$SocRo \times SocRo \times ExchProc \times Rel$
	Sociability Role Network	*SocRolNet*	$\wp(SocRo) \times \wp(SocRoProp) \times \wp(SocRoBeh) \times \wp(SocRoLnk)$
	Sociability Structure	*Soc*	$\wp(SocRo) \times \wp(SocRoProp) \times \wp(SocRoBeh) \times \wp(SocRoLnk) \times \wp(SocRoNet)$

Type Group	Type	Formal Name	Domain
Organizational Structure	Organizational Role	$OrgRo$	External
	Organizational Role Property	$OrgRoProp$	$OrgRol \times Prop$
	Organizational Role Behavior	$OrgRoBeh$	$OrgRol \times Proc$
	Organizational Role Interaction	$OrgRoInter$	$OrgRol \times OrgRo \times ExchProc$
	Organizational Role Relation	$OrgRoRel$	$OrgRol \times OrgRo \times Rel$
	Organizational Role Link	$OrgRolLnk$	$OrgRol \times OrgRo \times ExchProc \times Rel$
	Organizational Role Network	$OrgRolNet$	$\wp(OrgRol) \times \wp(OrgRoProp) \times \wp(OrgRoBeh) \times \wp(OrgRoLnk)$
	Organization Unit	$OrgUn$	$\wp(OrgRoNet) \times \wp(OrgRo) \times \wp(InPort) \times \wp(OutPort)$
	Organization Unit Property	$OrgUnProp$	$OrgUn \times Prop$
	Organization Unit Behavior	$OrgUnBeh$	$OrgUn \times Proc$
	Organization Unit Interaction	$OrgUnInter$	$OrgUn \times OrgUn \times ExchProc$
	Organization Unit Relation	$OrgUnRel$	$OrgUn \times OrgUn \times Rel$
	Organization Unit Network	$OrgUnlNet$	$\wp OrgUn \times \wp(OrgUnProp) \times \wp(OrgUnBeh \times \wp(OrgUnLnk$
	Organizational Structure	Org	$\wp OrgUn \times \wp(OrgUnProp) \times \wp(OrgUnBeh \times \wp(OrgUnLnk \times \wp(OrgUnNet)$
Material Environment	Material Object	$MatObj$	External
	Material Object Property	$MatObjProp$	$MatObj \times Prop$
	Material Object Behavior	$MatObjBeh$	$MatObj \times Proc$
	Material Object Interaction	$MatObjInter$	$MatObj \times MatObj \times ExchProc$
	Material Object Relation	$MatObjRel$	$MatObj \times MatObj \times Rel$
	Material Object Network	$MatObjNet$	$\wp(MatObj) \times \wp(MatObjProp) \times \wp(MatObjBeh) \times \wp(MatObjLnk)$
	Material Environment	$MatEnv$	$\wp(MatObj) \times \wp(MatObjProp) \times \wp(MatObjBeh) \times \wp(MatObjLnk \times \wp(MatObjNet)$
Symbolic Environment	Symbolic Object	$SymbObj$	External
	Symbolic Object Property	$SymbObjProp$	$SymbObj \times Prop$
	Symbolic Object Behavior	$SymbObjBeh$	$SymbObj \times Proc$
	Symbolic Object Interaction	$SymbObjInter$	$SymbObj \times SymbObj \times ExchProc$
	Symbolic Object Relation	$SymbObjRel$	$SymbObj \times SymbObj \times Rel$
	Symbolic Object Network	$SymbjNet$	$\wp(SymbObj) \times \wp(MatObjProp) \times \wp(MatObjBeh) \times \wp(MatObjLnk)$
	Symbolic Environment	$MatEnv$	$\wp(SymbObj) \times \wp(SymbObjProp) \times \wp(SymbObjBeh) \times \wp(SymbObjLnk \times \wp(SymbObjNet)$
	Norm	Nrm	$NrmExp$

Type Group	**Type**	**Formal Name**	**Domain**
Implementation relations	Implementation of *Soc* by *Pop*	$Imp_{SocRo/Pop}$	$\wp(Agent \times SocRo)$
	Implementation of *Org* by *Soc*	$Imp_{Org/Soc}$	$\wp(OrgRo \times SocRo)$
	Direct Implementation of *Org* by *Pop*	$Imp_{Org/Pop}$	$\wp(OrgRo \times Agent)$
Access link	Access link: *Agent* to *MatObj*	$Acc_{Agent/MatObj}$	$Agent \times MatObj \times ExchProc$
	Access link: *Agent* to *SymbObj*	$Acc_{Agent/SymbObj}$	$Agent \times SymbObj \times ExchProc$
	Access link: *OrgUn* to *MatObj*	$Acc_{OrgUn/MatObj}$	$OrgUn \times MatObj \times ExchProc$
	Access link: *OrgUn* to *SymbObj*	$Acc_{OrgUn/SymbObj}$	$OrgUn \times SymbObj \times ExchProc$
Agent Society	Agent Society	$AgSoc$	$Pop \times Soc \times Org \times MEnv \times SEnv \times ImpRel \times AccRel$

The above is a summary of the predefined types of the *TPO* type system. Other types may be obtained by operating on these predefined types through the type constructors (product, powerset, or any other type constructor defined anew for the system as, e.g., finite mappings to define types for *record*-like structures).

Chapter 6

TPO in Detail

6.1 Overview of the *TPO* Types

In this chapter, we formally introduce the types and typing rules of *TPO*. Section 6.2 introduces the *basic* types and their typing rules, Sects. 6.3 to 6.9 introduce the predefined *constructed* types and their typing rules.

6.2 The Predefined Basic Types

The *predefined basic types* are: time, property, relation, norm expression, event, input and output ports, import and export channels. We formally introduce them presently. For each of them we give their rules and meta-rules.

Except for the process and exchange process types, predefined basic types have no *destructor* rules, only *constructor* rules.

6.2.1 Time

The elements of the type T are called *time instants*, or simply *times*. They are formally denoted by the natural numbers ($\mathbb{N} = \{0, 1, 2, ...\}$), taken in their usual order. Type T is an *inanimate internal type*.

- Meta-Rule MR_{T}:

$$\frac{}{\mathrm{T} :: \mathit{Inanimate}}\ MR_{\mathrm{T}}$$

- Rule R_{T}:

$$\frac{t \in \mathbb{N}}{t : \mathrm{T}}\ R_{\mathrm{T}}$$

6.2.2 Predicate

The elements of the type *Pred* are called *predicate expressions*. The type *Pred* is an *inanimate external type*.

- Meta-Rule MR_{Pred}:

$$\frac{}{Pred :: Inanimate}\ MR_{Pred}$$

- Rule ER_{Pred}:

$$\frac{[p...]_{Pred}}{p : Pred}\ ER_{Pred}$$

6.2.3 Relation

The elements of the type *Rel* are called *relation expressions*. The type *Rel* is an *inanimate external type*.

- Meta-Rule MR_{Rel}:

$$\frac{}{Rel :: Inanimate}\ MR_{Rel}$$

- Rule ER_{Rel}:

$$\frac{[r...]_{Rel}}{r : Rel}\ ER_{Rel}$$

6.2.4 Norm Expression

The elements of *NrmExpr* are called *norm expressions*. The type *Nrm* is an *inanimate external type*.

- Meta-Rule MR_{Nrm}:

$$\frac{}{Nrm :: Inanimate}\ MR_{Nrm}$$

- Rule ER_{Nrm}:

$$\frac{[nrm...]_{Nrm}}{nrm : Nrm}\ ER_{Nrm}$$

6.2.5 Event

The elements of *Event* are called *events*. The type *Event* is an *inanimate external type*.

- Meta-Rule MR_{Event}:

$$\frac{}{Event :: Inanimate} \; MR_{Event}$$

- Rule ER_{Event}:

$$\frac{[ev...]_{Event}}{ev : Event} \; ER_{Event}$$

6.2.6 *Input Port* and *Output Port*

Input and *output ports* are elements that allow organization units to inter-operate with other components (agents, organization units) of agent societies.Types *InpPort* and *OutPort* are *inanimate external types.*

- Rule $MR_{InpPort}$:

$$\frac{}{InpPort :: Inanimate} \; MR_{InpPort}$$

- Rule $ER_{InpPort}$:

$$\frac{[inp...]_{InpPort}}{inp : InpPort} \; ER_{InpPort}$$

- Rule $MR_{OutPort}$:

$$\frac{}{OutPort :: Inanimate} \; MR_{OutPort}$$

- Rule $ER_{OutPort}$:

$$\frac{[out...]_{OutPort}}{out : OutPort} \; ER_{OutPort}$$

6.2.7 *Import Channel* and *Export Channel*

Import and *export channels* are elements that allow agent societies to inter-operate with other types of external systems. Types *ImpChnl* and *ExpChnl* are *inanimate external types.*

- Rule MR_{ImpChn}:

$$\frac{}{\mathit{ImpChn} :: \mathit{Inanimate}}\ MR_{ImpChn}$$

- Rule ER_{ImpChn}:

$$\frac{[\mathit{imp}...]_{\mathit{ImpChn}}}{\mathit{imp} : \mathit{ImpChn}}\ ER_{ImpChn}$$

- Rule MR_{ExpChn}:

$$\frac{}{\mathit{ExpChn} :: \mathit{Inanimate}}\ MR_{ExpChn}$$

- Rule ER_{ExpChn}:

$$\frac{[\mathit{exp}...]_{\mathit{ExpChn}}}{\mathit{exp} : \mathit{ExpChn}}\ ER_{ExpChn}$$

6.3 The *Process* and *Exchange Process* Types

The process and exchange process types are essential in the *TPO* type system.

6.3.1 Process

Processes are the fundamental *dynamic elements* of *TPO*. Any structure constructed according to *TPO* may encompass one or more processes. *Processes* are constituted by time-indexed sequences of *sets of events.* That is, we assume that *at each time instant*, the set of components of the society that cause the *process* may perform a *set of events* at that time, not just *one* event. The type *Proc* is an *inanimate internal type* given by the *function space* type constructor.

- Meta-Rule MR_{Proc}:

$$\frac{\mathrm{T} :: \mathit{Inanimate} \qquad \wp(\mathit{Event}) :: \mathit{Inanimate}}{\mathit{Proc} = \mathrm{T} \to \wp(\mathit{Event}) :: \mathit{Inanimate}}\ MR_{Proc}$$

- Rule R_{Proc}:

$$\frac{ev : T \to \wp(Event)}{ev : Proc}\ R_{Proc}$$

- Rule $DR[\wp(Event)]_{Proc}$

$$\frac{ev : Proc \qquad t : \mathrm{T}}{ev(t) : \wp(Event)}\ DR[\wp(Event)]_{Proc}$$

6.3.2 Exchange Process

An *exchange process* is a special type of process, namely, a process whose elements are given by *pairs of sets of events*. *Exchange processes* constitute the essential dynamic element of *interactions* between two system elements, which is the reason for giving its elements in terms of *pairs* of sets of events. We denote the first pair of events by the label "1" and the second component by the label "2". Notice that, formally, exchange processes are *not* a subtype of processes. The type *ExchProc* is an *inanimate internal type*, given by the *function space* type constructor.

- Meta-Rule $MR_{ExchProc}$:

$$\frac{\mathrm{T} :: Inanimate \qquad \wp(Event) \times \wp(Event) :: Inanimate}{ExchProc = \mathrm{T} \to \wp(Event) \times \wp(Event) :: Inanimate}\ MR_{ExchProc}$$

- Rule $R_{ExchProc}$:

$$\frac{ep : T \to \wp(Event) \times \wp(Event)}{ep : ExchProc}\ R_{ExchProc}$$

- Rule $DR[\wp(Event)_1]_{ExchProc}$:

$$\frac{ep : ExchProc \qquad t : \mathrm{T}}{ep(t)[1] : \wp(Event)}\ DR[\wp(Event)_1]_{ExchProc}$$

- Rule $DR[\wp(Event)_2]_{ExchProc}$:

$$\frac{ep : ExchProc \qquad t : \mathrm{T}}{ep(t)[2] : \wp(Event)}\ DR[\wp(Event)_2]_{ExchProc}$$

6.4 Type *Populational Structure*

The following are the component types of the type *populational structure*:

6.4.1 Agent

The type *Agent* is taken to be an *animate external type*, meaning that *TPO* is neutral with respect to the concept of agent existent in the agent societies and inter-societal agent systems.

• Meta-Rule MR_{Agent}:

$$\frac{}{Agent :: Animate} \; MR_{Agent}$$

• Rule ER_{Agent}:

$$\frac{[ag...]_{Agent}}{ag : Agent} \; ER_{Agent}$$

6.4.2 Agent Property, Behavior, Interaction and Relation

a) Agent Property

The type *AgProp* is an *inanimate external type*, given by the *product* type constructor.

• Meta-Rule MR_{AgProp}:

$$\frac{Ag :: Animate \qquad Prop :: Inanimate}{AgProp = Ag \times Prop :: Inanimate} \; MR_{AgProp}$$

• Rule R_{AgProp}

$$\frac{ag : Agent \qquad prop : Prop \qquad [\langle ag, prop\rangle ...]_{AgProp}}{\langle ag, prop\rangle : AgProp} \; R_{AgProp}$$

• Rule $DR[Agent]_{AgProp}$:

$$\frac{\langle ag, prop\rangle : AgProp}{ag : Agent} \; DR[Agent]_{AgProp}$$

• Rule $DR[Prop]_{AgProp}$:

$$\frac{\langle ag, prop\rangle : AgProp}{pred : Prop} \; DR[Prop]_{AgProp}$$

b) Agent Behavior

An object of the type *agent behavior* is a process. In an *agent behavior*, at each time, the agent is taken to be the cause of the set of actions that constitute the behavior at that time. Thus, since the component that causes the events of a behavior is known (the agent, in the case), those events are said to be *actions* of that agent. The type *AgBeh* is an *inanimate external type*, given by the *product* type constructor.

- Meta-Rule MR_{AgBeh}:

$$\frac{Ag :: Animate \qquad Proc :: Inanimate}{AgBeh = Ag \times Proc :: Inanimate}\ MR_{AgBeh}$$

- Rule R_{AgBeh}:

$$\frac{ag : Agent \qquad proc : Proc}{\langle ag, proc \rangle : AgBeh}\ R_{AgBeh}$$

- Rule $DR[Agent]_{AgBeh}$:

$$\frac{\langle ag, proc \rangle : AgBeh}{ag : Agent}\ DR[Agent]_{AgBeh}$$

- Rule $DR[Proc]_{AgBeh}$:

$$\frac{\langle ag, proc \rangle : AgBeh}{proc : Proc}\ DR[Proc]_{AgBeh}$$

c) Agent Interaction

An object of type *agent interaction* is an exchange process between two agents. In an *agent interaction*, the *first agent* of the pair is considered to be the cause of the *first set of events* of the pair of events that constitute the exchange process at that time. Correspondingly, the *second agent* is considered to be the cause of the *second set of events* of the pair. Since the components that cause the events of the exchange process are known (the interacting agents, in the case), those events are said to be *actions* of the agents that caused them. But, not every tuple $\langle ag_1, ag_2, ep \rangle$ is an *agent interaction*, a constraint between the exchange process ep and the behaviors of ag_1 and ag_2 has to be satisfied. The type *AgInter* is an *inanimate internal type*, given by the *product* type constructor.

- Meta-Rule $MR^*_{AgInter}$:

$$\frac{Ag :: Animate \qquad Ag :: Animate \qquad ExchProc :: Inanimate}{AgInter = Ag \times Ag \times ExchProc :: Inanimate}\ MR^*_{AgInter}$$

———————————— . ————————————

- Rule $R^*_{AgInter}$:

$$\frac{ag_1 : Agent \quad ag_2 : Agent \quad ep : ExchProc}{\langle ag_1, ag_2, ep\rangle : AgInter} \, R^*_{AgInter}$$

- Constraint $R^*_{AgInter}$:

$$\begin{array}{l} \forall \langle ag_1, ag_2, ep\rangle : AgInter \\ \qquad \exists \langle ag_1, beh_1\rangle, \langle ag_2, beh_2\rangle : AgBeh \\ \qquad\qquad \forall t : T[ep(t) = \langle beh_1(t), beh_2(t)\rangle] \end{array}$$

———————————— . ————————————

———————————— . ————————————

- Rule $DR[Agent_1]_{AgInter}$:

$$\frac{\langle ag_1, ag_2, ep\rangle : AgInter}{ag_1 : Agent} \, DR[Agent_1]_{AgInter}$$

- Rule $DR[Agent_2]_{AgInter}$:

$$\frac{\langle ag_1, ag_2, ep\rangle : AgInter}{ag_2 : Agent} \, DR[Agent_2]_{AgInter}$$

- Rule $DR[ExchProc]_{AgInter}$:

$$\frac{\langle ag_1, ag_2, ep\rangle : AgInter}{ep : ExchProc} \, DR[ExchProc]_{AgInter}$$

———————————— . ————————————

d) Agent Relation

The type *AgRel* is an *inanimate external type*, given by the *product* type constructor.

———————————— . ————————————

- Meta-Rule MR_{AgRel}:

$$\frac{Ag :: Animate \quad Ag :: Animate \quad Rel :: Inanimate}{AgRel = Ag \times Ag \times Rel :: Inanimate} \, MR_{AgRel}$$

———————————— . ————————————

———————————— . ————————————

- Rule ER_{AgRel}:

$$\frac{ag_1 : Agent \quad ag_2 : Agent \quad rel : Rel \quad [\langle ag_1, ag_2, rel\rangle ...]_{AgRel}}{\langle ag_1, ag_2, rel\rangle : AgRel} \, ER_{AgRel}$$

———————————— . ————————————

- Rule $DR[Agent_1]_{AgRel}$:

$$\frac{\langle ag_1, ag_2, rel\rangle : AgRel}{ag_1 : Agent}\ DR[Agent_1]_{AgRel}$$

- Rule $DR[Agent_2]_{AgRel}$:

$$\frac{\langle ag_2, ag_2, rel\rangle : AgRel}{ag_2 : Agent}\ DR[Agent_2]_{AgRel}$$

- Rule $DR[Rel]_{AgRel}$:

$$\frac{\langle ag_2, ag_2, rel\rangle : AgRel}{rel : Rel}\ DR[Rel]_{AgRel}$$

6.4.3 Populational Structure

At each time, the *populational structure* of an *agent society* is a composed by:

- the set of all *agents* existent in that agent society, at that time;
- the set of all *agent properties* of those agents;
- the set of all *agent behaviors* of those agents;
- the set of all *agent interactions* existent between those agents.

The type *Pop* is an *animate internal type*, given by the *product* type constructor.

- Meta-rule MR_{Pop}:

$$\frac{\begin{array}{c}\wp(Ag) :: Animate \\ \wp(AgProp) :: Inanimate \\ \wp(AgRel) :: Inanimate \\ \wp(AgInter) :: Inanimate\end{array}}{Pop = \wp(Ag) \times \wp(AgProp) \times \wp(AgRel) \times \wp(AgInter) :: Animate}\ MR_{Pop}$$

- Rule ER_{Pop}:

$$\frac{\begin{array}{c}AG : \wp(Ag) \\ AP : \wp(AgProp) \\ AB : \wp(AgBeh) \\ AI : \wp(AgInter)\end{array}}{\langle AG, AG, AB, AG\rangle : Pop}\ ER_{Pop}$$

---

* Rule $DR[\wp(Agent)]_{Pop}$:

$$\frac{\langle AG, AP, AB, AI\rangle : Pop}{AG : \wp(Ag)}\ DR[\wp(Ag)]_{Pop}$$

* Rule $DR[\wp(AgProp)]_{Pop}$:

$$\frac{\langle AG, AP, AB, AI\rangle : Pop}{AP : \wp(AgProp)}\ DR[\wp(Prop)]_{Pop}$$

* Rule $DR[\wp(AgBeh)]_{Pop}$:

$$\frac{\langle AG, AP, AB, AI\rangle : Pop}{AB : \wp(AgBeh)}\ DR[\wp(AgBeh)]_{Pop}$$

* Rule $DR[\wp(AgInter)]_{Pop}$:

$$\frac{\langle AG, AP, AB, AI\rangle : Pop}{AI : \wp(AgInter)}\ DR[\wp(AgInter)]_{Pop}$$

6.5 Type *Sociability Structure*

A *sociability structure* is composed by the set of *sociability roles* that agents may enact *for each other* (together with the *behaviors*, *properties*, *interactions* and *relations* that those sociability roles perform or have) and by the set of *sociability role networks* that the *sociability roles* form in the society.

6.5.1 Sociability Role

Sociability role is an *animate external type*, meaning that *TPO* is neutral with respect to the concept of sociability role existent in the agent societies and inter-social agent systems.

* Meta-rule $MR_{SocRole}$

$$\frac{}{SocRo :: Animate}\ MR_{SocRole}$$

* Rule ER_{SocRo}:

$$\frac{[ro...]_{SocRo}}{ro : SocRo}\ ER_{SocRo}$$

6.5.2 Sociability Role Property, Behavior, Interaction, and Relation

a) Sociability Role Property

The type *SocRoProp* is an *inanimate external type.*

- Meta-Rule $MR_{SocRoProp}$:

$$\frac{}{SocRoProp :: Inanimate}\ MR_{SocRoProp}$$

- Rule $ER_{SocRoProp}$:

$$\frac{ro : SocRo \quad prop : Prop \quad [\langle ro, prop\rangle ...]_{SocRoProp}}{\langle ro, prop\rangle : SocRoProp}\ ER_{SocRoProp}$$

- Rule $DR[SocRo]_{SocRoProp}$:

$$\frac{\langle ro, prop\rangle : SocRoProp}{ro : SocRo}\ ER[SocRo]_{SocRoProp}$$

- Rule $DR[Prop]_{SocRoProp}$:

$$\frac{\langle ro, prop\rangle : SocRoProp}{prop : Prop}\ ER[Prop]_{SocRoProp}$$

b) Sociability Role Behavior

Behaviors of sociability roles are processes. In a *sociability role behavior*, at each time, the *sociability role* is taken to be the cause of the *set of actions* that constitute the *behavior* at that time. The type *SocRoBeh* is an *inanimate external type.*

- Meta-rule $MR_{SocRoBeh}$:

$$\frac{}{SocRoBeh :: Inanimate}\ MR_{SocRoBeh}$$

- Rule $R_{SocRoBeh}$:

$$\frac{ro : SocRo \quad proc : Proc \quad [\langle ro, proc\rangle ...]_{SocRoBeh}}{\langle ro, proc\rangle : SocRoBeh}\ R_{SocRoBeh}$$

- Rule $\mathit{DR}[\mathit{SocRo}]_{\mathit{SocRoBeh}}$:

$$\frac{\langle ro, proc\rangle : \mathit{SocRoBeh}}{ro : \mathit{SocRo}}\ \mathit{DR}[\mathit{SocRo}]_{\mathit{SocRoBeh}}$$

- Rule $\mathit{DR}[\mathit{Proc}]_{\mathit{SocRoBeh}}$:

$$\frac{\langle ro, proc\rangle : \mathit{SocRoBeh}}{proc : \mathit{Proc}}\ \mathit{DR}[\mathit{Proc}]_{\mathit{SocRoBeh}}$$

---

c) Sociability Role Interaction

Interactions of sociability roles are *exchange processes* between two *sociability roles*. In a *sociability role interaction*, the *first sociability* of the pair is considered to be the cause of the *first set of actions* of the pair of *actions* that constitute the *interaction* at that time. Correspondingly, the *second sociability role* is considered to be the cause of the *second set of actions* of the pair. However, not every tuple $\langle ro_1, ro_2, ep\rangle$ is a *sociability role interaction*, a certain *constraint* between the interaction and the behaviors of the sociability roles has to be satisfied. The type *SocRoInter* is an *inanimate external type*.

- Meta-rule $\mathit{MR}^*_{\mathit{SocRoInter}}$:

$$\frac{}{\mathit{SocRoInter} :: \mathit{Inanimate}}\ \mathit{MR}^*_{\mathit{SocRoInter}}$$

- Rule $R^*_{\mathit{SocRoInter}}$:

$$\frac{\begin{array}{c} ro_1 : \mathit{SocRo} \\ ro_2 : \mathit{SocRo} \\ ep : \mathit{ExchProc} \\ [\langle ro_1, ro_2, ep\rangle ...]_{\mathit{SocRoInter}} \end{array}}{\langle ro_1, ro_2, ep\rangle : \mathit{SocRoInter}}\ R^*_{\mathit{SocRoInter}}$$

- Constraint $R^*_{\mathit{SocRoInter}}$:

$$\begin{array}{l} \forall \langle ro_1, ro_2, ep\rangle : \mathit{SocRoInter} \\ \qquad \exists \langle ro_1, beh_1\rangle, \langle ro_2, beh_2\rangle : \mathit{SocRoBeh} \\ \qquad\qquad \forall t : T[ep(t) = \langle beh_1(t), beh_2(t)\rangle] \end{array}$$

- Rule $\mathit{DR}[\mathit{SocRo}_1]_{\mathit{SocRoInter}}$:

$$\frac{\langle ro_1, ro_2, ep\rangle : \mathit{SocRoInter}}{ro_1 : \mathit{SocRo}}\ \mathit{DR}[\mathit{SocRo}_1]_{\mathit{SocRoInter}}$$

- Rule $\mathit{DR}[\mathit{SocRo}_2]_{\mathit{SocRoInter}}$:

$$\frac{\langle ro_1, ro_2, ep\rangle : \mathit{SocRoInter}}{ro_2 : \mathit{SocRo}}\ \mathit{DR}[\mathit{SocRo}_2]_{\mathit{SocRoInter}}$$

- Rule $\mathit{DR}[\mathit{ExchProc}]_{\mathit{SocRoInter}}$:

$$\frac{\langle ro_1, ro_2, ep\rangle : \mathit{SocRoInter}}{ep : \mathit{ExchProc}}\ \mathit{DR}[\mathit{ExchProc}]_{\mathit{SocRoInter}}$$

---

d) Sociability Role Relation

The type *SocRoRel* is an *inanimate external type.*

• Meta-Rule $MR_{SocRoRel}$:

$$\frac{}{\mathit{SocRoRel} :: \mathit{Inanimate}}\ MR_{SocRoRel}$$

• Rule $ER_{SocRoRel}$:

$$\frac{\begin{array}{c} ro_1 : \mathit{SocRo} \\ ro_2 : \mathit{SocRo} \\ rel : \mathit{Rel} \\ [\langle ro_1, ro_2, rel\rangle ...]_{SocRoRel} \end{array}}{\langle ro_1, ro_2, rel\rangle : \mathit{SocRoRel}}\ ER_{SocRoRel}$$

• Rule $DR[SocRo_1]_{SocRoRel}$:

$$\frac{\langle ro_1, ro_2, rel\rangle : \mathit{SocRoRel}}{ro_1 : \mathit{SocRo}}\ DR[SocRo_1]_{SocRoRel}$$

• Rule $DR[SocRo_2]_{SocRoRel}$:

$$\frac{\langle ro_1, ro_2, rel\rangle : \mathit{SocRoRel}}{ro_2 : \mathit{SocRo}}\ DR[SocRo_2]_{SocRoRel}$$

• Rule $DR[Rel]_{SocRoRel}$:

$$\frac{\langle ro_1, ro_2, rel\rangle : \mathit{SocRoRel}}{rel : \mathit{Rel}}\ DR[Rel]_{SocRoRel}$$

6.5.3 Sociability Role Network

Sociability role networks are composed by sets of *sociability roles*, and some of their behaviors, interactions and relations. In a *sociability role network*, no *sociability role behavior* may exist without the corresponding existence in the network of the *sociability role* that realizes it, and no *sociability role interaction* or *sociability role relation* may exist without the corresponding existence in the network of the two *sociability roles* that realize them. Notice that the decision of which sociability role behaviors, interactions and relations will belong to a *sociability role network* is a decision up to the external observer. But not every tuple $\langle SR, SRBeh, SRInter, SRRel\rangle$ constitutes a *sociability role network*, certain *constraints* have to be satisfied. Type *SocRoNet* is an *animate internal type.*

• Meta-rule $MR^*_{SocRoNet}$:

$$\frac{}{\mathit{SocRoNet} :: \mathit{Animate}}\ MR^*_{SocRoNet}$$

———————————— . ————————————

- Rule $R^*_{SocRoNet}$:

$$\frac{\begin{array}{c} SR : \wp(SocRo) \\ SRProp : \wp(SocRoProp) \\ SRBeh : \wp(SocRoBeh) \\ SRInter : \wp(SocRoInter) \\ SRRel : \wp(SocRoRel) \end{array}}{\langle SR, SRProp, SRBeh, SRInter, SRRel\rangle : SocRoNet}\ R^*_{SocRoNet}$$

- Constraints $R^*_{SocRoNet}$:

$\forall\langle SR, SRProp, SRBeh, SRRel\rangle : SocRoNet\ \ [SR \neq \emptyset]$

$$\begin{array}{l} \forall\langle SR, SRProp, SRBeh, SRRel\rangle : SocRoNet \\ \qquad [\forall\langle ro, prop\rangle \in SRProp : [ro \in SR \\ \qquad\qquad\qquad \wedge \langle ro, prop\rangle : SocRoProp] \\ \qquad \wedge \forall\langle ro, beh\rangle \in SRBeh : [ro \in SR \\ \qquad\qquad\qquad \wedge \langle ro, beh\rangle : SocRoBeh] \\ \qquad \wedge \forall\langle ro_1, ro_2, ep\rangle \in SRInter[ro_1, ro_2 \in SR \\ \qquad\qquad\qquad\qquad \wedge \langle ro_1, ro_2, ep\rangle : SocRoInter \\ \qquad \wedge \forall\langle ro_1, ro_2, rel\rangle \in SRRel[ro_1, ro_2 \in SR \\ \qquad\qquad\qquad\qquad \wedge \langle ro_1, ro_2, rel\rangle : SocRoRel]] \end{array}$$

———————————— . ————————————

———————————— . ————————————

- Rule $DR[\wp(SocRo)]_{SocRoNet}$:

$$\frac{\langle SR, SRProp, SRBeh, SRInter, SRRel\rangle : SocRoNet}{SR : \wp(SocRo)}\ DR[\wp(SocRo)]_{SocRoNet}$$

- Rule $DR[\wp(SocRoProp)]_{SocRoNet}$:

$$\frac{\langle SR, SRProp, SRBeh, SRInter, SRRel\rangle : SocRoNet}{SRProp : \wp(SocRoProp)}\ DR[\wp(SocRoProp)]_{SocRoNet}$$

- Rule $DR[\wp(SocRoBeh)]_{SocRoNet}$:

$$\frac{\langle SR, SRProp, SRBeh, SRInter, SRRel\rangle : SocRoNet}{SRBeh : \wp(SocRoBeh)}\ DR[\wp(SocRoBeh)]_{SocRoNet}$$

- Rule $DR[\wp(SocRoInter)]_{SocRoNet}$:

$$\frac{\langle SR, SRProp, SRBeh, SRInter, SRRel\rangle : SocRoNet}{SRInter : \wp(SocRoLnk)}\ DR[\wp(SocRoInter)]_{SocRoNet}$$

- Rule $DR[\wp(SocRoRel)]_{SocRoNet}$:

$$\frac{\langle SR, SRProp, SRBeh, SRInter, SRRel\rangle : SocRoNet}{SRRel : \wp(SocRoRel)}\ DR[\wp(SocRoRel)]_{SocRoNet}$$

———————————— . ————————————

6.5.4 Sociability Structure

At each time, the *sociability structure* of an *agent society* is composed by:

- the set of all *sociability roles* existent in that society, at that time;
- the set of all *sociability role properties* of those sociability roles;
- the set of all *sociability role behaviors* of those sociability roles;
- the set of all *sociability role interactions* existent between those sociability roles;
- the set of all *sociability role relations* existent between those sociability roles;
- the set of all *sociability role networks* existent in that society, at that time.

The type *Soc*, of sociability structures, is an *animate internal type.*

- Meta-rule MR_{Soc}:

$$\frac{}{\mathit{Soc} :: \mathit{Animate}}\ MR_{Soc}$$

- Rule R_{Soc}:

$$\frac{\begin{array}{c} \mathit{SR} : \wp(\mathit{SocRo}) \\ \mathit{SRProp} : \wp(\mathit{SocRoProp}) \\ \mathit{SRBeh} : \wp(\mathit{SocRoBeh}) \\ \mathit{SRInter} : \wp(\mathit{SocRoLnk}) \\ \mathit{SRRel} : \wp(\mathit{SocRoRel}) \\ \mathit{SRNet} : \wp(\mathit{SocRoNet}) \end{array}}{\langle \mathit{SR}, \mathit{SRBeh}, \mathit{SRInter}, \mathit{SRRel}, \mathit{SRNet} \rangle : \mathit{Soc}}\ R_{Soc}$$

- Rule $DR[\mathit{SocRo}]_{Soc}$:

$$\frac{\langle \mathit{SR}, \mathit{SRBeh}, \mathit{SRInter}, \mathit{SRRel}, \mathit{SRNet} \rangle : \mathit{Soc}}{\mathit{SR} : \wp(\mathit{SocRo})}\ DR[\mathit{SocRo}]_{Soc}$$

- Rule $DR[\wp(\mathit{SocRoBeh})]_{Soc}$:

$$\frac{\langle \mathit{SR}, \mathit{SRBeh}, \mathit{SRInter}, \mathit{SRRel}, \mathit{SRNet} \rangle : \mathit{Soc}}{\mathit{SRBeh} : \wp(\mathit{SocRoBeh})}\ DR[\wp(\mathit{SocRoBeh})]_{Soc}$$

- Rule $DR[\wp(\mathit{SocRoInter})]_{Soc}$:

$$\frac{\langle SR, SRBeh, SRInter, SRRel, SRNet\rangle : Soc}{SRInter : \wp(SocRoInter)}\ DR[\wp(SocRoInter)]_{Soc}$$

• Rule $DR[\wp(SocRoRel)]_{Soc}$:

$$\frac{\langle SR, SRBeh, SRInter, SRRel, SRNet\rangle : Soc}{SRRel : \wp(SocRoRel)}\ DR[\wp(SocRoRel)]_{Soc}$$

• Rule $DR[\wp(SocRoNet)]_{Soc}$:

$$\frac{\langle SR, SRBeh, SRInter, SRRel, SRNet\rangle : Soc}{SRNet : \wp(SocRoNet)}\ DR[\wp(SocRoNet)]_{Soc}$$

———————— . ————————

6.6 Type *Organizational Structure*

An *organizational structure* is composed by:

- its *organizational micro-level* (Org_ω): the set of *organizational roles* (*OrgRo*) and corresponding *organizational role behaviors*, *properties*, *interactions* and *links*;
- its *organizational meso-level* (Org_μ): the set of *organization units* (*OrgUn*), and their *behaviors* and *links*, that may be formed in the society by networking *organizational roles* and encapsulating them by means of *interfaces*, so that such organization units exchange objects with the elements that are external to them only by means of *input* and *output ports*;
- its organizational macro-level (Org_Ω): the set of *organization unit networks* (*OrgUnNet*) that are maximal[1] among the *organization unit networks* that *organization units* may form.

6.6.1 Organizational Role

Type *OrgRo* is an inanimate external type.

———————— . ————————

• Rule MR_{OrgRo}:

$$\frac{}{OrgRo :: Animate}\ MR_{OrgRo}$$

———————— . ————————

———————— . ————————

• Rule ER_{OrgRo}:

$$\frac{[ro...]_{OrgRo}}{ro : OrgRo}\ ER_{OrgRo}$$

———————— . ————————

[1]That is, that have not been encapsulated within other *organization units*, see below.

6.6.2 Organizational Role Property, Behavior, Interaction, Relation and Link

Organizational Role Property

Type *OrgRoProp* is an *inanimate external type.*

• Meta-rule $MR_{OrgRoProp}$

$$\frac{}{OrgRoProp :: Inanimate}\ MR_{OrgRoProp}$$

• Rule $ER_{OrgRoProp}$:

$$\frac{ro : OrgRo \qquad prop : Prop \qquad [\langle ro, prop\rangle ...]_{OrgRoProp}}{\langle ro, prop\rangle : OrgRoProp}\ ER_{OrgRoProp}$$

• Rule $DR[OrgRo]_{OrgRoProp}$:

$$\frac{\langle ro, prop\rangle : OrgRoProp}{ro : OrgRo}\ DR[OrgRo]_{OrgRoProp}$$

• Rule $DR[Prop]_{OrgRoProp}$:

$$\frac{\langle ro, prop\rangle : OrgRoProp}{prop : Prop}\ DR[Prop]_{OrgRoProp}$$

Organizational Role Behavior

Type *OrgRoBeh* is an *inanimate external type.*

• Meta-rule $MR_{OrgRoBeh}$:

$$\frac{}{OrgRoBeh :: Inanimate}\ MR_{OrgRoBeh}$$

• Rule $R_{OrgRoBeh}$:

$$\frac{ro : OrgRo \qquad proc : Proc \qquad [\langle ro, proc\rangle ...]_{OrgRoBeh}}{\langle ro, proc\rangle : OrgRoBeh}\ R_{OrgRoBeh}$$

• Rule $DR[OrgRole]_{OrgRoBeh}$:

$$\frac{\langle ro, proc\rangle : OrgRoBeh}{ro : OrgRole}\ DR[OrgRole]_{OrgRoBeh}$$

- Rule $DR[Proc]_{OrgRoBeh}$:

$$\frac{\langle ro, proc\rangle : OrgRoBeh}{proc : Proc}\ DR[Proc]_{OrgRoBeh}$$

———————————— . ————————————

Organizational Role Interaction

An object of type *organizational role interaction* is an *exchange process* between two organizational roles. In an *organizational role interaction*, the *first organizational role* of the pair is considered to be the cause of the *first set of actions* of the pair of actions that constitute the interaction at that time. Correspondingly, the *second organizational role* is considered to be the cause of the *second set of actions* of the pair. But, not every tuple $\langle ro_1, ro_2, ep\rangle$ is an *organizational role interaction*, a certain *compatibility constraint* between the exchange process *ep* and the behaviors of the organizational roles ro_1 and ro_2 has to be satisfied. Type *OrgRoInter* is an *inanimate external type*.

———————————— . ————————————

- Meta-rule $MR^*_{OrgInter}$:

$$\frac{}{OrgInter :: Inanimate}\ MR^*_{OrgInter}$$

———————————— . ————————————

———————————— . ————————————

- Rule $R^*_{OrgRoInter}$:

$$\frac{\begin{array}{c} ro_1 : OrgRole \\ ro_2 : OrgRole \\ ep : ExchProc \\ [\langle ro_1, ro_2, ep\rangle ...]_{OrgRoInter} \end{array}}{\langle ro_1, ro_2, ep\rangle : OrgRoInter}\ R^*_{OrgRoInter}$$

- Constraint $R^*_{OrgRoInter}$:

$$\begin{array}{l} \forall\langle ro_1, ro_2, ep\rangle : OrgRoInter \\ \qquad \exists\langle ro_1, beh_1\rangle, \langle ro_2, beh_2\rangle : OrgRoBeh \\ \qquad\qquad \forall t : T[ep(t) = \langle beh_1(t), beh_2(t)\rangle] \end{array}$$

———————————— . ————————————

———————————— . ————————————

- Rule $DR[OrgRo_1]_{OrgRoInter}$:

$$\frac{\langle ro_1, ro_2, ep\rangle : \mathit{OrgRoInter}}{ro_1 : \mathit{OrgRole}}\ DR[\mathit{OrgRo}_1]_{\mathit{OrgRoInter}}$$

• Rule $DR[\mathit{OrgRo}_2]_{\mathit{OrgRoInter}}$:

$$\frac{\langle ro_1, ro_2, ep\rangle : \mathit{OrgRoInter}}{ro_2 : \mathit{OrgRole}}\ DR[\mathit{OrgRo}_2]_{\mathit{OrgRoInter}}$$

• Rule $DR[\mathit{ExchProc}]_{\mathit{OrgRoInter}}$:

$$\frac{\langle ro_1, ro_2, ep\rangle : \mathit{OrgRoInter}}{ep : \mathit{ExchProc}}\ DR[\mathit{ExchProc}]_{\mathit{OrgRoInter}}$$

Organizational Role Relation

Type *OrgRoRel* is an *inanimate external type.*

• Meta-rule MR^*_{OrgRoRel}

$$\frac{}{\mathit{OrgRoRel} :: \mathit{Inanimate}}\ MR^*_{\mathit{OrgRoRel}}$$

• Rule ER_{OrgRoRel}:

$$\frac{\begin{array}{c} ro_1 : \mathit{OrgRo} \\ ro_2 : \mathit{OrgRo} \\ rel : \mathit{Rel} \\ [\langle ro_1, ro_2, rel\rangle ...]_{\mathit{OrgRoRel}} \end{array}}{\langle ro_1, ro_2, rel\rangle : \mathit{OrgRoProp}}\ ER_{\mathit{OrgRoRel}}$$

• Rule $DR[\mathit{OrgRo}_1]_{\mathit{OrgRoRel}}$:

$$\frac{\langle ro_1, ro_2, rel\rangle : \mathit{OrgRoProp}}{ro_1 : \mathit{OrgRo}}\ DR[\mathit{OrgRo}_1]_{\mathit{OrgRoRel}}$$

• Rule $DR[\mathit{OrgRo}_2]_{\mathit{OrgRoRel}}$:

$$\frac{\langle ro_1, ro_2, rel\rangle : \mathit{OrgRoProp}}{ro_2 : \mathit{OrgRo}}\ DR[\mathit{OrgRo}_2]_{\mathit{OrgRoRel}}$$

• Rule $DR[\mathit{Rel}]_{\mathit{OrgRoRel}}$:

$$\frac{\langle ro_1, ro_2, rel\rangle : \mathit{OrgRoProp}}{rel : \mathit{Rel}}\ DR[\mathit{Rel}]_{\mathit{OrgRoRel}}$$

6.6.3 Organizational Role Network

An object of the type *organizational role network* is composed by a set of *organizational roles*, and some of their properties, behaviors, interactions and relations. In an *organizational role network*, no *organizational role property* or *behavior* may exist without the corresponding existence in the network of the *organizational role* that supports them, and no *organizational role interaction* or *organizational role relation* may exist without the corresponding existence in the network of the two *organizational roles* that realize them. Notice that the decision of which *organizational role behaviors* and which *organizational roles interactions* will belong to an *organizational role network* is a decision up to the external observer (modeler or specifier). But not every tuple $\langle RO, ROProp, ROBeh, ROInter, RORel \rangle$ constitutes an *organizational role network*, certain *constraints* have to be satisfied. Type *OrgRoNet* is an animate type.

- Meta-rule $MR^*_{OrgRoNet}$:

$$\frac{}{OrgRoNet :: Animate}\; MR^*_{OrgRoNet}$$

- Rule $R^*_{OrgRoNet}$:

$$\frac{\begin{array}{c} RO : \wp(OrgRole) \\ ROProp : \wp(OrgRoProp) \\ ROBeh : \wp(OrgRoBeh) \\ ROInter : \wp(OrgRoInter) \\ RORel : \wp(OrgRoRel) \end{array}}{\langle RO, ROProp, ROBeh, ROInter, RORel \rangle : OrgRoNet}\; R^*_{OrgRoNet}$$

- Constraints $R^*_{OrgRoNet}$:

$$\forall \langle RO, ROProp, ROBeh, ROInter, RORel \rangle : OrgRoNet \;\; [Ro \neq \emptyset]$$

$$\begin{array}{l} \forall \langle RO, ROProp, ROBeh, ROInter, RORel \rangle : OrgRoNet \\ \qquad [\forall \langle ro, prop \rangle \in ROProp : ro \in Ro \\ \qquad\qquad\qquad \wedge \langle ro, prop \rangle : OrgRoProp \\ \qquad \wedge \forall \langle ro, beh \rangle \in ROBeh : ro \in Ro \\ \qquad\qquad\qquad \wedge \langle ro, beh \rangle : OrgRoBeh \\ \qquad \wedge \forall \langle ro_1, ro_2, ep \rangle \in RoInter : ro_1, ro_2 \in Ro \\ \qquad\qquad\qquad\qquad \wedge \langle ro_1, ro_2, ep \rangle : OrgRoInter \\ \qquad \wedge \forall \langle ro_1, ro_2, rel \rangle \in RORel : ro_1, ro_2 \in Ro \\ \qquad\qquad\qquad\qquad \wedge \langle ro_1, ro_2, rel \rangle : OrgRoRel \end{array}$$

- Rule $DR[\wp(OrgRole)]_{OrgRoNet}$:

$$\frac{\langle RO, ROProp, ROBeh, ROInter, RORel \rangle : OrgRoNet}{RO : \wp(OrgRole)}\; DR[\wp(OrgRole)]_{OrgRoNet}$$

• Rule $DR[\wp(OrgRoProp)]_{OrgRoNet}$:

$$\frac{\langle RO, ROProp, ROBeh, ROInter, RORel\rangle : OrgRoNet}{ORProp : \wp(OrgRoProp)}\ DR[\wp(OrgRoProp)]_{OrgRoNet}$$

• Rule $DR[\wp(OrgRoBeh)]_{OrgRoNet}$:

$$\frac{\langle RO, ROProp, ROBeh, ROInter, RORel\rangle : OrgRoNet}{ORBeh : \wp(OrgRoBeh)}\ DR[\wp(OrgRoBeh)]_{OrgRoNet}$$

• Rule $DR[\wp(OrgRoInter)]_{OrgRoNet}$:

$$\frac{\langle Ro, Beh, Lnk\rangle : OrgRoNet}{ORInter : \wp(OrgRoInter)}\ DR[\wp(OrgRoInter)]_{OrgRoNet}$$

• Rule $DR[\wp(OrgRoRel)]_{OrgRoNet}$:

$$\frac{\langle Ro, Beh, Lnk\rangle : OrgRoNet}{ORRel : \wp(OrgRoRel)}\ DR[\wp(OrgRoRel)]_{OrgRoNet}$$

———————————— . ————————————

6.6.4 Organization Unit

Organization units may be *basic organization units* or *structured organization units. Basic organization units* are composed of *organizational role networks. Structured organization units* are composed of *organization unit networks* (defined in Sect. 6.6.6). *Organization units* have *interfaces*, constituted by *organizational roles* of their component organizational units, to encapsulate its internal elements, and inter-operate with other elements of the agent society (*agents*, *sociability roles*, *organization units*, etc.) through *input* and *output ports.* Type *OrgUn* is an *animate internal type.*

———————————— . ————————————

• Meta-rule MR^*_{OrgUn}:

$$\frac{}{OrgUn :: Animate}\ MR^*_{OrgUn}$$

———————————— . ————————————

———————————— . ————————————

• Rule $R^*_{OrgUnit_1}$:

$$\frac{\begin{array}{c}\langle OR, ORProp, ORBeh, ORInter, ORRel\rangle : OrgRoNet \\ Interf \subseteq OR \\ Inp : \wp(InpPort) \\ Out : \wp(OutPort)\end{array}}{\langle\langle OR, ORProp, ORBeh, ORInter, ORRel\rangle, Interf, Inp, Out\rangle : OrgUnit}\ R^*_{OrgUnit_1}$$

• Constraints $R^*_{OrgUnit_1}$:

$$\forall\langle\langle OR, ORProp, ORBeh, ORInter, ORRel\rangle, Interf, Inp, Out\rangle : OrgUnit\ \ [OR \neq \emptyset]$$

———————————— . ————————————

———————————— . ————————————

- Rule $R_{OrgUnit_2}$:

$$\frac{\begin{array}{c}\langle OU, \mathit{OUProp}, \mathit{OUBeh}, \mathit{OUInter}, \mathit{OURel}\rangle : \mathit{OrgUnNet}\\ \mathit{Interf}' \subseteq \mathit{INTERF}\\ \mathit{Inp}' : \wp(\mathit{InpPort})\\ \mathit{Out}' : \wp(\mathit{OutPort})\end{array}}{\langle\langle OU, \mathit{OUProp}, \mathit{OUBeh}, \mathit{OUInter}, \mathit{OURel}\rangle, \mathit{Interf}', \mathit{Inp}', \mathit{Out}'\rangle : \mathit{OrgUnit}}\ R_{OrgUnit_2}$$

where:

$$\begin{aligned}\mathit{INTERF} = &\{\mathit{Interf} \mid \langle\langle OR, \mathit{ORProp}, \mathit{ORBeh}, \mathit{ORInter}, \mathit{ORRel}\rangle, \mathit{Interf}, \mathit{Inp}, \mathit{Out}\rangle \in OU\}\\ &\cup \{\mathit{Interf} \mid \langle\langle OU', \mathit{OUProp}, \mathit{OUBeh}, \mathit{OUInter}, \mathit{OURel}\rangle, \mathit{Interf}, \mathit{Inp}, \mathit{Out}\rangle \in OU\}\end{aligned}$$

———————————— . ————————————

———————————— . ————————————

- Rule $DR[\mathit{OrgRoNet}]_{OrgUnit_1}$:

$$\frac{\langle \mathit{ORNet}, \mathit{Interf}, \mathit{Inp}, \mathit{Out}\rangle : \mathit{OrgUnit}}{\mathit{ORNet} : \mathit{OrgRoNet}}\ DR[\mathit{OrgRoNet}]_{OrgUnit_1}$$

- Rule $DR[\wp(\mathit{OrgRo})]_{OrgUnit_1}$:

$$\frac{\langle \mathit{ORNet}, \mathit{Interf}, \mathit{Inp}, \mathit{Out}\rangle : \mathit{OrgUnit}}{\mathit{Interf} : \wp(\mathit{OrgRo})}\ DR[\wp(\mathit{OrgRo})]_{OrgUnit_1}$$

- Rule $DR[\wp(\mathit{InpPort})]_{OrgUnit_1}$:

$$\frac{\langle \mathit{ORNet}, \mathit{Interf}, \mathit{Inp}, \mathit{Out}\rangle : \mathit{OrgUnit}}{\mathit{Inp} : \wp(\mathit{InpPort})}\ DR[\mathit{InpPort}]_{OrgUnit_1}$$

- Rule $DR[\wp(\mathit{OutPort})]_{OrgUnit_1}$:

$$\frac{\langle \mathit{ORNet}, \mathit{Interf}, \mathit{Inp}, \mathit{Out}\rangle : \mathit{OrgUnit}}{\mathit{Out} : \wp(\mathit{OutPort})}\ DR[\mathit{OuttPort}]_{OrgUnit_1}$$

- Rule $DR[\mathit{OUNet}]_{OrgUnit_2}$:

$$\frac{\langle \mathit{OUNet}, \mathit{Interf}, \mathit{Inp}, \mathit{Out}\rangle : \mathit{OrgUnit}}{\mathit{OUNet} : \mathit{OrgUnNet}}\ DR[\mathit{OUNet}]_{OrgUnit_2}$$

- Rule $DR[\wp(\mathit{OrgRole})]_{OrgUnit_2}$:

$$\frac{\langle \mathit{OUNet}, \mathit{Interf}, \mathit{Inp}, \mathit{Out}\rangle : \mathit{OrgUnit}}{\mathit{Interf} : \wp(\mathit{OrgRole})}\ DR[\wp(\mathit{OrgRole})]_{OrgUnit_2}$$

- Rule $DR[\wp(\mathit{InpPort})]_{OrgUnit_2}$:

$$\frac{\langle \mathit{OUNet}, \mathit{Interf}, \mathit{Inp}, \mathit{Out}\rangle : \mathit{OrgUnit}}{\mathit{Inp} : \wp(\mathit{InpPort})}\ DR[\wp(\mathit{InpPort})]_{OrgUnit_2}$$

- Rule $DR[\wp(\mathit{OutPort})]_{OrgUnit_2}$:

$$\frac{\langle \mathit{OUNet}, \mathit{Interf}, \mathit{Inp}, \mathit{Out}\rangle : \mathit{OrgUnit}}{\mathit{Out} : \wp(\mathit{OutPort})}\ DR[\wp(\mathit{OutPort})]_{OrgUnit_2}$$

———————————— . ————————————

6.6.5 Organization Unit Property, Behavior, Interaction, Relation and Link

Organization Unit Property

Type *OrgUnProp* is an *inanimate external type.*

- Meta-rule $MR_{OrgUnProp}$

$$\frac{}{OrgUnProp :: Inanimate}\ MR_{OrgUnProp}$$

- Rule $ER_{OrgUnProp}$:

$$\frac{ou : OrgUn \quad prop : Prop \quad [\langle ou, prop\rangle ...]_{OrgUnProp}}{\langle ou, prop\rangle : OrgUnProp}\ ER_{OrgUnProp}$$

- Rule $DR[OrgUn]_{OrgUnProp}$:

$$\frac{\langle ou, prop\rangle : OrgUnProp}{ou : OrgUn}\ DR[OrgUn]_{OrgUnProp}$$

- Rule $DR[Prop]_{OrgUnProp}$:

$$\frac{\langle ou, prop\rangle : OrgUnProp}{prop : Prop}\ DR[Prop]_{OrgUnProp}$$

Organization Unit Behavior

Type *OrgUnBeh* is an *inanimate external type.*

- Meta-rule $MR_{OrgUnBeh}$:

$$\frac{}{OrgUnBeh :: Inanimate}\ MR_{OrgUnBeh}$$

- Rule $R_{OrgUnBeh}$:

$$\frac{ou : OrgUnit \quad proc : Proc \quad [\langle ou, proc\rangle ...]_{OrgUnBeh}}{\langle ou, proc\rangle : OrgUnBeh}\ R_{OrgUnBeh}$$

- Rule $DR[OrgUnit]_{OrgUnBeh}$:

$$\frac{\langle ou, proc\rangle : OrgUnBeh}{ou : OrgUnit}\ DR[OrgUnit]_{OrgUnBeh}$$

- Rule $DR[ExchProc]_{OrgUnBeh}$:

$$\frac{\langle ou, proc\rangle : OrgUnBeh}{proc : ExchProc}\ DR[ExchProc]_{OrgUnBeh}$$

Organization Unit Interaction

An object of type *organization unit interaction* is an exchange process between two *organization units.* In an *organizational role interaction*, the *first organization unit* of the pair is considered to be the cause of the *first set of actions* of the pair of actions that constitute the interaction at that time. Correspondingly, the *second organization unit* is considered to be the cause of the *second set of actions* of the pair. But, not every tuple $\langle ou_1, ou_2, ep \rangle$ is an *organization unit interaction*, a certain *compatibility constraint* between the exchange process *ep* and the behaviors of the organization units ou_1 and ou_2 has to be satisfied. Type *OrgUnInter* is an *inanimate type.*

———————— . ————————

• Meta-rule $MR^*_{OrgUnInter}$:

$$\frac{}{OrgUnInter :: Inanimate}\ MR^*_{OrgUnInter}$$

———————— . ————————

———————— . ————————

• Rule $R^*_{OrgUnInter}$:

$$\frac{\begin{array}{c} ou_1 : OrgUnit \\ ou_2 : OrgUnit \\ ep : ExchProc \\ [\langle ou_1, ou_2, ep \rangle ...]_{OrgUnInter} \end{array}}{\langle ou_1, ou_2, ep \rangle : OrgUnInter}\ R^*_{OrgUnInter}$$

• Constraint $R^*_{OrgUnInter}$:

$$\begin{array}{l} \forall \langle ou_1, ou_2, ep \rangle : OrgUnInter \\ \qquad \exists \langle ou_1, beh_1 \rangle, \langle ou_2, beh_2 \rangle : OrgUnBeh \\ \qquad\qquad \forall t : T[ep(t) = \langle beh_1(t), beh_2(t) \rangle] \end{array}$$

———————— . ————————

———————— . ————————

• Rule $DR[OrgRo_1]_{OrgUnInter}$:

$$\frac{\langle ou_1, ou_2, ep \rangle : OrgUnInter}{ou_1 : OrgRo}\ DR[OrgRo_1]_{OrgUnInter}$$

• Rule $DR[OrgRo_2]_{OrgUnInter}$:

$$\frac{\langle ou_1, ou_2, ep \rangle : OrgUnInter}{ou_2 : OrgRo}\ DR[OrgRo_2]_{OrgUnInter}$$

• Rule $DR[ExchProc]_{OrgUnInter}$:

$$\frac{\langle ou_1, ou_2, ep \rangle : OrgUnInter}{ep_2 : ExchProc}\ DR[ExchProc]_{OrgUnInter}$$

———————— . ————————

Organization Unit Relation

Type *OrgUnRel* is an *inanimate external type.*

- Meta-rule $MR_{OrgUnRel}$

$$\frac{}{OrgUnRel :: Inanimate}\ MR_{OrgUnRel}$$

- Rule $ER_{OrgUnRel}$:

$$\frac{\begin{array}{c} ou_1 : OrgUn \\ ou_2 : OrgUn \\ rel : Rel \\ [\langle ou_1, ou_2, rel\rangle ...]_{OrgUnRel} \end{array}}{\langle ou_1, ou_2, rel\rangle : OrgUnRel}\ ER_{OrgUnRel}$$

- Rule $DR[OrgUn_1]_{OrgUnRel}$:

$$\frac{\langle ou_1, ou_2, rel\rangle : OrgUnRel}{ou_1 : OrgUn}\ DR[OrgUn_1]_{OrgUnRel}$$

- Rule $DR[OrgUn_2]_{OrgUnRel}$:

$$\frac{\langle ou_1, ou_2, rel\rangle : OrgUnRel}{ou_2 : OrgUn}\ DR[OrgUn_2]_{OrgUnRel}$$

- Rule $DR[Rel]_{OrgUnRel}$:

$$\frac{\langle ou_1, ou_2, rel\rangle : OrgUnRel}{rel : Rel}\ DR[Rel]_{OrgUnRel}$$

6.6.6 Organization Unit Network

An object of the type *organization unit network* is composed by a set of *organization units*, and some of their *organization unit properties*, *behaviors*, *interactions* and *relations*. In an *organization unit network*, no *organization unit property* or *behavior* may exist without the corresponding existence in the network of the *organization unit* that supports them, and no *organization unit interaction* or *relation* may exist without the corresponding existence in the network of the two *organization units* that realize them. Notice that the decision of which *organization unit behaviors* and which *organization unit interactions* will belong to an *organization unit network* is a decision up to the external observer (modeler or specifier). But not every tuple $\langle OU, OUProp, OUBeh, OUInter, OURel\rangle$ constitutes an *organization unit network*, certain *constraints* have to be satisfied. Type *OrgUnNet* is an *animate internal type.*

———————— . ————————

- Meta-rule $MR^*_{OrgUnNet}$:

$$\frac{}{\mathit{OrgUnNet} :: \mathit{Animate}}\ MR^*_{OrgUnNet}$$

———————— . ————————

———————— . ————————

- Rule $R^*_{OrgUnNet}$:

$$\frac{\begin{array}{c} OU : \wp(\mathit{OrgUn}) \\ \mathit{OUProp} : \wp(\mathit{OrgUnProp}) \\ \mathit{OUBeh} : \wp(\mathit{OrgUnBeh}) \\ \mathit{OUInter} : \wp(\mathit{OrgUnInter}) \\ \mathit{OURel} : \wp(\mathit{OrgUnRel}) \end{array}}{\langle OU, \mathit{OUProp}, \mathit{OUBeh}, \mathit{OUInter}, \mathit{OURel}\rangle : \mathit{OrgUnNet}}\ R^*_{OrgUnNet}$$

- Constraints $R^*_{OrgUnNet}$:

$$\forall OU, \mathit{OUProp}, \mathit{OUBeh}, \mathit{OUInter}, \mathit{OURel}\rangle : \mathit{OrgUnNet}\ \ [OU \neq \emptyset]$$

$$\begin{array}{l} \forall OU, \mathit{OUProp}, \mathit{OUBeh}, \mathit{OUInter}, \mathit{OURel}\rangle : \mathit{OrgUnNet} \\ \qquad [\forall\langle ou, prop\rangle \in \mathit{OUProp}[ou \in OU \wedge \langle ou, prop\rangle : \mathit{OrgUnProp}] \\ \qquad \wedge \forall\langle ou, beh\rangle \in \mathit{OUBeh}[ou \in OU \wedge \langle ou, beh\rangle : \mathit{OrgUnBeh}] \\ \qquad \wedge \forall\langle ou_1, ou_2, ep\rangle \in \mathit{OUInter}[ou_1, ou_2 \in OU \\ \qquad\qquad\qquad \wedge \langle ou_1, ou_2, ep\rangle : \mathit{OrgUnInter}] \\ \qquad \wedge \forall\langle ou_1, ou_2, rel\rangle \in \mathit{OURel}[ou_1, ou_2 \in OU \\ \qquad\qquad\qquad \wedge \langle ou_1, ou_2, rel\rangle : \mathit{OrgUnRel}]] \end{array}$$

———————— . ————————

———————— . ————————

- Rule $DR[\wp(\mathit{OrgUn})]_{OrgUnNet}$:

$$\frac{OU, \mathit{OUProp}, \mathit{OUBeh}, \mathit{OUInter}, \mathit{OURel}\rangle : \mathit{OrgUnNet}}{OU : \wp(\mathit{OrgUn})}\ DR[\wp(\mathit{OrgUn})]_{OrgUnNet}$$

- Rule $DR[\wp(\mathit{OrgUnProp})]_{OrgUnNet}$:

$$\frac{OU, \mathit{OUProp}, \mathit{OUBeh}, \mathit{OUInter}, \mathit{OURel}\rangle : \mathit{OrgUnNet}}{\mathit{OUProp} : \wp(\mathit{OrgUnProp})}\ DR[\wp(\mathit{OrgUnProp})]_{OrgUnNet}$$

- Rule $DR[\wp(\mathit{OrgUnBeh})]_{OrgUnNet}$:

$$\frac{OU, \mathit{OUProp}, \mathit{OUBeh}, \mathit{OUInter}, \mathit{OURel}\rangle : \mathit{OrgUnNet}}{\mathit{OUBeh} : \wp(\mathit{OrgUnBeh})}\ DR[\wp(\mathit{OrgUnBeh})]_{OrgUnNet}$$

- Rule $DR[\wp(\mathit{OrgUnInter})]_{OrgUnNet}$:

$$\frac{OU, \mathit{OUProp}, \mathit{OUBeh}, \mathit{OUInter}, \mathit{OURel}\rangle : \mathit{OrgUnNet}}{\mathit{OUInter} : \wp(\mathit{OrgUnInter})}\ DR[\wp(\mathit{OrgUnInter})]_{OrgUnNet}$$

- Rule $DR[\wp(\mathit{OrgUnRel})]_{OrgUnNet}$:

$$\frac{OU, \mathit{OUProp}, \mathit{OUBeh}, \mathit{OUInter}, \mathit{OURel}\rangle : \mathit{OrgUnNet}}{\mathit{OURel} : \wp(\mathit{OrgUnRel})}\ DR[\wp(\mathit{OrgUnRel})]_{OrgUnNet}$$

———————— . ————————

6.6.7 Organizational Structure

The *organizational structure* of an *agent society*, at a given time, is composed by:

- the set of all *organizational units* existent in that agent society, at that time;
- the set of all *organizational unit properties* of those organizational units;
- the set of all *organizational unit behaviors* of those organizational units;
- the set of all *organizational unit interactions* existent between those organizational units;
- the set of all *organizational unit relations* existent between those organizational units;
- the set of all *organizational unit networks* existent in that agent society, at that time.

Type *Org* is an *animate internal type.*

———————— . ————————

• Meta-rule MR_{Org}:

$$\frac{}{Org :: Animate}\ MR_{Org}$$

———————— . ————————

———————— . ————————

• Rule ER_{Org}:

$$\frac{\begin{array}{c} OU : \wp(OrgUn) \\ OUProp : \wp(OrgUnProp) \\ OUBeh : \wp(OrgUnBeh) \\ OUInter : \wp(OrgUnInter) \\ OURel : \wp(OrgUnRel) \\ OUNet : \wp(OrgUnNet) \end{array}}{\langle OU, OUProp, OUBeh, OUInter, OURel, OUNet\rangle : Org}\ ER_{Org}$$

———————— . ————————

———————— . ————————

• Rule $DR[\wp(OrgUn)]_{Org}$:

$$\frac{\langle ..., OU, OUProp, OUBeh, OULnk, OUNet\rangle : Org}{OU : \wp(OrgUn)}\ DR[\wp(OrgUn)]_{Org}$$

• Rule $DR[\wp(OrgUnProp)]_{Org}$:

$$\frac{\langle ..., OU, OUProp, OUBeh, OULnk, OUNet\rangle : Org}{OUProp : \wp(OrgUnProp)}\ DR[\wp(OrgUnProp)]_{Org}$$

• Rule $DR[\wp(OrgUnBeh)]_{Org}$:

$$\frac{\langle ..., OU, OUProp, OUBeh, OULnk, OUNet\rangle : Org}{OUBeh : \wp(OrgUnBeh)}\ DR[\wp(OrgUnBeh)]_{Org}$$

• Rule $DR[\wp(OrgUnInter)]_{Org}$:

$$\frac{\langle ..., OU, OUProp, OUBeh, OULnk, OUNet\rangle : Org}{OUInter : \wp(OrgUnInter)}\ DR[\wp(OrgUnInter)]_{Org}$$

- Rule $DR[\wp(OrgUnRel)]_{Org}$:

$$\frac{\langle ..., OU, OUProp, OUBeh, OULnk, OUNet\rangle : Org}{OURel : \wp(OrgUnRel)}\ DR[\wp(OrgUnRel)]_{Org}$$

- Rule $DR[\wp(OrgUnNet)]_{Org}$:

$$\frac{\langle ..., OU, OUProp, OUBeh, OULnk, OUNet\rangle : Org}{OUNet : \wp(OrgUnNet)}\ DR[\wp(OrgUnNet)]_{Org}$$

6.7 Type *Material Environment*

The *material environment* of an agent society or inter-societal agent system is composed of *material objects* (and their *properties*, *interactions* and *relations*), which *agents*, *organizations*, etc. of the agent society or inter-societal agent system make use of to support their own *properties*, *behaviors*, *interactions* and *relations*.

6.7.1 Material Object

The type *MatObj* is an *animate external type*.

- Meta-rule MR_{MatObj}:

$$\frac{}{MatObj :: Animate}\ MR_{MatObj}$$

- Rule ER_{MatObj}:

$$\frac{[mo...]_{MatObj}}{mo : MatObj}\ ER_{MatObj}$$

6.7.2 Material Object Property, Behavior, Interaction and Relation

Material Object Property

The type *MatObjProp* is an *inanimate external type*.

- Meta-rule $MR_{MatObjProp}$:

$$\frac{}{MatObjProp :: Inanimate}\ MR_{MatObjProp}$$

• Rule $ER_{MatObjProp}$:

$$\frac{mo : MatObj \qquad prop : Prop \qquad [\langle mo, prop\rangle ...]_{MatObjProp}}{\langle mo, prop\rangle : MatObjProp}\ ER_{MatObjProp}$$

• Rule $DR[MatObj]_{MObjProp}$:

$$\frac{\langle mo, prop\rangle : MatObjProp}{mo : MatObj}\ DR[MatObj]_{MatObjProp}$$

• Rule $DR[Prop]_{MObjProp}$:

$$\frac{\langle mo, prop\rangle : MatObjProp}{prop : Prop}\ DR[Prop]_{MatObjProp}$$

Material Object Behavior

The type *MatObjBeh* is an *inanimate external type.*

• Meta-rule $MR_{MatObjBeh}$:

$$\frac{}{MatObjBeh :: Inanimate}\ MR_{MatObjBeh}$$

• Rule $ER_{MatObjBeh}$:

$$\frac{mo : MatObj \qquad proc : Proc \qquad [\langle mo, proc\rangle ...]_{MatObjBeh}}{\langle mo, proc\rangle : MatObjBeh}\ R_{MatObjBeh}$$

• Rule $DR[MatObj]_{MatObjBeh}$:

$$\frac{\langle mo, proc\rangle : MatObjBeh}{mo : MatObj}\ DR[MatObj]_{MatObjBeh}$$

• Rule $DR[Proc]_{MatObjBeh}$:

$$\frac{\langle mo, proc\rangle : MatObjBeh}{proc : Proc}\ DR[Proc]_{MatObjBeh}$$

Material Object Interaction

The type *MatObjInter* is an *inanimate external type.*

———————————— . ————————————

• Meta-rule $MR_{MatObjInter}$:

$$\frac{}{MatObjInter :: Inanimate}\ MR_{MatObjInter}$$

———————————— . ————————————

———————————— . ————————————

• Rule $ER_{MatObjInter}$:

$$\frac{\begin{array}{c} mo_1 : MatObj \\ mo_2 : MatObj \\ ep : ExchProc \\ [\langle mo_1, mo_2, ep\rangle ...]_{MatObjInter} \end{array}}{\langle mo_1, mo_2, ep\rangle : MatObjInter}\ R_{MatObjInter}$$

———————————— . ————————————

———————————— . ————————————

• Rule $DR[MatObj_1]_{MatObjInter}$:

$$\frac{\langle mo_1, mo_2, ep\rangle : MatObjInter}{mo_1 : MatObj}\ DR[MatObj_1]_{MatObjInter}$$

• Rule $DR[MatObj_2]_{MatObjInter}$:

$$\frac{\langle mo_1, mo_2, ep\rangle : MatObjInter}{mo_2 : MatObj}\ DR[MatObj_2]_{MatObjInter}$$

• Rule $DR[ExchProc]_{MatObjInter}$:

$$\frac{\langle mo_1, mo_2, ep\rangle : MatObjInter}{ep : ExchProc}\ DR[ExchProc]_{MatObjInter}$$

———————————— . ————————————

Material Object Relation

The type *MatObjRel* is an *inanimate external type.*

———————————— . ————————————

• Meta-rule $MR_{MatObjRel}$:

$$\frac{}{MatObjRel :: Inanimate}\ MR_{MatObjRel}$$

———————————— . ————————————

- Rule $ER_{MatObjRel}$:

$$\frac{\begin{array}{c} mo_1 : MatObj \\ mo_2 : MatObj \\ rel : Rel \\ [\langle mo_1, mo_2, rel\rangle ...]_{MatObjRel} \end{array}}{\langle mo_1, mo_2, rel\rangle : MatObjRel}\ ER_{MatObjRel}$$

- Rule $DR[MatObj_1]_{MatObjRel}$:

$$\frac{\langle mo_1, mo_2, rel\rangle : MatObjRel}{mo_1 : MatObj}\ DR[MatObj_1]_{MatObjRel}$$

- Rule $DR[MatObj_2]_{MatObjRel}$:

$$\frac{\langle mo_1, mo_2, rel\rangle : MatObjRel}{mo_2 : MatObj}\ DR[MatObj_2]_{MatObjRel}$$

- Rule $DR[Rel]_{MatObjRel}$:

$$\frac{\langle mo_1, mo_2, rel\rangle : MatObjRel}{rel_2 : Rel}\ DR[Rel]_{MatObjRel}$$

6.7.3 Material Object Network

A *material object network* encompasses a non-empty set of material objects, and some of their behaviors, properties, interactions and relations. The type *MatObjNet* is an *inanimate type.*

- Meta-rule $MR^*_{MatObjNet}$:

$$\frac{}{MatObjNet :: Inanimate}\ MR^*_{MatObjNet}$$

- Rule $R^*_{MatObjNet}$:

$$\frac{\begin{array}{c} MO : \wp(MatObj) \\ MOProp : \wp(MatObjProp) \\ MOBeh : \wp(MatObjBeh) \\ MOInter : \wp(MatObjInter) \\ MORel : \wp(MatObjRel) \end{array}}{\langle MO, MOProp, MOBeh, MOInter, MORel\rangle : MatObjNet}\ R^*_{MatObjNet}$$

- Constraints $R^*_{MatObjNet}$:

$$\forall \langle MO, MOProp, MOBeh, MOInter, MORel\rangle : MatObjNet\ \ [MO \neq \emptyset]$$

$$\begin{array}{l}\forall\langle MO, MOProp, MOBeh, MOInter, MORel\rangle : MatObjNet \\ \qquad [\forall\langle mo, prop\rangle \in MOProp : mo \in MO \\ \qquad \wedge \forall\langle mo, beh\rangle \in MOBeh : mo \in MO \\ \qquad \wedge \forall\langle mo_1, mo_2, ep\rangle \in MOInter : mo_1, mo_2 \in MO \\ \qquad \wedge \forall\langle mo_1, mo_2, rel\rangle \in MORel : mo_1, mo_2 \in MO]\end{array}$$

———————— . ————————

———————— . ————————

- Rule $DR[\wp(MatObj)]_{MatObjNet}$:

$$\frac{\langle MO, MOProp, MOBeh, MOInter, MORel\rangle : MatObjNet}{MO : \wp(MatObj)}\ DR[\wp(MatObj)]_{MatObjNet}$$

- Rule $DR[\wp(MatObjProp)]_{MatObjNet}$:

$$\frac{\langle MO, MOProp, MOBeh, MOInter, MORel\rangle : MatObjNet}{MOBeh : \wp(MatObjProp)}\ DR[\wp(MatObjProp)]_{MatObjNet}$$

- Rule $DR[\wp(MatObjBeh)]_{MatObjNet}$:

$$\frac{\langle MO, MOProp, MOBeh, MOInter, MORel\rangle : MatObjNet}{MOBeh : \wp(MatObjBeh)}\ DR[\wp(MatObjBeh)]_{MatObjNet}$$

- Rule $DR[\wp(MatObjInter)]_{MatObjNet}$:

$$\frac{\langle MO, MOProp, MOBeh, MOInter, MORel\rangle : MatObjNet}{MOInter : \wp(MatObjInter)}\ DR[\wp(MatObjInter)]_{MatObjNet}$$

- Rule $DR[\wp(MatObjRel)]_{MatObjNet}$:

$$\frac{\langle MO, MOProp, MOBeh, MOInter, MORel\rangle : MatObjNet}{MORel : \wp(MatObjRel)}\ DR[\wp(MatObjRel)]_{MatObjNet}$$

———————— . ————————

6.7.4 Material Environment

The *material environment* of an *agent society*, at a given time, is composed by:

- the set of all *material objects* existent in that society, at that time;
- the set of all *material object properties* of those material objects;
- the set of all *material object behaviors* of those material objects;
- the set of all *material object interactions* existent between those material objects;
- the set of all *material object relations* existent between those material objects;
- the set of all *material object networks* existent in that society, at that time.

The type *MEnv* is an *animate internal type.*

———————— . ————————

- Meta-rule MR^*_{MEnv}

$$\frac{}{MEnv :: Animate}\ MR^*_{MEnv}$$

———————— . ————————

- Rule R^*_{MEnv}:

$$\frac{\begin{array}{c} MO : \wp(MatObj) \\ MOProp : \wp(MatObjProp) \\ MOBeh : \wp(MatObjBeh) \\ MOInter : \wp(MatObjInter) \\ MORel : \wp(MatObjRel) \\ MONet : \wp(MatObjNet) \end{array}}{\langle MO, MOProp, MOBeh, MOInter, MORel, MONet\rangle : MEnv}\, R_{MEnv}$$

- Constraint R^*_{MEnv}:

$$\begin{array}{l} \forall\langle MO, MOProp, MORel, MONet\rangle : MEnv \\ \qquad [\forall\langle mo, prop\rangle \in MOProp[mo \in MO] \\ \qquad \wedge\, \forall\langle mo_1, mo_2, rel\rangle \in MORel[mo_1, mo_2 \in MO] \\ \qquad \wedge\, \forall\langle MO', MOProp, MORel\rangle \in MONet[MO' \subseteq MO]] \end{array}$$

- Rule $DR[\wp(MatObj)]_{MEnv}$:

$$\frac{\langle MO, MOProp, MOBeh, MOInter, MORel, MONet\rangle : MEnv}{MO : \wp(MatObj)}\, DR[\wp(MatObj)]_{MEnv}$$

- Rule $DR[\wp(MatObjProp)]_{MEnv}$:

$$\frac{\langle MO, MOProp, MOBeh, MOInter, MORel, MONet\rangle : MEnv}{MOProp : \wp(MatObjProp)}\, DR[\wp(MatObjProp)]_{MEnv}$$

- Rule $DR[\wp(MatObjBeh)]_{MEnv}$:

$$\frac{\langle MO, MOProp, MOBeh, MOInter, MORel, MONet\rangle : MEnv}{MOBeh : \wp(MatObjBeh)}\, DR[\wp(MatObjBeh)]_{MEnv}$$

- Rule $DR[\wp(MatObjInter)]_{MEnv}$:

$$\frac{\langle MO, MOProp, MOBeh, MOInter, MORel, MONet\rangle : MEnv}{MOInter : \wp(MatObjInter)}\, DR[\wp(MatObjInter)]_{MEnv}$$

- Rule $DR[\wp(MatObjRel)]_{MEnv}$:

$$\frac{\langle MO, MOProp, MOBeh, MOInter, MORel, MONet\rangle : MEnv}{MORel : \wp(MatObjRel)}\, DR[\wp(MatObjRel)]_{MEnv}$$

- Rule $DR[\wp(MatObjNet)]_{MEnv}$:

$$\frac{\langle MO, MOProp, MOBeh, MOInter, MORel, MONet\rangle : MEnv}{MONet : \wp(MatObjNet)}\, DR[\wp(MatObjNet)]_{MEnv}$$

6.8 Type *Symbolic Environment*

The *symbolic environment* of an agent society or inter-societal agent system is the set of *symbolic objects* (and their *properties* and *relations*), which the *agents*, *organizations*, etc. of the society or inter-societal agent system make use of to support their own *properties*, *behaviors*, *interactions* and *relations*.

6.8.1 Symbolic Object

Symbolic objects may be *basic symbolic objects* or a *structured symbolic object.* Structured symbolic objects are *networks* of symbolic objects (defined in Sect. 6.8.3). The type *SymbObj* is an *inanimate external type.*

- Meta-rule $MR_{SymbObj}$:

$$\frac{}{SymbObj :: Inanimate}\ MR_{SymbObj}$$

- Rule $ER_{SymbObj_1}$:

$$\frac{[so...]_{SymbObj}}{so : SymbObj}\ ER_{SymbObj_1}$$

- Rule $ER_{SymbObj_2}$:

$$\frac{son : SymbObjNet}{\langle son\rangle : SymbObj}\ ER_{SymbObj_2}$$

6.8.2 Symbolic Object Property, Behavior, Interaction and Relation

Symbolic Object Property

Symbolic object property is an *inanimate external type.*

- Meta-rule $MR_{SymbObjProp}$:

$$\frac{}{SymbObjProp :: Inanimate}\ MR_{SymbObjProp}$$

- Rule $ER_{SymbObjProp}$:

$$\frac{so : SObj \quad prop : Prop \quad [\langle so : prop\rangle...]_{SymbObjProp}}{\langle so, prop\rangle : SymbObjProp}\ ER_{SymbObjProp}$$

- Rule $DR[SymbObj]_{SymbObjProp}$:

$$\frac{\langle so, prop\rangle : SymbObjProp}{so : SymbObj}\ DR[SymbObj]_{SymbObjProp}$$

- Rule $DR[Prop]_{SymbObjProp}$:

$$\frac{\langle so, prop\rangle : SymbObjProp}{prop : Prop}\ DR[Prop]_{SymbObjProp}$$

Symbolic Object Relation

Symbolic object relation is an *inanimate external type*.

* Meta-rule $MR_{SymbObjRel}$:

$$\frac{}{SymbObjRel :: Inanimate}\ MR_{SymbObjRel}$$

* Rule $ER_{SObjRel}$:

$$\frac{\begin{array}{c} so_1 : SymbObj \\ so_2 : SymbObj \\ rel : Rel \\ [\langle so_1, so_2, rel\rangle ...]_{SymbObjRel} \end{array}}{\langle so_1, so_2, rel\rangle : SymbObjRel}\ ER_{SymbObjRel}$$

* Rule $DR[SymbObj_1]_{SymbObjRel}$:

$$\frac{\langle so_1, so_2, rel\rangle : SymbObjRel}{so_1 : SymbObj}\ DR[SymbObj_1]_{SymbObjRel}$$

* Rule $DR[SymbObj_2]_{SymbObjRel}$:

$$\frac{\langle so_1, so_2, rel\rangle : SymbObjRel}{so_2 : SymbObj}\ DR[SymbObj_2]_{SymbObjRel}$$

* Rule $DR[Rel]_{SymbObjRel}$:

$$\frac{\langle so_1, so_2, rel\rangle : SymbObjRel}{rel : Rel}\ DR[Rel]_{SymbObjRel}$$

6.8.3 Symbolic Object Network

A *symbolic object network* encompasses a non-empty set of symbolic objects, and some of their properties and relations. The type *SymbObjNet* is an *inanimate internal type*.

* Meta-rule $MR^*_{SymbObjNet}$:

$$\frac{}{SymbObjNet :: Inanimate}\ MR^*_{SymbObjNet}$$

- Rule $R^*_{SymbObjNet}$:

$$\frac{\begin{array}{c} SO : \wp(SymbObj) \\ SOProp : \wp(SymbObjProp) \\ SORel : \wp(SymbObjRel) \end{array}}{\langle SO, SOProp, SORel \rangle : SymbObjNet} \; R^*_{SymbObjNet}$$

- Constraints $R^*_{SymbObjNet}$:

$\forall \langle SO, SOProp, SORel \rangle : SymbObjNet \;\; [SO \neq \emptyset]$

$$\begin{array}{l} \forall \langle SO, SOProp, SORel \rangle : SymbObjNet \\ \qquad [\forall \langle so, prop \rangle \in SOProp : so \in SO \\ \qquad \wedge \forall \langle so_1, so_2, rel \rangle \in SORel : so_1, so_2 \in MO] \end{array}$$

- Rule $DR[\wp(SymbObj)]_{SymbObjNet}$:

$$\frac{\langle SO, SOProp, SORel \rangle : SymbObjNet}{SO : \wp(SymbObj)} \; DR[\wp(SymbObj)]_{SymbObjNet}$$

- Rule $DR[\wp(SymbObjProp)]_{SymbObjNet}$:

$$\frac{\langle SO, SOProp, SORel \rangle : SymbObjNet}{SOBeh : \wp(SymbObjProp)} \; DR[\wp(SymbObjProp)]_{SymbObjNet}$$

- Rule $DR[\wp(SymbObjRel)]_{SymbObjNet}$:

$$\frac{\langle SO, SOProp, SORel \rangle : SymbObjNet}{SOLnk : \wp(SymbObjLnk)} \; DR[\wp(SymbObjLnk)]_{SymbObjNet}$$

6.8.4 Symbolic Environment

At each timehe *simbolic environment* of an agent society is the set of *symbolic objects* (and their *properties* and *relations*), and all *symbolic object networks* existent in that agent society, at that time. The type *SEnv* is an *inanimate external type.*

- Meta-rule MR^*_{SEnv}:

$$\frac{}{SEnv :: Inanimate} \; MR^*_{SEnv}$$

- Rule R^*_{SEnv}:

$$\frac{\begin{array}{c} SO : \wp(SymbObj) \\ SOProp : \wp(SymbObjProp) \\ SORel : \wp(SymbObjRel) \\ SONet : \wp(SymbObjNet) \end{array}}{\langle SO, SOProp, SORel, SONet\rangle : SEnv}\ R^*_{SEnv}$$

- Constraints R^*_{SEnv}:

$$\begin{array}{l} \forall\langle SO, SOProp, SORel, SONet\rangle : SEnv \\ \qquad [\forall\langle so, prop\rangle \in SOProp[so \in SO] \\ \qquad \wedge \forall\langle so_1, so_2, rel\rangle \in SORel[so_1, so_2 \in SO] \\ \qquad \wedge \forall\langle SO', SOProp, SORel\rangle \in SONet[SO' \subseteq SO]] \end{array}$$

- Rule $DR[\wp(SymbObj)]_{SEnv}$:

$$\frac{\langle SO, SOProp, SORel, SONet\rangle : SEnv}{SO : \wp(SymbObj)}\ DR[\wp(SymbObj)]_{SEnv}$$

- Rule $DR[\wp(SymbObjProp)]_{SEnv}$:

$$\frac{\langle SO, SOProp, SORel, SONet\rangle : SEnv}{SOProp : \wp(SymbObjProp)}\ DR[\wp(SymbObjProp)]_{SEnv}$$

- Rule $DR[\wp(SymbObjRel)]_{SEnv}$:

$$\frac{\langle SO, SOProp, SORel, SONet\rangle : SEnv}{SORel : \wp(SymbObjRel)}\ DR[\wp(SymbObjRel)]_{SEnv}$$

- Rule $DR[\wp(SymbObjNet)]_{SEnv}$:

$$\frac{\langle SO, SOProp, SORel, SONet\rangle : SEnv}{SONet : \wp(SymbObjNet)}\ DR[\wp(SymbObjNet)]_{SEnv}$$

6.8.5 Norm Expressions

Norm expressions are a subtype of the type *SymbObj*. The type *NrmExpr* is an *inanimate external type*.

$$\frac{}{NrmExp :: SymbObj}\ MR_{NrmExp}$$

$$\frac{[nrmexpr...]}{nrmexpr : NrmExpr}\ R_{NrmExp}$$

6.9 Type *Agent Society*

An *agent society* is a system composed of an object of type *populational structure* (*Pop*), an object of type *sociability structure* (*Soc*), an object of type *organizational structure* (*Org*), with its organizational levels (micro Org_ω, meso Org_μ and macro Org_Ω), an object of type *material environment* (*MEnv*) and an object of type *symbolic environment* (*SEnv*).

Such *architectural components* are connected to each other through two *implementation relations*, each establishing connections between two different architectural levels:

- between the *populational structure* and the *sociability structure*, allowing the *agents* to implement *sociability roles*, by the agents' behaviors, interactions and relations implementing sociability roles' behaviors, interactions and relations;
- between the *sociability structure* and the *organizational structure*, allowing *sociability roles* to implement *organization units*, by the sociability roles' behaviors, interactions and relations implementing the behaviors, interactions and relations of the organizational roles of the organization units[2].

Also, a set of *access links* allow animate elements of one architectural component to access animate or inanimate elements of other architectural components.

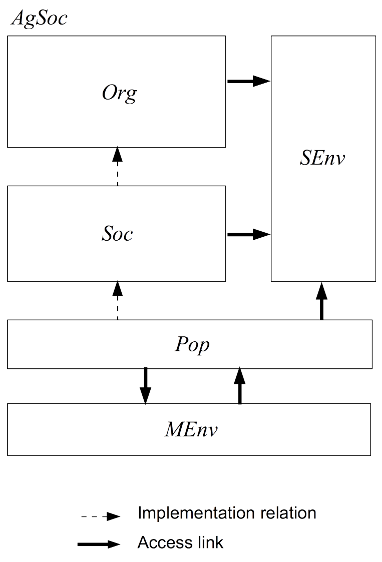

Figure 6.1: The implementation relations and access links between the components of an agent society.

Figure 6.1 illustrates the construction of objects of the type *AgSoc*. The solid arrows show the *access links* between the components. But, the figure does not show explicitly neither the access links that each of the three organizational levels (Org_ω, Org_μ and Org_Ω) may have to the material and symbolic environments (*MEnv* and *SEnv*) nor the access links through which the material environment can access the symbolic environment. The dashed arrows show the *implementation relations.*

6.9.1 The Implementation Relations

Implementation of *Soc* by *Pop*

The *sociability roles* of the sociability structure *Soc* may be implemented by the *agents* of the populational structure *Pop*. A compatibility constraint has to be satisfied, however, to the effect

[2]Clearly, the implementation of *organizational roles* by *sociability roles* implies the unpersonification of the former. Sect. 6.9.1 defines a *direct* implementation relation of organizational roles by agents, which personifies the organizational roles.

that for each sociability role behavior implemented by an agent, the sociability role behavior is at most a part of one of that agent's behaviors, and that for each exchange process executed by a pair of sociability roles, the exchange process is at most a part of an exchange process executed by the pair of agents that implement those sociability roles[3]. Notice that, here, we do not impose constraints on the relation between the *sociability role relations* and the *agent relations*. The type $Imp_{Soc/Pop}$ is an *inanimate internal type*.

———————— . ————————

- Meta-rule $MR^*_{Imp_{Soc/Pop}}$:

$$\frac{}{Imp_{Soc/Pop} :: Inanimate}\ MR^*_{Imp_{Soc/Pop}}$$

———————— . ————————

———————— . ————————

- Rule $R^*_{Imp_{Soc/Pop}}$:

$$\frac{\begin{array}{c}\langle AG, AGProp, AGBeh, AGInter, AGRel\rangle : Pop \\ \langle SR, SRProp, SRBeh, SRInter, SRRel, SRN\rangle : Soc \\ \langle ag, sr\rangle : AG \times SR\end{array}}{\langle ag, sr\rangle : Imp_{Soc/Pop}}\ R_{Imp_{Soc/Pop}}$$

- Constraint $Imp_{Soc/Pop}$:

$$\begin{array}{l}\forall \langle ag, sr\rangle : Imp_{Soc/Pop} \Rightarrow \\ \quad [\forall \langle sr, srproc\rangle \in SRBeh\ \exists \langle ag, agproc\rangle \in Beh \forall t : T[srproc(t) \subseteq agproc(t)] \\ \quad \wedge \forall \langle sr, sr', srep\rangle \in SRInter\ \exists \langle ag, ag', agep\rangle \in AGInter \\ \qquad\qquad [\langle ag', sr'\rangle : Imp_{Soc/Pop} \wedge \forall t : T[srep(t) \subseteq agep(t)]]]\end{array}$$

where we take that $(X, Y) \subseteq (Z, W)$ means $(X \subseteq Z) \wedge (Y \subseteq W)$.

———————— . ————————

———————— . ————————

- Rule $DR[Agent]_{Imp_{Soc/Pop}}$:

$$\frac{\langle ag, sr\rangle : Imp_{Soc/Pop}}{ag : Agent}\ DR[Agent]_{Imp_{Soc/Pop}}$$

ponto Rule $DR[SocRo]_{Imp_{Soc/Pop}}$:

$$\frac{\langle ag, sr\rangle : Imp_{Soc/Pop}}{sr : SocRo}\ DR[SocRo]_{Imp_{Soc/Pop}}$$

———————— . ————————

[3]In [6], we presented a more general notion of *implementation relation*, establishing also implementation relations between the *actions* performed at the various organizational levels, thus requiring that the behaviors and exchange processes of a given implemented level be a homomorphic image (through such action implementation relations) of the behaviors and exchange processes of the level that implements it. This remark applies also to the other types of implementation relations, defined below.

Implementation of *Org* by *Soc*

The *organizational roles* of the organizational structure *Org* may be implemented by the *sociability roles* of the sociability structure *Soc*. A compatibility constraint has to be satisfied, however, to the effect that for each organizational role implemented by a sociability role, the organizational role behavior is at most a part of one of that sociability role's behaviors, and that for each exchange process excuted by a pair of organizational roles, the exchange process is a at most a part of an exchange process executed by the pair of sociability roles that implement those organizational roles (But, see footnote 3). Notice that, here, we do not impose constraints on the relation between the *organizational role relations* and the *sociability role relations*. The type $Imp_{Org/Soc}$ is an *inanimate internal type*.

——————— . ———————

- Meta-rule $MR^*_{Imp_{Org/Soc}}$:

$$\frac{}{Imp_{Org/Soc} :: Inanimate}\; MR^*_{Imp_{Org/Soc}}$$

——————— . ———————

——————— . ———————

- Rule $R^*_{Imp_{Org/Soc}}$:

$$\frac{\begin{array}{c}\langle SR, SRBeh, SRInter, SRRel, SRNet\rangle : Soc\\ \langle OR, ORProp, ORBeh, ORInter, ORRel, ORNet, OU, OUProp, OUBeh, OUInter, OURel, OUNet\rangle : Org\\ \langle sr, or\rangle : SR \times OR\end{array}}{\langle sr, or\rangle : Imp_{Org/Soc}}\; R^*_{Imp_{Org/Soc}}$$

- Constraints $R^*_{Imp_{Org/Soc}}$:

$$\begin{array}{l}\forall\langle sr, or\rangle : Imp_{Org/Soc}\\ \quad[\forall\langle or, orproc\rangle \in ORBeh \exists\langle sr, srproc\rangle \in SRBeh[\forall t : T[orproc(t) \subseteq srproc(t)]\\ \quad\wedge \forall\langle or_1, or_2, orep\rangle \in ORInter \exists\langle sr_1, sr_2, srep\rangle \in SRInter\\ \qquad\qquad\qquad\qquad [\forall t : T[orep(t) \subseteq srep(t)]]]\end{array}$$

——————— . ———————

——————— . ———————

- Rule $DR[SocRo]_{Imp_{Org/Soc}}$:

$$\frac{\langle sr, or\rangle : Imp_{Org/Soc}}{sr : SocRo}\; DR[SocRo]_{Imp_{Org/Soc}}$$

- Rule $DR[OrgRo]_{Imp_{Org/Soc}}$:

$$\frac{\langle sr, or\rangle : Imp_{Org/Soc}}{or : OrgRo}\; DR[OrgRo]_{Imp_{Org/Soc}}$$

——————— . ———————

Direct Implementation of *Org* by *Pop*

The direct implementation of *Org* by *Pop* is the implementation that bypasses the intermediate structure *Soc*. It may be used whenever the modeling of the *self-presentation structure* of the agents of *Pop*, given by *Soc*, is not necessary.

———————— . ————————

- Meta-rule $MR^*_{Imp_{Org/Pop}}$:

$$\frac{}{Imp_{Org/Pop} :: Inanimate}\ MR^*_{Imp_{Org/Pop}}$$

———————— . ————————

———————— . ————————

- Rule $R^*_{Imp_{Org/Pop}}$:

$$\frac{\begin{array}{c}\langle AG, AGProp, AGBeh, AGInter, AGRel\rangle : Pop\\ \langle OR, ORProp, ORBeh, ORInter, ORRel, ORNet, OU, OUProp, OUBeh, OUInter, OURel, OUNet\rangle : Org\\ \langle ag, or\rangle : AG \times OR\end{array}}{\langle ag, or\rangle : Imp_{Org/Pop}}\ R^*_{Imp_{Org/Pop}}$$

- Constraints $R^*_{Imp_{Org/Pop}}$:

$$\begin{array}{l}\forall\langle ag, or\rangle : Imp_{Org/Pop}\\ \quad[\forall\langle or, orproc\rangle \in ORBeh\exists\langle ag, agproc\rangle \in AGBeh[\forall t : T[orproc(t) \subseteq agproc(t)]]\\ \quad\wedge \forall\langle or_1, or_2, orep\rangle \in ORInter\exists\langle ag_1, ag_2, agep\rangle \in AGInter\\ \qquad\qquad\qquad\qquad[\forall t : T[orep(t) \subseteq agep(t)]]]\end{array}$$

———————— . ————————

———————— . ————————

- Rule $DR[Agent]_{Imp_{Org/Pop}}$:

$$\frac{\langle ag, or\rangle : Imp_{Org/Pop}}{ag : Agent}\ DR[Agent]_{Imp_{Org/Pop}}$$

- Rule $DR[OrgRo]_{Imp_{Org/Pop}}$:

$$\frac{\langle ag, or\rangle : Imp_{Org/Pop}}{or : OrgRo}\ DR[OrgRo]_{Imp_{Org/Pop}}$$

———————— . ————————

6.9.2 The Access Links to *MEnv* and *SEnv*

The following are the main types of *access links* to the material and symbolic environment, namely, those serving the agents and the organizational units of an agent society:

- access link between *Agent* and *MatObj*;
- access link between *Agent* and *SymbObj*;
- access link between *OrgUn* and *MatObj*;
- access link between *OrgUn* and *SymbObj*.

Access link between *Agent* **and** *MatObj*

The *access link* between an agent and a material object is based on an exchange process. The type $AccLnk_{Agent/MatObj}$ is an *inanimate external type.*

- Rule $MR_{AccLnk_{Agent/MatObj}}$:

$$\frac{}{AccLnk_{Agent/MatObj} :: Inanimate}\ MR_{AccLnk_{Agent/MatObj}}$$

- Rule $ER_{AccLnk_{Agent/MatObj}}$:

$$\frac{\begin{array}{c} ag : Agent \\ mo : MatObj \\ ep : ExchProc \\ [\langle ag, mo, ep\rangle ...]_{Acc_{Agent/MatObj}} \end{array}}{\langle ag, mo, ep\rangle : AccLnk_{Agent/MatObj}}\ ER_{Acc_{Agent/MatObj}}$$

- Rule $DR[Agent]_{AccLnk_{Agent/MatObj}}$:

$$\frac{\langle ag, mo, ep\rangle : AccLnk_{Agent/MatObj}}{ag : Agent}\ DR[Agent]_{Acc_{Agent/MatObj}}$$

- Rule $DR[MatObj]_{AccLnk_{Agent/MatObj}}$:

$$\frac{[\langle ag, mo, ep\rangle ...]_{AccLnk_{Agent/MatObj}}}{mo : MatObj}\ DR[MatObj]_{Acc_{Agent/MatObj}}$$

- Rule $DR[ExchProc]_{AccLnk_{Agent/MatObj}}$:

$$\frac{\langle ag, mo, ep\rangle : AccLnk_{Agent/MatObj}}{ep : ExchProc}\ DR[ExchProc]_{Acc_{Agent/MatObj}}$$

Access link between *Agent* **and** *SymbObj*

The *access link* between an agent and a symbolic object is based on an exchange process. The type $AccLnk_{Agent/SymbObj}$ is an *inanimate external type.*

- Rule $MR_{Acc_{Agent/SymbObj}}$:

$$\frac{}{Acc_{Agent/SymbObj} :: Inanimate}\ MR_{Acc_{Agent/SymbObj}}$$

- Rule $ER_{AccLnk_{Agent/SymbObj}}$:

$$\frac{\begin{array}{c} ag : Agent \\ so : SymbObj \\ ep : ExchProc \\ [\langle ag, so, ep\rangle ...]_{AccLnk_{Agent/SymbObj}} \end{array}}{\langle ag, so, ep\rangle : AccLnk_{Agent/SymbObj}}\ ER_{Acc_{Agent/SymbObj}}$$

- Rule $DR[Agent]_{AccLnk_{Agent/SymbObj}}$:

$$\frac{\langle ag, so, ep\rangle : AccLnk_{Agent/SymbObj}}{ag : Agent}\ DR[Agent]_{Acc_{Agent/SymbObj}}$$

- Rule $DR[SymbObj]_{AccLnk_{Agent/SymbObj}}$:

$$\frac{[\langle ag, so, ep\rangle ...]_{AccLnk_{Agent/SymbObj}}}{so : SymbObj}\ DR[SymbObj]_{Acc_{Agent/SymbObj}}$$

- Rule $DR[ExchProc]_{AccLnk_{Agent/SymbObj}}$:

$$\frac{\langle ag, so, ep\rangle : AccLnk_{Agent/SymbObj}}{ep : ExchProc}\ DR[ExchProc]_{Acc_{Agent/SymbObj}}$$

Access link between *OrgUn* and *MatObj*

The *access link* between an organization unit and a material object is based on an exchange process. The type $AccLnk_{OrgUn/MatObj}$ is an *inanimate external type.*

- Rule $MR_{AccLnk_{OrgUn/MatObj}}$:

$$\frac{}{AccLnk_{OrgUn/MatObj} :: Inanimate}\ MR_{AccLnk_{OrgUn/MatObj}}$$

- Rule $ER_{AccLnk_{OrgUn/MatObj}}$:

$$\frac{\begin{array}{c} ou : OrgUn \\ mo : MatObj \\ ep : ExchProc \\ [\langle ou, mo, ep\rangle ...]_{Acc_{OrgUn/MatObj}} \end{array}}{\langle ou, mo, ep\rangle : AccLnk_{OrgUn/MatObj}}\ ER_{Acc_{OrgUn/MatObj}}$$

- Rule $DR[OrgUn]_{AccLnk_{OrgUn/MatObj}}$:

$$\frac{\langle ou, mo, ep\rangle : AccLnk_{OrgUn/MatObj}}{ou : OrgUn}\ DR[OrgUn]_{Acc_{OrgUn/MatObj}}$$

- Rule $DR[MatObj]_{AccLnk_{OrgUn/MatObj}}$:

$$\frac{[\langle ou, mo, ep\rangle ...]_{AccLnk_{OrgUn/MatObj}}}{mo : MatObj}\ DR[MatObj]_{Acc_{OrgUn/MatObj}}$$

- Rule $DR[ExchProc]_{AccLnk_{OrgUn/MatObj}}$:

$$\frac{\langle ou, mo, ep\rangle : AccLnk_{OrgUn/MatObj}}{ep : ExchProc}\ DR[ExchProc]_{Acc_{OrgUn/MatObj}}$$

Access link between *OrgUn* and *SymbObj*

The *access link* between an organization unit and a symbolic object is based on an exchange process. The type $AccLnk_{Agent/MatObj}$ is an *inanimate external type.*

- Rule $MR_{Acc_{OrgUn/SymbObj}}$:

$$\frac{}{Acc_{OrgUn/SymbObj} :: Inanimate}\ MR_{Acc_{OrgUn/SymbObj}}$$

- Rule $ER_{AccLnk_{OrgUn/SymbObj}}$:

$$\frac{\begin{array}{c} ou : OrgUn \\ so : SymbObj \\ ep : ExchProc \\ [\langle ou, so, ep\rangle ...]_{AccLnk_{OrgUn/SymbObj}} \end{array}}{\langle ou, so, ep\rangle : AccLnk_{OrgUn/SymbObj}}\ ER_{Acc_{OrgUn/SymbObj}}$$

- Rule $DR[OrgUn]_{AccLnk_{OrgUn/SymbObj}}$:

$$\frac{\langle ou, so, ep\rangle : AccLnk_{OrgUn/SymbObj}}{ou : OrgUn}\ DR[Agent]_{Acc_{OrgUn/SymbObj}}$$

- Rule $DR[SymbObj]_{AccLnk_{OrgUn/SymbObj}}$:

$$\frac{[\langle ou, so, ep\rangle ...]_{AccLnk_{OrgUn/SymbObj}}}{so : SymbObj}\ DR[SymbObj]_{Acc_{OrgUn/SymbObj}}$$

- Rule $DR[ExchProc]_{AccLnk_{OrgUn/SymbObj}}$:

$$\frac{\langle ou, so, ep\rangle : AccLnk_{OrgUn/SymbObj}}{ep : ExchProc}\ DR[ExchProc]_{Acc_{OrgUn/SymbObj}}$$

6.9.3 Agent Society

A *agent society* $AgSoc$ is a structure with components Pop, Soc, Org, $MEnv$ and $SEnv$, and the implementation relations and access links between them. The type $AgSoc$ is an *animate internal type*.

———————————— . ————————————

• Meta-rule MR^*_{AgSoc}:

$$\frac{}{AgSoc :: Animate}\ MR^*_{AgSoc}$$

———————————— . ————————————

———————————— . ————————————

• Rule R^*_{AgSoc}:

$$\frac{\begin{array}{c} P : Pop \\ S : Soc \\ O : Org \\ ME : MEnv \\ SE : SEnv \\ Imp : IMP \\ Acc : ACC \end{array}}{\langle P, S, O, ME, SE, Imp, Acc\rangle : AgSoc}\ R^*_{AgSoc}$$

where:

$$IMP = \wp(Imp_{Soc/Pop} \cup Imp_{Org/Soc} \cup Imp_{Org/Pop})$$

$$ACC = \wp(Acc_{Agent/MatObj} \cup Acc_{Agent/SymbObj} \cup Acc_{OrgUn/MatObj} \cup Acc_{OrgUn/SymbObj})$$

• Constraints R^*_{AgSoc}:

$$\forall sr \in S\cdot SocRo\ \exists ag \in P\cdot\wp(Agent)\ [\langle ag, sr\rangle \in Imp]$$

$$\begin{array}{l}\forall or \in O\cdot OrgRo \\ \quad [(\exists ag \in P\cdot\wp(Agent)\ [\langle ag, or\rangle \in Imp]) \vee (\exists sr \in S\cdot\wp(SocRo)\ [\langle sr, or\rangle \in Imp])]\end{array}$$

———————————— . ————————————

where we are writing $C\cdot e$ to denote the element e of the architectural component C.

———————————— . ————————————

———————————— . ————————————

• Rule $DR[Pop]_{AgSoc}$:

$$\frac{\langle P, S, O, ME, SE, Imp, Acc\rangle : AgSoc}{P : Pop}\ DR[Pop]_{AgSoc}$$

• Rule $DR[Soc]_{AgSoc}$:

$$\frac{\langle P, S, O, ME, SE, Imp, Acc\rangle : AgSoc}{S : Soc}\ DR[Soc]_{AgSoc}$$

• Rule $DR[Org]_{AgSoc}$:

$$\frac{\langle P, S, O, \mathit{ME}, \mathit{SE}, \mathit{Imp}, \mathit{Acc}\rangle : \mathit{AgSoc}}{O : \mathit{Org}}\ \mathit{DR}[\mathit{Org}]_{\mathit{AgSoc}}$$

- Rule $\mathit{DR}[\mathit{MEnv}]_{\mathit{AgSoc}}$:

$$\frac{\langle P, S, O, \mathit{ME}, \mathit{SE}, \mathit{Imp}, \mathit{Acc}\rangle : \mathit{AgSoc}}{\mathit{ME} : \mathit{MEnv}}\ \mathit{DR}[\mathit{MEnv}]_{\mathit{AgSoc}}$$

- Rule $\mathit{DR}[\mathit{SEnv}]_{\mathit{AgSoc}}$:

$$\frac{\langle P, S, O, \mathit{ME}, \mathit{SE}, \mathit{Imp}, \mathit{Acc}\rangle : \mathit{AgSoc}}{\mathit{SE} : \mathit{SEnv}}\ \mathit{DR}[\mathit{SEnv}]_{\mathit{AgSoc}}$$

- Rule $\mathit{DR}[\mathit{IMP}]_{\mathit{AgSoc}}$:

$$\frac{\langle P, S, O, \mathit{ME}, \mathit{SE}, \mathit{Imp}, \mathit{Acc}\rangle : \mathit{AgSoc}}{\mathit{Imp} : \mathit{IMP}}\ \mathit{DR}[\mathit{IMP}]_{\mathit{AgSoc}}$$

- Rule $\mathit{DR}[ACC]_{\mathit{AgSoc}}$:

$$\frac{\langle P, S, O, \mathit{ME}, \mathit{SE}, \mathit{Imp}, \mathit{Acc}\rangle : \mathit{AgSoc}}{\mathit{Acc} : ACC}\ \mathit{DR}[ACC]_{\mathit{AgSoc}}$$

Chapter 7

Re-typing Classical Organizational Models with *TPO*

7.1 AALAADIN

The Aalaadin model (later known as the AGR model, see [15]) was introduced in [14], by Jacques Ferber and Olivier Gutknecht, as a meta-model for MAS organizations. It was of great historical importance for the MAS area, for it was the first explicitly proposed full-fledged organizational model for the concept of *group of agents*, and the first to explicitly propose the *externalist* point of view, that is, the point of view that abstracts away the details of the structure and operation of the agents.

As such, it is at the root of a whole series of works that proliferated in the late 1990's and early 2000's, about the notion of *MAS organization*, some of which we review in this part of the report, re-typing them with *TPO*.

In this section, we re-type with *TPO* the Aalaadin mode. In Sect. 7.2, we review and type to *TPO* the *AGRE* model, an extension of *AGR* (i.e., Aalaadin) with the concept of *environment*.

Aalaadin is composed of two main parts: the *core model* and the so-called *analytical/methodological model*. We discuss them in that order.

7.1.1 The *Core Model*

The three concepts that constitute the *core model* of Aalaadin are *agent*, *group* and *role*. The classical diagram that represents Aalaadin is like that in Fig. 7.1 (outside the *core concepts* box are the *analytical/methodological concepts*).

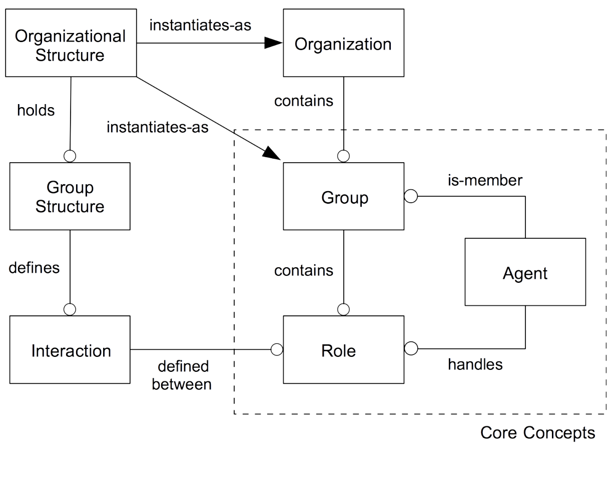

Figure 7.1: The Aalaadin model.

In our re-typing, the core concepts correspond to the *TPO* types of *agent* (*Agent*), *organization unit* (*OrgUn*) and *organizational role* (*OrgRo*).

The re-typing of the relations between the core concepts of *Aalaadin* goes as follows (see formal presentations below):

- *is-member*: relates *agents* to *groups*; we re-type it as the *implementation of groups by agents*, denoted by $Imp_{OrgUn/Ag}$, a relation that is not basic in *TPO*, but that we introduce below as a composition of basic *TPO* relations;
- *handles*: relates *agents* to *roles*; we re-type it as the *direct implementation of organizational roles by agents* ($Imp_{OrgRo/Agent}$);
- *contains*: relates *groups* to *roles*; we re-type it as the *membership of roles* (elements of the set *OR*) to *organization units* (instances *ou* of the type *OrgUn*), which is implicit in the form: $ou = \langle OR, ...\rangle$); we denote the type of such membership relations by $Memb_{OrgRo,OrgUn}$.

In Aalaadin, there is no notion that corresponds to that of *organization unit behavior*, since groups are, in principle, just to "a way to tag a set of agents", they need not "contain" roles. Only when they "contain" roles they correspond to full-fledged *organization units*, although without a notion of *organization unit behavior*[1].

In consequence, in *Aalaadin*, there is no notion of group interaction, i.e., no object of the type *organization unit interaction* (*OrgUnInter*). That is, in Aalaadin, groups do not interact with each other *as groups*: only their member agents interact.

Roles, on the other hand, are bound to "abstract interaction schemes". These interaction schemes correspond to the *TPO* type *organizational role interaction* (*OrgRoInter*). But, those abstract interaction schemes are not explicitly included in the *core model*, got no formalized presentation, and are considered to be part of the *analytical/methodological concepts*.

Finally, notice that, in *Aalaadin*, groups cannot be constituted by hierarchically ordered, lower-level groups: groups may intersect, may share all of their members, but groups cannot, *as groups*, be part of other groups.

7.1.2 The *Core Model* Formally Re-typed in *TPO*

To formally account for the re-typing of the core model of Aalaadin with *TPO*, we need first to define, in terms of the basic relations constitutive of the types of *TPO*, the composite type $Imp_{OrgUn/Agent}$. This may be done by defining that relation, in a recursive way, as follows[2].

Defining the Implementation Relation $Imp_{OrgUn/Agent}$

As the basic form of organization units is $ou = \langle OR, ...\rangle$, where *OR* is the set of *organizational roles* that constitute *ou*, we may define the basic case of the relation $imp(ag, ou) : Imp_{OrgUn/Agent}$ as:

$$\text{Base: } imp(ag, ou) \Leftrightarrow ou = \langle OR, ...\rangle \wedge [\exists or \in OR \wedge imp(ag, or)]$$

where $imp(ag, or) : Imp_{OrgRo/Agent}$.

The recursive case of the definition of $imp(ag, ou) : Imp_{OrgUn/Agent}$ corresponds to the cases in which $ou = \langle OU, ...\rangle$ with $OU : \wp(OrgUn)$, that is, when *ou* is constituted by lower-level, already constituted, organization units. In such cases, $imp(ag, ou') : Imp_{OrgUn/Agent}$ holds for some $ou' : OrgUn$, that is, for an organization unit of a lower-level then *ou*.

Thus, we define the recursive case of $imp(ag, ou) : Imp_{OrgUn/Agent}$ by:

$$\text{Recursive step: } imp(ag, ou) \Leftrightarrow ou = \langle OU, ...\rangle \wedge \exists ou' \in OU[imp(ag, ou')]$$

[1]This notion of *group as a name of a set of roles* (or a *set of role names*), not as an *active entity* in an agent society (that is, an entity with activity of its own, conceptually separable from the activities of its members, as in *PopOrg*) pervades all the models that we review here and, in general, all the models that followed the track initiated by Aalaadin.

[2]Notice that recursion is not used in the original definition of *AALAADIN*. The recursive definition of $Imp_{OrgUn/Agent}$ serves also the purpose of illustrating the extensibility of *TPO*.

Re-typing the *Core Model*

The re-typing of the core model of AALAADIN with *TPO* may be as follows, with the re-typing of *Aaladin* concepts as types of *TPO* denoted by "$\hookrightarrow$":

$$\text{Role} \hookrightarrow \mathit{Rol} \subseteq \mathit{OrgRo} \tag{7.1}$$
$$\text{Group} \hookrightarrow \mathit{Group} \subseteq \mathit{OrgUn} \tag{7.2}$$
$$\text{contains} \hookrightarrow \mathit{constains} : \mathit{Memb}_{\mathit{OrgRo},\mathit{OrgUn}} \tag{7.3}$$
$$\text{handles} \hookrightarrow \mathit{handles} : \mathit{Imp}_{\mathit{OrgRo}/\mathit{Agent}} \tag{7.4}$$
$$\text{is-member} \hookrightarrow \mathit{is\text{-}member} : \mathit{Imp}_{\mathit{OrgUn}/\mathit{Agent}} \tag{7.5}$$
$$\text{Core Model} \hookrightarrow \mathit{CoreModel} = \wp(\mathit{Rol}) \times \wp(\mathit{Group}) \times \{(\mathit{contains}, \mathit{handles}, \mathit{is\text{-}member})\} \tag{7.6}$$

We remark that, as defined in Chap. 5.2, to the left of the re-typing sign ("$\hookrightarrow$") we have the original *concepts* and *objects* of AALAADIN, while to the right of that sign we have the corresponding *types* and *objects* of *TPO*.

Notice that *is-member*(ag, grp) can be deduced from *handles*(ag, ro), given enough information about the roles *contained* in *grp*.

One additional aspect of the core model of AALAADIN should yet be mentioned. It concerns the procedure that an agent should follow to become a member of a group. In [15], this procedure is elaborated and shown to be an important part of the mechanism by which groups assure a certain level of security for the organization of a multiagent system.

The re-typing of such feature amounts to the definition of a particular behavior for a particular role that agents may play in a group, namely, the role of *gatekeeper*, that every group should have, together with an auxiliary *entrance group*, which the *gatekeeper* controls. We do not provide the re-typing of such feature here, however.

7.1.3 The *Analytical/Methodological Model*

The analytical/methodological model is constituted by "several concepts [...] that are not represented directly in multiagent organizations". It encompasses "all possible roles, valid interactions, and structures of groups and organizations" and "only serves as an analysis and design tool".

The following are the analytical/methodological concepts:

1. *Group structure*: "identifies all the roles and interactions that can appear within a group";

   Formally, a *group structure* is given by $S = \langle R, G, L \rangle$ where:

   – R is a finite set of *role identifiers*;
   – G is the *interaction graph* given by $G : R \times R \to \mathcal{L}$, where $\mathcal{L}$ is a set of labels;
   – L is the *interaction language*, the "formalism for the individual interaction definitions", each such definition called a "protocol", one protocol for each interaction between two roles.

   We note that *group structures*:

   - encompass "role identifiers", not *roles* themselves;
   - characterize interactions in intensional terms, by "protocols", not extensively, as *processes*;
   - the mapping of labels (*labels*) to protocols (p) is not given any formal name, being indirectly specified by the logical formula: $\forall(r_i, r_j, \mathit{label}) \in G, \exists! p \in L$;
   - although not stated in a formal way, the group structures are taken to be dynamical, with the set of contained role identifiers allowed to vary in time.

2. *Organizational structure*: defined to be a "set of group structures expressing the design of a multi-agent organization scheme".

In other words, in AALAADIN, multiagent systems are taken to have an "organization", whose scheme is expressed as a set of group structures, conjoined in an *organizational structure.*

Formally, an organizational structure is a tuple of the form $O = \langle S, \mathit{Rep} \rangle$ where[3]:

- S is a set of valid group structures;
- Rep is the *representative graph*, that is, a graph with edges relating two roles, each in a different group, $S_1 = \langle R_1, G_1, L_1 \rangle$ and $S_2 = \langle R_2, G_2, L_2 \rangle$ of S, such that an agent may play the two roles simultaneously.

We remark that:

- in *Aalaadin*, *group* and *organization* are not terms to be taken at the same conceptual level: *groups* are forms of components of multiagent systems, *organizations* are forms that multiagent systems, taken as a whole, may have. In the *TPO* type system, on the other hand, both *groups* and *organizations*, formally construed as *organization units*, are *components* of multiagent systems[4], the forms of the latter given by their *architectures*;
- the need to define an *organization structure* by referring the representative graph of an organization structure to the *roles* that belong to the groups that constitute the organization, not to the groups themselves, is in consonance with the fact that groups are not animate entities, in the AALAADIN model, but only containers for roles and agents.

7.1.4 Re-typing of the *Analytical/Methodological Model*

The re-typing of *analytical/methodological model* may proceed as follows:

1. Group structures:

We re-type a *group structure* with *TPO* in terms of a reduced form of *organization unit*:

$$\mathit{OrgUn}' = \wp(\mathit{OrgRo}) \times \wp(\mathit{OrgRoInter}) \tag{7.7}$$

$$\text{Group Structure} \hookrightarrow \mathit{GroupStruc} \subseteq \mathit{OrgUn}' \tag{7.8}$$

where OrgUn' is a reduction of the *TPO* type $\mathit{OrgUnit}$[5].

Notice that *TPO*, being a set-theoretic type-system, is *extensional*, so that interactions are modeled as *processes* (time-indexed sequences of pairs of sets of actions), not *intensionally*, by expressions ("protocols") of a protocol language.

We, thus, re-type protocols as *symbolic objects* of the *symbolic environment* of the organization (which is never actualized as an architectural component of the agent society, in the case of AALAADIN):

$$\text{Protocol} \hookrightarrow \mathit{Protocol} \subseteq \mathit{SymbObj} \tag{7.9}$$

2. Organizational structures:

With the re-typing of *groups* as *organization units*, and re-typing the representative relation by:

$$\text{Label} \hookrightarrow \mathit{Label} \subseteq \mathit{SymbObj} \tag{7.10}$$

$$\text{Rep} \hookrightarrow \mathit{Rep} \subseteq \mathit{OrgUn}' \times \mathit{OrgUn}' \times \mathit{Label} \tag{7.11}$$

[3]The writing of this formal definition of *organizational structure* in [14] is somewhat confusing, probably due to typos. What follows is our reading of what is printed there.

[4]The concept of *group* can be introduced as a separate concept, in *TPO*, to encompass any kind of *aggregation* of social actors (agents, organizational roles, organization units, organization unit networks, agent societies, etc.). Formally, in such use, the concept of a *group* would be a *structural operator*, acting at any of the architectural levels of agent societies and inter-societal agent systems. See the development of this *operatory notion of group* in Chap. **??**.

[5]For comparison, see Sect. 6.6.4, where:

$$\mathit{OrgUnit} = (\wp(\mathit{OrgRo}) \times \wp(\mathit{OrgRoInter}) \times \wp(\mathit{OrgRoRel}) \times \wp(\mathit{OrgRo}) \times \mathit{InpPort} \times \mathit{OutPort}$$

we may have the re-typing of the *organizational structures* as *organization unit networks*:

$$OrgUnRel' = OrgUn' \times OrgUn' \times Rel \tag{7.12}$$
$$OrgUnNet' = \wp(OrgUn') \times \wp(OrgUnLnk) \tag{7.13}$$
$$\text{Organizational Structure} \hookrightarrow OrgStruct \subseteq OrgUnNet' \tag{7.14}$$

where *OrgUnNet′* is a reduced form of the TPO type *OrgUnNet*. For comparison, see Sect. 6.6.6.

7.1.5 Aalaadin Re-typed with *TPO*

Figure 7.2 summarizes the above re-typing of the Aalaadin model with *TPO*.

$$\text{Role} \hookrightarrow Rol \subseteq OrgRo \tag{7.1}$$
$$\text{Group} \hookrightarrow Group \subseteq OrgUn \tag{7.2}$$
$$\text{contains} \hookrightarrow contains : Memb_{OrgUn/OrgRo} \tag{7.3}$$
$$\text{handles} \hookrightarrow handles : Imp_{OrgRo/Agent} \tag{7.4}$$
$$\text{is-member} \hookrightarrow \textit{is-member} : Imp_{OrgUn/Agent} \tag{7.5}$$
$$\text{Core Model} \hookrightarrow CoreModel = \wp(Role) \times \wp(Group) \times \{contains, handles, \textit{is-member}\} \tag{7.6}$$

The Analytical/Methodological Model:

$$OrgUn' = \wp(OrgRo) \times \wp(OrgRoInter) \tag{7.7}$$
$$\text{Group Structure} \hookrightarrow GroupStruct \subseteq OrgUn' \tag{7.8}$$
$$\text{Protocol} \hookrightarrow Protocol \subseteq SymbObj \tag{7.9}$$
$$\text{Label} \hookrightarrow Label \subseteq SymbObj \tag{7.10}$$
$$\text{Rep} \hookrightarrow Rep \subseteq OrgUn' \times OrgUn' \times SymbObj \tag{7.11}$$
$$OrgUnRel' = OrgUn' \times OrgUn' \times Rel \tag{7.12}$$
$$OrgUnNet' = \wp(OrgUn') \times \wp(OrgUnRel') \tag{7.13}$$
$$\text{Organizational Structure} \hookrightarrow OrgStruct \subseteq OrgUnNet' \tag{7.14}$$

Figure 7.2: The Aalaadin model typed with *TPO*.

7.2 *AGR* and *AGRE*

AGRE (Agent-Group-Role-Environment) [16] is the extension of *AGR* (the Aalaadin model that we studied in Sect. 7.1) with an *environment* component.

In a general sense, an *environment* constitutes "the conditions under which an entity exists" [16](p.49, citing [37]). The aim of *AGRE* is to allow for the design of *social* and *physical environments* in an integrated way. *Groups* are already a form of social environment, for they are "a context for a pattern of activity" [16](p.50).

7.2.1 *AGR*: AALAADIN Revised

Differently from the original paper about Aalaadin, the paper about *AGRE* introduces in *AGR* the concepts of *group type* and *role type*, of an abstract and descriptive level (called the *organizational level*[6]), and let *groups* and *roles* be associated in a more concrete way, making of *AGR*

[6]The concept of *organizational level* was already introduced, in general terms, in [15].

a more complete model than Aalaadin. This is shown in the class diagram of Fig. 7.3 (adapted from [16](p.51)).

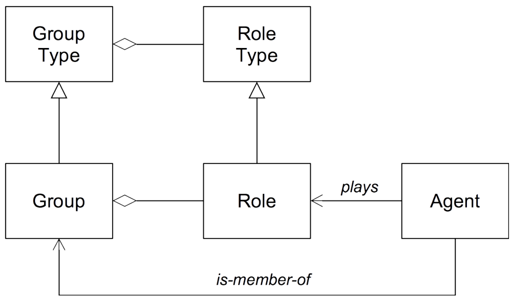

Figure 7.3: The *AGR* class structure.

A *role type* describes the expected behavior of an agent playing a role of that type in a group. As such, a role type defines an *interface* that the agent shows inside the group. As in [15], *interaction protocols* and *structural constraints* between types may be specified.

A *group type* in *AGR* describes:

- the set of *agents* that constitute the group;
- what are the *roles* of the agents in the group;
- the communication language that agents may use in the group;
- the norms that apply in the group.

7.2.2 *AGRE*

AGRE takes that *agents* are situated in *spaces* and introduces the concept of *mode* to talk about the way agents perform *actions* in those spaces.

A *mode*, or *interface*, is loosely defined as "the way of existence of an agent in a space" [16](p.51). It defines "an agent's location and the way it perceives and acts within a space". The two types of modes are *body* (the mode of an agent in a physical space) and *role* (the mode of an agent in a group).

The concrete level of *AGRE* (the level where agents and groups exist as concrete entities) acquires a much more complex class structure than that of *AGR*, as shown in Fig. 7.4 (adapted from [16]).

Worlds can be of two types: *organizations*, which are social environments composed of *groups*, and *physical worlds*, which are physical environments composed of *spaces*.

Agents may enter *worlds* (either organizations or physical worlds), through appropriate primitive operations and, once inside a world, they may enter one or more of its *spaces* (either *groups* or

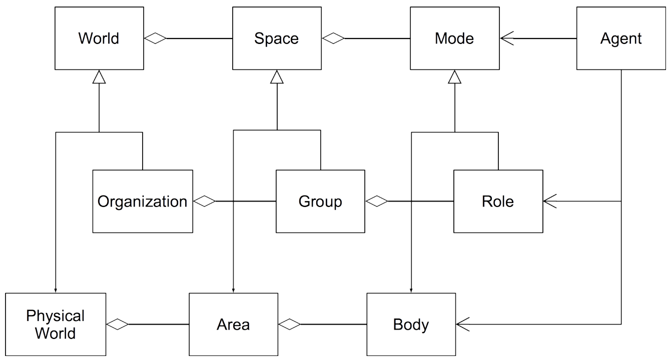

Figure 7.4: The *AGRE* class structure.

areas, respectively). Once inside a space, agents may, again though appropriate primitives, acquire one or more of its *modes*(*roles* or *bodies*, respectively, with the restriction that an agent can have at most one body, in any area it enters).

So, tuples of the form $\langle world, group, role\rangle$ and $\langle world, area, body\rangle$ constitute *agent properties* that guide the behaviors that agents may have in each of those types of contexts.

Agents may perform actions, in a given space, in accordance with the actions allowed by each mode it has in that space. Communication actions can only be performed when playing roles in a group, and in connection to the roles of that group.

7.2.3 Re-typing the Primitive Operations of *AGRE* with *TPO*

AGRE defines a set of *primitive operations*, to allow agents to handle the main components of agent systems. Some of those primitive operations, mentioned in the paper [16], are the following:

- The operation of *registering at a world*, which agents must perform when they are created. The paper gives no further information about that operation. One may guess that it could be re-typed more or less as:

$$\text{Agent registers in a world } \hookrightarrow \mathit{reg} : \mathit{Agent} \times \mathit{World} \rightarrow \{\mathit{Success}, \mathit{Failure}\}$$

- Once in a world (for instance, in an organization), an agent may *register in a group* by requesting to enact a role of a given type, in that group. The formal expression of that operation is (using a Java syntax):

```
Role r = o.requestRole(GroupName, RoleType, RoleName, Authorizations);
```

  which we re-type as:

$$\texttt{requestRole} \hookrightarrow \mathit{requestRole} : \mathit{OrgUn}' \times \mathit{SymbObj} \times \mathit{SymbObj} \times \wp(\mathit{SymbObj}) \rightarrow \mathit{OrgRo}$$

- If an agent enters a *physical world*, it can request to enter an *area*, by requesting a *body* of a certain *body type*, to use in that area. This is done by the operation:

```
Body b = p.requestBody(AreaName, BodyType, Location, Authorizations);
```

  which we re-type as:

$$\text{requestBody} \hookrightarrow \mathit{requestBody} : \mathit{SymbObj} \times \mathit{SymbObj} \times \mathit{SymbObj} \times \wp(\mathit{SymbObj}) \rightarrow \mathit{MatObj}$$

7.2.4 *AGRE* Re-typed with *TPO*

Figure 7.5 gives the re-typing of the *AGRE* model with *TPO*, where we make use of some of the types defined for the re-typing of the Aalaadin model (see Fig. 7.2).

7.3 *Electronic Institutions*

The *Electronic Institution* (*EI*) model of organization of agent societies evolved from Pablo Noriega's thesis [38], through a number of further works by the IIIA/Barcelona (e.g., [39], [40] and [41]). We take the presentation of electronic institutions given in [42] and [41] as the bases for our re-typing of the *EI* model with *TPO*.

According to [42] (see also [39]), the concept of *institution* adopted in the *EI* model derives from [43] (see also [44]), which follows the custom in Economics of taking an institution to be a *system of norms*, not an *entity* acting within a society. We remark, however, that notwithstanding that statement the *electronic institutions*, with their *institutional agents*, do exactly that, i.e., they realize a model of such concrete acting entities.

$$\begin{aligned}
\text{Agent} &\hookrightarrow \mathit{Agent} && (7.15)\\
\text{Role} &\hookrightarrow \mathit{OrgRo} && (7.16)\\
\text{Body} &\hookrightarrow \mathit{Bod} \subseteq \mathit{MatObj} && (7.17)\\
\text{Group} &\hookrightarrow \mathit{Group} \subseteq \mathit{OrgUn}' && (7.18)\\
\text{Area} &\hookrightarrow \mathit{Area} \subseteq \wp(\mathit{MatObj}) && (7.19)\\
\text{Group type} &\hookrightarrow \mathit{GroupType} \subseteq \wp(\mathit{OrgUn}') && (7.20)\\
\text{Space} &\hookrightarrow \mathit{Space} \subseteq \wp(\mathit{MatObj}) && (7.21)\\
\text{Organization} &\hookrightarrow \mathit{Organization} \subseteq \mathit{OrgUnNet}' && (7.22)\\
\text{Physical World} &\hookrightarrow \mathit{PhysWorld} \subseteq \wp(\mathit{Area}) && (7.23)\\
\text{World} &\hookrightarrow \mathit{World} = \mathit{PhysWorld} \cup \mathit{Organization} && (7.24)
\end{aligned}$$

Figure 7.5: The *AGRE* model typed with *TPO*.

The aim of an *institution*, in the sense of [43], is to establish "a stable structure for human interactions" [42](p.34), and an *electronic institution* aims to do that with the support of electronic means, particularly, computational means and, specially, agent systems.

But while an *institution*, in the sense of [43], obliges individuals by operating mainly at the level of their minds, an *electronic institution* operates at the level of the individual's actions, by blocking actions that are not according to the norms. This is so, because electronic institutions are used as intermediary between the interacting individuals, with means to automatically *block* forbidden actions and to impose *sanctions* without considering arguments before their application, a mode of operation first called *regimentation* in [45].

7.3.1 *Electronic Institutions* Informally Presented

The core notions of an *EI* are:

- *Agents* and *roles*: *agents* are the players of the *EI*, interacting through *speech acts*, and *roles* are *standardized patterns of behaviors.* Agents and roles are differentiated between *institutional* and *non-institutional* agents and roles, the former having the purpose of enacting the regimentation processes that implement the norms of the institutions, the latter agents (called "delegates") having the purpose of representing the institutions' users (which are expected to conform to those norms, but that sometimes do not, which is the reason for the regimentation).

- *Dialogical framework*: constituted by the *ontology* and the *communication language* used by the agents inside the *EI.*

- *Scene*: a *multi-agent protocol* articulating the interactions between agents, specifying their possible dialogs, but in accordance with the roles they enact, i.e., a *role-based framework* for the agents' interactions.

- *Performative structure*: a workflow (network of scenes) determining the sequences of scenes that agents can traverse, in their functioning in the system. Performances of utterances in one scene of the performative structure may imply commitments to the performance of other utterances in further scenes.

- *Normative rules*: sets of rules that impose obligations and restrictions on actions that agents may perform in the scenes of the performative structure. Actions by *institutional agents* should ensure that actions by *non-institutional agents* abide by the institutional rules.

In summary, an *EI* is "a 'workflow' (performative structure) of multi-agent protocols (scenes) along with a collection of (normative) rules that can be triggered off by agents' actions (speech acts)" [42](p.35), whose purpose is to mediate all interactions among the participating (non-institutional) agents, so that they become regimented by the *EI*, according to the norms that it implements.

In what follows, we detail the fundamental concepts of *Electronic Institutions*, as they are embedded in the specification formalism introduced in [41]. As we do that, we type such concepts with *TPO*.

7.3.2 Re-typing the *Electronic Institution* Model with *TPO*

The main idea that should guide the re-typing of the *EI* model with *TPO* is that formulations of *EI*s are to be taken as *specifications* of electronic institutions, given at the conceptual level, or as *descriptions* of existent electronic institutions, not institutions effectively existent.

Even more, *EI* specifications are supposed to be compiled into machine readable forms, so that they may be used by the agents of the *EI*, when interacting with each other. That makes *EI* specifications be treated as, essentially, *symbolic structures* to be stored in a symbolic environment to which those agents should have access.

We take the following fundamental concepts of the *EI* model from the formal language that was introduced in [41] for the *specification* of electronic institutions.

Agents and Roles

- The relation between *agents* and *roles* is denoted by $a : r$, where a is an agent and r is a role. In the terminology of the specification language, $a : r$ is read by saying that agent a "is of the type" r. Since *agents* treat roles as their *properties*, and *roles* are defined as "patterns of dialogic action" [41](p.3), we re-type *EI agents*, *roles* and the relation "is of type" with *TPO* as:

$$\begin{aligned} \text{Agent} &\hookrightarrow \mathit{Agent} \\ \text{Role} &\hookrightarrow \mathit{Rol} \subseteq \mathit{SymbObj} \\ \text{is of type} &\hookrightarrow \textit{is-of-type} \subseteq \mathit{AgProp} \end{aligned}$$

- Roles are *hierarchically organized*, through a relation $r \succeq r'$, determining that r *subsumes* r' (which means that if a can play role r than it can also play role r'). We re-type the "subsumption" relation "$\succeq$" with *TPO* as:

$$\text{Role Subsumption} \hookrightarrow \mathit{RolSubsump} \subseteq \mathit{SymbObjRel}$$

- Roles may be related to each other by means of a relation of "static separation of duties" (*ssd*), meaning that if $(r, r') \in \mathit{ssd}$ then r and r' cannot be played by the same agent. We re-type the relation of "static separation of duties" with *TPO* as:

$$\text{Static Separation of Duties} \hookrightarrow \mathit{StaticSepDut} \subseteq \mathit{SymbObjRel}$$

- Roles may be either *institutional* or *non-institutional.* We take *non-institutional* roles to be the default type of role, and we re-type *institutional* roles as:

$$\text{Institutional Role} \hookrightarrow \mathit{InstRole} \subseteq \mathit{SymbObjProp}$$

Dialogic Frameworks

Dialogic frameworks, shared by two or more agents, allow them to communicate with each other. Dialogic frameworks are composed of:

- a *communication language*;
- a language to represent *domain content*;
- an *ontology*, to standardize their vocabulary about the domain.

We re-type *dialogic frameworks* as *networks of symbolic objects* whose elements are symbolic objects that represent:

- *terms* of the ontology;
- *sentences* of the representation language;
- *expressions* of the communication language;

which are related to each other by syntactical rules that define the properties and relations among them, establishing the structure of the network.

In particular, we notice that [41] determines that communication expressions are of the form $\iota(\alpha_i : \rho_i, \alpha_j : \rho_j, \phi, \tau)$ where:

- $\alpha_i : \rho_i$ and $\alpha_j : \rho_j$ are symbolic references to typed agents;
- ϕ is a sentence of the representation language;
- τ is a time instant;
- ι is an illocutionary particle.

which we re-type with *TPO* as:

$$\begin{aligned}
\text{OntologyTerm} &\hookrightarrow \mathit{OntologyTerm} \subseteq (\mathit{SymbObj})\\
\text{Ontology} &\hookrightarrow \mathit{Ontoloty} = \wp(\mathit{OntologyTerm})\\
\text{Symbolic Reference to Typed Agent} &\hookrightarrow \mathit{SymbRefTypedAg} \subseteq \mathit{SymbObj}\\
\text{Representation Sentence} &\hookrightarrow \mathit{RepSent} \subseteq \mathit{SymbObj}\\
\text{Representation Language} &\hookrightarrow \mathit{RepLang} = \wp(\mathit{RepSent})\\
\text{Illocutionary Particle} &\hookrightarrow \mathit{IllocPart} \subseteq \mathit{SymbObj}\\
\text{Time} &\hookrightarrow T\\
\text{Communication Expression} &\hookrightarrow \mathit{CommExpr} = \mathit{SymbRefTypedAg} \times \mathit{SymbRefTypedAg}\\
&\quad \times \mathit{RepSent} \times \mathit{IllocPart} \times T\\
\text{Communication Language} &\hookrightarrow \mathit{CommLang} = \wp(\mathit{CommExpr})
\end{aligned}$$

so that:

$$\text{Dialogical Framework} \hookrightarrow \mathit{DialogFrmwrk} = \mathit{Ontology} \times \mathit{RepLang} \times \mathit{CommLang}$$

Scenes

A *scene* of an *EI* is an “agent group meeting” [41](p.4) where agents interact according to a *protocol* (dialogical framework). The protocol (“pattern of multi-role conversations”) defines a set of possible dialogical interactions through which the agents, playing pre-determined roles, perform the activity of the electronic institution specified by the scene. Scenes prescribe definite points at which the agents may enter or leave them. As discussed below, the overall activity of an electronic institution is organized by the *performative structure*, as a network of scenes.

Scenes are composed of:

- a hierarchically ordered *set of roles*;
- a *dialogical framework*;
- a set of (identifiers of) *scene states*, with an *initial* and a *final* scene state;
- an *ordering relation* between states, each pair of states *labeled* with a communication expression;
- for each role, a *set of access and exit states*;

- for each role, an indication of the *maximum* and *minimum cardinality* of the set of agents that may play it.

The communication expression that labels a given pair of states stands for the dialogical action that promotes the *transition* of the scene from one state to the other.

We re-type *EI scenes* with *TPO* as follows:

$$\begin{aligned}
\text{Role Hierarchy} &\hookrightarrow \mathit{RoleHierarch} \subseteq \mathit{SymbObjRel} \\
\text{Scene State} &\hookrightarrow \mathit{SceneState} \subseteq \mathit{SymbObj} \\
\text{Initial Scene State} &\hookrightarrow \mathit{InitSceneState} \subseteq \mathit{SymbObj} \\
\text{Final Scene State} &\hookrightarrow \mathit{FinSceneState} \subseteq \mathit{SymbObj} \\
\text{State Ordering Relation} &\hookrightarrow \mathit{StateOrder} \subseteq \mathit{SymbObjRel} \\
\text{Role Access States} &\hookrightarrow \mathit{RoleAccState} \subseteq \mathit{SymbObjRel} \\
\text{Role Exit States} &\hookrightarrow \mathit{RoleExitState} \subseteq \mathit{SymbObjRel} \\
\text{Role Maximum Cardinality} &\hookrightarrow \mathit{RoleMaxCard} \subseteq \mathit{SymbObjRel} \\
\text{Role Minimum Cardinality} &\hookrightarrow \mathit{RoleMinCard} \subseteq \mathit{SymbObjRel} \\
\text{Scene} &\hookrightarrow \mathit{Scene} = \wp(\mathit{Role}) \times \mathit{DialogFrmwrk} \times \wp(\mathit{SceneState}) \\
&\quad \times \mathit{InitSceneState} \times \mathit{FinSceneState} \times \mathit{StateOrder} \times \wp(\mathit{RoleAccState}) \\
&\quad \times \wp(\mathit{RoleExitState}) \times \mathit{RoleMaxCard} \times \mathit{RoleMinCard}
\end{aligned}$$

Performative Structures

Performative structures "model the relationships between scenes" [41](p.5). They may:

- capture *causal dependencies* between scenes;
- *synchronize* agent actions, before they start to act together in another scene;
- be realized in *parallel*;
- define *choice points* for agents, in terms of alternative exit states;
- establish the *role policy* between scenes, that is, the way agents may leave roles in a scene and adopt other roles in another scene.

Agents may perform the following *operations*, regarding the scenes of a performative structure [41](p.6):

- *start* a new scene;
- *enter* an active scene;
- *participate* in multiple scenes, simultaneously;
- *leave* an active scene;
- *abandon* the performative structure.

To allow for such possibilities, performative structures have *transition scenes*, where agents may perform such transition operations. *Transition* and *non-transition scenes* interleave in their connections, so that any connection between two *non-transition scenes* (or, simply *scenes*) is mediated by a *transition scene* (or, simply, *transition*).

Transitions are classified in the following way, according to the way they relate their *access* and *exit points*:

- *And/And*, meaning that agents are forced to synchronize at their access points, before leaving together the exit points;
- *Or/Or*, meaning that agents may go asynchronously (i.e., independently of each other) through the access and exit points of the transition;

- *And/Or*, meaning that agent are forced to synchronize at their access points, but may leave asynchronously through the exit points;
- *Or/And*, meaning that the agents may enter the transition asynchronously, but may leave them only synchronously.

Connection arcs connect scenes to transitions, and transitions to scenes. For each scene or transition, they indicate which agents, playing which roles, are to path through it.

We may, thus, re-type *types of transitions* and *connection arcs* with *TPO* as:

$$\begin{aligned}\text{Connection Arc} &\hookrightarrow \mathit{ConnecArc} \subseteq \mathit{SymbObj}\\ \text{Transition} &\hookrightarrow \mathit{Transition} \subseteq \mathit{SymbObj}\\ \text{Agent/Role Path} &\hookrightarrow \mathit{AgRolePath} \subseteq \mathit{SymbObjProp}\\ \text{Transition Type} &\hookrightarrow \mathit{TransitionType} \subseteq \mathit{SymbObjProp}\end{aligned}$$

In addition, agents are restricted to follow connection arcs only if certain constraints (given by logical expressions) are satisfied, to the effect that certain conditions should hold for the agents to follow certain connection arcs. We re-type such arc constraints as:

$$\text{Arc Constraint} \hookrightarrow \mathit{ArcCnstr} \subseteq \mathit{SymbObj}$$

Scenes may have a limit in the number of its simultaneous instantiations, re-typed with *TPO* as:

$$\text{Scene Upper Bound} \hookrightarrow \mathit{SceneUpBound} \subseteq \mathit{SymbObj}$$

A *performative structure* is defined, then, as composed of the following elements:

- a *set of scenes*;
- a *set of transitions*;
- the *root scene*;
- the *output scene*;
- a set of *exit arcs*, linking a scene to a transition;
- a set of *access arcs*, linking a transition to a scene;
- a set of *agent/role paths* for each connection arc;
- for each transition, its *transition type*;
- for each connection arc, its *arc type*;
- for each connection arc, its *constraint*;
- for each scene, the *bound* in the number of its simultaneous instantiations.

We re-type *EI performative structures* with *TPO* as:

$$\begin{aligned}
\text{Set of Scenes} &\hookrightarrow \mathit{SetScenes} \subseteq \wp(\mathit{Scene})\\
\text{Set of Transitions} &\hookrightarrow \mathit{SetTrans} \subseteq \wp(\mathit{Transitions})\\
\text{Root Scene} &\hookrightarrow \mathit{RootScene} \subseteq \mathit{SymbObj}\\
\text{Output Scene} &\hookrightarrow \mathit{OutScene} \subseteq \mathit{SymbObj}\\
\text{Set of Exit Arcs} &\hookrightarrow \mathit{SetExitArcs} \subseteq \wp(\mathit{Arc})\\
\text{Set of Acess Arcs} &\hookrightarrow \mathit{SetAccArcs} \subseteq \wp(\mathit{Arc})\\
\text{Set of Transition Types} &\hookrightarrow \mathit{SetTransTypes} \subseteq \wp(\mathit{TransitionType})\\
\text{Set of Arc Types} &\hookrightarrow \mathit{SetArcTypes} \subseteq \wp(\mathit{AcType})\\
\text{Set of Arc Constraints} &\hookrightarrow \mathit{SetArcCnstrs subseteq} \wp(\mathit{ArcCnstr})\\
\text{Set of Scene Bound} &\hookrightarrow \mathit{SetSceneBound} \subseteq \wp(\mathit{SceneBound})\\
\text{Performative Structure} &\hookrightarrow \mathit{PerformStrct} = \mathit{SetScenes} \times \mathit{SetTransitions} \times \mathit{RootScene} \times \mathit{OutputScene}\\
&\quad \times \mathit{SetExitArcs} \times \mathit{SetAccArcs} \times \mathit{SetTransTypes} \times \mathit{ArcTypes}\\
&\quad \times \mathit{SetArcConstr} \times \mathit{SetSceneBounds}
\end{aligned}$$

Normative Rules

The *EI* model defines the concept of an *agent context* as the set of conditions that determine the agent's obligations, permissions and prohibitions, either indirectly through the *form* of the performative structure and its scenes, or directly through *constraints* and *norms* imposed on the agent's actions:

- inside scenes, the *scene protocol* and the *constraints* on the arcs that link actions indicate what the agent can or cannot say, according to the roles it plays in the scene;
- between scenes, the *transitions* and the *constraints* over the arcs determine the possible paths of the agents in the performative structure;
- between scenes, *norms* determine how actions performed in one scene imply *obligations* (*commitments*) to be fulfilled in other scenes, through the performance of other actions in the latter.

Obligations are represented by deontic expressions of the form:

- `obliged(x,`ψ`,s)`

meaning that agent x is obliged to do, in scene s, the illocutionary act determined by the communication expression ψ.

Norms are expressed by *rules* of the form:

- $(s_1, \gamma_1) \wedge ... \wedge (s_m, \gamma_m) \wedge \phi_1 \wedge ... \wedge \phi_n \Rightarrow \phi_{n+1} \wedge ... \wedge \phi_r$

where:

- s_i is a scene;
- γ_i is an illocution performed in s_i;
- ϕ_i is the logical expression of a condition;

such that conclusions, if necessary, may be assessed by constraints in the current or next scenes. In particular, the $\phi_{n+1}, ..., \phi_r$ may be expressions of obligations to be fulfilled in next scenes.

We re-type *normative rules* with *TPO* as:

$$\text{Normative Rule} \hookrightarrow \mathit{NormRule} \subseteq \mathit{SymbObj}$$

Electronic Institutions

An *electronic institution* is defined, then, as being composed of:

- a *performative structure*;
- the indication of which of the roles of the performative structure are *institutional* roles;
- the *hierarchy of subsumption* of roles;
- the set of *normative rules*;
- the policy of *static separation of duties*.

We re-type *electronic institutions* with *TPO* by:

Electronic Institution $\hookrightarrow EI = \mathit{PerformStrct} \times \wp(\mathit{InstRole}) \times \wp(\mathit{NormRule}) \times \mathit{StaticSepDut}$

7.3.3 The *EI* Model Re-typed with *TPO*

Figures 7.6 to 7.9 give the re-typing of the *Electronic Institution* model with *TPO*.

$$\text{Agent} \hookrightarrow \mathit{Agent} \tag{7.1}$$
$$\text{Role} \hookrightarrow \mathit{Rol} \subseteq \mathit{SymbObj} \tag{7.2}$$
$$\text{is of type} \hookrightarrow \mathit{is\text{-}of\text{-}type} \subseteq \mathit{AgProp} \tag{7.3}$$
$$\text{Role Subsumption} \hookrightarrow \mathit{RoleSubsumptsubseteqSymbObjRel} \tag{7.4}$$
$$\text{Static Separation of Duties} \hookrightarrow \mathit{StaticSpeDutsubseteqSymbObjRel} \tag{7.5}$$
$$\text{Institutional Role} \hookrightarrow \mathit{InstRolesubseteqSymbObjProp} \tag{7.6}$$
$$\text{OntologyTerm} \hookrightarrow \mathit{OntologyTerm} \subseteq (\mathit{SymbObj}) \tag{7.7}$$
$$\text{Ontology} \hookrightarrow \mathit{Ontoloty} = \wp(\mathit{OntologyTerm}) \tag{7.8}$$
$$\text{Symbolic Reference to Typed Agent} \hookrightarrow \mathit{SymbRefTypedAg} \subseteq \mathit{SymbObj} \tag{7.9}$$
$$\text{Representation Sentence} \hookrightarrow \mathit{RepSent} \subseteq \mathit{SymbObj} \tag{7.10}$$
$$\text{Representation Language} \hookrightarrow \mathit{RepLang} = \wp(\mathit{RepSent}) \tag{7.11}$$
$$\text{Illocutionary Particle} \hookrightarrow \mathit{IllocPart} \subseteq \mathit{SymbObj} \tag{7.12}$$
$$\text{Time} \hookrightarrow T \tag{7.13}$$
$$\text{Communication Expression} \hookrightarrow \mathit{CommExpr} = \mathit{SymbRefTypedAg} \times \mathit{SymbRefTypedAg} \times \mathit{RepSent} \times \mathit{IllocPart} \times T \tag{7.14}$$
$$\text{Communication Language} \hookrightarrow \mathit{CommLang} = \wp(\mathit{CommExpr}) \tag{7.15}$$
$$\text{Dialogical Framework} \hookrightarrow \mathit{DialogFrmwrk} = \mathit{Ontology} \times \mathit{RepLang} \times \mathit{CommLang} \tag{7.16}$$

Figure 7.6: *Basic Elements* and *Dialogical Frameworks* of Electronic Institutions re-typed with *TPO*.

$$\begin{aligned}
\text{Role Hierarchy} &\hookrightarrow \mathit{RoleHierarch} \subseteq \mathit{SymbObjRel} && (7.17)\\
\text{Scene State} &\hookrightarrow \mathit{SceneState} \subseteq \mathit{SymbObj} && (7.18)\\
\text{Initial Scene State} &\hookrightarrow \mathit{InitSceneState} \subseteq \mathit{SymbObj} && (7.19)\\
\text{Final Scene State} &\hookrightarrow \mathit{FinSceneState} \subseteq \mathit{SymbObj} && (7.20)\\
\text{State Ordering Relation} &\hookrightarrow \mathit{StateOrder} \subseteq \mathit{SymbObjRel} && (7.21)\\
\text{Role Access States} &\hookrightarrow \mathit{RoleAccState} \subseteq \mathit{SymbObjRel} && (7.22)\\
\text{Role Exit States} &\hookrightarrow \mathit{RoleExitState} \subseteq \mathit{SymbObjRel} && (7.23)\\
\text{Role Maximum Cardinality} &\hookrightarrow \mathit{RoleMaxCard} \subseteq \mathit{SymbObjRel} && (7.24)\\
\text{Role Minimum Cardinality} &\hookrightarrow \mathit{RoleMinCard} \subseteq \mathit{SymbObjRel} && (7.25)\\
\text{Scene} &\hookrightarrow \mathit{Scene} = \wp(\mathit{Role}) \times \mathit{DialogFrmwrk} \times \wp(\mathit{SceneState})\\
&\times \mathit{InitSceneState} \times \mathit{FinSceneState} \times \mathit{StateOrder} \times \wp(\mathit{RoleAccState})\\
&\times \wp(\mathit{RoleExitState}) \times \mathit{RoleMaxCard} \times \mathit{RoleMinCard} && (7.26)
\end{aligned}$$

Figure 7.7: *Scenes* of Electronic Institutions re-typed with *TPO*.

$$\begin{aligned}
\text{Connection Arc} &\hookrightarrow \mathit{SymbObj} && (7.27)\\
\text{Transition} &\hookrightarrow \mathit{SymbObj} && (7.28)\\
\text{Agent/Role Path} &\hookrightarrow \mathit{SymbObjProp} && (7.29)\\
\text{Transition Type} &\hookrightarrow \mathit{SymbObjProp} && (7.30)\\
\text{Arc Constraint} &\hookrightarrow \mathit{SymbObj} && (7.31)\\
\text{Scene Upper Bound} &\hookrightarrow \mathit{SymbObjs} && (7.32)\\
\text{Set of Scenes} &\hookrightarrow \mathit{SetScenes} \subseteq \wp(\mathit{Scene}) && (7.33)\\
\text{Set of Transitions} &\hookrightarrow \mathit{SetTrans} \subseteq \wp(\mathit{Transitions}) && (7.34)\\
\text{Root Scene} &\hookrightarrow \mathit{RootScene} \subseteq \mathit{SymbObj} && (7.35)\\
\text{Output Scene} &\hookrightarrow \mathit{OutScene} \subseteq \mathit{SymbObj} && (7.36)\\
\text{Set of Exit Arcs} &\hookrightarrow \mathit{SetExitArcs} \subseteq \wp(\mathit{Arc}) && (7.37)\\
\text{Set of Acess Arcs} &\hookrightarrow \mathit{SetAccArcs} \subseteq \wp(\mathit{Arc}) && (7.38)\\
\text{Set of Transition Types} &\hookrightarrow \mathit{SetTransTypes} \subseteq \wp(\mathit{TransitionType}) && (7.39)\\
\text{Set of Arc Types} &\hookrightarrow \mathit{SetArcTypes} \subseteq \wp(\mathit{AcType}) && (7.40)\\
\text{Set of Arc Constraints} &\hookrightarrow \mathit{SetArcCnstrs subseteq} \wp(\mathit{ArcCnstr}) && (7.41)\\
\text{Set of Scene Bound} &\hookrightarrow \mathit{SetSceneBound} \subseteq \wp(\mathit{SceneBound}) && (7.42)\\
\text{Performative Structure} &\hookrightarrow \mathit{PerformStrct} = \mathit{SetScenes} \times \mathit{SetTransitions} \times \mathit{RootScene} \times \mathit{OutputScene}\\
&\times \mathit{SetExitArcs} \times \mathit{SetAccArcs} \times \mathit{SetTransTypes} \times \mathit{ArcTypes}\\
&\times \mathit{SetArcConstr} \times \mathit{SetSceneBounds} && (7.43)
\end{aligned}$$

Figure 7.8: *Performative Structures* of Electronic Institutions re-typed in *TPO*.

$$\begin{aligned}
\text{Normative Rule} &\hookrightarrow \mathit{NormRule} \subseteq \mathit{SymbObj} && (7.44)\\
\text{Electronic Institution} &\hookrightarrow \mathit{EI} = \mathit{PerformStrct} \times \wp(\mathit{InstRole}) \times \wp(\mathit{NormRule}) \times \mathit{StaticSepDut} && (7.45)
\end{aligned}$$

Figure 7.9: *Electronic Institutions* re-typed with *TPO*.

7.4 *OperA*

The *OperA* model originated from Virginia Dignum's thesis [46]. We describe it as shown in Fig. 7.10 (adapted from [46](p.55)):

- the *social* and the *interaction models* are the *concrete* parts of the model, constituted by agents that enact roles and that interact with each other;
- the *organizational model* is the *abstract* part of the model, constituted by the components that determine how the social and interaction models are realized;
- we say that the *social model* implements the *social structure* and that the *interaction model* implements the *interaction structure*;
- we say that the *binding* between the social model and the interaction model implements the binding between the social structure and the interaction structure;
- the *social structure* specifies how the *roles* of the organizational structure relate to each other;
- the *interaction structure* specifies how the agents that enact the roles specified by the social structure should interact with each other, in terms of sequences of *scenes* that realize the goals of the system;
- the *communicative structure* specifies the *communication language* and the *ontologies* to be used by the agents that enact the roles, when communicating with each other;
- the *normative structure* specifies the *norms* that agents should comply to, when enacting the roles of the organization, which are norms that concern: the roles, the interactions inside the scenes, and the transitions between scenes.

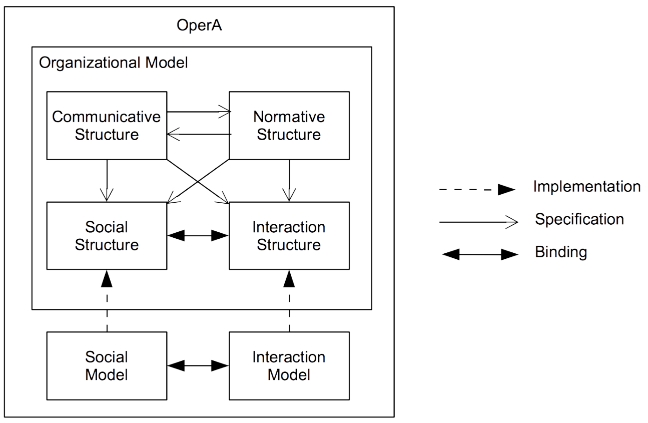

Figure 7.10: The structure of the OperA model.

Clearly, the *OperA* model builds on the *EI* model. We describe each of its components as follows.

7.4.1 The *Social Structure*

The *social structure* determines the *roles*, *relations between roles*, and *groups of roles* of the agent society.

The *Roles*

Roles are “abstract representations of a policy, service or a function” [46](p.59). The elements of a role are the following:

- the set of desired *results* (final and intermediate) of the enaction of the role, given in terms of logical expressions (said to be “landmarks” of the activity of the role);
- the *rights* of the role, also given in terms of logical expressions;
- the set of *norms* (given by deontic logical expressions) that apply to the role and that are independent of interactions the role may be involved in;
- an indication of the type of agent (*external* or *institutional*) that can enact the role: like in *EI*s, institutional roles can only be enacted by agents controlled by the organization, while external roles can be enacted by any kind of agent, possibly according to some *access rules* specified by the organization[7].

Clearly, as all components of the definition of a role in *OperA* are symbolic elements. *OperA* roles exist only as elements of *specifications* of agent societies, at “specification time”, not as their *effective components*, existent at “run time”.

As a possible exception to such condition of “unreality” of roles and their components in agent societies (i.e., the condition of their existence only as “descriptive devices”), the components of roles (specially their *rights*, *norms* and *access rules*) may happen to exist as *symbolic items*, in the symbolic environments of systems realized according to the model.

Assuming the realization of *OperA* roles in the organizational structure of agent societies, and of *OperA* role rights and norms in the symbolic environment, we can re-type *OperA* roles with *TPO* in the following way:

$$\begin{aligned}
\text{Role Landmark} &\hookrightarrow \mathit{RoleLndmrk} \subseteq \mathit{SymbObj}\\
\text{Role Right} &\hookrightarrow \mathit{RoleRight} \subseteq \mathit{SymbObj}\\
\text{Role Norm} &\hookrightarrow \mathit{RoleNorm} \subseteq \mathit{SymbObj}\\
\text{Role Type} &\hookrightarrow \mathit{RoleType} \subseteq \mathit{SymbObj}\\
\text{Role} &\hookrightarrow \mathit{Rol} = \wp(\mathit{RoleLndmrk}) \times \wp(\mathit{RoleRight}) \times \wp(\mathit{RoleNorm}) \times \mathit{RoleType}
\end{aligned}$$

The *Groups*

As in *AGR* (see Chaps. 7.1 and 7.2), in *OperA* a group is a means to refer to a set of roles, not an active entity in the agent society. Also as in *AGR*, *OperA* groups can specify *norms* for the agents that enact roles that belong to the group. And the *group norms* of a group are expected to be consistent with the *role norms* of the roles of the group.

We re-type *OperA* groups with *TPO* as:

$$\begin{aligned}
\text{Group Norm} &\hookrightarrow \mathit{GroupNorm} \subseteq \mathit{SymbObj}\\
\text{Group} &\hookrightarrow \mathit{Group} = \wp(\mathit{Rol}) \times \wp(\mathit{GroupNorm})
\end{aligned}$$

The *Dependence Relations*

Dependence and *power relations* between roles can be established, determining how one role depends on another role, for the realization of its objectives, and how it influences the realization of the other role’s objectives by means of *goal delegations* and *adoptions*.

Dependence relations are of the form $r_1 \Phi_\gamma r_2$, where:

- Φ denotes the dependence relation;
- r_1 is the dependent role and r_2 is the role on which r_1 depends;

[7]Norms, however, are not *regimented*, for the *OperaA* platform is not expected to *block* prohibited actions by the agents. *Enforcement* of norms on the basis of sanctions is, instead, the way normativity is to be achieved in the systems realized according to the model.

- γ is the goal regarding which r_1 depends on r_2.

We re-type *OperA* dependence relations between roles with *TPO* as:

$$\begin{aligned}
\text{Goal} &\hookrightarrow \mathit{Goal} \subseteq \mathit{SymbObj} \\
\text{Dependence Relation} &\hookrightarrow \mathit{DepRel} = \mathit{Rol} \times \mathit{Rol} \times \mathit{Goal}
\end{aligned}$$

The Whole Social Structure

Overall, the re-typing of *OperA*'s *social structure* gives:

$$\begin{aligned}
\text{Role Landmark} &\hookrightarrow \mathit{RoleLndmrk} \subseteq \mathit{SymbObj} \\
\text{Role Right} &\hookrightarrow \mathit{RoleRight} \subseteq \mathit{SymbObj} \\
\text{Role Norm} &\hookrightarrow \mathit{RoleNorm} \subseteq \mathit{SymbObj} \\
\text{Role Type} &\hookrightarrow \mathit{RoleType} \subseteq \mathit{SymbObj} \\
\text{Role} &\hookrightarrow \mathit{Rol} = \wp(\mathit{RoleLndmrk}) \times \wp(\mathit{RoleRight}) \times \wp(\mathit{RoleNorm}) \times \mathit{RoleType} \\
\text{Group Norm} &\hookrightarrow \mathit{GroupNorm} \subseteq \mathit{SymbObj} \\
\text{Group} &\hookrightarrow \mathit{Group} = \wp(\mathit{Rol}) \times \wp(\mathit{GroupNorm}) \\
\text{Goal} &\hookrightarrow \mathit{Goal} \subseteq \mathit{SymbObj} \\
\text{Dependence Relation} &\hookrightarrow \mathit{DepRel} = \mathit{Rol} \times \mathit{Rol} \times \mathit{Goal} \\
\text{Social Structure} &\hookrightarrow \mathit{SocStrct} = \wp(\mathit{Group}) \times \wp(\mathit{DependenceRelation})
\end{aligned}$$

7.4.2 The *Interaction Structure*

In *OperA*, like in *EI*s, the *activities* performed by the *role enacting agents* are organized in *scenes*, whose functionings follow *scene scripts*.

However, as *OperA* does not contemplate explicitly neither the concept of *action* nor the concept of *interaction*, scene scripts are defined in terms of inter-relations of goals to be achieved by the agents that enact roles that belong to the scenes. Besides, *scene norms* are also provided, to regulate the interactions of the agents that enact the scene roles. That is, in *OperA*, *scene scripts* are described by sets of *roles*, *goals* and *norms*, and relations between them.

On the other hand, as *scene scripts* are means to specify the way the goals of the organization are to be achieved, and such goals may require to be decomposed in sub-goals, scene scripts may have to be organized into sequences, which together achieve the societies goals.

Similarly to the function of *performative structures* is *EI*s, the *interaction structure* of the organizational structure of an *OperA* model is the means for *sequencing of the scene scripts* in that model. This sequencing is specified by means of relations between scenes, called *scene transitions*. Sequences of scene transitions constitute *transition scripts*. *Initial* and *final scenes* of each interaction structure should be specified.

Also, *enabling relations* are established between roles, to guide such transitions. Both *role enabling relations* and *scene transitions* support parallelism in the enacting of roles and in the execution of scenes. Upper bounds in the number of instances of scenes that may occur simultaneously may be specified. *Conflict relations* between roles may be defined, to prevent agents to simultaneously enact certain roles, in certain scenes. Here, we treat these relations as *scene norms*.

We re-type *OperA scene scripts* with *TPO* in the following way:

$$\begin{aligned}
\text{Scene Role} &\hookrightarrow \mathit{SceneRole} \subseteq \mathit{SymbObj} \\
\text{Scene Goal} &\hookrightarrow \mathit{SceneGoal} \subseteq \mathit{SymbObj} \\
\text{Scene Goals Ordering} &\hookrightarrow \mathit{SceneGoalsOrd} \subseteq \mathit{SceneGoal} \times \mathit{SceneGoal} \\
\text{Scene Norm} &\hookrightarrow \mathit{SceneNorm} \subseteq \mathit{SymbObj} \\
\text{Scene Script} &\hookrightarrow \mathit{SceneScript} = \wp(\mathit{SceneRole}) \times \wp(\mathit{SceneGoal}) \times \mathit{SceneGoalsOrd} \\
&\qquad \times \wp(\mathit{SceneNorm})
\end{aligned}$$

For simplicity, we omit here the analysis of the details of the scene transitions, involving scene connectors, etc. We treat scene transitions just as relations between scenes. Thus, we re-type *OperA interaction structures* with *TPO* in the following way:

$$\begin{aligned}
\text{Scene Transitions} &\hookrightarrow \mathit{SceneTrans} = \mathit{SceneScripts} \times \mathit{SceneScripts} \\
\text{Role Enabling Relation} &\hookrightarrow \mathit{RoleEnabRel} = \mathit{SceneRole} \times \mathit{SceneRole} \\
\text{Initial Scene} &\hookrightarrow \mathit{InitialScene} = \mathit{SceneScript} \\
\text{Final Scene} &\hookrightarrow \mathit{finalScene} = \mathit{SceneScript} \\
\text{Interaction Structure} &\hookrightarrow \wp(\mathit{SceneScript}) \times \mathit{SceneTrans} \times \mathit{RoleEnabRel} \\
&\quad\; \times \mathit{InitialScene} \times \mathit{FinalScene}
\end{aligned}$$

7.4.3 The *Normative Structure*

In *OperA*, the norms of an organization "define the rights and obligations of the agents in an organization, related to the roles they play, or to a particular area of activity" [46](p.72).

Norms express their meanings in concrete terms, referred to the structure and operation of the organization, by using the *ontology* and *communication framework* specified by the *communicative structure* of the organization (examined here in Sect. 7.4.4).

As mentioned above, norms are part of the specification of *roles*, *groups* and *scene scripts*. The re-typing of *OperA* norms with *TPO* is simply:

$$\text{Normative Structure} \hookrightarrow \mathit{NormStrct} = \wp(\mathit{RoleNorm}) \cup \wp(\mathit{GroupNorm}) \cup \wp(\mathit{SceneNorm}) \tag{7.46}$$

7.4.4 The *Communicative Structure*

The *communicative structure* specifies the *communication primitives* that the agents may use, when enacting roles. Communication primitives include *communicative acts*, for operations of communication, and *domain language*, for content representation.

Communicative acts relate two roles through the indication of the *content* that one may communicate to the other (in the form of an *expression* involving terms of an *ontology*), and the type of *performance* expected from the communicative act (in the form of the type of *performative* associated with the communicative act: request, inform, commit, and declare).

As such, communicative acts constitute a ternary relation. The re-typing of *OperA* communicative structure with TPO amounts to:

$$\begin{aligned}
\text{Communitactive Expression} &\hookrightarrow \mathit{CommExpr} \subseteq \mathit{SymbObj} \\
\text{Communicative Act} &\hookrightarrow \mathit{CommAct} = \mathit{SceneRole} \times \mathit{SceneRole} \times \mathit{CommExpr} \\
\text{Communicative Structure} &\hookrightarrow \mathit{CommStrct} = \wp(\mathit{CommunicativeAct})
\end{aligned}$$

7.4.5 The *Social Model*

The *social model* specifies, in *OperA*, the *role implementation relation*, by which agents implement directly organizational roles.

Such specification is given in terms of a *social contract* between the agent and the society (or one of its representatives), that is, a set of *agreements* about the conditions of the agent's participation in the society (expected behavior, duration of the participation, resources it may use, capabilities it should mobilize, monitorings and sanctions it should admit, etc.).

A social model is, then, "a set of social contracts mapping agents in [the population] to roles in OM" [46](p.80).

Social contracts in *OperA* are structures composed of:

- an *agent*, which enacts a role;
- the *role* enacted by the agent;

- a set of *contract clauses*, each contract clause given by a deontic expression, stating a permission, obligation or prohibition, related to the activity of the agent as enactor of the role.

Agents may have more than one social contract, in a society, one for each role it enacts. But agents enact roles only in the context of scenes, so the relation between agents, roles and scenes should be specified separately. That is done by means of the relation called *role enacting agent* which relates those three elements.

Re-typing *OperA*'s *social model* with *TPO* gives:

$$\begin{aligned}
\text{Social Contract Clause} &\hookrightarrow \mathit{SocCntrctClaus} \subseteq \mathit{SymbObj} \\
\text{Social Contract} &\hookrightarrow \mathit{SocCntrct} = \mathit{Agent} \times \mathit{Rol} \times \wp(\mathit{SocCntrctClaus}) \\
\text{Role Enacting Agent} &\hookrightarrow \mathit{RoleEnactAg} = \mathit{Agent} \times \mathit{Rol} \times \mathit{SceneScript} \\
\text{Social Model} &\hookrightarrow \mathit{SocModel} = \wp(\mathit{RoleEnactAg}) \times \wp(\mathit{SocialContract})
\end{aligned}$$

OperA specifies two operations concerned with the *dynamics* of the role implementation relation, namely, the operations of *setting up* of a social contract, and *ending* a social contract.

Each agent entering a society should go through a *start scene* in which a particular *facilitator agent*, member of the organization, negotiates with the entering agent the *social contract* that it will have to follow, while enacting a role in the society. Analogously, to leave a society, agents have to go through an *end scene*, where the contract is ended.

Start and *end* scenes are particular scenes that belong to every *OperA* OM, but their content vary from application to application.

7.4.6 The *Interaction Model*

The *interaction model* of a society is a set of *interaction scenes*, constituted in terms of particular agents, etc., by means of the *interaction contracts*, to which those agents must agree.

The *interaction model* constitutes the second step in the realization of an organization, after the establishment of the *social model*, constituted by the *social contracts*, which specify the realization of roles by means of agents.

An *interaction contract* determines the way an *interaction script* from the interaction structure is realized by an *interaction scene* of the interaction model. It specifies, for a given interaction scene, the operational results expected from the scene, the partnerships between the role enacting agents, the interaction protocol that the agents should follow in the interaction scene, and the social norms that apply during the interaction. The organizational goals and norms, from the organizational and interaction structure, are inherited by the instances of an interaction scene.

Interaction contracts are composed of:

- a set of *role enacting agents*, that is, agents that enact roles according to established social contracts;
- a set of *contract clauses*, a set of deontic expressions representing the interaction norms that apply to scene;
- the *protocol* to be followed by the role enacting agents, which specifies how the *landmarkss* of the interaction script should be achieved.

The re-typing of the *OperA* interaction model with *TPO* gives:

$$\begin{aligned}
\text{Interaction Contract Clause} &\hookrightarrow \mathit{InterCntrctClause} \subseteq \mathit{SymbObj} \\
\text{Landmark Protocol} &\hookrightarrow \mathit{LndmrkProtocol} \subseteq \wp(\mathit{SymbObj}) \\
\text{Interaction Contract} &\hookrightarrow \mathit{InterCntrct} = \mathit{RoleEnactAg} \times \mathit{RoleEnactAg} \\
&\qquad\qquad \times \wp(\mathit{InterCntrctClause}) \times \mathit{LndmrkProtocol} \\
\text{Interaction Model} &\hookrightarrow \mathit{InterModel} = \wp(\mathit{RoleEnactAg}) \times \wp(\mathit{InterCntrct})
\end{aligned}$$

7.4.7 *OperaA* Re-typed with *TPO*

Figures 7.11 and 7.12 summarize the above re-typing of the *OperA* model with *TPO*.

The Organizational Structure:

$$\begin{aligned}
\text{Role Landmark} &\hookrightarrow \mathit{RoleLndmrk} \subseteq \mathit{SymbObj} && (7.1)\\
\text{Role Right} &\hookrightarrow \mathit{RoleRight} \subseteq \mathit{SymbObj} && (7.2)\\
\text{Role Norm} &\hookrightarrow \mathit{RoleNorm} \subseteq \mathit{SymbObj} && (7.3)\\
\text{Role Type} &\hookrightarrow \mathit{RoleType} \subseteq \mathit{SymbObj} && (7.4)\\
\text{Role} &\hookrightarrow \mathit{Rol} = \wp(\mathit{RoleLndmrk}) \times \wp(\mathit{RoleRight})\\
&\qquad \times \wp(\mathit{RoleNorm}) \times \mathit{RoleType} && (7.5)\\
\text{Group Norm} &\hookrightarrow \mathit{GroupNorm} \subseteq \mathit{SymbObj} && (7.6)\\
\text{Group} &\hookrightarrow \mathit{Group} = \wp(\mathit{Rol}) \times \wp(\mathit{GroupNorm}) && (7.7)\\
\text{Goal} &\hookrightarrow \mathit{Goal} \subseteq \mathit{SymbObj} && (7.8)\\
\text{Dependence Relation} &\hookrightarrow \mathit{DepRel} = \mathit{Rol} \times \mathit{Rol} \times \mathit{Goal} && (7.9)\\
\text{Social Structure} &\hookrightarrow \mathit{SocStrct} = \wp(\mathit{Group}) \times \wp(\mathit{DependenceRelation}) && (7.10)
\end{aligned}$$

The Interaction Structure:

$$\begin{aligned}
\text{Scene Role} &\hookrightarrow \mathit{SceneRole} \subseteq \mathit{SymbObj} && (7.11)\\
\text{Scene Goal} &\hookrightarrow \mathit{SceneGoal} \subseteq \mathit{SymbObj} && (7.12)\\
\text{Scene Goals Ordering} &\hookrightarrow \mathit{SceneGoalsOrd} \subseteq \mathit{SceneGoal} \times \mathit{SceneGoal} && (7.13)\\
\text{Scene Norm} &\hookrightarrow \mathit{SceneNorm} \subseteq \mathit{SymbObj} && (7.14)\\
\text{Scene Script} &\hookrightarrow \mathit{SceneScript} = \wp(\mathit{SceneRole}) \times \wp(\mathit{SceneGoal}) \times \mathit{SceneGoalsOrd}\\
&\qquad \times \wp(\mathit{SceneNorm}) && (7.15)\\
\text{Scene Transitions} &\hookrightarrow \mathit{SceneTrans} = \mathit{SceneScripts} \times \mathit{SceneScripts} && (7.16)\\
\text{Role Enabling Relation} &\hookrightarrow \mathit{RoleEnabRel} = \mathit{SceneRole} \times \mathit{SceneRole} && (7.17)\\
\text{Initial Scene} &\hookrightarrow \mathit{InitialScene} = \mathit{SceneScript} && (7.18)\\
\text{Final Scene} &\hookrightarrow \mathit{finalScene} = \mathit{SceneScript} && (7.19)\\
\text{Interaction Structure} &\hookrightarrow \wp(\mathit{SceneScript}) \times \mathit{SceneTrans} \times \mathit{RoleEnabRel}\\
&\qquad \times \mathit{InitialScene} \times \mathit{FinalScene} && (7.20)
\end{aligned}$$

The Normative Structure:

$$\begin{aligned}
\text{Normative Structure} &\hookrightarrow \mathit{NormStrct} = \wp(\mathit{RoleNorm}) \cup \wp(\mathit{GroupNorm}) \cup \wp(\mathit{SceneNorm}) && (7.21)
\end{aligned}$$

The Communicative Structure:

$$\begin{aligned}
\text{Communitactive Expression} &\hookrightarrow \mathit{CommExpr} \subseteq \mathit{SymbObj} && (7.22)\\
\text{Communicative Act} &\hookrightarrow \mathit{CommAct} = \mathit{SceneRole} \times \mathit{SceneRole} \times \mathit{CommExpr} && (7.23)\\
\text{Communicative Structure} &\hookrightarrow \mathit{CommStrct} = \wp(\mathit{CommunicativeAct}) && (7.24)
\end{aligned}$$

Figure 7.11: *OperA* re-typed with *TPO* (part I)

The Social Model:

$$\text{Social Contract Clause} \hookrightarrow \mathit{SocCntrctClaus} \subseteq \mathit{SymbObj} \tag{7.25}$$

$$\text{Social Contract} \hookrightarrow \mathit{SocCntrct} = \mathit{Agent} \times \mathit{Rol} \times \wp(\mathit{SocCntrctClaus}) \tag{7.26}$$

$$\text{Role Enacting Agent} \hookrightarrow \mathit{RoleEnactAg} = \mathit{Agent} \times \mathit{Rol} \times \mathit{SceneScript} \tag{7.27}$$

$$\text{Social Model} \hookrightarrow \mathit{SocModel} = \wp(\mathit{RoleEnactAg}) \times \wp(\mathit{SocialContract}) \tag{7.28}$$

The Interaction Model:

$$\text{Interaction Contract Clause} \hookrightarrow \mathit{InterCntrctClause} \subseteq \mathit{SymbObj} \tag{7.29}$$

$$\text{Landmark Protocol} \hookrightarrow \mathit{LndmrkProtocol} \subseteq \wp(\mathit{SymbObj}) \tag{7.30}$$

$$\begin{aligned}\text{Interaction Contract} \hookrightarrow \mathit{InterCntrct} = {} & \mathit{RoleEnactAg} \times \mathit{RoleEnactAg} \\ & \times \wp(\mathit{InterCntrctClause}) \times \mathit{LndmrkProtocol}\end{aligned} \tag{7.31}$$

$$\text{Interaction Model} \hookrightarrow \mathit{InterModel} = \wp(\mathit{RoleEnactAg}) \times \wp(\mathit{InterCntrct}) \tag{7.32}$$

Figure 7.12: *OperA* re-typed with *TPO* (part II)

7.5 *MOISE+*

MOISE+ [47] (see also [48]) is a MAS organization model, proposed by Jomi Fred Hübner and partners, that allows the separate specification of three aspects of a MAS organization:

- the *structural* aspect, corresponding to the so-called *structural dimension* of the organization, and captured by its *structural specification*;
- the *functional* aspect, corresponding to the so-called *functional dimension* of the organization, and captured by its *functional specification*;
- the link between those two aspects, corresponding to the so-called *deontic dimension* of the organization, and captured by its *deontic specification.*

The structural and functional dimensions are allowed to be specified independently of each other. The deontic dimension is to be specified on the bases of those two specifications.

7.5.1 The *Structural Specification*

The *structural dimension* of *MOISE+* adopts the basic pattern established by the Aalaadin model (see Chap. 7.1), namely, constituting the organizational structure of MAS as a set of *roles* and *groups*.

As in the Aalaadin model, in the *MOISE+* model a role is just a *name* for one or more agents that undertake certain tasks, and relations to other agents, in the organization.

However, while Aalaadin relates roles to each other by means of *interaction processes* (determined by *protocols*), *MOISE+* relates roles to each other by means of certain types of *role relations*, which do not involve interaction processes, necessarily, but which restrict the behaviors that the agents that play them may perform in the organization.

In addition, *MOISE+* defines a special type of role, called *abstract role*, whose purpose is to support a *specialization relation*, and a corresponding operation of *inheritance of properties*, between roles.

Besides the *specialization* relation, other types of role relations are defined in *MOISE+*, affecting the way the agents that play the roles may interact with each other:

- *acquaintance* relation, which allows the agents that play roles that are related by that relation to represent information about each other (identifiers, etc.);

- *communication* relation, which allows the agents that play roles that are related by that relation to communicate with each other;

- *authority* relation, which establishes a relation of authority between the agents that play roles that are related by that relation.

Clearly, although treated just as "links" between the agents that they relate, such relations also impose *norms* on the agents: for instance, norms of *permission* of communication and norms of *authorization.*

In addition, the *MOISE+* model defines a *compatibility relation* between roles, restricting the set of roles that an agent may play, simultaneously, in the organization.

Also, although a *group* is presented conceptually as capable of "operating as if it was a single entity" [47](p.44), what happens is that, in their formal definition, groups are constituted just as sets of roles, without behaviors and interactions of their own.

Differently from the concept of group in AALAADIN, however, groups in *MOISE+* can be recursively structured in terms of other groups. But, accordingly, the sub-groups of a group do not constitute operational units inside that group, due to their also being without behaviors and interactions of their own.

Formally, a group is a structure $G = (R, SG, L_{intra}, L_{inter}, C_{intra}, C_{inter}, np, ng)$ where:

- R is the set of roles that constitute the group;

- SG is the set of sub-groups of the group;

- L_{intra} and L_{inter} are the sets of internal and external relations (links) of the group, respectively;

- C_{intra} and C_{inter} are the sets of internal and external relations of compatibility of roles, intra and inter groups, respectively (the latter called, here, simply *group compatibility*);

- np is a function determining the cardinality of the set of agents allowed to implement each role of the group;

- ng is a function determining the cardinality of the set of agents allowed to implement each sub-group of the group.

The *structural specification* of MAS organizations is formally given as structures of the form $SS = (RG, R, \sqsubset)$ where:

- RG is the set of *root groups* of the organization, that is, groups that are not sub-groups of other groups;

- R is the set of all *roles* of the organization, with at least one role, namely, R_{soc}, the root of the specialization relation;

- $\sqsubset$ is the relation of specialization between roles, which determines their inheritance of properties.

We re-type the *MOISE+* organizational model with *TPO* in the following way:

- a *MOISE+ organization* is typed with an *organization unit network*;

- a *MOISE+ group* is typed with an *organization unit*;

- a *role* is typed with an *organizational role*;

- a *role link* is typed with an *organizational role relation* and an *organizational norm*;

- a *role compatibility relation* is typed with an *organizational role relation* and an *organizational norm*;

- a *group link* is typed with an *organizational unit relation* and an *organizational norm*;

- a *group compatibility relation* is typed with an *organizational unit relation* and an *organizational norm*;

- a delimitation of *role cardinality* is typed with an *organizational role property*;

- a delimitation of *group cardinality* is typed with an *organization unit property*;

- the *role specialization relation* and the *role property inheritance rule* are typed with an *organizational role relation* and an *organizational norm*, respectively.

Formally, we have the following re-typing of the *MOISE+ structural dimension* with the *TPO* type system:

The Structural Specification:

$$\begin{aligned}
\text{Role} &\hookrightarrow \mathit{Rol} \subseteq \mathit{SymbObj} \\
\text{Role Norm} &\hookrightarrow \mathit{RoleNrm} \subseteq \mathit{SymbObj} \\
\text{Group} &\hookrightarrow \mathit{Group} = \wp(\mathit{Role}) \\
\text{Group Population} &\hookrightarrow \mathit{GroupPop} = \wp(\mathit{Group}) \\
\text{Organizational Norm} &\hookrightarrow \mathit{OrgNrm} \subseteq \mathit{SymbObj} \\
\text{Group Norm} &\hookrightarrow \mathit{GroupNrmSymbObj} \\
\text{Intra Link} &\hookrightarrow \mathit{IntraLink} = \mathit{Rol} \times \mathit{Rol} \times \mathit{OrgNrm} \\
\text{Inter Link} &\hookrightarrow \mathit{InterLink} = \mathit{Rol} \times \mathit{Rol} \times \mathit{OrgNrm} \\
\text{Intra Compatibility Relation} &\hookrightarrow \mathit{IntraCompatRel} = \mathit{Rol} \times \mathit{Rol} \times \mathit{RoNrm} \\
\text{Inter Compatibility Relation} &\hookrightarrow \mathit{InterCompRel} = \mathit{Rol} \times \mathit{Rol} \times \mathit{GroupNrm} \\
\text{Role Cardinality} &\hookrightarrow \mathit{RoleCard} \subseteq \mathit{SymbObjProp} \\
\text{Group Cardinality} &\hookrightarrow \mathit{GroupCard} \subseteq \mathit{SymbObjProp} \\
\text{Group} &\hookrightarrow \mathit{Group} = \mathit{GroupPop} \times \wp(\mathit{IntraLink}) \times \wp(\mathit{intraCompatRel}) \\
&\quad \times \wp(\mathit{RoleCard}) \\
\text{Organization} &\hookrightarrow \mathit{GroupPop} \times \wp(\mathit{interLink}) \times \wp(\mathit{InterCompatRel}) \\
&\quad \times \wp(\mathit{GroupCard})
\end{aligned}$$

7.5.2 The *Functional Specification*

The *functional dimension* of the *MOISE+* organizational model deals with the *aims* that a multiagent system has to attain. It specifies such aims by means of a set of so-called *social schemes*, each social scheme being a set of *goals*, structured as a *compound plan.* The goals of a social scheme are taken to be goals of the whole multiagent system.

Roles are assigned goals in terms of ordered sets of goals, called *missions.* The goals that constitute a mission of a role may occupy any of the locations of goals in a social scheme, so that the *assignment of missions to roles* amounts to a *distribution* of the compound plan of a social scheme among the roles that participate in its execution.

Goals assigned to roles by means of missions become individual goals, that is, goals of the agents that play those roles. And individual agents commit to the execution of missions by committing themselves to the playing of roles: the missions assigned to a given role become missions of the agent that commits to playing that role.

Global goals are endowed with three kinds of attributes:

- *satisfaction level*, indicating how much of the global goal has already been satisfied, at a given time;

- *allocation level*, indicating if there is already an agent allocated to the achievement of the goal, at a given time;

- *activation level*, indicating if all the pre-conditions for the achievement of the goal have already been satisfied, at a given time.

Social schemes are structured in terms of *goal trees*, with a single goal as the root of the tree, said to be the *social goal of the scheme*, which we denote by g_{soc}.

Three operators, which allow the composition of goals, and two properties constitute the structure of social schemes:

- *sequential* composition: $g_1 = g_2, g_3$;
- *choice*: $g_1 = g_2 \mid g_3$;
- *parallelism*: $g_1 = g_2 \mid\mid g_3$;
- *success probability*: the expected probability of success of the plan;
- *success rate*: the historical rate of success of the plan, updated each time the plan is executed.

As sub-goals that are components of goal sequences constitute pre-conditions for the achievement of other sub-goals, social schemes provide means for the coordination of the agents committed to the missions that involve the goals of the schemes.

Goals that occupy the leaves of a social scheme do not have indications, in the social scheme, of how to be achieved. It is up to the agents committed to such goals to find ways to satisfy them.

Formally, a *social scheme* is a structure $Sch = (G, P, M, mo, nm)$ where:

- G is the set of all goals of the scheme, including a root goal (g_{soc});
- P is the set of plans that structure the scheme, one plan for each goal;
- M is the set of missions, spread over the set of goals, with a mission m_{soc}, corresponding to the achievement of the social goal g_{soc};
- $mo : M \to \wp(G)$ is the function that assign sets of goals to missions;
- nm is a function that determines the minimum and maximum number of agents that may commit to a given mission.

The root goal (g_{soc}) serves the purpose of structuring several independent and partial social schemes, possibly related to separate goals of separate parts of the organization, with one single social scheme. The agents that commit themselves to g_{soc} get the *right to start* the execution of the scheme.

A *preference order* ($\prec$) may be imposed on missions that are committed to the same agent. That is, if an agent commits to two missions, and one of them is preferred, in comparison to the other, then the agent is supposed to give preference to the goals of the more preferred mission, whenever possible.

Formally, a *functional specification* is given, then, by a structure $FS = (S, \prec)$ where:

- S is a social scheme;
- $\prec$ is a preference order imposed on the missions of $\prec$.

The re-typing of the *functional dimension* of *MOISE+* may proceed as follows:

- *social schemes* and *plans* are typed as *symbolic object nets*, related to each other inside the networks by appropriate *symbolic object relations*;
- *goals* are typed as *symbolic objects*;
- *missions* are typed as sets of *symbolic objects*;
- *success probabilities* and *success rates* of goals, and *commitment cardinalities* of missions are typed as *symbolic object properties*;
- *preference orders* between missions are typed as *symbolic object relations.*

Formally, the re-typing of *MOISE+ functional specifications* with the *TPO type system* is given by:

$$\begin{aligned}
\text{Goal} &\hookrightarrow \mathit{Goal} \subseteq \mathit{SymbObj}\\
\text{Sequential Operator} &\hookrightarrow \mathit{SeqOp} \subseteq \mathit{SymbObjRel}\\
\text{Choice Operator} &\hookrightarrow \mathit{ChoiceOp} \subseteq \mathit{SymbObjRel}\\
\text{Parallel Operator} &\hookrightarrow \mathit{ParOp} \subseteq \mathit{SymbObjRel}\\
\text{Plan} &\hookrightarrow \mathit{Plan} \subseteq \mathit{SymbObjNet}\\
\text{Mission} &\hookrightarrow \mathit{Mission} \subseteq \wp(\mathit{Goal})\\
\text{Success Probability} &\hookrightarrow \mathit{SuccProb} \subseteq \mathit{SymbObjProp}\\
\text{Success Rate} &\hookrightarrow \mathit{SuccRate} \subseteq \mathit{SymbObjProp}\\
\text{Mission Cardinality} &\hookrightarrow \mathit{MissionCard} = \mathit{Mission} \times \mathit{SymbObj}\\
\text{Set of Goals} &\hookrightarrow \mathit{SetGoals} = \wp(\mathit{Goal})\\
\text{Set of Plans} &\hookrightarrow \mathit{SetPlans} = \wp(\mathit{Plan})\\
\text{Set of Missions} &\hookrightarrow \mathit{SetMissions} = \wp(\mathit{Mission})\\
\text{Mission Preference Order} &\hookrightarrow \mathit{MissionPrefOrd} = \mathit{Mission} \times \mathit{Mission}\\
\text{Social Scheme} &\hookrightarrow \mathit{SocScheme} = \wp(\mathit{Goal}) \times \wp(\mathit{Plan}) \times \wp(\mathit{Mission}) \times \mathit{MissionPrefOrd}
\end{aligned}$$

7.5.3 The *Deontic Specification*

The *deontic dimension* links the *structural dimension* with the *functional dimension* by specifying, for each role, which missions an agent that plays the role has to achieve, or is permitted to achieve.

Both permissions and obligations to achieve goals may be qualified by temporal constraints, in the form of time intervals in which those permissions are valid, or those obligations should be completed.

Formally, a *deontic specification* is a structure of the form $D = (P, O)$ where:

- $P \subseteq R \times M \times TI$ is a set of *permissions*, where R is the set of *roles* of the structural dimension, M is the set of *missions* of the functional dimension and TI is the set of possible time intervals;

- $O \subseteq R \times M \times TI$ is a set of *obligations*, defined on the same domain as P, containing at least one obligation, namely, $obl(R_{soc}, m_{soc})$.

Notice that since R_{soc} is the root of the specialization relation between roles, and all the roles inherit properties from R_{soc}, all the roles are obligated to the mission m_{soc}, which concerns the root goal of the social scheme, g_{soc}.

We re-type the *deontic specification* with *TPO* as follows:

$$\begin{aligned}
\text{Time} &\hookrightarrow T\\
\text{Permission} &\hookrightarrow \mathit{Permission} = \mathit{Rol} \times \mathit{Mission} \times \wp(T)\\
\text{Obligation} &\hookrightarrow \mathit{Obligation} = \mathit{Rol} \times \mathit{Mission} \times \wp(T)\\
\text{Organizational Norm} &\hookrightarrow \mathit{OrgNorm} = \mathit{Permission} \cup \mathit{Obligation}\\
\text{Deontic Structure} &\hookrightarrow \mathit{DeonStrct} = \wp(\mathit{OrgNrm})
\end{aligned}$$

7.5.4 *MOISE+* Typed with *TPO*

Figure 7.13 summarize the above re-typing of the *MOISE+* model with *TPO*.

The Structural Specification:

$$\begin{aligned}
\text{Role} &\hookrightarrow \mathit{Rol} \subseteq \mathit{SymbObj} && (7.1)\\
\text{Role Norm} &\hookrightarrow \mathit{RoleNrm} \subseteq \mathit{SymbObj} && (7.2)\\
\text{Group} &\hookrightarrow \mathit{Group} = \wp(\mathit{Role}) && (7.3)\\
\text{Group Population} &\hookrightarrow \mathit{GroupPop} = \wp(\mathit{Group}) && (7.4)\\
\text{Organizational Norm} &\hookrightarrow \mathit{OrgNrm} \subseteq \mathit{SymbObj} && (7.5)\\
\text{Group Norm} &\hookrightarrow \mathit{GroupNrmSymbObj} && (7.6)\\
\text{Intra Link} &\hookrightarrow \mathit{IntraLink} = \mathit{Rol} \times \mathit{Rol} \times \mathit{OrgNrm} && (7.7)\\
\text{Inter Link} &\hookrightarrow \mathit{InterLink} = \mathit{Rol} \times \mathit{Rol} \times \mathit{OrgNrm} && (7.8)\\
\text{Intra Compatibility Relation} &\hookrightarrow \mathit{IntraCompatRel} = \mathit{Rol} \times \mathit{Rol} \times \mathit{RoNrm} && (7.9)\\
\text{Inter Compatibility Relation} &\hookrightarrow \mathit{InterCompRel} = \mathit{Rol} \times \mathit{Rol} \times \mathit{GroupNrm} && (7.10)\\
\text{Role Cardinality} &\hookrightarrow \mathit{RoleCard} \subseteq \mathit{Rol} \times \mathit{SymbObj} && (7.11)\\
\text{Group Cardinality} &\hookrightarrow \mathit{GroupCard} \subseteq \mathit{GroupSymbObj} && (7.12)\\
\text{Group} &\hookrightarrow \mathit{Group} = \mathit{GroupPop} \times \wp(\mathit{IntraLink}) \times \wp(\mathit{intraCompatRel})\\
&\quad \times \wp(\mathit{RoleCard}) && (7.13)\\
\text{Organization} &\hookrightarrow \mathit{GroupPop} \times \wp(\mathit{interLink}) \times \wp(\mathit{InterCompatRel})\\
&\quad \times \wp(\mathit{GroupCard}) && (7.14)
\end{aligned}$$

The Functional Specification:

$$\begin{aligned}
\text{Goal} &\hookrightarrow \mathit{Goal} \subseteq \mathit{SymbObj} && (7.15)\\
\text{Sequential Operator} &\hookrightarrow \mathit{SeqOp} \subseteq \mathit{SymbObjRel} && (7.16)\\
\text{Choice Operator} &\hookrightarrow \mathit{ChoiceOp} \subseteq \mathit{SymbObjRel} && (7.17)\\
\text{Parallel Operator} &\hookrightarrow \mathit{ParOp} \subseteq \mathit{SymbObjRel} && (7.18)\\
\text{Plan} &\hookrightarrow \mathit{Plan} \subseteq \mathit{SymbObjNet} && (7.19)\\
\text{Mission} &\hookrightarrow \mathit{Mission} \subseteq \wp(\mathit{Goal}) && (7.20)\\
\text{Success Probability} &\hookrightarrow \mathit{SuccProb} \subseteq \mathit{SymbObjProp} && (7.21)\\
\text{Success Rate} &\hookrightarrow \mathit{SuccRate} \subseteq \mathit{SymbObjProp} && (7.22)\\
\text{Mission Cardinality} &\hookrightarrow \mathit{MissionCard} = \mathit{Mission} \times \mathit{SymbObj} && (7.23)\\
\text{Set of Goals} &\hookrightarrow \mathit{SetGoals} = \wp(\mathit{Goal}) && (7.24)\\
\text{Set of Plans} &\hookrightarrow \mathit{SetPlans} = \wp(\mathit{Plan}) && (7.25)\\
\text{Set of Missions} &\hookrightarrow \mathit{SetMissions} = \wp(\mathit{Mission}) && (7.26)\\
\text{Mission Preference Order} &\hookrightarrow \mathit{MissionPrefOrd} = \mathit{Mission} \times \mathit{Mission} && (7.27)\\
\text{Social Scheme} &\hookrightarrow \mathit{SocScheme} = \wp(\mathit{Goal}) \times \wp(\mathit{Plan}) \times \wp(\mathit{Mission})\\
&\qquad \times \mathit{MissionPrefOrd} && (7.28)
\end{aligned}$$

The Deontic Specification:

$$\begin{aligned}
\text{Time} &\hookrightarrow T && (7.29)\\
\text{Permission} &\hookrightarrow \mathit{Permission} = \mathit{Rol} \times \mathit{Mission} \times \wp(T) && (7.30)\\
\text{Obligation} &\hookrightarrow \mathit{Obligation} = \mathit{Rol} \times \mathit{Mission} \times \wp(T) && (7.31)\\
\text{Organizational Norm} &\hookrightarrow \mathit{OrgNorm} = \mathit{Permission} \cup \mathit{Obligation} && (7.32)\\
\text{Deontic Structure} &\hookrightarrow \mathit{DeonStrct} = \wp(\mathit{OrgNrm}) && (7.33)
\end{aligned}$$

Figure 7.13: *MOISE+* typed with *TPO*

7.6 *JaCaMo*

The *JaCaMo* model [49] integrates three previously available models: the *Jason* agent model [50], the *Moise+* organizational model[8], and the *CArtAgO* environment model [51].

7.6.1 *Jason*

Jason is an agent platform whose *agent model* evolved from the *BDI agent model.* The essential components of an agent, from the *JaCaMo* perspective, are [49](p.751):

- *beliefs*, i.e., the set of information that the agent assumes to be true about the state of its exterior, as well as about the state of its interior;
- *goals*, i.e., the set of states (exterior as well as interior) that the agent intends to achieve;
- *plans*, i.e., the set of structured sets of actions that the agent may put to work to achieve its goals;
- *actions*, the set of either internal or external *primitive plans* that the agent may execute to achieve *basic goals*, that is, goals that do not decompose into simpler goals;
- *events*, the set of possible *changes* either in the *current set of belief*s or in the *current set of goals*, which trigger the execution of plans.

An *agent action* is a one-shot, concrete *behavior* that an agent may realize, at any time. Events, being changes in sets of beliefs or sets of goals, are *relations* between sets of beliefs, or relations between sets of goals. All the other elements are of a *symbolic character* and may be taken as properties that agents may have, at each time. We thus re-type the *Jason agent model* with *TPO* as:

$$\begin{aligned}
\text{Agent} &\hookrightarrow \mathit{Agent}\\
\text{Agent Belief} &\hookrightarrow \mathit{AgentBel} \subseteq \mathit{AgProp}\\
\text{Agent Goal} &\hookrightarrow \mathit{AgentGoal} \subseteq \mathit{AgProp}\\
\text{Agent Plan} &\hookrightarrow \mathit{AgentPlan} \subseteq \mathit{AgProp}\\
\text{Agent Action} &\hookrightarrow \mathit{AgentAction} \subseteq \mathit{AgBeh}\\
\text{Agent Event} &\hookrightarrow \mathit{AgentEvent} = \wp(\mathit{AgentBel} \times \mathit{AgentBel}) \cup \wp(\mathit{AgentGoal} \times \mathit{AgentGoal})\\
\text{Jason Agent} &\hookrightarrow \mathit{JasonAg} = \mathit{Agent} \times \wp(\mathit{AgentBel}) \times \wp(\mathit{AgentGoal})\\
&\qquad\qquad \times \wp(\mathit{AgentPlan}) \times \wp(\mathit{AgentAction}) \times \wp(\mathit{AgentEvent})
\end{aligned}$$

Taken as a whole, the set of agents running on a Jason platform constitutes the *population* of the system that is specified by the *JaCaMo* model whose execution is realized by that platform.

Thus, we re-type the population of *Jason* agents with *TPO* as:

$$\text{Jason Population} \hookrightarrow \mathit{JasonPop} = \wp(\mathit{JasonAgent})$$

7.6.2 *CArtAgO*

CArtAgO supports the development of *environments* for agent systems, by means of the reification of environments' objects in terms of the so-called *artifacts*, which from the perspective of the agents are characterized by the following features [19]:

- their *properties*, which agent can observe in them;
- the *operations*, which agents can perform on them;
- the *events* they may generate toward the agents, corresponding to the realization of some operation, by that agent or by another one.

[8]The reading of this section depends on the previous reading of Sect. 7.5.

From the agent society perspective (see Sect. I), artifacts are *material objects*.The contents of artifacts may, or may not, constitute *symbolic objects*, from the perspective of agent society.

We re-type artifacts with *TPO* as follows:

$$\text{Artifact} \hookrightarrow \mathit{Artifact} \subseteq \mathit{MatObj} \tag{7.34}$$
$$\text{Artifact Property} \hookrightarrow \mathit{ArtProp} \subseteq \mathit{MatObjProp} \tag{7.35}$$
$$\text{Artifact Operation} \hookrightarrow \mathit{ArtOp} \subseteq \mathit{MatObj} \times \mathit{MatObj} \tag{7.36}$$
$$\text{Artifact Event} \hookrightarrow \mathit{ArtEv} \subseteq \mathit{MatObjBeh} \tag{7.37}$$

where:

- *operations* are typed as one-shot artifact *behaviors*;
- *events* are typed as relations expressing modifications in the configurations of *properties* of the artifacts.

In *JaCaMo*, *CArtAgO* supports the development of an *organizational environment*, by means of the reification of *MOISE+* organizational entities (groups and their components, social schemes and their components) in terms of *organizational artifacts*, i.e., artifacts specially designed for such purpose [52].

With *CArtAgO*, *JaCaMo* is capable of not only making agents conform to an organization model which is taken in the conceptual sense and considered to exist at the specification level (as in all of the models that we have examined above), but is also capable of *realizing* such model, in terms of *organizational artifacts*, i.e., as a set of *concrete entities* that the agents can handle and that can actively interfere in the agents' workings, by means of the signals that they can generate.

Three main types of artifacts support such reification:

- *OrgBoards*, that keep track of the state of reification of the overall organization, noting the groups and social schemes that were reified;
- *GroupBoards*, which manage the life-cycle of a group;
- *SchemeBoards*, which manage the execution of a social scheme.

On the basis of those three types of artifacts, and relations between them, *MOISE+* models can be reified in *JaCaMo*.

Notice that the *material environments* that *CArtAgO* can realize are application dependent, and do not participate in the definition of the *JaCaMo* organizational model. Notice also that the *organizational environments* provided by *CArtAgO* to *JaCaMo* comprise exactly the types of the *MOISE+* model that we typed in Chap. 7.5 (with the proviso that, as already mentioned, the organizational entities of the *MOISE+* are effectively reified in *JaCaMo*, as artifacts, and are not anymore just *symbolic entities* existing at the conceptual level).

7.6.3 The Conceptual Mappings Between the Three Models

A central concern of *JaCaMo* is the integration of the three models, of agents (*Jason*), environments (*CArtAgO*) and organizations (*MOISE+*). This is done by mapping concepts from each model into concepts of the other models.

The main mappings are the following:

- *operations* that can be performed on *artifacts* are made available as *actions* that *agents* may perform (the so-called *external* actions);
- *observable events* that *artifacts* may generate are allowed to produce *events* in *agents*, concerning the activation of plans;
- *observable properties* that *artifacts* can expose are made available as beliefs in *agents* that deliberate to *focus* their attention on such artifacts through a specific *focus* operation;

- *organizational goals* that belong to the *social scheme* of the organization are mapped into *individual goals* that *agents* may attempt to achieve[9].

We re-type such conceptual mappings in *TPO* in terms of a *subsumption of types*:

$$\begin{aligned} \mathit{ArtOp} &\subseteq \mathit{AgBeh} \\ \mathit{ArtEv} &\subseteq \mathit{AgEv} \\ \mathit{ArtProp} &\subseteq \mathit{AgBel} \\ \mathit{Goal} &\subseteq \mathit{AgentGoal} \end{aligned}$$

where it should be noticed that the types *ArtBeh*, *ArtEv* and *ArtProp* subsume the corresponding types for every type of *artifact* and so subsume, in particular, the corresponding types for *organizational* artifacts [49]. That is, they subsume the corresponding types for *roles* and *groups.*

7.6.4 *JaCaMo* Re-typed with *TPO*

Figures 7.14 to 7.16 put together the re-typing of the *Jason* agent model, the *CArtAgO* environment model and the *MOISE+* organizational model (repeated here, for convenience, from Chap. 7.5, but considering the *CArtAgO* reification of organizational entities), along with the re-typing of the *conceptual mappings* defined between them.

AGENT DIMENSION

$$\begin{aligned} \text{Agent} &\hookrightarrow \mathit{Agent} && (7.1) \\ \text{Agent Belief} &\hookrightarrow \mathit{AgentBel} \subseteq \mathit{AgProp} && (7.2) \\ \text{Agent Goal} &\hookrightarrow \mathit{AgentGoal} \subseteq \mathit{AgProp} && (7.3) \\ \text{Agent Plan} &\hookrightarrow \mathit{AgentPlan} \subseteq \mathit{AgProp} && (7.4) \\ \text{Agent Action} &\hookrightarrow \mathit{AgentAction} \subseteq \mathit{AgBeh} && (7.5) \\ \text{Agent Event} &\hookrightarrow \mathit{AgentEvent} = \wp(\mathit{AgentBel} \times \mathit{AgentBel}) \cup \wp(\mathit{AgentGoal} \times \mathit{AgentGoal}) && (7.6) \\ \text{Jason Agent} &\hookrightarrow \mathit{JasonAg} = \mathit{Agent} \times \wp(\mathit{AgentBel}) \times \wp(\mathit{AgentGoal}) \\ &\qquad \times \wp(\mathit{AgentPlan}) \times \wp(\mathit{AgentAction}) \times \wp(\mathit{AgentEvent}) && (7.7) \\ \text{Jason Population} &\hookrightarrow \mathit{JasonPop} = \wp(\mathit{JasonAgent}) && (7.8) \end{aligned}$$

Figure 7.14: *JaCaMo* re-typed with *TPO* (the *agent* dimension)

ENVIRONMENT DIMENSION

$$\begin{aligned} \text{Artifact} &\hookrightarrow \mathit{Artifact} \subseteq \mathit{MatObj} && (7.9) \\ \text{Artifact Property} &\hookrightarrow \mathit{ArtProp} \subseteq \mathit{MatObjProp} && (7.10) \\ \text{Artifact Operation} &\hookrightarrow \mathit{ArtOp} \subseteq \mathit{MatObj} \times \mathit{MatObj} && (7.11) \\ \text{Artifact Event} &\hookrightarrow \mathit{ArtEv} \subseteq \mathit{MatObjBeh} && (7.12) \end{aligned}$$

Figure 7.15: *JaCaMo* re-typed with *TPO* (the *environment* dimension)

[9]Such mapping is effected through the *SchemeBoard* artifact, when an agent registers in a *GroupBoard* artifact to enact a role and the agent accesses the *SchemeBoard* to acquire the goals that constitute the missions to which that role is committed.

CONCEPTUAL MAPPING

$$ArtOp \subseteq AgBeh \tag{7.13}$$
$$ArtEv \subseteq AgEv \tag{7.14}$$
$$ArtProp \subseteq AgBel \tag{7.15}$$
$$Goal \subseteq AgentGoal \tag{7.16}$$

Figure 7.16: *JaCaMo* re-typed with *TPO* (the *conceptual mappings* between dimensions)

7.7 Discussion: An Appreciation of *CMSC*

The *Conceptual Map for Social Coordination* model [17] (which we abbreviate here as *CMSC model*) originated from the Workshop on Models for Social Coordination, held in Veldhoven in 2014 [53], where the main research groups working on the subject of organizational models for multiagent systems met to discuss the possibilities of convergence of their different approaches. The *CMSC* model was introduced in that book as an attempt to spell out a *common denominator* for the models presented in the workshop by the various groups.

Figure 7.18 gives our view of the model. We remark the following, about the *CMSC* model and the diagram that pictures it:

- The double-line boxes indicate abstract entities whose purpose in that conceptual diagram is to support the inheritance of properties. The concrete entities are those pictured as single-line boxes.
- The central entity in the *CMSC* model is *Organization*, and the model adheres to the line of thought inaugurated by the AALAADIN model, and adopted by all other models that we have examined above. Such models take *Organization Theory* as the conceptual framework, considering an agent system to be an *organization*, existing for the purpose of achieving goals determined by the system designer.
- The *CMSC* model is neutral regarding the idea of the organizational entities that they conceptualize being real entities existent in agent systems, or descriptive items existent only at the conceptual level of description or specification of such systems.
- The *CMSC* model contemplates most the concepts that we have examined above, which were introduced in the research area of multiagent system organization since AALAADIN (and even before, with the work on Distributed Artificial Intelligence systems [54]), namely:
  - agent
  - role
  - norm
  - action
  - state
  - event
  - organization (i.e., the whole system)

but not concepts like, e.g., *group* and *environment*, which shows the difficulty in fitting, in the intersection of the various models that were examined, any organizational component that has an *heterogeneous internal structure* and that, thus, can not be treated as as having a *simple type* (the concept of *organization*, as mentioned in the list, enters the *CMSC* to refer to the whole system, not to any of its components).

Thus, even the central notion of *organization* is given no internal structure in the *CMSC* model, differently from concepts like *goal* and *activity*, which can easily be given internal structure,

ORGANIZATION DIMENSION
The Structural Specification:

$$\text{Role} \hookrightarrow \mathit{Rol} \subseteq \mathit{SymbObj} \tag{7.17}$$
$$\text{Role Norm} \hookrightarrow \mathit{RoleNrm} \subseteq \mathit{SymbObj} \tag{7.18}$$
$$\text{Group} \hookrightarrow \mathit{Group} = \wp(\mathit{Role}) \tag{7.19}$$
$$\text{Group Population} \hookrightarrow \mathit{GroupPop} = \wp(\mathit{Group}) \tag{7.20}$$
$$\text{Organizational Norm} \hookrightarrow \mathit{OrgNrm} \subseteq \mathit{SymbObj} \tag{7.21}$$
$$\text{Group Norm} \hookrightarrow \mathit{GroupNrmSymbObj} \tag{7.22}$$
$$\text{Intra Link} \hookrightarrow \mathit{IntraLink} = \mathit{Rol} \times \mathit{Rol} \times \mathit{OrgNrm} \tag{7.23}$$
$$\text{Inter Link} \hookrightarrow \mathit{InterLink} = \mathit{Rol} \times \mathit{Rol} \times \mathit{OrgNrm} \tag{7.24}$$
$$\text{Intra Compatibility Relation} \hookrightarrow \mathit{IntraCompatRel} = \mathit{Rol} \times \mathit{Rol} \times \mathit{RoNrm} \tag{7.25}$$
$$\text{Inter Compatibility Relation} \hookrightarrow \mathit{InterCompRel} = \mathit{Rol} \times \mathit{Rol} \times \mathit{GroupNrm} \tag{7.26}$$
$$\text{Role Cardinality} \hookrightarrow \mathit{RoleCard} \subseteq \mathit{Rol} \times \mathit{SymbObj} \tag{7.27}$$
$$\text{Group Cardinality} \hookrightarrow \mathit{GroupCard} \subseteq \mathit{GroupSymbObj} \tag{7.28}$$
$$\begin{aligned}\text{Group} \hookrightarrow\ & \mathit{Group} = \mathit{GroupPop} \times \wp(\mathit{IntraLink}) \times \wp(\mathit{intraCompatRel}) \\ & \times \wp(\mathit{RoleCard})\end{aligned} \tag{7.29}$$
$$\begin{aligned}\text{Organization} \hookrightarrow\ & \mathit{GroupPop} \times \wp(\mathit{interLink}) \times \wp(\mathit{InterCompatRel}) \\ & \times \wp(\mathit{GroupCard})\end{aligned} \tag{7.30}$$

The Functional Specification:

$$\text{Goal} \hookrightarrow \mathit{Goal} \subseteq \mathit{SymbObj} \tag{7.31}$$
$$\text{Sequential Operator} \hookrightarrow \mathit{SeqOp} \subseteq \mathit{SymbObjRel} \tag{7.32}$$
$$\text{Choice Operator} \hookrightarrow \mathit{ChoiceOp} \subseteq \mathit{SymbObjRel} \tag{7.33}$$
$$\text{Parallel Operator} \hookrightarrow \mathit{ParOp} \subseteq \mathit{SymbObjRel} \tag{7.34}$$
$$\text{Plan} \hookrightarrow \mathit{Plan} \subseteq \mathit{SymbObjNet} \tag{7.35}$$
$$\text{Mission} \hookrightarrow \mathit{Mission} \subseteq \wp(\mathit{Goal}) \tag{7.36}$$
$$\text{Success Probability} \hookrightarrow \mathit{SuccProb} \subseteq \mathit{SymbObjProp} \tag{7.37}$$
$$\text{Success Rate} \hookrightarrow \mathit{SuccRate} \subseteq \mathit{SymbObjProp} \tag{7.38}$$
$$\text{Mission Cardinality} \hookrightarrow \mathit{MissionCard} = \mathit{Mission} \times \mathit{SymbObj} \tag{7.39}$$
$$\text{Set of Goals} \hookrightarrow \mathit{SetGoals} = \wp(\mathit{Goal}) \tag{7.40}$$
$$\text{Set of Plans} \hookrightarrow \mathit{SetPlans} = \wp(\mathit{Plan}) \tag{7.41}$$
$$\text{Set of Missions} \hookrightarrow \mathit{SetMissions} = \wp(\mathit{Mission}) \tag{7.42}$$
$$\text{Mission Preference Order} \hookrightarrow \mathit{MissionPrefOrd} = \mathit{Mission} \times \mathit{Mission} \tag{7.43}$$
$$\begin{aligned}\text{Social Scheme} \hookrightarrow\ & \mathit{SocScheme} = \wp(\mathit{Goal}) \times \wp(\mathit{Plan}) \times \wp(\mathit{Mission}) \\ & \qquad \times \mathit{MissionPrefOrd}\end{aligned} \tag{7.44}$$

The Deontic Specification:

$$\text{Time} \hookrightarrow T \tag{7.45}$$
$$\text{Permission} \hookrightarrow \mathit{Permission} = \mathit{Rol} \times \mathit{Mission} \times \wp(T) \tag{7.46}$$
$$\text{Obligation} \hookrightarrow \mathit{Obligation} = \mathit{Rol} \times \mathit{Mission} \times \wp(T) \tag{7.47}$$
$$\text{Organizational Norm} \hookrightarrow \mathit{OrgNorm} = \mathit{Permission} \cup \mathit{Obligation} \tag{7.48}$$
$$\text{Deontic Structure} \hookrightarrow \mathit{DeonStrct} = \wp(\mathit{OrgNrm}) \tag{7.49}$$

Figure 7.17: *JaCaMo* re-typed with *TPO* (the *organization* dimension, see Fig. 7.13)

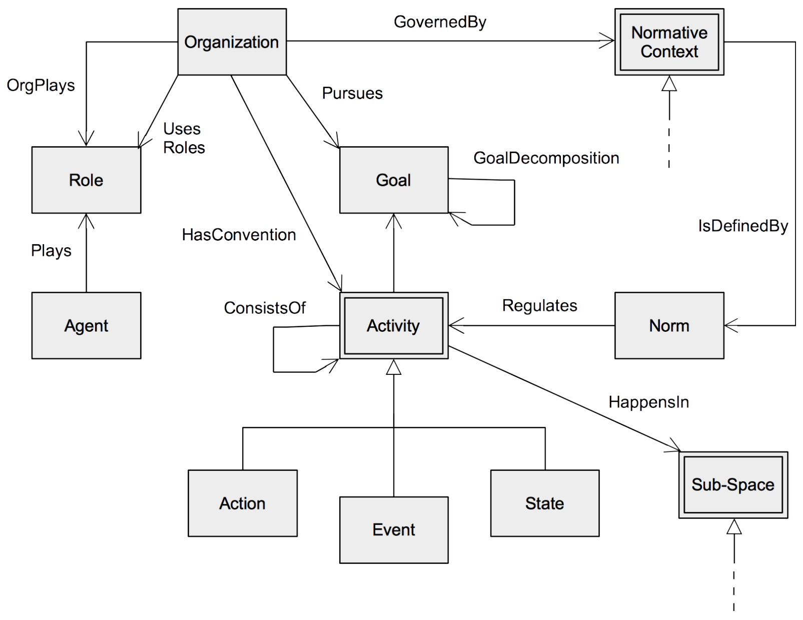

Figure 7.18: Overview of the *CMSC* model (adapted from [17](p.14)).

because they structure themselves recursively in an homogeneous way, that is, in terms of their own concepts.

We submit that the reason for such meagerness of the *CMSC* model is that its principle is ill conceived: it's not by looking at the *intersection* of a set of organizational models, it's not by looking for elements that are common to all of them, that a *fundamental organizational model* for agent systems will be found.

On the contrary, it is by looking at a *large* model, capable of accommodating each of those *particular* models as a sub-model, on the basis of a rich enough set of concepts, but without worrying about guaranteeing their compatibility with each other. For that compatibility cannot be achieved, given that the particular models are *particular views* of agent systems and, as particular views, need not be given in terms that are compatible with other views.

From a categorial point of view, that means that the right *common* model, capable of leveraging the cooperation between the variety of research groups working on the subject, and the unification of the variety of organizational models that they have developed, is not the *initial* object of the category of MAS organizational models, the least among them, but its *terminal* object, the *limit* of all such models.

We claim that the extended and consolidated *PopOrg* model, presented in this report in the form of the *TPO* type system and the *SML* modeling language, is one such limit model. The examination of some of the most important classical models, and their re-typing with *TPO*, which we presented in this Part I of the report, aimed to argue for that idea. Part II intends to show that the argument can also be argued by translating instances of those models to instances of the society modeling language *SML*, which was defined on the basis of *TPO*.

Part III

The Society Modeling Language *SML*

Chapter 8

General Features of *SML*

The language that we introduce in this Part II of the report (*SML* - Society Modeling Language) is an attempt to give a concrete representation for the societal model underlying the *TPO* type system introduced in Part I. As such, *SML* is proposed as a *descriptive* language, not as a *programming* language.

Figure 8.1 illustrate a view of the stages of development of descriptive languages:

- From a formal model of the kind of systems in question (agent societies, in our case), a formal type system may be defined.
- Such type system serves as the basis for the definition of an abstract descriptive language, formulating in general syntactical forms the relations between the various types of the type system.
- The abstract syntax, on its turn, serves as the basis for the development of both a formal semantical model for the descriptive language, and one or more extended syntaxes for its concrete presentation.
- The figure shows also the variety of comings and goings between the various stages of the development process.
- Notice that the type system and the semantical model are compatible with any form of concrete syntax that complies with the abstract syntax.

It seems that there has been no descriptive language proposed, up to now, concerning the description of full-fledged agent societies, in the sense we considered in Chap. I. *SML* seems to be - to the best of our knowledge - the first language proposed with such aim.

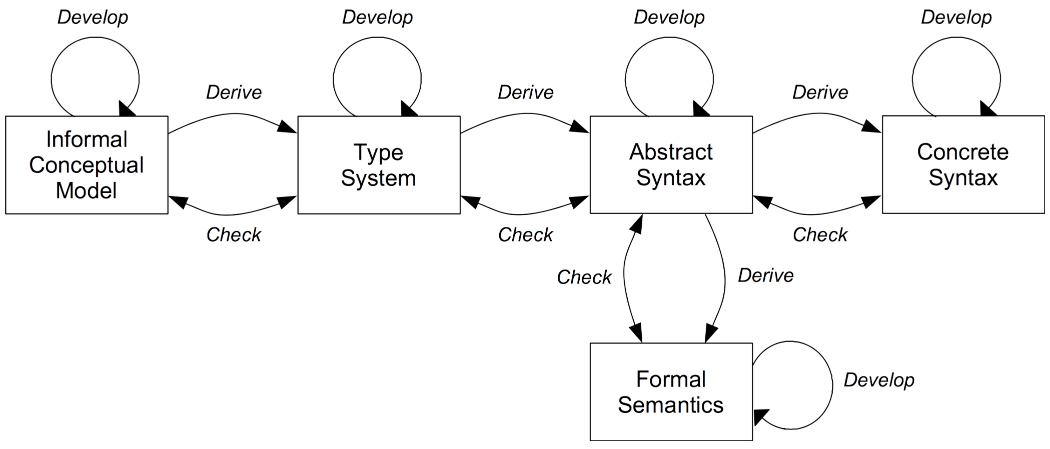

Figure 8.1: A view of the stages in the logical development descriptive languages.

Chapter 9

SML in Detail

In this chapter we introduce in sequence: the abstract syntax of *SML*, the abstract syntax of *SML* typed with *TPO*, and a concrete syntax for *SML*. We leave the presentation of the semantics of *SML* for further work.

9.1 The Abstract Syntax of *SML*

The *abstract* syntax of *SML* is presented in Fig. 9.1 and 9.2. Notice that:

- expressions in normal style are comments;
- expressions in **Bold** style are **non-terminals** of the grammar;
- expressions in SmallCaps style are terminals of the grammar (corresponding semantically to external objects and types);
- the expression '[E]' denotes an optional occurrence of the expression E;
- the expression '$[\mathrm{E}]^*$' denotes zero or more repetitions of the expression E;
- the expression '$[\mathrm{E}]^\omega$' denotes an infinite repetition of the expression E;
- the expression '1..n' denotes a range of integer values, from 1 to n (the most common use of such range of values is to indicate the range of possible instantiations of a given *SML* construct, in its implementation[1]);
- the operation of *sequencing of elements* is indicated by the simple juxtaposition of elements (e.g.: "$X\ Y\ \mathbf{Z}$" joins the terminal X with the terminal Y and the non-terminal **Z** in a single sequence);
- the symbol "ε" denotes the empty expression, which is the neutral element for the operation of sequencing of elements;
- tuples of elements are formed by separating elements by commas (e.g.: "$X, Y, \mathbf{Z}$" denotes the triple formed by X, Y and **Z**);
- grammar rules are of the form "head := body";
- the vertical succession of elements in the body of a rule indicates sequencing of those elements;
- the operator | is the choice operator;
- groups of rules are given a title and are separated from each other by an horizontal line.

[1]This feature is tentatively introduced here to overcome the lack of *syntactical* types in this draft version of *SML*.

Notice that, for generality, the abstract syntax of *SML* (and, in consequence, its concrete syntax) supports the modeling of agent societies with empty populations, that is, with no agent present in the society. This is to allow for the compatibility of *SML* with many existent models of agent societies (including most of the classical models that we examine in this report) which concern themselves only with the *organizational* structure of agent societies, being silent about their *populational* structures.

Also, notice that the overall syntactic style of the abstract and concrete syntax of *SML* are loosely inspired by the syntax of the *Python* programming language (see, e.g., `www.python.org`).

9.2 Typing the Abstract Syntax of *SML* with *TPO*

We type *SML* with the types of the *TPO* type system by associating *TPO* types with syntactical elements of the abstract syntax of *SML*. The typing result is given in Figs. 9.3 and 9.4, where *types* are denoted by expressions in the *Italic* style.

Notice that the typing of the *SML* abstract syntax serves as a modified form of the so-called *direct* technique of giving semantical content to a language, the technique adopted, e.g., for the first version of *OWL* (*Web Ontology Language*), as presented in [55]. The modification we introduce in the technique of *direct semantics* is simply the use of *set-based types* as semantic domains for the *abstract* syntactical structures.

Thus, in Figs. 9.3 and 9.4, an expression like $[\![E]\!]$ gives the meaning of the abstract expression E in terms of the *set-based type* of the objects that are denoted by the *instances* of E.

Notice, on the other hand, that in Figs. 9.3 and 9.4 we abuse the notation, semantically overloading the expression of abstract syntactical elements: if E denotes an abstract syntactical element, we may use E itself for its type, when we place E to the right side of the typing operator ":". As an example: if E_1 and E_2 are names of abstract syntactical categories, we denote by $E_2 : \wp(E_1)$ the typing of the abstract syntactical category E_2 with the power-set of the *type* of the abstract syntactical category E_1.

9.3 The Concrete Syntax of *SML*

In Figs. 9.5 and 9.6, we give the *concrete* of *SML*, in which we codify the examples of agent systems presented in Chap. 10. The notational conventions for the *concrete* syntax of *SLM* are the same of the *abstract* syntax, with the addition of:

- `terminal` expressions are denoted by fonts with the `Teletype` style, enclosed within `'quotes'`;
- statements of the *SML* language are structured in a vertical succession of elements, indicating the sequencing of those elements, with reserved words marked by *colons*;
- indentation is syntactically relevant, serving to denote statements and statement blocks;
- **identifiers** are denoted by expressions of the form **...ID**, in the **Bold Face** style.

Externals:

Event, Prop, Rel
Agent, SocRo, OrgRo
InPort, OutPort
MatObj, SymbObj

Process, Exchange Process:

Proc :=
  [Event]$^\omega$

ExchProc :=
  [Event Event]$^\omega$

Populational Structure:

AG :=
  [Agent]*

AGProp :=
  [Agent Prop]*

AGBeh :=
  [Agent **Proc**]*

AGInter :=
  [Agent Agent **ExchProc**]*

AGRel :=
  [Agent Agent Rel]*

Pop :=
  AG, **AGProp**, **AGBeh**, **AGInter**, **AGRel**

Sociability Structure:

SR :=
  [SocRo]*

SRProp :=
  [SocRo Prop]*

SRBeh :=
  [SocRo **Proc**]*

SRInter :=
  [SocRo SocRo **ExchProc**]*

SRNet :=
  [**SR**, **SRProp**, **SRBeh**, **SRInter**, **SRRel**]*

Soc :=
  SR, **SRProp**, **SRBeh**, **SRInter**, **SRRel**, **SRNet**

Organizational Structure:

OR :=
  [OrgRo]*

ORProp :=
  [OrgRo Prop]*

ORBeh :=
  [OrgRo **Proc**]*

ORInter :=
  [OrgRo OrgRo **ExchProc**]*

ORRel :=
  [OrgRo OrgRo Rel]

ORNet :=
  OR, **ORProp**, **ORBeh**, **ORInter**, **ORRel**

OU :=
  [**OrgUn**]*

OUProp :=
  [**OrgUn** Prop]*

OUBeh :=
  [**OrgUn** **Proc**]*

OUInter :=
  [**OrgUn** **OrgUn** **ExchProc**]*

OURel :=
  [**OrgUn** **OrgUn** Rel]*

Interf :=
  [**OrgRo**]*

Inp :=
  [InpPort]*

Out :=
  [OutPort]*

OrgUn :=
  ORNet, **Interf**, **Inp**, **Out**
  | **OUNet**, **Interf**, **Inp**, **Out**

OUNet :=
  [**OU**, **OUProp**, **OUBeh**, **OUInter**, **OURel**]*

Org :=
  OU, **OUProp**, **OUBeh**, **OUInter**, **OURel**, **OUNet**

Figure 9.1: Abstract Syntax of *SML* (part 1)

Material Environment:

MO :=
  [MatObj]*

MOProp :=
  [MatObj Prop]*

MOBeh :=
  [MatObj **Proc**]*

MOInter :=
  [MatObj MatObj **ExchProc**]*

MORel :=
  [MatObj MatObj Rel]*

MONet :=
  [**MO**, **MOProp**, **MOBeh**, **MOInter**, **MORel**]*

MEnv :=
  MO, **MOProp**, **MOBeh**, **MOInter**, **MORel**, **MONet**

Symbolic Environment:

SO :=
  [SymbObj]*

SOProp :=
  [SymObj Prop]*

SORel :=
  [SymbObj SymbObj Rel]*

SEnv :=
  SO, **SOProp**, **SORel**

Implementation Relations:

ImpSocPop :=
  [Agent SocRo]*

ImpOrgSoc :=
  [SocRo OrgRo]*

ImpOrgPop :=
  [Agent OrgRo]*

Imp :=
  ImpSocPop
  | **ImpOrgSoc**
  | **ImpOrgPop**

Access Links:

AccAgMObj :=
  [Agent MatObj **ExchProc**]*

AccAgSObj :=
  [Agent SymbObj **ExchProc**]*

AccOrgUnMObj :=
  [**OrgUn** MatObj **ExchProc**]*

AccOrgUnSObj :=
  [**OrgUn** SymbObj **ExchProc**]*

Acc :=
  AccAgMObj
  | **AccAgSObj**
  | **AccOrgUnMObj**
  | **AccOrgUnSObj**

Agent Society:

AgSoc :=
  Pop, **Soc**, **Org**, **MEnv**, **SEnv**, **Imp**, **Acc**

Figure 9.2: Abstract Syntax of *SML* (part 2)

Externals:

Event : *Event*
Prop : *Prop*
Rel : *Rel*
Agent : *Agent*
SocRo : *SocRo*
OrgRo : *OrgRo*
InpPort : *InpPort*
OutPort : *OutPort*
MatObj : *MatObj*
SymbObj : *SymbObj*

Process, Exchange Process:

Proc : *Proc*

ExchProc : *ExchProc*

Populational Structure:

AG : $\wp(Agent)$

AGProp : $\wp(Agent \times Prop)$

AGBeh : $\wp(Agent \times Proc)$

AGInter : $\wp(Agent \times Agent \times ExchProc)$

AGRel : $\wp(Agent \times Agent \times Rel)$

Pop : $\mathbf{AG} \times \mathbf{AGProp} \times \mathbf{AGBeh} \times \mathbf{AGInter} \times \mathbf{AGRel}$

Sociability Structure:

SR : $\wp(SocRo)$

SRProp : $\wp(SocRo \times Prop)$

SRBeh : $\wp(SocRo \times Proc)$

SRInter : $\wp(SocRo \times SocRo \times ExchProc)$

SRNetwork : $\mathbf{SR} \times \mathbf{SRProp} \times \mathbf{SRBeh} \times \mathbf{SRInter} \times \mathbf{SRRel}$

SRNet : $\wp(\mathbf{SRNetwork})$

Soc : $\mathbf{SR} \times \mathbf{SRProp} \times \mathbf{SRBeh} \times \mathbf{SRInter} \times \mathbf{SRRel} \times \mathbf{SRNet}$

Organizational Structure:

OR : $\wp(OrgRo)$

ORProp : $\wp(OrgRo \times Prop)$

ORBeh : $\wp(OrgRo \times Proc)$

ORInter : $\wp(OrgRo \times OrgRo \times ExchProc)$

ORRel : $\wp(OrgRo \times OrgRo \times Rel)$

OrgRoNet : $\mathbf{OR} \times \mathbf{ORProp} \times \mathbf{ORBeh} \times \mathbf{ORInter} \times]\mathbf{ORRel}$

ORNet : $\wp(\mathbf{OrgRoNet})$

OU : $\wp(OrgUn)$

OUProp : $\wp(OrgUn \times Prop)$

OUBeh : $\wp(OrgUn \times Proc)$

OUInter : $\wp(OrgUn \times OrgUn \times ExchProc)$

OURel : $\wp(OrgUn \times OrgUn \times Rel)$

OUNet : $\wp(\mathbf{OU} \times \mathbf{OUProp} \times \mathbf{OUBeh} \times \mathbf{OUInter} \times \mathbf{OURel})$

Interf : $\wp(OrgRo)$

Inp : $\wp(InpPort)$

Out : $\wp(OutPort)$

OrgUn : $\mathbf{ORNet} \times \mathbf{Interf} \times \mathbf{Inp} \times \mathbf{Out} \cup \mathbf{OUNet} \times \mathbf{Interf} \times \mathbf{Inp} \times \mathbf{Out}$

Org : $\mathbf{OU} \times \mathbf{OUProp} \times \mathbf{OUBeh} \times \mathbf{OUInter} \times \mathbf{OURel} \times \mathbf{OUNet}$

Figure 9.3: The Abstract Syntax of *SML* Typed with *TPO* (part 1)

Material Environment:

$\mathbf{MO} : \wp(\mathit{MatObj})$

$\mathbf{MOProp} : \wp(\mathit{MatObj} \times \mathit{Prop})$

$\mathbf{MOBeh} : \wp(\mathit{MatObj} \times \mathit{Proc})$

$\mathbf{MOInter} : \wp(\mathit{MatObj} \times \mathit{MatObj} \times \mathit{ExchProc})$

$\mathbf{MORel} : \wp(\mathit{MatObj} \times \mathit{MatObj} \times \mathit{Rel})]$

$\mathbf{MONet} : \mathbf{MO} \times \mathbf{MOProp} \times \mathbf{MOBeh} \times \mathbf{MOInter} \times \mathbf{MORel}$

$\mathbf{MEnv} : \mathbf{MO} \times \mathbf{MOProp} \times \mathbf{MOBeh} \times \mathbf{MOInter} \times \mathbf{MORel} \times \mathbf{MONet}$

Symbolic Environment:

$\mathbf{SO} : \wp(\mathit{SymbObj})$

$\mathbf{SOProp} : \wp(\mathit{SymbObj} \times \mathit{Prop})$

$\mathbf{SORel} : \wp(\mathit{SymbObj} \times \mathit{SymbObj} \times \mathit{Rel})$

$\mathbf{SEnv} : \mathbf{SO} \times \mathbf{SOProp} \times \mathbf{SORel}$

Implementation Relations:

$\mathbf{ImpSocPop} : \wp(\mathit{Agent} \times \mathit{SocRo})$

$\mathbf{ImpOrgSoc} : \wp(\mathit{SocRo} \times \mathit{OrgRo})$

$\mathbf{ImpOrgPop} : \wp(\mathit{Agent} \times \mathit{OrgRo})$

$\mathbf{Imp} : \mathbf{ImpSocPop} \cup \mathbf{ImpOrgSoc} \cup \mathbf{ImpOrgPop}$

Access Links:

$\mathbf{AccAgMObj} : \wp(\mathit{Agent} \times \mathit{MatObj} \times \mathit{ExchProc})$

$\mathbf{AccAgSObj} : \wp(\mathit{Agent} \times \mathit{SymbObj} \times \mathit{ExchProc})$

$\mathbf{AccOrgUnMObj} : \wp(\mathbf{OrgUn} \times \mathit{SymbObj} \times \mathit{ExchProc})$

$\mathbf{AccOrgUnSObj} : \wp(\mathbf{OrgUn} \times \mathit{SymbObj} \times \mathit{ExchProc})$

$\mathbf{Acc} : \mathbf{AccAgMObj} \cup \mathbf{AccAgSObj} \cup \mathbf{AccOrgUnMObj} \cup \mathbf{AccOrgUnSObj}$

Agent Society:

$\mathbf{AgSoc} : \mathbf{Pop} \times \mathbf{Soc} \times \mathbf{Org} \times \mathbf{MEnv} \times \mathbf{SEnv} \times \mathbf{Imp} \times \mathbf{Acc}$

Figure 9.4: The Abstract Syntax of *SML* Typed with *TPO* (part 2)

```
————————————————————
Externals:

EVENT, PROP, REL,
PROC,EXCHPROC,
AGENT, SOCRO, ORGRO,
INPPORT, OUTPORT,
MATOBJ, SYMBOBJ := ‘<’ ExternalID ‘>’
————————————————————
Populational Structure:

AG :=
    ‘Agents:’
        [AgentID = AGENT]*

AGProp :=
    ‘AgProps:’
        [PROP ‘(’ AgentID ‘)’ ]*

AGBeh :=
    ‘AgBehs:’
        [AgentID ‘-->’PROC]*

AGInter :=
    ‘AgInters:’
        [AgentID ‘<--’ EXCHPROC ‘-->’ AgentID]*

AGRel :=
    ‘AgRels:’
        [REL ‘(’ AgentID ‘,’ AgentID ‘)’ ]*

Pop :=
    ‘Pop’ PopID ‘:’
        AG
        AGProp
        AGBeh
        AGInter
        AGRel
————————————————————
Sociability Structure:

SR :=
    ‘SocRos:’
        [SocRoID [ ‘[’ 1..n ‘]’ ] = SOCRO]*

SRProp :=
    ‘SocRoProps:’
        [PROP ‘(’ SocRoID ‘)’]*

SRBeh :=
    ‘SocRoBehs:’
        [SocRoID ‘-->’ PROC]*

SRInter :=
    ‘SocRoInters:’
        [SocRoID ‘<--’ EXCHPROC ‘-->’ SocRoID]*

SRRel :=
    ‘SocRoRels:’
        [REL ‘(’ SocRoID ‘,’ SocRoID ‘)’]*

SRNet :=
    ‘SRNets:’
        [SRNetworkID ‘(’ SR, SRProp,
            SRBeh,SRInter, SRRel ‘)’]*

Soc :=
    ‘Soc’ SocID ‘:’
        SR
        SRProp
        SRBeh
        SRInter
        SRRel
        SRNet

————————————————————
Organizational Structure:

OR :=
    ‘OrgRos:’
        [OrRoID [ ‘[’ 1..n ‘]’ ] = ORGRO]*

ORProp :=
    ‘ORProps:’
        [PROP ‘(’ OrgRoID ‘)’]*

ORBeh :=
    ‘ORBehs:’
        [OrgRoID ‘-->’ PROC]*

ORInter :=
    ‘ORInters’
        [OrgRoID ‘<--’ EXCHPROC ‘-->’ OrgRoID]*

ORRel :=
    ‘ORRels:’
        [REL ‘(’ OrgRoID ‘,’ OrgRoID ‘)’]*

ORNet :=
    ‘ORNet:’
        OR
        ORProp
        ORBeh
        ORInter
        ORRel

OUProp :=
    ‘OUProps:’
        [PROP ‘(’ OrgUnID ‘)’]*

OUBeh :=
    ‘OUBehs:’
        [OrgUnID ‘-->’ PROC]*

OUInter :=
    ‘OUInters’
        [OrgUnID ‘<--’ EXCHPROC ‘-->’ OrgUnID]*

OURel :=
    ‘OURels:’
        [REL ‘(’ OrgUnID ‘,’ OrgUnID ‘)’]*

OUNet :=
    ‘OUNet:’
        OU
        OUProp
        OUBeh
        OUInter
        OURel

Interf :=
    ‘Interf:’
        [OrgRoID]*

Inp :=
    ‘Inps:’
        [InpPortID INPPORT]*

Out :=
    ‘Outs:’
        [OutPortID OUTPORT]*

OrgUn [ ‘[’ 1..n ‘]’ ] :=
    ‘OrgUn’ OrUnID ‘:’
        ORNet | OUNet
        Interf
        Inp
        Out

OU :=
    [OrgUn]+
```

Figure 9.5: The Concrete Syntax of *SML* (part 1)

```
Org :=
    'Org' OrgID ':'
        OU
        OUProp
        OUBeh
        OUInter
        OURel
        OUNet
―――――――――――――――――――――――――
Material Environment:

MO :=
    'MOs:'
        [MatObjID = MatObj]*

MOProp :=
    'MOProp:'
        [Prop(MatObjID)]*

MOBeh :=
    'MOBeh:'
        [MatObjID '-->' Proc]*

MOInter :=
    'MOInter:'
        [MatObjID '<--' ExchProc '-->' MatObjID]*

MORel :=
    'MORel:'
    [Rel(MatObjID,MatObjID)]*

MONet :=
    'MONet' MONetID ':'
        [MO
        MOProp
        MOBeh
        MOInter
        MORel]*

MEnv :=
    'MEnv' MEnvID ':'
        MO
        MOProp
        MOBeh
        MOInter
        MORel
        MONet
―――――――――――――――――――――――――
Symbolic Environment:

SO :=
    'SOs:'
        [SymbObjID = SymbObj]*

SOProp :=
    SymObjProp:
        [Prop '(' SymObjID ')']*

SORel :=
    'SymbObjRel:'
        [Rel '(' SymbObjID ',' SymbObjID ')']*

SEnv :=
    'SEnv' SEnvID ':'
        SO
        SOProp
        SORel

―――――――――――――――――――――――――
Implementation Relations:

ImpSocPop :=
    'ImpSocPop:'
        [AgentID '-->>' SocRoID]*

ImpOrgSoc :=
    'ImpOrgSoc:'
        [SocRoID '-->>' OrgRoID]*

ImpOrgPop :=
    'ImpOrgPop:'
        [AgentID '-->>' OrgRoID]*

Imp :=
    'Imp' ImpID ':'
        ImpSocPop
        ImpOrgSoc
        ImpOrgPop

―――――――――――――――――――――――――
Access Links:

AccAgMObj :=
    'AccAgMObj:'
        [AgentID '<<--' ExchProc '-->>' MatObjID]*

AccAgSObj :=
    'AccAgSObj:'
        [AgentID '<<--' ExchProc '-->>' SymbObjID]*

AccOrgUnMObj :=
    'AccOrgUnMObj:'
        [OrgUnID '<<--' ExchProc '-->>' MatObjID]*

AccOrgUnSObj :=
    'AccOrgUnMObj:'
        [OrgUnID '<<--' ExchProc '-->>' MatObjID]*

Acc :=
    'Acc' AccID ':'
        AccAgMObj
        AccAgSObj
        AccOrgUnMObj
        AccOrgUnSObj
―――――――――――――――――――――――――
Agent Society:

AgSoc :=
    'AgSoc' AgSocID ':'
    PopID
    SocID
    OrgID
    MEnvID
    SEnvID
    ImpID
    AccID
―――――――――――――――――――――――――
```

Figure 9.6: The Concrete Syntax of *SML* (part 2)

Chapter 10

Modeling Example Systems in *SML*

10.1 Modeling an *Aalaadin* Example in *SML*

In this section, we model in *SML* the AALADIN *market organization example* provided by[14]. It is an organization in which:

- there are three groups of agents, namely: *Service Providers* group, *Client* group and *Temporary Contract* group;
- the *Service Providers* group and the *Client* group have their own internal interaction protocols;
- an agent has to play two roles simultaneously, both roles called *Broker*, one in the *Service Providers* group and one in the *Client* group;
- the two *Broker* roles specify that the task of the agent that simultaneously play them is to insert into a *Temporary Contract* group both agents that are playing the role *Client* in the *Client* group and agents that are playing the role *Service* in the *Service Providers* group, so that they can enter in contact.

Figure 10.1 shows the original figure given in [14] to informally present the *market organization* example. The upper part shows the *organizational structure* of the system, the lower part illustrates an *instantiation* of such organizational structure.

In Fig. 10.3 we give, in the *SML* modeling language, a formal presentation of both parts of the *market organization example.*

Remark that:

- the power-set operator is denoted by `P(X)`, for any set `X`;
- comments are placed between double-quotes;
- the expression `BOT` indicates that the type or type component is undefined;
- the expression `S.C` names the component `C` of the structure `S`.

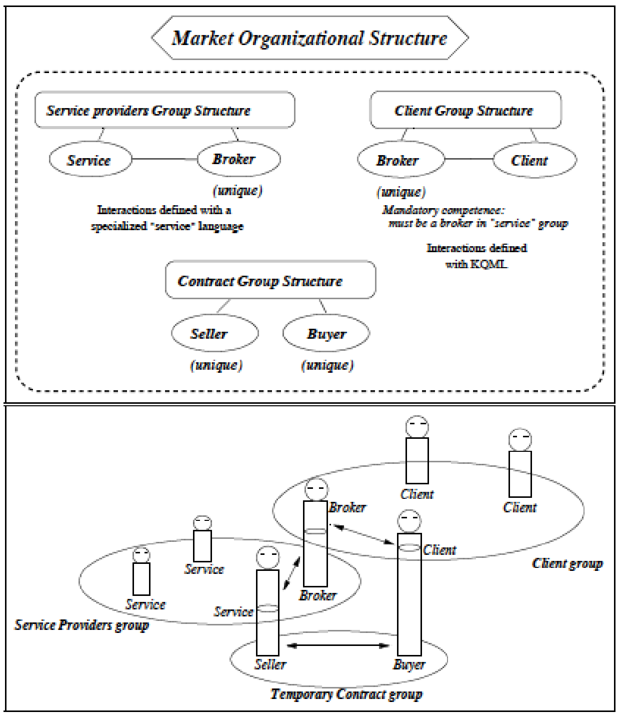

Figure 10.1: The AALAADIN *market organization example* [14].

```
Org O:
   OrgUn Clients:
      OrgRos:
            client
            broker-client
      OrgRoInters:
            client <-- <ep1> --> broker-client
   OrgUn Service-Providers:
      OrgRos:
            service
            broker-service
      OrgRoInters:
            service <-- <ep2> --> broker-service
   OrgUn Temporary-Contract:
      OrgRos:
            seller
            buyer
      OrgRoInters:
            seller <-- <ep3> --> buyer
AgSoc MarketOrganization:
   O
```

Figure 10.2: An SML summary description of the organizational model of the AALAADIN *market organization* example.

```
---
Pop P:
     Agent:
          ag1 = <extag1>
          ag2 = <extag2>
          ag3 = <extag3>
     AgProp: BOT
     AgBeh: BOT
     AgInter: BOT
     AgRel: BOT
---
Soc S:
     SocRo : BOT
     SocRoProp : BOT
     SocRoBeh : BOT
     SocRoInter : BOT
     SocRoRel : BOT
     SocRoNet : BOT
---
Org O:
     OrgUnit Clients:
         ORNet:
              OrgRos:
                   client = BOT
                   broker-client = BOT
              OrgRoProps: BOT
              OrgRoBehs: BOT
              OrgRoInters:
                   client <-- <ep1> --> broker-client
              OrgRoRels: BOT
     OrgUnit Service-Providers:
         ORNet:
              OrgRos:
                   service = BOT
                   broker-service = BOT
              OrgRoProps: BOT
              OrgRoBehs: BOT
              OrgRoInters:
                   service <-- <ep2> --> broker-service
              OrgRoRels: BOT
     OrgUnit Temporary-Contract:
        ORNet:
              OrgRos:
                   seller = BOT
                   buyer = BOT
              OrgRoProps: BOT
              OrgRoBehs: BOT
              OrgRoInters:
                   seller <-- <ep3> --> buyer
              OrgRoRels: BOT
---
MEnv ME: BOT
---
SEnv SE BOT
---
Imp I:
     ImpSocPop : BOT
     ImpOrgSoc : BOT
     ImpOrgPop:
     "direct implementation of organizational roles by agents"
          ag1 -->> client
          ag1 -->> buyer
          ag2 -->> broker-client
          ag2 -->> broker-service
          ag3 -->> service
          ag3 -->> seller
---
Acc A: BOT
---
AgSoc Market-Org-Example:
          P
          S
          O
          ME
          SE
          I
          A
---
```

Figure 10.3: An *SML* detailed description of the implementation of the Aalaadin *market organization* example.

10.2 Modeling an ***Electronic Institution*** Example in ***SML***

[in construction]

10.3 Modeling an ***OperA*** Example in ***SML***

[in construction]

10.4 Modeling a *MOISE+* Example in *SML*

In this section, we model in *SML* an example of organizational specification in *MOISE+*, namely, the specification of a graduate school and the committee charged of selecting students for a graduate course. The example is adapted from the one given in [47].

10.4.1 The *Structural Specification* of the Graduate School

The *MOISE+* structural specification of the *School* system is presented, informally, in Fig. 10.4.

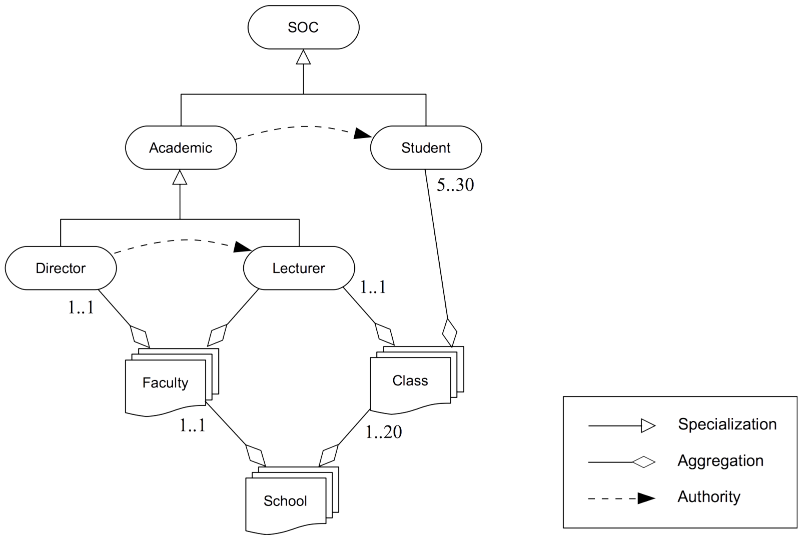

Figure 10.4: The structural specification of a school.

Formally, we may denote the *MOISE+* structural specification of *School* as follows:

$SS_{School} = (G_0, R, \sqsubseteq)$

where:

- SS_{School}: the structural specification of *School*;
- $G_0 = \{G_{School}\}$: the set of root groups of *School*;
- $R = \{Soc, Academic, Student, Director, Lecturer\}$: the set of roles of *School*;
- $\{Soc \sqsupset Academic, Soc \sqsupset Student, Academic \sqsupset Director,$ $Academic \sqsupset Lecturer\}$: the relation of specialization of the roles of *School*.

$G_{School} = (R, SG, L_{Intra}, L_{inter}, C_{Intra}, C_{Inter}, rc, gc)$

where:

- G_{School}: the group *School*;
- $R = \emptyset$: the set of roles under G_{school};
- $SG = \{G_{Faculty}, G_{Class}\}$: the set of sub-groups of G_{School};
- $L_{Intra} = \emptyset$: the set of intra-links of the roles G_{school};
- $L_{Inter} = \emptyset$: the set of inter-links of the roles G_{School};
- $C_{Intra} = \emptyset$: the set of intra-compatibility relations of the roles of G_{School};
- $C_{Inter} = \emptyset$: the set of inter-compatibility relations of the roles of G_{School};
- $rc = \emptyset$: the role cardinality of the roles of G_{School};
- $gc = \{G_{Faculty} \mapsto (1..1), G_{Class} \mapsto (1..20)\}$: the cardinality of the sub-groups of G_{School}.

$G_{Faculty} = (R, SG, L_{Intra}, L_{inter}, C_{Intra}, C_{Inter}, rc, gc)$

where:

- $G_{Faculty}$: the group *Faculty*;
- $R = \{Academic, Director, Lecturer\}$: the set of roles under $G_{Faculty}$;
- $SG = \emptyset$: the set of sub-groups of $G_{Faculty}$;
- $L_{Intra} = \{link(Director, Lecturer, auth)\}$: the set of intra-links of the roles of $G_{Faculty}$;
- $L_{Inter} = \{link(Academic, Student, auth)\}$: the set of inter-links of the roles of $G_{Faculty}$;
- $C_{Intra} = \emptyset$: the set of intra-compatibility relations of the roles of $G_{Faculty}$;
- $C_{Inter} = \emptyset$: the set of inter-compatibility relations of the roles of $G_{Facutly}$;
- $rc = \{Director \mapsto (1..1)\}$: the role cardinality of $G_{Faculty}$;
- $gc = \emptyset$: the cardinality of the sub-groups of $G_{Faculty}$.

$G_{Class} = (R, SG, L_{Intra}, L_{inter}, C_{Intra}, C_{Inter}, rc, gc)$

where:

- G_{Class}: the group *Class*;
- $R = \{Lecturer, Student\}$: the set of roles under G_{Class};
- $SG = \emptyset$: the set of sub-groups of G_{Class};
- $L_{Intra} = \emptyset$: the set of intra-links of the roles of G_{Class};
- $L_{Inter} = \{link(Academic, Student, auth)\}$: the set of inter-links of the roles of G_{Class};
- $C_{Intra} = \emptyset$: the set of intra-compatibility relations of the roles of G_{Class};
- $C_{Inter} = \emptyset$: the set of inter-compatibility relations of the roles of G_{Class};
- $rc = \{Lecturer \mapsto (1..1), Student \mapsto (5..30)\}$: the role cardinality of G_{Class};
- $gc = \emptyset$: the cardinality of the sub-groups of G_{Class}.

Modeled in *SML*, the structure of the *School* organization may appear as follows[1]:

```
---
Org O:
    OrgUn School:
        OUNet:
            OrgUn Faculty[1..1]:
                OrgRoNet:
                    OrgRos:
                        Soc = BOT
                        Academic = BOT
                        Director[1..1] = BOT
                        Lecturer = BOT
                    OrgRoRel:
                        spec(Soc,Academic)
                        spec(Academic,Director)
                        spec(Academic,Lecturer)
                        auth(Director,Lecture)
            OrgUn Class[1..20]:
                OrgRoNet:
                    OrgRos:
                        Lecturer[1..1] = BOT
                        Student[5..30] = BOT
            OrgUnRels:
                auth(Faculty.Academic,Class.Student)
```

10.4.2 The *Functional Specification* of the School

We follow [47] and treat the functional dimension of the *School* example regarding a committee created to select students for a graduate course.

Figure 10.5 shows the structural specification of the *School*, extended with the group *Committee*. Two new roles were created, to account for the operation of the new group, namely, *President* and *Member*, both specialization of the role *Lecturer*. The respective role cardinalities of the new roles are indicated in the figure.

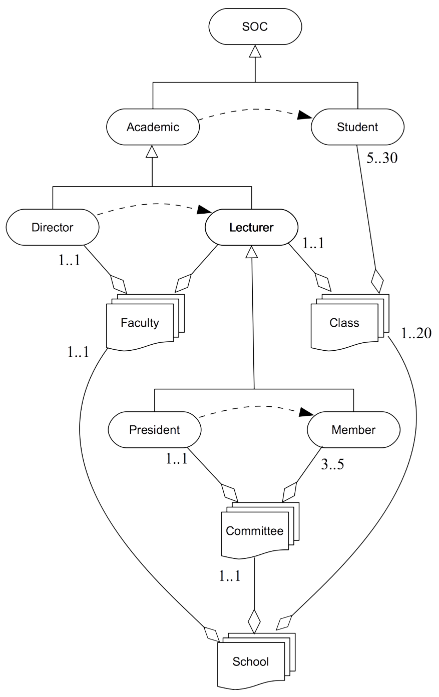

Figure 10.5: The structural specification of the *School*.

The formal *MOISE+* definition of the *new* structural specification of the *School* is the following (cf. above the previous formal specification SS_{School}):

[1]We show here only the relevant elements. The translation of the full structural, functional and deontic specification of the *School* is given in Sect. 10.4.4.

$SS_{School} = (G_0, R, \sqsubseteq)$

where:

- SS: the structural specification of $School$;
- $G_0 = \{G_{School}\}$: the set of root groups of $School$;
- $R = \{Soc, Academic, Student, Director, Lecturer, President, Member\}$: the set of roles of $School$;
- $\{Soc \sqsubset Academic, Soc \sqsubset Student, Academic \sqsubset Director, Academic \sqsubset Lecturer, Lecturer \sqsubset President, Lecturer \sqsubset Member\}$: the relation of specialization of the roles of $School$.

$G_{School} = (R, SG, L_{Intra}, L_{inter}, C_{Intra}, C_{Inter}, rc, gc)$

where:

- G_{School}: the group $School$;
- $R = \emptyset$: the set of roles under G_{school};
- $SG = \{G_{Faculty}, G_{Class}, G_{Committee}\}$: the set of sub-groups of G_{School};
- $L_{Intra} = \emptyset$: the set of intra-links of the roles G_{school};
- $L_{Inter} = \emptyset$: the set of inter-links of the roles G_{School};
- $C_{Intra} = \emptyset$: the set of intra-compatibility relations of the roles of G_{School};
- $C_{Inter} = \emptyset$: the set of inter-compatibility relations of the roles of G_{School};
- $rc = \emptyset$: the role cardinality of the roles of G_{School};
- $gc = \{Faculty \mapsto (1..1), Class \mapsto (1..20)\}$: the cardinality of the sub-groups ofG_{School}.

$G_{Faculty} = (R, SG, L_{Intra}, L_{inter}, C_{Intra}, C_{Inter}, rc, gc)$

where:

- $G_{Faculty}$: the group $Faculty$;
- $R = \{Academic, Director, Lecturer\}$: the set of roles under $G_{Faculty}$;
- $SG = \emptyset$: the set of sub-groups of $G_{Faculty}$;
- $L_{Intra} = \{link(Director, Lecturer, auth)\}$: the set of intra-links of the roles of $G_{Faculty}$;
- $L_{Inter} = \{link(Academic, Student, auth)\}$: the set of inter-links of the roles of $G_{Faculty}$;
- $C_{Intra} = \emptyset$: the set of intra-compatibility relations of the roles of $G_{Faculty}$;
- $C_{Inter} = \emptyset$: the set of inter-compatibility relations of the roles of $G_{Facutly}$;
- $rc = \{Director \mapsto (1..1)\}$: the role cardinality of $G_{Faculty}$;
- $gc = \emptyset$: the cardinality of the sub-groups of $G_{Faculty}$.

$G_{Class} = (R, SG, L_{Intra}, L_{inter}, C_{Intra}, C_{Inter}, rc, gc)$

where:

- G_{Class}: the group *Class*;
- $R = \{Academic, Lecturer, Student\}$: the set of roles under G_{Class};
- $SG = \emptyset$: the set of sub-groups of G_{Class};
- $L_{Intra} = \emptyset$: the set of intra-links of the roles of G_{Class};
- $L_{Inter} = \{link(Academic, Student, auth), link(Director, Lecturer, auth)\}$: the set of inter-links of the roles of G_{Class};
- $C_{Intra} = \emptyset$: the set of intra-compatibility relations of the roles of G_{Class};
- $C_{Inter} = \emptyset$: the set of inter-compatibility relations of the roles of G_{Class};
- $rc = \{Lecturer \mapsto (1..1), Student \mapsto (5..30)\}$: the role cardinality of G_{Class};
- $gc = \emptyset$: the cardinality of the sub-groups of G_{Class}.

$G_{Committee} = (R, SG, L_{Intra}, L_{inter}, C_{Intra}, C_{Inter}, rc, gc)$

where:

- $G_{Committee}$: the group *Committee*;
- $R = \{President, Member\}$: the set of roles under $G_{Committee}$;
- $SG = \emptyset$: the set of sub-groups of $G_{Committee}$;
- $L_{Intra} = \{link(President, Member, auth)\}$: the set of intra-links of $G_{Committee}$;
- $L_{Inter} = \{link(Director, President, auth), link(Director, Member, auth)$: the set of inter-links of $G_{Committee}$;
- $C_{Intra} = \emptyset$: the set of intra-compatibility relations of the roles of $G_{Committee}$;
- $C_{Inter} = \emptyset$: the set of inter-compatibility relations of the roles of $G_{Committee}$;
- $rc = \{President \mapsto (1..1), Member \mapsto (3..5)\}$: the role cardinality of $G_{Committee}$;
- $gc = \emptyset$: the cardinality of the sub-groups of $G_{Committee}$.

The *SML* modeling of the extended structure of the *School* is:

```
---
Org O:
     OrgUn School:
         OUNet:
            OrgUn Faculty[1..1]:
                 OrgRoNet:
                      OrgRos:
                          Soc = BOT
                          Academic = BOT
                          Director[1..1] = BOT
                          Lecturer = BOT
                      OrgRoRel:
                          spec(Soc,Academic)
                          spec(Academic,Director)
                          spec(Academic,Lecturer)
                          auth(Director,Lecture)
            OrgUn Class[1..20]:
                 OrgRoNet:
                     OrgRos:
                         Lecturer[1..1] = BOT
                         Student[5..30] = BOT
            OrgUn Committee:
                 OrgRoNet:
                      OrgRos:
                          President[1..1] = BOT
                          Member[3..5] = BOT
                      OrgRolRel:
                           auth(President,Member)
            OrgUnRels:
                  auth(Faculty.Academic,Class.Student)
                  auth(Faculty.Director,Committee.President)
                  auth(Faculty.Director,Committee.Member)
```

The functional specification of the task of the *Committee* group is given informally by the social scheme of Fig. 10.6.

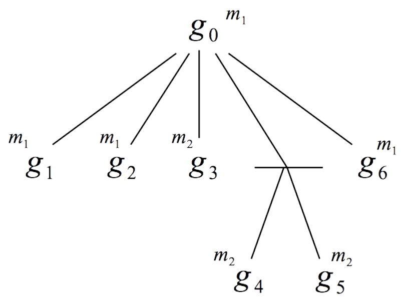

Figure 10.6: The social scheme of the selection committee.

The following are the goals of that social scheme:

Goal	**Description**
g_0	an application to a graduate course is processed
g_1	the application is received by the *President*
g_2	the members of the $G_{Committee}$ are indicated
g_3	the $G_{Committee}$ meets to analyze the documentation
g_4	the candidate is approved by $G_{Committee}$
g_5	the candidate is rejected by $G_{Committee}$
g_6	the result is communicated to the applicant

The functional specification of the *School*, concerning the task of the *Committee* is formally given by:

$FS_{G_{Committee}} = (SS_{G_{Committee}}, MP)$

where:

- $Sch_{G_{Committee}}$: the social scheme of $G_{Committee}$;
- $MP = \emptyset$: the preference relation between missions.

$Sch_{G_{Committee}} = (G, P, M, mo, mc)$

where:

- $Sch_{G_{Committee}}$: the social scheme of $G_{Committee}$;
- $G = \{g_0, g_1, g_2, g_3, g_4, g_5, g_6\}$: the set of goals of $G_{Committee}$;
- $P = \{g_0 = g_1, g_2, g_3, (g_4 \mid g_5), g_6\}$: the plan for the goal g_0;
- $M = \{m_1, m_2\}$: the set of missions of $G_{Committee}$;
- $mo = \{m_1 \mapsto \{g_0, g_1, g_2, g_6\}, m_2 \mapsto \{g_3, g_4, g_5\}\}$: the set of goals of each mission;
- $mc = \{m_1 \mapsto (1.1), m_2 \mapsto (3..5)\}$: the agent cardinality of each mission.

The modeling in *SML* of this *MOISE+* functional specification amounts to its representation as a symbolic structure in the symbolic environment of the agent society:

```
---
SEnv SE:
    SOs:
        "goals:"
            g0 = g1,g2,g3,(g4|g5),g6
            g1 = <g1>
            g2 = <g2>
            g3 = <g3>
            g4 = <g4>
            g5 = <g5>
            g6 = <g6>
        "missions:"
            m1[1..1] = g0,g1,g2,g6
            m2[3..5] = g3,g4,g5
```

10.4.3 The *Deontic Specification* of the School

The *deontic specification* links the structural and the functional specifications by determining which missions a role has the obligation or permission to execute.

Formally, we determine the deontic specification of the *School* system, without taking temporal limits into account, as follows:

$DS = (P, O)$

where:

- $P = \{perm(President, m_1)\}$: the set of permissions of the *President*;
- $O = \{oblig(President, m_2), oblig(Member, m_2)\}$: the set of obligations of the *President* and of any *Member*.

The modeling of this MOISE+ deontic specification to SML amounts to adding norms to the symbolic environment *SEnv*, resulting in the following:

```
---
SEnv SE:
    SOs:
        "goals:"
            g0 = <g1,g2,g3,(g4|g5),g6>
            g1 = <g1>
            g2 = <g2>
            g3 = <g3>
            g4 = <g4>
            g5 = <g5>
            g6 = <g6>
        "missions:"
            m1[1..1] = <g0,g1,g2,g6>
            m2[3..5] = <g3,g4,g5>
        "norms:"
            nrm1 = <permit(President,m1)>
            nrm2 = <oblig(President,m2))>
            nrm3 = <oblig(Memb,m2))>
```

10.4.4 The Full Modeling of the *MOISE+* Specification of the School in *SML*

The full modeling in *SML* of the *MOISE+* specification of the *School* system is shown in Fig. 10.7.

```
---
Org O:
     OrgUn School:
         OUNet:
            OrgUn Faculty[1..1]:
                 OrgRoNet:
                      OrgRos:
                          Soc = BOT
                          Academic = BOT
                          Director[1..1] = BOT
                          Lecturer = BOT
                      OrgRoRel:
                          spec(Soc,Academic)
                          spec(Academic,Director)
                          spec(Academic,Lecturer)
                          auth(Director,Lecture)
            OrgUn Class[1..20]:
                 OrgRoNet:
                     OrgRos:
                         Lecturer[1..1] = BOT
                         Student[5..30] = BOT
            OrgUn Committee:
                 OrgRoNet:
                      OrgRos:
                           President[1..1] = BOT
                           Member[3..5] = BOT
                      OrgRolRel:
                           auth(President,Member)
            OrgUnRels:
                  auth(Faculty.Academic,Class.Student)
                  auth(Faculty.Director,Committee.President)
                  auth(Faculty.Director,Committee.Member)
---
SEnv SE:
    SOs:
        "goals:"
            g0 = <g1,g2,g3,(g4|g5),g6>
            g1 = <g1>
            g2 = <g2>
            g3 = <g3>
            g4 = <g4>
            g5 = <g5>
            g6 = <g6>
        "missions:"
            m1[1..1] = <g0,g1,g2,g6>
            m2[3..5] = <g3,g4,g5>
        "norms:"
            nrm1 = <permit(President,m1)>
            nrm2 = <oblig(President,m2))>
            nrm3 = <oblig(Memb,m2))>
---
AgSoc School:
     O
     SE
---
```

Figure 10.7: An *SML* detailed description of organizational model of the *MOISE+ School* example.

Chapter 11

Conclusion

This report introduced *TPO*, a type-based approach to agent societies and inter-societal agent systems. The concepts of agent society and inter-societal agent systems themselves were only made precise on the basis of this formal presentation.

The *TPO* type system, with its basic types, type constructors, typing rules and rule constraints, was presented on a step-by-step way, following the bottom-up construction of the system.

[to be finished]

Acknowledgments

To:

- Jomi Hübner and Rafael Bordini, for ancient talks about open multiagent systems.
- Yves Demazeau, for discussions that sparkled the very first idea of the PopOrg model.
- Graçaliz Dimuro, for the direct collaboration in the development of most of the previous versions of the PopOrg model.
- Helder Coelho, for the partnership in the work on the moral aspects of agent societies, which is not included here.
- Rafael Bordini and Renata Vieira, for the interest on the work on the ideological aspects of agent societies, which is not included here.
- UFRGS, PUCRS, UCPel and FURG, which provided financial and operational support for this work.
- CNPq, CAPES and FAPERGS, which provided partial financial support for this work.
- All other colleagues, students and paper reviewers that commented, suggested, criticized or otherwise collaborated with this work.
- Jomi Hübner, for discussions that sparkled the present report.

Bibliography

[1] Costa, A.C.R.: Proposal for a notion of modularity in multiagent systems. In van Riemskijk, M.B., Dalpiaz, F., Dix, J., eds.: Informal Proceedings of EMAS 2014, AAMAS @ Paris (2014)

[2] Costa, A.C.R.: Agent organizations and agent societies as interoperable modules for agent and conventional software systems. *Open publication on* www.ResearchGate.net, DOI: 10.13140/RG.2.2.35919.69284 (2017)

[3] Costa, A.C.R.: Two concepts of module, for agent societies and inter-societal agent systems. In: Informal Proceedings of the Workshop Engineering Multi-Agent Systems - EMAS@AAMAS2017, São Paulo, IFAMAS (2017)

[4] Costa, A.C.R., Demazeau, Y.: Toward a formal model of multi-agent systems with dynamic organizations. In: Proc. of ICMAS 96 – 2nd. Int'l Conf. on Mutiagent Systems, Kyoto, IEEE (1996) 431

[5] Demazeau, Y., Costa, A.C.R.: Populations and organizations in open multi-agent systems. In: 1st National Symposium on Parallel and Distributed Artificial Intelligence (PDAI'96), Hyderabad, India (1996)

[6] Costa, A.C.R., Dimuro, G.P.: A minimal dynamical organization model. In Dignum, V., ed.: Hanbook of Multi-Agent Systems: Semantics and Dynamics of Organizational Models. IGI Global, Hershey (2009) 419–445

[7] Costa, A.C.R.: Symbolic environments and the cultural aspects of augmented worlds. *Open publication on* www.ResearchGate.net (2016)

[8] Costa, A.C.R.: Ecosystems as agent societies, landscapes as multi-societal agent systems. In Adamatti, D.F., ed.: Multiagent Based Simulations Applied to Biological and Environmental Systems. IGI Global, Hershey (2017) 25–43

[9] Squazzoni, F.: A (computational) social science perspective on societal transitions. Computational and Mathematical Organization Theory **14** (2008) 266–282

[10] Costa, A.C.R.: The cultural level of agent societies. Invited talk at WESAAC 2011 - 5o. Workshop-School of Agent Systems, their Environments, and Applications. Curitiba, Brazil. Proceedings (2011) (In Portuguese).

[11] Costa, A.C.R.: Situated ideological systems: A core formal concept, some computational notation, some applications. Axiomathes **27** (2015) 15–78

[12] McCarthy, J.: Ascribing mental qualities to machines. Technical Report, Computer Science Dept., Stanford University (1979) Available at: http://www-formal.stanford.edu/jmc/.

[13] Dennet, D.: The intentional Stance. MIT Press, Cambridge (1987)

[14] Ferber, J., Gutknecht, O.: Aalaadin: a meta-model for the analysis and design of organizations in multi-agent systems. In Demazeau, Y., ed.: International Conference on Multi-Agent Systems - ICMAS 98, Paris, IEEE Press (1998) 128–135

[15] Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: An organizational view of multiagent systems. In Giorgini, P., Müller, J., Odell, J., eds.: Agent Oriented Software Engineering - AOSE IV, Berlin, Springer (2004) 214–230 (LNCS 2935).

[16] Ferber, J., Michel, F., Baez-Barranco, J.A.: Agre: Integrating environments with organizations. In Weyns, D., Parunak, H.V.D., Michel, F., eds.: Environments for Multi-Agent Systems, First International Workshop, E4MAS 2004. LNAI vol. 3374, Springer (2005) 48–56

[17] Aldewereld, H., Ávarez Napagao, S., Garcia, M.E., Sanz, J.G., Jiang, J., Cardoso, H.L.: Conceptual map for social coordination. In Aldewereld, H., Boissier, O., Dignum, V., Noriega, P., Padget, J., eds.: Social Coordination Frameworks for Social Technical Systems. Springer, Berlin (2016) 11–23

[18] Shoham, Y.: Agent oriented programming. Artificial Intelligence **60** (1993) 51–92

[19] Ricci, A., Viroli, M., Omicini, A.: Programming MAS with Artifacts. In Bordini, R.P., Dastani, M., Dix, J., El Fallah Seghrouchni, A., eds.: Programming Multi-Agent Systems. Volume 3862 of LNAI. Springer (2006) 206–221 3rd International Workshop (PROMAS 2005), AAMAS 2005, Utrecht, The Netherlands, 26/jul/2005. Revised and Invited Papers.

[20] Costa, A.C.R.: On the legal aspects of agent societies. *Open publication on* www.ResearchGate.net - DOI: 10.13140/2.1.4345.7923 (2014)

[21] Costa, A.C.R.: Situated legal systems and their operational semantics. Artificial Intelligence & Law **43** (2015) 43–102

[22] Costa, A.C.R.: Moral systems of agent societies: Some elements for their analysis and design. In: Workshop on Ethical Issues in the Design of Intelligent Agents - EDIA@ECAI 2016, EURAI/University of Delft (2016) Paper 10

[23] Bourdieu, P.: Outline of a Theory of Practice. Cambridge Univ. Press, Cambridge (1977)

[24] Turner, J.H.: Theoretical Principles of Sociology. Springer (2010)

[25] Enderton, H.: Introduction to Mathematical Logic. Hartcourt, San Diego (2001)

[26] Piaget, J.: Essai sur la théorie des valeur qualitatives en sociologie statique ("synchronique"). Publications de la Faculté des Science Économiques et Sociales de l'Université de Genève (1941) (English translation in: Piaget, J. Sociological Studies, Routledge, 1995.).

[27] Piaget, J.: Les relations entre la morale et le droit. Publications de la Faculté des Science Économiques et Sociales de l'Université de Genève (1944) (English translation in: Piaget, J. Sociological Studies, Routledge, 1995.).

[28] Searle, J.R.: The Construction of Social Reality. The Free Press, New York (1995)

[29] Orwell, G.: 1984. New American Library, New York (1961)

[30] Simmel, G.: Individuality and Social Forms. Univ. Chicago Press, Chicago (1971)

[31] Mauss, M.: Essai sur le Don. PUF, Paris (2007)

[32] Goffman, E.: The Presentation of the Self in Every Day Life. University of Edinburgh, Edinburg (1956)

[33] Herder, F.: The Psychology of Interpersonal Relations. Wiley, New York (1958)

[34] Homans, G.: Social Behavior – Its Elementary Forms. Harcourt, Brace & World, New York (1961)

[35] Costa, A.C.R.: Elements for an ideology modeling language. (Submitted for publication) (2015)

[36] Cardelli, L.: Types for data oriented languages. In Schmidt, J.W., Ceri, S., Missikof, M., eds.: Advances in Database Techonology - EDBT' 88. Springer, Berlin (1988) 1–14 (LNCS 303).

[37] Weyns, D., Parunak, H.V.D., Michel, F., Holvoet, T., J., F.: Environments for multiagent systems: State-of-the-art and research challenges. In: Environments for Mutiagent Systems, Berlin, Springer (2005) (LNAI 2477).

[38] Noriega, P.: Agent Mediated Auctions: The Fishmarket Metaphor. PhD thesis, Universitat Autônoma de Barcelona Facultat de Ciências, Barcelona (1997)

[39] Esteva, M., Cruz, D., Sierra, C.: ISLANDER: an electronic institutions editor. In: Proceedings AAMAS'02, New York, ACM (2002) 1045–1052

[40] Esteva, M., Rosell, B., Rodríguez-Aguilar, J.A., Arcos, J.: AMELI: an agent-based middleware for electronic institutions. In: Proceedings AAMAS'04, New York, ACM (2004) 236–243

[41] Esteva, M., Rodríguez, J.A., Arcos, J.L., Sierra, C., Garcia, P.: Formalising agent mediated electronic institutions. In: Congrés Catalã d'Intelligência Artificial, Vilanova i la Geltru, Catalonia (2000) 329–=338

[42] Sierra, C., Rodríguez-Aguilar, J.A., Noriega, P., Arcos, J.L.: Engineering multi-agent systems as electronic institutions. UPGRADE - The European Journal for the Informatics Professionsl **V** (2004) 33–39

[43] North, D.C.: Institutions, Institutional Change and Economic Performance. Cambridge University Press (1990)

[44] North, D.C.: Institutions. Journal of Economic Perspectives **5** (1991) 97–112

[45] Jones, A.J.L., Sergot, M.: On the characterization of law and computer systems: The normative systems perspective. In Meyer, J.J.C., Wieringa, R.J., eds.: Deontic Logic in Computer Science: Normative System Specification. Wiley, New York (1993) 275–307

[46] Dignum, V.: A model for organizational interaction: based on agents, founded in logic. PhD thesis, University of Utrecht, Utrecht (2003)

[47] Hübner, J.F.: Um Modelo de Reorganização de Sistemas Multiagentes. PhD thesis, Escola Politécnica - USP, São Paulo (2003)

[48] Hübner, J.F., Sichman, J.S., Boissier, O.: Developing organised multi-agent systems using the MOISE+ model: Programming issues at the system and agent levels. International Journal of Agent-Oriented Software Engineering **1** (2007) 370–395

[49] Boissier, O., Bordini, R., Hübner, J.F., Ricci, A., , Santi, A.: Multi-agent oriented programming with JaCaMo. Science of Computer Programming **78** (2013) 747–761

[50] Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak using Jason. Wiley, London (2007)

[51] Ricci, A., Viroli, M., Omicini, A.: CArtAgO: An infrastructure for engineering computational environments in MAS. In Weyns, D., Parunak, H.V.D., Michel, F., eds.: E4MAS2006 – 3rd International Workshop on Environments for Multi-Agent Systems, AAMAS (2006) 102–119

[52] Piunti, M., Ricci, A., Boissier, O., Hübner, J.: Embodied organisations in MAS environments. In Braubach, L., van der Hoek, W., Petta, P., Pokahr, A., eds.: Proceedings of 7th German conference on Multi-Agent System Technologies (MATES 09), Berlin, Springer (2009) 115–127 (LNCS 5774).

[53] Aldewereld, H., Boissier, O., Dignum, V., Noriega, P., Padget, J., eds.: Social Coordination Frameworks for Social Technical Systems. Springer, Berlin (2016)

[54] Weiss, G., ed.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge (1999)

[55] Patel-Schneider, P., Hayes, P., Horrocks, I.: OWL Web Ontology Language Abstract Syntax and Semantics (2004) Online at http://www.w3.org/TR/owl-semantics/.