Multi Agent Cooperative Robotics Laboratory

Kansas State University
234 Nichols Hall
Manhattan, KS 66506-2302

Phone: (785) 532-6350
Fax: (785) 532-7353
http://macr.cis.ksu.edu/

Technical Report

Using Category Theory to Compose Multiagent Organization Design Models

by

Walamitien H. Oyenan and Scott A. DeLoach

MACR-TR-2010-01

March 25, 2010

Table of Contents

Abstract

Organization-based Multiagent Systems (OMAS) have been viewed as an effective paradigm for addressing the design challenges posed by today's complex systems. In those systems, the organizational perspective is the main abstraction, which provides a clear separation between agents and systems, allowing a reduction in the complexity of the overall system. To ease the development of OMAS, several methodologies have been proposed. However, existing multiagent approaches do not provide a rigorous composition process to integrate multiagent organization designs. In this report, we propose a formalization of OMAS designs using concepts from category theory and show how large and complex organization designs can be built from simpler ones.

1. Introduction

Design models are often created by different teams and need to be merged into one main design. Similarly, complex designs can be decomposed into smaller more manageable design models and then integrated later. Moreover, several projects might require the reuse of some previous designs. Unfortunately, most of the current Agent-Oriented Software Engineering (AOSE) methodologies simply suggest the decomposition of organization designs and fail to provide a rigorous process to recombine them. In most cases, the designer informally uses the agent interaction mechanisms to integrate organization designs.

In this report, we define a general framework for the formal composition of Organization-based Multiagent Systems (OMAS) design models. The composition is done solely at the design level, resulting in a single composite organization design that can then be used at runtime. The main organizational models used in this work are the goal and role models, which are key models that provide OMAS their adaptability. These models (in various forms) have been used in several OMAS framework. My work is based on the Organization Model for Computational Adaptive Systems (OMACS) [7]. There are many other organizational models for OMAS [10] and the approach proposed in this report could well be adapted for any of them. OMACS proposes a formal framework for describing organizational models for MAS and is supported by a rigorous methodology tailored from the O-MaSE process framework [13]. OMACS defines an organization as a set of goals (G) that the organization is attempting to accomplish, a set of roles (R) that must be played to achieve those goals, a set of capabilities (C) required to play those roles and a set of agents (A) who are assigned to roles in order to achieve organizational goals. There are more entities defined in OMACS that are not relevant for this report. The reader is referred to [9] for the complete model.

The goal models and role models represent the persistent part of a multiagent organization, the organization structure [11], which can then be populated later with heterogonous agents to produce a dynamic organization. Hence, this work does not deal with the actual agents that will participate in the organization.

We propose a framework allowing the composition design models, by treating composition as an algebraic operator over models and their relationships. Using a category theoretic approach, we formalize the goal model, the role model, and finally the entire organization. We define each model as a category and then show that these models can be composed by computing their pushout in the appropriate category. By using this mathematical framework, we obtain a formal construction of the composition of organizations that is guaranteed to result in a correct organizational design.

2. Organization Design Models

OMACS defines an organization as a set of goals (G) that the organization is attempting to accomplish, a set of roles (R) that must be played to achieve those goals, a set of capabilities (C) required to play those roles and a set of agents (A) who are assigned to roles in order to achieve organizational goals. The complete OMACS model is defined in [9]. In this report, we use a generalization of the OMACS model and only consider the goals, roles and the relationship that exists between them. These entities represent the persistent part of the organization that can be populated with heterogonous agents to produce a dynamic organization. In the following subsections, we formally define the models that will be used throughout this work.

2.1. Goal Model

In a typical multiagent organization, organizational goals and roles are organized in a goal tree [18, 23, 32, 33] and in a role model [21, 33, 34] respectively. For this report, we chose to organize our goals using a Goal Model for Dynamic Systems (GMoDS) [8]. In a GMoDS goal model, goals are organized in a goal tree such that subgoals of a goal are either an OR-decomposition or an AND-decomposition of that goal. A goal represents a desirable state of a system and is represented by the tuple $g = \langle name, type \rangle$ where *name* is the name of the goal, and *type* represents the decomposition of the goal, which can be *OR, AND, or LEAF* (leaf goals are of type LEAF and cannot be decomposed). In addition, the GMoDS goal model contains two time-based relationships between goals: the *precedes* and *triggers* relations. We say *goal g_1 precedes goal g_2*, if g_1 must be satisfied before g_2 can be pursued by the organization. Moreover, during the pursuit of specific goals, *events* may occur that cause the instantiation of new goals. Instantiated goals may be parameterized to allow a context sensitive meaning. If an event e can occur during the pursuit of goal g_1 that instantiates goal g_2, we say *g_1 triggers g_2 based on e*.

We extend GMoDs [8] to incorporate the notion of external goals and internal goals. Internal goals (G_i) are the actual goals that the organizations will try to achieve. They are organized in a tree. External goals (G_x) are just placeholders for goals from other organizations and they do not impact the satisfiability of a goal model as they will never be assigned to an agent. External goals are not part of the decomposition tree. They can only trigger internal goals and be triggered by internal goals. In addition, without loss of generality, we assume that all goal models have the same root. This root is represented by an empty AND goal called *generic root* (*g_root*). Formally, the goal model can be represented as mathematical structure composed of a rooted tree and a graph. The tree correspond to the AND-OR decompositions between goals. Its edges represent the *parent* relationship. The graph represents the *time-based* relationships between goals.

Definition 1: Goal Model

A *goal model* is a tuple GM = $\langle$ G, ET, EG, g_root $\rangle$ where:

- G : set of *organizational goals* such that $G = G_x \cup G_i$, where G_x represents the set of *external goals* and G_i the set of *internal goals*. We have $G_x \cap G_i = \varnothing$;
- $ET \in G_i \times G_i$: set of *parent* edges such that $\langle G_i, ET, g\_root\rangle$ is a rooted tree
- $EG \in G \times G$: set of *time-based* edges such that $\langle G, EG\rangle$ is a directed graph
- $g\_root \in G_i$: the *root* of the goal model.

Given a goal model GM, the set $G_L \subset G_i$ represents the leaf goals. The rooted tree is called *induced tree* and the graph is called *induced graph*.

Moreover, we define three functions over the nodes goal model: parent, precedes and triggers.

Definition 2: Functions on goals

Given a goal model $GM = \langle G, ET, EG, g\_root\rangle$ and a set of events Ev, we have:

- parent: $G_i \rightarrow G_i$; defines the parent of a given goal
- precedes: $G_i \rightarrow 2^{Gi}$; indicates all the goals preceded by a given goal
- triggers: $Ev \rightarrow 2^{G \times G}$; $\langle g_1,g_2\rangle \in$triggers(e) iff g_1 triggers g_2 based on e.

Following this definition, we can characterize the internal and external goals as follows:

$G_i = \{g \in G \mid g\_root \in parent^*(g)\}$;

$G_x = \{g \in G - \{g\_root\} \mid parent(g) = \varnothing\}$.

Moreover, we have:

$ET = \{\langle g_1,g_2\rangle \in G_i \times G_i \mid g_1 = parent(g_2)\}$.

$EG = \{\langle g_1,g_2\rangle \in G \times G \mid (g_2 \in precedes(g_1)) \vee (\exists e \in Ev \mid \langle g_1,g_2\rangle \in triggers(e))\}$

Example 1

Figure 1 represents the goal model $GM\_example = \langle G, ET, EG, g\_root\rangle$, where:

- $G = \{g\_root, g_1, g_2, g_3, g_4, g_5, g_6, g_7, eg_1\}$ with $G_i = \{g\_root, g_1, g_2, g_3, g_4, g_5, g_6, g_7\}$ and $G_x = eg_1\}$
- $ET = \{ \langle g_1,g\_root\rangle, \langle g_2,g\_root\rangle, \langle g_3,g_1\rangle, \langle g_4,g_1\rangle, \langle g_5,g_1\rangle , \langle g_6,g_2\rangle , \langle g_7,g_2\rangle\}$
- $EG = \{\langle eg_1,g_1\rangle, \langle g_3,g_4\rangle, \langle g_4,g_5\rangle, \langle g_5,g_2\rangle\}$
- $root = \{g\_root\}$

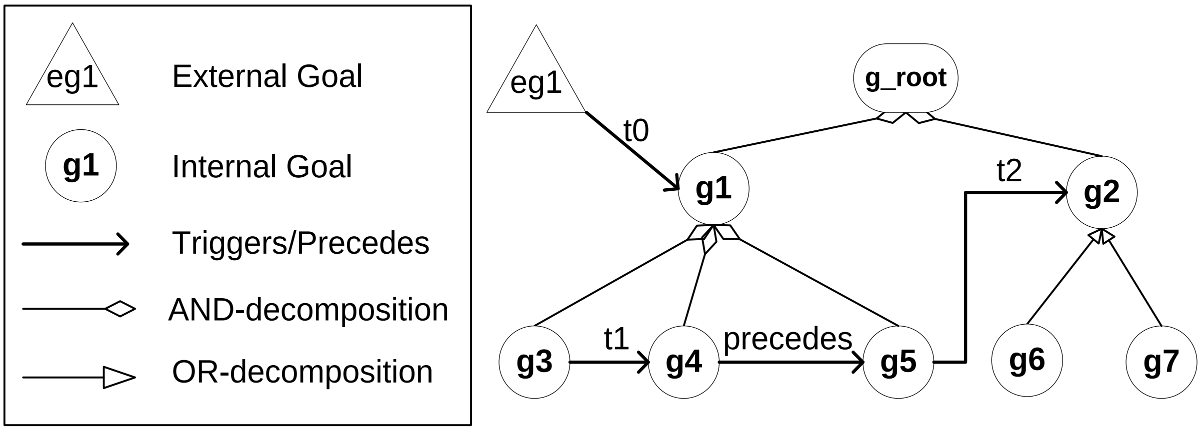

Figure 1. Goal Model Example

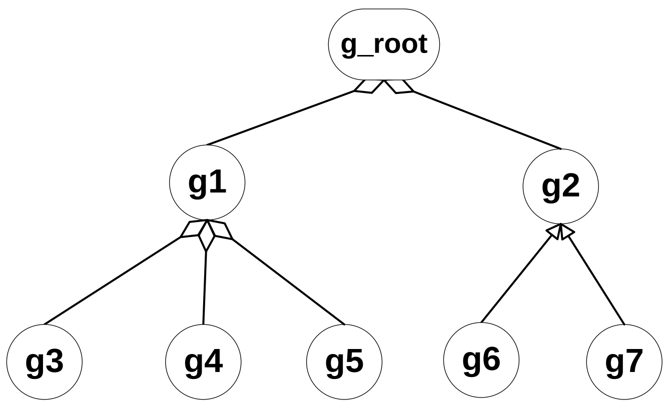

Figure 2. Example of Induced Tree

Moreover, for each goal g in the goal model in Figure 1, the type of the goal (g.type) is defined based on the decomposition arrow. Hence, g_root.type=g_1.type=AND; g_2.type=OR; g_3.type = g_4.type = g_5.type = g_6.type = g_7.type = LEAF.

Figure 2 and Figure 3 respectively show the induced tree and the induced graph for this goal model.

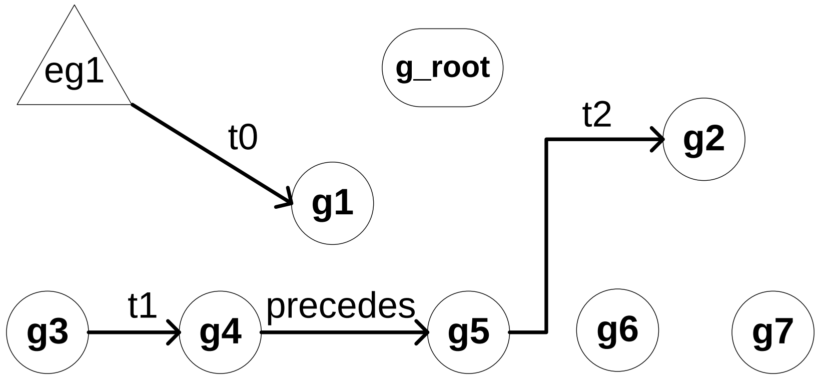

Figure 3. Example of Induced Graph

2.2. Role Model

We also organize the roles using a role model that is essentially a set of roles connected by protocols. There are two types of roles: internal roles and external roles. *Internal roles* are roles that are defined inside the organization. *External roles* represent placeholder for roles from external organizations. They represent an interface to the outside world, which will allow organizations to cater for interactions with unknown roles at design time. Eventually, either later on in the design or at runtime, external roles will be replaced by concrete roles (internal roles) from other organizations. Formally, a role model can be viewed as a directed graph having roles as nodes and protocol as edges such that an edge p from role r_1 to role r_2 would indicate a protocol p for which r_1 is the initiator and r_2 the responder. We assume that in a role model, protocols names are unique. This can be enforced by having a protocol naming scheme that takes into account the participants of that protocol. In addition, we assume that given two roles, there is at most one protocol between them. This assumption is valid as if there is more than one protocol between two roles, those protocols can be combined into one protocol having several alternate cases [19].

Definition 3: Role Model

A *role model* is a tuple RM = ⟨R, P, participant⟩ where:

- R: set of *roles*

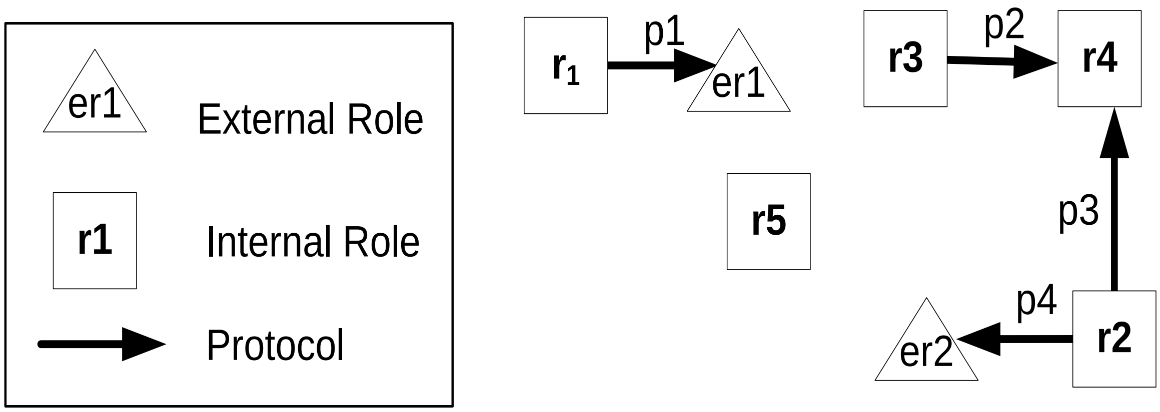

Figure 4. Example of Role Model

- P: set of *protocols*
- participants: $P \rightarrow R \times R$; indicates the pair of roles connected by a given protocol

In addition, we have $R = R_i \cup R_x$ ($R_i \cap R_x = \emptyset$), where R_x represents the set of *external* roles and R_i the set of *internal* roles.

This role model corresponds to a directed graph having roles as nodes and protocols as edges.

Example 2

Figure 4 represents the role model RM_example = $\langle R, P, participant \rangle$, where:

- $R = \{ r_1, r_2, r_3, r_4, r_5, er_1, er_2 \}$
- $P = \{ p_1, p_2, p_3, p_4 \}$
- participant = $\{(p_1, \langle r_1, er_1 \rangle), (p_2, \langle r_3, r_4 \rangle), (p_3, \langle r_2, r_4 \rangle), (p_4, \langle r_2, er_2 \rangle)\}$

2.3. Organization Structure

Finally, we define an organization as a goal model and a role model such that each leaf goal is achieved by a role.

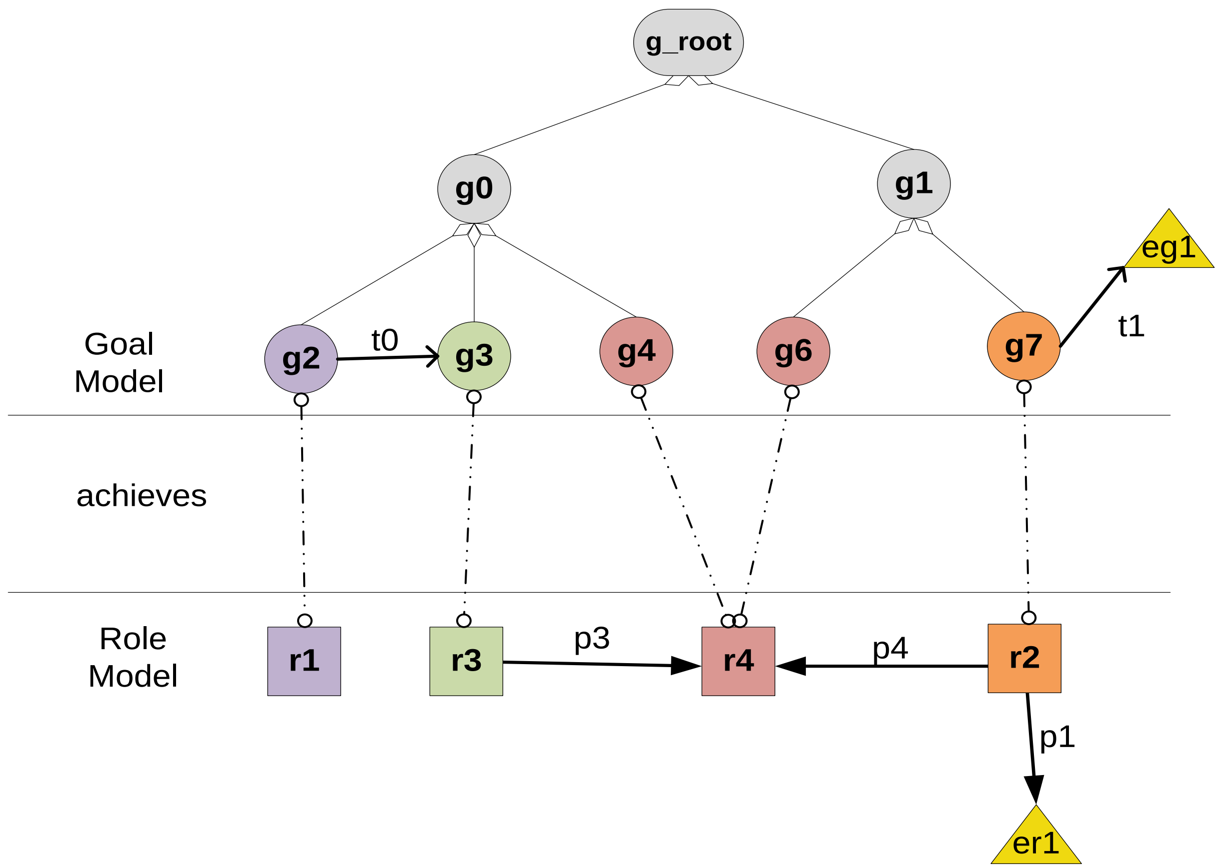

Figure 5. Example of Organization

Definition 4: Organization

An *organization* is a tuple O = ⟨GM, RM, achieves⟩ where:

- GM: Goal Model
- RM: Role Model
- achieves $\subseteq$ R $\times$ G_l: set of role-goal pairs such that the role achieves the goal.

Essentially, we can view an organization design as a directed graph with multiple node types and multiple edge types following the structure imposed by the goal and role model. The type of nodes and edges matches the corresponding organizational notions. Hence, the nodes can be of type *goal* or *role* while the edges can be of type *achieve*, *protocol*, *parent* or *time-based*.

Example 3

Figure 5 represents the organization ORG_example = ⟨GM, RM, achieves⟩, where:

- GM = ⟨G, ET, EG, g_root⟩ as depicted in the top part of Figure 5
- RM = ⟨R, P, participant⟩ as depicted in the bottom part of Figure 5
- achieves = $\{(r_1, g_2), (r_3, g_3), (r_4,g_4), (r_4,g_6), (r_2,g_7)\}$

3. Category Theory Preliminaries

Category Theory is a mathematical tool originally used to establish a uniform framework in order to study the relations between different mathematical structures appearing in various areas of mathematics such as algebra, topology and logic [14, 24].

There is a clear difference in approaches between set theory and category theory. Set theory characterizes a mathematical object by describing its inner structure, its members. However, category theory takes a different approach. Mathematical objects are black boxes only defined by their interactions with other objects. For this reason, Fiadeiro [12] talks about "the social life of objects" as the basis of category theory. Hence, category theory can be viewed as more "abstract" than set theory. Since in this language there is no way to look at the internal membership structure of objects, all the concepts must be defined by their relations with other objects, and these relations are established by the existence and the equality of particular morphisms. In computer science, category theory is very helpful and can be applied in areas such as algebraic specification, type theory, automata theory, programming language semantics, and graph rewriting [2].

In this section, we briefly introduce the key notions of category theory that are used. Those preliminaries do not constitute a proper introduction to category theory. The reader is referred to [14, 24], for a more elaborate introduction to category theory concepts. A computer science introduction to category theory is provided in [1, 2, 12, 29]. The definitions in this section are adapted from [12] and [15].

Definition 5: Graph

A *Graph* $G = \langle V, E\rangle$ is a mathematical structure consisting of two finite sets V and E. The elements of V are called *vertices* (or *nodes*) and the elements of E are called *edges*. Each edge has two vertices associated to it, which are called endpoints. If two vertices u and v are joined by an edge, this edge is denoted $|u,v|$.

G is a *directed graph* if the set of edges contains ordered pairs of vertices. A *path* represents a sequence of vertices such that from each vertex there is an edge to the next vertex in the sequence. A *cycle* is a path such that the start vertex and end vertex are the same. A graph is called *connected* if every pair of distinct vertices in the graph can be connected through some path.

Definition 6: Rooted Tree

A *rooted tree* $T = \langle V, E, r\rangle$ is a connected acyclic graph $\langle V, E\rangle$ in which vertex r has been designated the root.

Definition 7: Graph Homomorphism

Given two graphs $G_1 = \langle V_1, E_1\rangle$ and $G_2 = \langle V_2, E_2\rangle$, a *graph homomorphism* h from G_1 to G_2 consists of two functions $f : V_1 \to V_2$ and $g : E_1 \to E_2$, such that:

- if $e = |a,b| \in E_1$ then $g(e) = |f(a), f(b)| \in E_2$ (preserve edges)

Definition 8: Tree Homomorphism

Given two rooted trees $T_1 = \langle V_1, E_1, r_1\rangle$ and $T_2 = \langle V_2, E_2, r_2\rangle$, a *tree homomorphism* f from T_1 to T_2 consists of two functions $f : V_1 \to V_2$ and $h: E_1 \to E_2$, such that:

- $f(r_1) = r_2$ (preserve root)
- if $e=|a,b| \in E_1$ then $h(e) = |f(a),f(b)| \in E_2$ (preserve edges).

Definition 9: Category

A *category* C is given by a collection of *objects* and a collection of *morphisms* ("arrows") that have the following structure:

- Each morphism has a domain and a codomain that are objects; we write $f : X \to Y$ if $X = \mathrm{dom}(f)$ and $Y = \mathrm{cod}(f)$;

- Given two morphisms f and g such that cod(f) = dom(g), the composition of f and g, written g ○ f, is defined and has domain dom(f) and codomain cod(g);
- The composition is associative, that is: given $f : X \rightarrow Y$, $g : Y \rightarrow Z$ and $h : Z \rightarrow W$, h ○ (g ○ f) = (h ○ g) ○ f;
- For every object X there is an identity morphism $id_x : X \rightarrow X$, satisfying id_x ○ g = g for every $g : Y \rightarrow X$ and f ○ id_x for every $f : X \rightarrow Y$.

Essentially, a category is a mathematical structure that has objects and morphisms, with an associative composition operation on the morphisms and an identity morphism for each object. In other words, categories are graphs (with multiple directed edges) with a composition and identity structure.

Definition 10: Pushout

Let $f : X \rightarrow Y$ and $g : X \rightarrow Z$ be morphisms of a category C. A *pushout* of f and g consists of an object W and a pair of morphisms f': $Y \rightarrow W$ and g': $Z \rightarrow W$ such that:

- f' ○ f = g' ○ g
- For any other morphisms f'': $Y \rightarrow V$ and g'': $Z \rightarrow V$ such that f'' ○ f = g'' ○ g, there is a unique morphism k: $W \rightarrow V$ in C such that k ○ f' = f'' and k ○ g' = g'' .

This definition is illustrated in Figure 6.

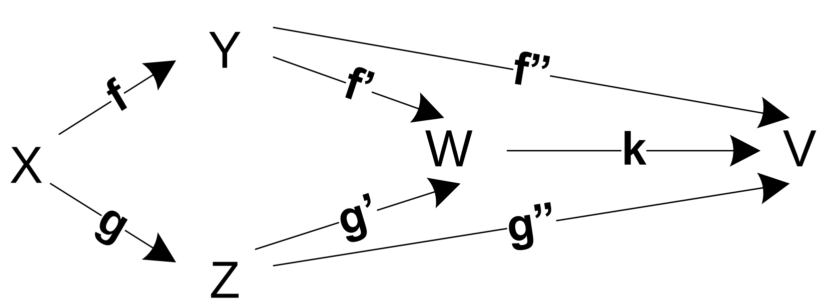

Figure 6. Pushout in a category

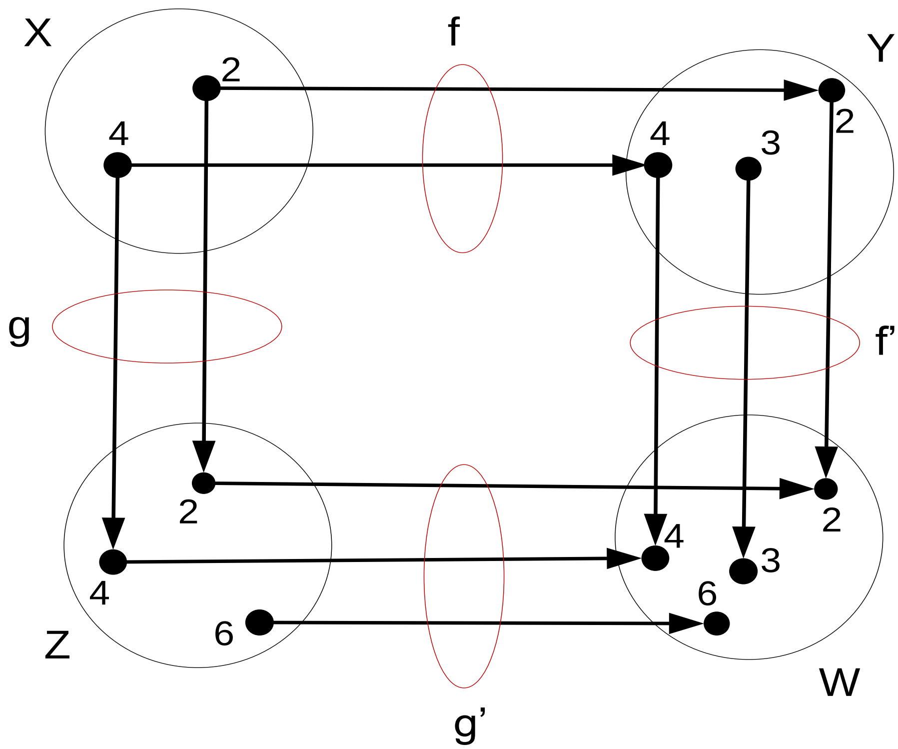

Figure 7. Pushout in the SET category

Examples 4

Figure 7 shows a pushout in SET. The proof that this diagram represents a pushout is outlined as follows. We have $f: X \to Y$ and $g : X \to Z$, $f': Y \to W$ and $g': Z \to W$ four functions such that:

$f = \{(2,2), (4,4)\}$, $g = \{(2,2), (4,4)\}$, $f' = \{(2,2), (4,4), (6,6)\}$, $g' = \{(2,2), (4,4), (3,3)\}$.

It easy to see that $f' \circ f = g' \circ g$. Moreover, assume that there exists two functions $f'': Y \to V$ and $g'': Z \to V$ such that $f'' \circ f = g'' \circ g$. If $k: W \to V$ is a function such that $k \circ f' = f''$ and $k \circ g' = g''$, then the fact that $f'' \circ f = g'' \circ g$ leaves no choice for the choice of k, which ensures uniqueness.

Remark that the object W computed by pushout of f and g in Figure 7 is just the union of Z and Y.

In general, the pushout allows us to merge objects based on their relationships without violating the requirements that are imposed on their structure and adding any unnecessary duplication of elements. In fact, as pointed out by Goguen [14], pushouts represent a construction to interconnect systems to form a larger systems.

4. Category of Goal Models

In this section, we define the category of Goal Model. We then introduce the key notions that will allow the composition of goal model via pushout.

Definition 11: Goal Model Homomorphism

Given two goal models $GM_1 = \langle G_1, ET_1, EG_1, root_1 \rangle$ and $GM_2 = \langle G_2, ET_2, EG_2, root_2 \rangle$, a *goal model homomorphism* from GM_1 to GM_2 is a function $\Gamma = \langle \mathbf{f, g, h} \rangle$ where:

- $f : G_1 \rightarrow G_2$, such that $f(root_1) = root_2$
- $g : ET_1 \rightarrow ET_2$, such that if $|a,b| \in ET_1$ then $g(|a,b|) = |f(a),f(b)| \in ET_2$
- $h : EG_1 \rightarrow EG_2$, such that if $|a,b| \in EG_1$ then $h(|a,b|) = |f(a), f(b)| \in EG_2$.

Note that |a,b| denotes an edge. This distinct notation allows edges to be easily differentiated from any other tuples.

Proposition 12: Category of goal models

Goal models along with goal model homomorphisms define the **category GOAL_MODEL**.

Proof:

Let us prove that goal models along with goal model homomorphisms form a category. For that, we need to identify the objects, morphisms and identity morphisms and verify that the composition of morphism exists and is associative.

Objects**:** The objects are goal models.

Morphisms**:** The morphisms are goal model homomorphisms.

Identity**:** The identity morphism is a function $id_{GM} = \langle id_G, id_{ET}, id_{EG} \rangle$ such that id_G is an identity function that maps each goal to itself, id_{ET} is an identity function that maps each induced tree edge to itself and id_{EG} is an identity function that maps each induced graph edge to itself.

Composition**:** Let X,Y,Z be three goal models and $\Gamma_1 = \langle f_1,g_1,h_1 \rangle$, $\Gamma_2 = \langle f_2,g_2,h_2 \rangle$ be two goal model homomorphisms such that: $\Gamma_1 : X \rightarrow Y$ and $\Gamma_2 : Y \rightarrow Z$.

The composition is defined as follow: $\Gamma_2 \circ \Gamma_1 = \langle f_2 \circ f_1, g_2 \circ g_1, h_2 \circ h_1 \rangle$.

Associativity**:** The goal model homomorphism consists of functions between sets. Hence, the assiociativity property is derived from the corresponding property of functions between sets [22]. □

Definition 13: Configurations of Goal Models

A *configuration of goal models* specifies all the mappings that will be used to merge two goal models. Given two goal models GM_1 and GM_2, a *configuration of goal models* GM_1 and GM_2 is a triplet $\mathbf{config_{goal} = \langle GM_0, \Gamma_1, \Gamma_2 \rangle}$ where:

- GM_0 is a goal model
- Γ_1 is a goal model homomorphism from GM_0 to GM_1
- Γ_2 is a goal model homomorphism from GM_0 to GM_2

Definition 14: Goal model composition

The *composition* of two goal models GM_1, GM_2 over a goal model configuration $\langle GM_0, \Gamma_1, \Gamma_2 \rangle$ is the goal model resulting from the pushout of Γ_1 and Γ_2 in category GOAL_MODEL.

Example 5

The composition of goal models GM_1, GM_2 over the configuration $\langle GM_0, \Gamma_1, \Gamma_2 \rangle$ is depicted in Figure 8. Goal model GM_3 represents the composed model and it is obtained by pushout. The mappings for the functions comprised in the goal model homomorphisms are shown in Figure 9.

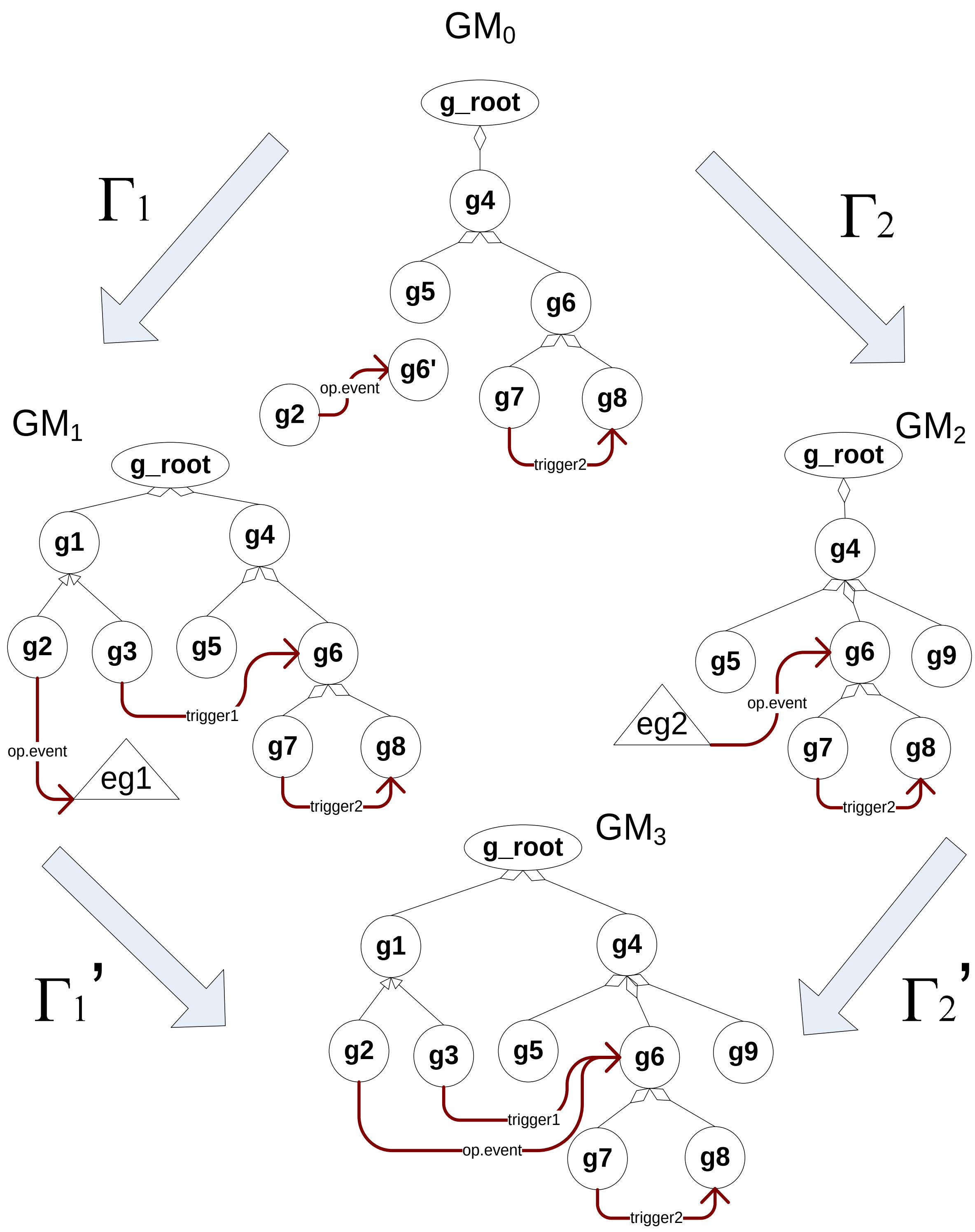

Figure 8. Overview of Pushout of Goal Models

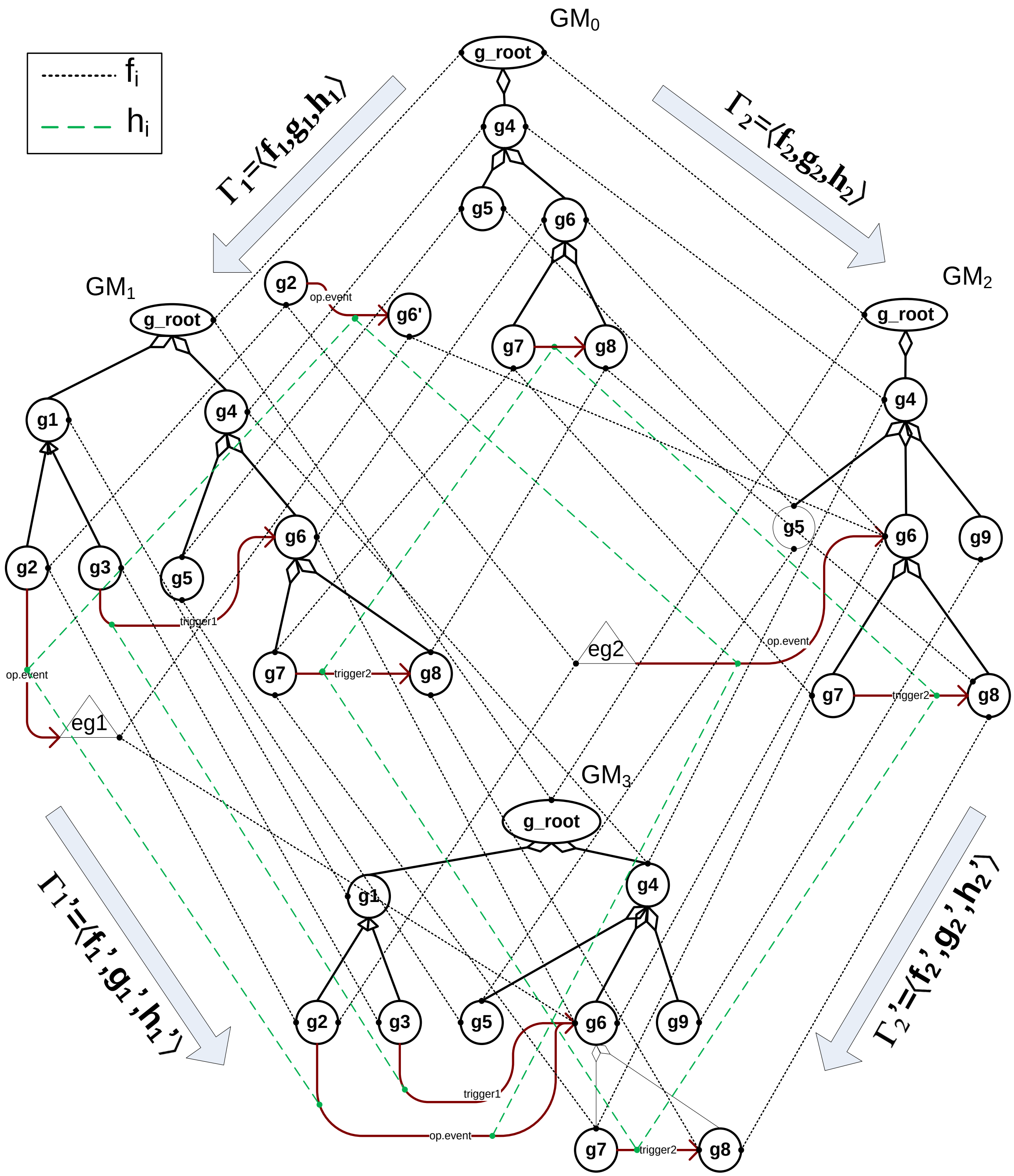

Figure 9. Pushout of Goal Models with detailed functions. Only functions mapping goals (f_i) and induced graph edges (h_i) are shown. Functions mapping tree edges (g_i) are not shown.

The elements for the pushout of goal models shown in Figure 9 are defined as follows:

Goal Model GM_0:

$GM_0 = \langle G_0, ET_0, EG_0, g\_root \rangle$,

$G_0 = \{ g\_root , g_2, g_4, g_5, g_6, g_6', g_7, g_8\}$,

$ET_0 = \{ |g\_root,g_4|, |g_4,g_5|, |g_4,g_6| , |g_6,g_7| , |g_6,g_8|\}$

$EG_0 = \{ |g_2,g_6'|, |g_7,g_8|\}$

Goal Model GM_1:

$GM_1 = \langle G_1, ET_1, EG_1, g\_root \rangle$,

$G_1 = \{ g\_root , g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, eg_1\}$,

$ET_1 = \{|g\_root, g_1|, |g\_root,g_4|, |g_1,g_2|, |g_1,g_3|, |g_4,g_5|, |g_4,g_6| , |g_6,g_7| , |g_6,g_8|\}$

$EG_1 = \{ |g_2,eg_1|, |g_3,g_6|, |g_7,g_8|\}$

Goal Model GM_2:

$GM_2 = \langle G_2, ET_2, EG_2, g\_root \rangle$,

$G_2 = \{ g\_root , g_4, g_5, g_6, g_7, g_8, g_9, eg_2\}$,

$ET_2 = \{ |g\_root,g_4|, |g_4,g_5|, |g_4,g_6| , |g_4,g_9| , |g_6,g_7|, |g_6,g_8|\}$

$EG_2 = \{ |eg_2, g_6|, |g_7,g_8|\}$

Goal Model GM_3:

$GM_3 = \langle G_3, ET_3, EG_3, g\_root \rangle$,

$G_3 = \{ g\_root , g_4, g_5, g_6, g_7, g_8, g_9, eg_2\}$,

$ET_3 = \{ \langle g\_root,g_4 \rangle, \langle g_4,g_5 \rangle, \langle g_4,g_6 \rangle , \langle g_4,g_9 \rangle , \langle g_6,g_7 \rangle, \langle g_6,g_8 \rangle\}$

$EG_3 = \{ \langle eg_2, g_6 \rangle, \langle g_7,g_8 \rangle\}$

Homomorphism Γ_1: (mappings f_1 and h_1 for Γ_1 are shown in Figure 9)

$\Gamma_1 = \langle f_1, g_1, h_1 \rangle$ such that: f_1: $G_0 \rightarrow G_1$, g_1: $ET_0 \rightarrow ET_1$, h_1: $EG_0 \rightarrow EG_1$. We have:

- $f_1 = \{\langle g\_root, g\_root \rangle, \langle g_4, g_4 \rangle, \langle g_5, g_5 \rangle, \langle g_6, g_6 \rangle, \langle g_7, g_7 \rangle, \langle g_8, g_8 \rangle, \langle g_2, g_2 \rangle, \langle g_6', eg_1 \rangle\}$;

- $g_1 = \{ \langle |g\_root,g_4|, |g\_root,g_4| \rangle, \langle |g_4,g_5|, |g_4,g_5| \rangle, \langle |g_4,g_6|, |g_4,g_6| \rangle, \langle |g_6,g_7|, |g_6,g_7| \rangle, \langle |g_6,g_8|, |g_6,g_8| \rangle \}$;
- $h_1 = \{\langle |g_2,g_6'|, |g_2, eg_1|\rangle, \langle |g_7,g_8|, |g_7,g_8|\rangle\}$;

Homomorphism Γ_2: (mappings f_2 and h_2 for Γ_2 are shown in Figure 9)

$\Gamma_2 = \langle f_2, g_2, h_2\rangle$ such that $f_2 : G_0 \rightarrow G_2$; g_2: $ET_0 \rightarrow ET_2$; h_2: $EG_0 \rightarrow EG_2$. We have:

- $f_2 = \{\langle g\_root, g\_root\rangle, \langle g_4, g_4\rangle, \langle g_5, g_5\rangle, \langle g_6, g_6\rangle, \langle g_7, g_7\rangle,\langle g_8, g_8\rangle,\langle g_2, eg_2\rangle,\langle g_6', g_6\rangle\}$;
- $g_2 =\{ \langle |g\_root,g_4|, |g\_root,g_4| \rangle, \langle |g_4,g_5|, |g_4,g_5| \rangle, \langle |g_4,g_6|, |g_4,g_6| \rangle, \langle |g_6,g_7|, |g_6,g_7| \rangle, \langle |g_6,g_8|, |g_6,g_8| \rangle \}$;
- $h_2 = \{\langle |g_2,g_6'|, |eg_2, g_6|\rangle, \langle |g_7,g_8|, |g_7,g_8|\rangle\}$;

Homomorphism Γ_1': (mappings f_1' and h_1' for Γ_1' are shown in Figure 9)

$\Gamma_1' = \langle f_1', g_1', h_1'\rangle$ such that f_1': $G_1 \rightarrow G_3$, g_1': $ET_1 \rightarrow ET_3$, h_1': $EG_1 \rightarrow EG_3$. We have:

- $f_1' = \{\langle g\_root, g\_root\rangle, \langle g_1, g_1\rangle, \langle g_2, g_2\rangle, \langle g_3, g_3\rangle, \langle g_4, g_4\rangle, \langle g_5, g_5\rangle, \langle g_6, g_6\rangle, \langle g_7, g_7\rangle, \langle g_8, g_8\rangle, \langle eg_1, g_6\rangle\}$;
- $g_1' = \{\langle |g\_root, g_1|, |g\_root, g_1|\rangle, \langle |g_1, g_2|, |g_1, g_2|\rangle, \langle |g_1, g_3|, |g_1, g_3|\rangle, \langle |g\_root,g_4|, |g\_root,g_4| \rangle, \langle |g_4,g_5|, |g_4,g_5| \rangle, \langle |g_4,g_6|, |g_4,g_6| \rangle, \langle |g_6,g_7|, |g_6,g_7| \rangle, \langle |g_6,g_8|, |g_6,g_8| \rangle\}$;
- $h_1' = \{\langle |g_2, eg_1|, |g_2,g_6|\rangle, \langle |g_7,g_8|, |g_7,g_8|\rangle, \langle |g_3,g_6|, |g_3,g_6|\rangle\}$;

Homomorphism Γ_2': (mappings f_2' and h_2' for Γ_2' are shown in Figure 9)

$\Gamma_2' = \langle f_2', g_2', h_2'\rangle$ such that f_2': $G_2 \rightarrow G_3$, g_2': $ET_2 \rightarrow ET_3$, h_2': $EG_2 \rightarrow EG_3$. We have:

- $f_2' = \{\langle g\_root, g\_root\rangle, \langle g_4, g_4\rangle, \langle g_5, g_5\rangle, \langle g_6, g_6\rangle, \langle g_7, g_7\rangle, \langle g_8, g_8\rangle, \langle g_9, g_9\rangle, \langle eg_2, g_2\rangle\}$;
- $g_2' = \{ \langle |g\_root,g_4|, |g\_root,g_4| \rangle, \langle |g_4,g_5|, |g_4,g_5| \rangle, \langle |g_4,g_6|, |g_4,g_6| \rangle, \langle |g_6,g_7|, |g_6,g_7| \rangle, \langle |g_6,g_8|, |g_6,g_8| \rangle, \langle |g_4, g_9|, |g_4, g_9|\rangle\}$;
- $h_2' = \{\langle |eg_2, g_6|, |g_2,g_6|\rangle, \langle |g_7,g_8|, |g_7,g_8|\rangle\}$;

GM_3 along with homomorphism Γ_1' and Γ_2' represent the pushout of GM_0 with homomorphism Γ_1 and Γ_2 . In fact, we have:

- $f_1' \circ f_1 = \{\langle g\_root, g\_root\rangle, \langle g_4, g_4\rangle, \langle g_5, g_5\rangle, \langle g_6, g_6\rangle, \langle g_7, g_7\rangle,\langle g_8, g_8\rangle,\langle g_2, g_2\rangle,\langle g_6', g_6\rangle\}$
- $f_2' \circ f_2 = \{\langle g\_root, g\_root\rangle, \langle g_4, g_4\rangle, \langle g_5, g_5\rangle, \langle g_6, g_6\rangle, \langle g_7, g_7\rangle,\langle g_8, g_8\rangle,\langle g_2, g_2\rangle,\langle g_6', g_6\rangle\}$
- $g_1' \circ g_1 = \{\langle |g\_root,g_4|, |g\_root,g_4| \rangle, \langle |g_4,g_5|, |g_4,g_5| \rangle, \langle |g_4,g_6|, |g_4,g_6| \rangle, \langle |g_6,g_7|, |g_6,g_7| \rangle, \langle |g_6,g_8|, |g_6,g_8| \rangle \}$;
- $g_2' \circ g_2 = \{\langle |g\_root,g_4|, |g\_root,g_4| \rangle, \langle |g_4,g_5|, |g_4,g_5| \rangle, \langle |g_4,g_6|, |g_4,g_6| \rangle, \langle |g_6,g_7|, |g_6,g_7| \rangle, \langle |g_6,g_8|, |g_6,g_8| \rangle \}$;
- $h_1' \circ h_1 = \{\langle |g_2,g_6'|, |g_2, g_6|\rangle, \langle |g_7,g_8|, |g_7,g_8|\rangle\}$;
- $h_2' \circ h_2 = \{\langle |g_2,g_6'|, |g_2, g_6|\rangle, \langle |g_7,g_8|, |g_7,g_8|\rangle\}$;

As $\Gamma_1' \circ \Gamma_1 = \langle f_1' \circ f_1, g_1' \circ g_1, h_1' \circ h_1\rangle$ and $\Gamma_2' \circ \Gamma_2 = \langle f_2' \circ f_2, g_2' \circ g_2, h_2' \circ h_2\rangle$, we have $\mathbf{\Gamma_1' \circ \Gamma_1 = \Gamma_2' \circ \Gamma_2}$. □

5. Category of Role Models

In this section, we define the category of Role Model. We then introduce the concepts that will allow the composition of role model via pushout.

Definition 15: Role models Homomorphism

Given two role models $RM_1 = \langle R_1, P_1, participant_1 \rangle$ and $RM_2 = \langle R_2, P_2, participant_2\rangle$, a *role model homomorphism* from RM_1 to RM_2 is a function $\mathbf{\Delta = \langle i, j\rangle}$ with $i : R_1 \rightarrow R_2$, $j : P_1 \rightarrow P_2$ such that:

- $\forall p \in P_1 | participant_1(p)= (r_1,r_2), j(p)= (i(r_1),i(r_2))$.

Proposition 16: Category of Role Models

Role models along with role model homomorphisms define the **category ROLE_MODEL**.

Proof:

Let us prove that role models along with role model homomorphisms form a category. For that, we need to identify the objects, morphisms and identity morphisms and verify that the composition of morphism exists and is associative.

Objects**:** The objects are role models.

Morphisms**:** The morphisms are role model homomorphisms.

Identity**:** The identity morphism is a function $id_{RM} = \langle id_R, id_P \rangle$ such that id_R is an identity function that maps each role to itself, id_P is an identity function that maps each protocol to itself.

Composition**:** Let RM_1, RM_2, RM_3 be three role models and $\Delta_1 = \langle i_1, j_1 \rangle$, $\Delta_2 = \langle i_2, j_2 \rangle$ be two role model homomorphisms such that: Δ_1: $RM_1 \rightarrow RM_2$ and Δ_2: $RM_2 \rightarrow RM_3$. The composition is defined as follow: $\Delta_2 \circ \Delta_1 = \langle i_2 \circ i_1, j_2 \circ j_1 \rangle$.

Associativity**:** The role model homomorphism consists of functions between sets. Hence, the assiociativity property is derived from the corresponding property of functions between sets [22]. □

Definition 17: Configurations of Role Models

A *configuration of role models* specifies all the mappings that will be used to merge two role models. Given two role models RM_1 and RM_2, a *configuration of role models* RM_1 and RM_2 is a triplet $\mathbf{config_{role} = \langle RM_0, \Delta_1, \Delta_2 \rangle}$ where:

- RM_0 is a role model
- Δ_1 corresponds to a role model homomorphism from RM_0 to RM_1
- Δ_2 corresponds to a role model homomorphism from RM_0 to RM_2

Definition 18: Role model composition

The *composition of two role models* RM_1, RM_2 over a role model configuration $\langle RM_0, \Delta_1, \Delta_2 \rangle$ is the role model resulting from the pushout of Δ_1 and Δ_2 in category ROLE_MODEL.

Example 6

The composition of role models RM_1, RM_2 over the configuration $\langle RM_0, \Delta_1, \Delta_2 \rangle$ is depicted in Figure 10. Role model RM_3 represents the composed model and it is obtained by pushout. The mappings for the functions comprised in the role model homomorphisms are shown in Figure 11.

The elements for the pushout of role models shown in Figure 11 are defined as follows:

Role Model RM_0:

$RM_0 = \langle R_0, P_0, participant_0 \rangle$,

$R_0 = \{ r_1, r_2, r_4, r_6, r_6' \}$

$P_0 = \{ p_1, p_2 \}$

$participant_0 = \{\langle p_1, \langle r_1, r_6' \}\rangle, \langle p_2, \langle r_2, r_6 \}\rangle\}$

Role Model RM_1:

$RM_1 = \langle R_1, P_1, participant_1 \rangle$,

$R_1 = \{ r_1, r_2, r_3, r_4, r_6, er_1, er_2 \}$

$P_1 = \{ p_1, p_2, p_3, p_4 \}$

$participant_1 = \{\langle p_1, \langle r_1, er_1 \rangle\rangle, \langle p_2, \langle r_2, r_6 \rangle\rangle, \langle p_3, \langle r_3, r_4 \rangle\rangle, \langle p_4, \langle r_3, er_2 \rangle\rangle\}$

Role Model RM_2:

$RM_2 = \langle R_2, P_2, participant_2 \rangle$,

$R_2 = \{ r_2, r_4, r_5, r_6, er_3 \}$

$P_2 = \{ p_1, p_2, p_5 \}$

$participant_1 = \{\langle p_1, \langle er_3, r_6 \rangle\rangle, \langle p_2, \langle r_2, r_6 \rangle\rangle, \langle p_5, \langle r_2, r_5 \rangle\rangle\}$

Role Homomorphism Δ_1: (mappings i_1 and j_1 for Δ_1 are shown in Figure 11)

$\Delta_1 = \langle i_1, j_1 \rangle$ such that $i_1: R_0 \rightarrow R_1$, $j_1: P_0 \rightarrow P_1$. We have:

- $i_1 = \{\langle r_1, r_1 \rangle, \langle r_2, r_2 \rangle, \langle r_4, r_4 \rangle, \langle r_6, r_6 \rangle, \langle r_6', er_1 \rangle\}$;
- $j_1 = \{\langle p_1, p_1 \rangle, \langle p_2, p_2 \rangle\}$;

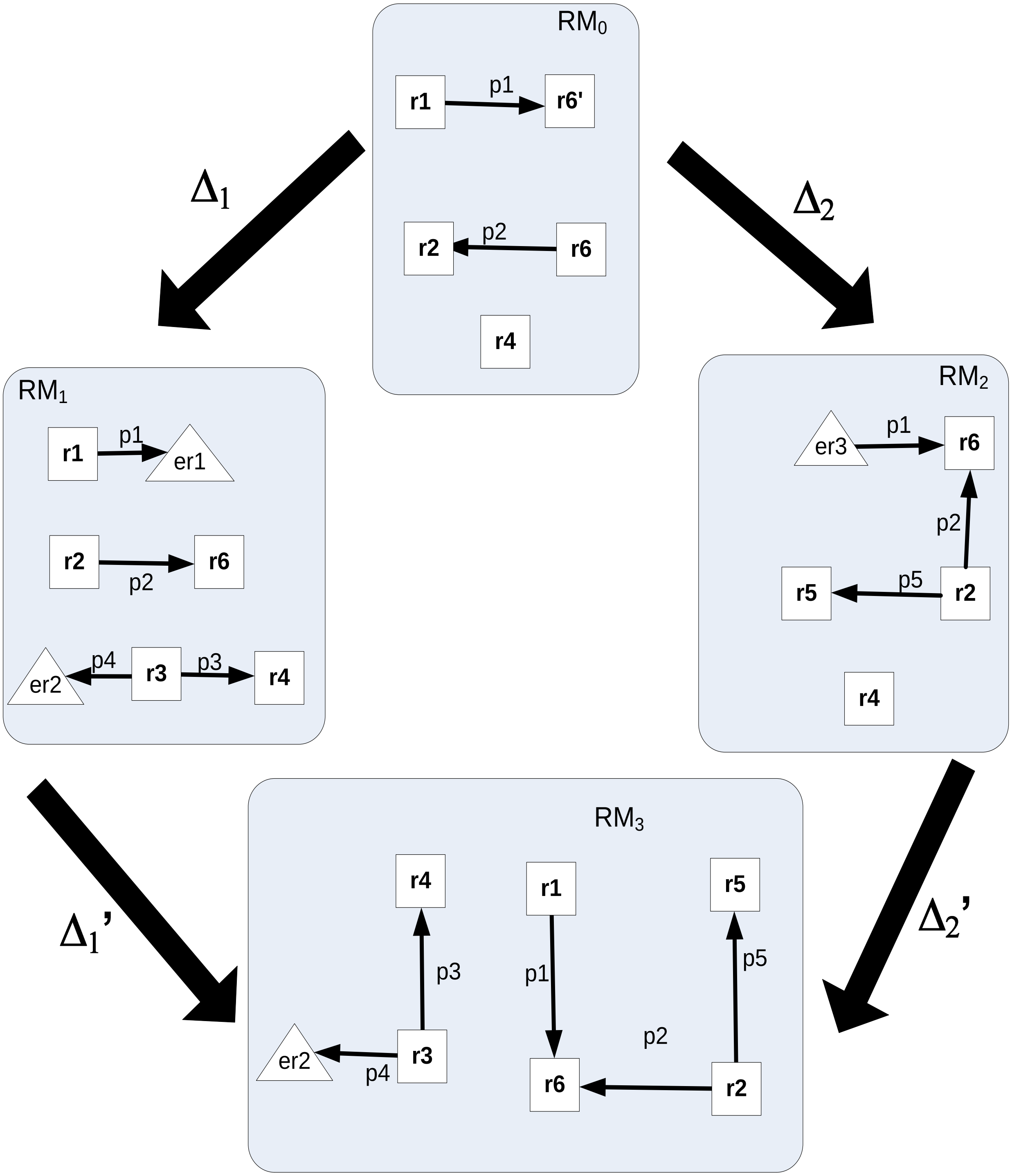

Figure 10. Overview of Pushout of Role Models

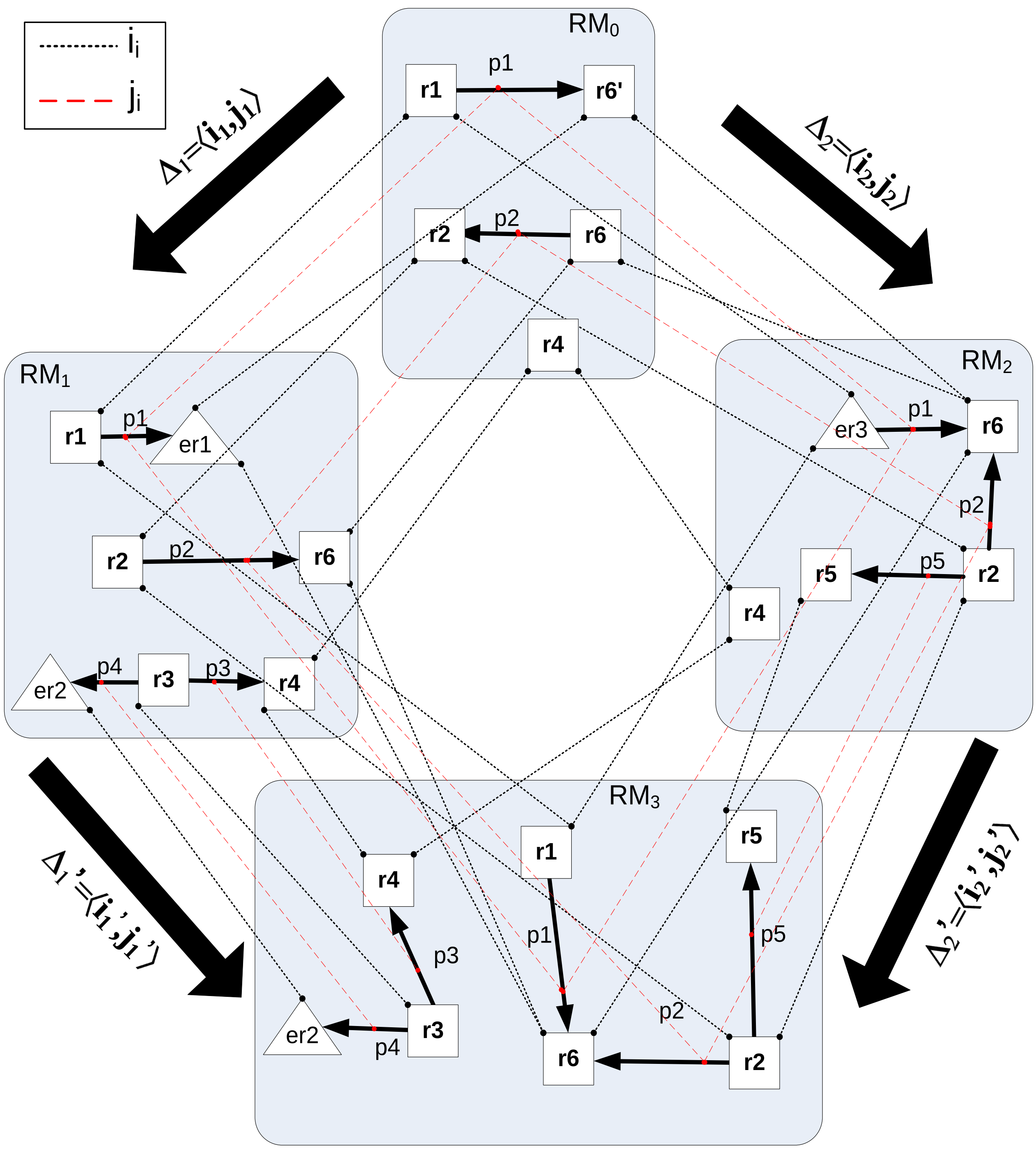

Figure 11. Pushout of Role Models with detailed functions. Functions mapping roles (i_i) and protocols (j_i) are shown.

Role Homomorphism Δ_2: (mappings i_2 and j_2 for Δ_2 are shown in Figure 11)

$\Delta_2 = \langle i_2, j_2 \rangle$ such that $i_2: R_0 \rightarrow R_2$, $j_2: P_0 \rightarrow P_2$. We have:

- $i_2 = \{\langle r_1, er_3\rangle, \langle r_2, r_2\rangle, \langle r_4, r_4\rangle, \langle r_6, r_6\rangle, \langle r_6', r_6\rangle\}$;
- $j_2 = \{\langle p_1, p_1\rangle, \langle p_2, p_2\rangle\}$;

Role Homomorphism Δ_1': (mappings i_1' and j_1' for Δ_1' are shown in Figure 11)

$\Delta_1' = \langle i_1', j_1'\rangle$ such that $i_1': R_1 \rightarrow R_3$, $j_1': P_1 \rightarrow P_3$. We have:

- $i_1' = \{\langle r_1, r_1\rangle, \langle r_2, r_2\rangle, \langle r_3, r_3\rangle, \langle r_4, r_4\rangle, \langle r_6, r_6\rangle, \langle er_1, r_6\rangle, \langle er_2, er_2\rangle\}$;
- $j_1' = \{\langle p_1, p_1\rangle, \langle p_2, p_2\rangle, \langle p_3, p_3\rangle, \langle p_4, p_4\rangle\}$;

Role Homomorphism Δ_2': (mappings i_2' and j_2' for Δ_2' are shown in Figure 11)

$\Delta_2' = \langle i_2', j_2' \rangle$ such that $i_2': R_2 \rightarrow R_3$, $j_2': P_2 \rightarrow P_3$. We have:

- $i_2' = \{\langle r_2, r_2\rangle, \langle r_4, r_4\rangle, \langle r_5, r_5\rangle, \langle r_6, r_6\rangle, , \langle er_3, r_1\rangle\}$;
- $j_2' = \{\langle p_1, p_1\rangle, \langle p_2, p_2\rangle, \langle p_5, p_5\rangle\}$;

RM_3 along with homomorphism Δ_1' and Δ_2' represent the pushout of RM_0 with homomorphism Δ_1 and Δ_2 . In fact, we have:

- $i_1' \circ i_1 = \{\langle r_1, r_1\rangle, \langle r_2, r_2\rangle, \langle r_4, r_4\rangle, \langle r_6, r_6\rangle, \langle r_6', r_6\rangle\}$
- $i_2' \circ i_2 = \{\langle r_1, r_1\rangle, \langle r_2, r_2\rangle, \langle r_4, r_4\rangle, \langle r_6, r_6\rangle, \langle r_6', r_6\rangle\}$
- $j_1' \circ j_1 = \{\langle p_1, p_1\rangle, \langle p_2, p_2\rangle\}$
- $j_2' \circ j_2 = \{\langle p_1, p_1\rangle, \langle p_2, p_2\rangle\}$

As $\Delta_1' \circ \Delta_1 = \langle i_1' \circ i_1, j_1' \circ j_1\rangle$ and $\Delta_2' \circ \Delta_2 = \langle i_2' \circ i_2, j_2' \circ j_2\rangle$, we have $\boldsymbol{\Delta_1' \circ \Delta_1 = \Delta_2' \circ \Delta_2}$.
□

6. Composition of Organization Models

In this section, we define the category of Organizations. We then introduce the concepts that will allow the composition of organizations via pushout.

Definition 19: Organizations Homomorphism

Given two organizations $ORG_1 = \langle GM_1, RM1, achieves_1\rangle$, $ORG_2 = \langle GM_2, RM2, achieves_2\rangle$, an *organization homomorphism* $\Phi = \langle \Gamma, \Delta, k\rangle$ from ORG_1 to ORG_2 consists of:

- $\Gamma = \langle f, g, h\rangle$: goal model homomorphism from GM_1 to GM_2
- $\Delta = \langle i, j\rangle$: role model homomorphism from RM_1 to RM_2
- $k : achieves_1 \rightarrow achieves_2$, such that if $|r,g| \in achieves_1$, then $k(|r,g|) \in achieves_2$ and $k(|r,g|) = |\, i(r), f(g)|$

Proposition 20: Category of Organizations

Organizations along with organization homomorphisms define the **category ORG_MODEL**.

Proof:

Let us prove that organizations along with organization homomorphisms form a category. For that, we need to identify the objects, morphisms and identity morphisms and verify that the composition of morphism exists and is associative.

Objects**:** The objects are organizations.

Morphisms**:** The morphisms are organization homomorphisms.

Identity**:** The identity morphism is a function $id_{ORG} = \langle id_{GM}, id_{RM}, id_k \rangle$ such that id_{GM} is an identity function that maps each goal model to itself (as defined in Section 4.), id_{RM} is an identity function that maps each role model to itself (as defined in Section 5.) and id_k is an identity function that maps each *achieves* edge to itself.

Composition**:** Let ORG_1, ORG_2, ORG_3 be three organizations and $\Phi_1 = \langle \Gamma_1, \Delta_1, k_1\rangle$, $\Phi_2 = \langle \Gamma_2, \Delta_2, k_2 \rangle$ be two organization homomorphisms such that: $\Phi_1: ORG_1 \rightarrow ORG_2$ and $\Phi_2: ORG_2 \rightarrow ORG_3$. The composition is defined as follow:

$$\Phi_2 \text{ o } \Phi_1 = \langle \Gamma_2 \text{ o } \Gamma_1, \Delta_2 \text{ o } \Delta_1, k_2 \text{ o } k_1\rangle.$$

Associativity**:** An organization homomorphism consists of functions between sets (Goal model homomorphisms and role model homomorphisms are set functions). Hence, the assiociativity property is derived from the corresponding property of functions between sets [22]. □

Definition 21: Configuration of Organizations

A *configuration of organizations* specifies all the mappings that will be used to merge two organizations. Given two organizations ORG_1 and ORG_2, a *configuration of organizations* ORG_1 and ORG_2 is a triplet **config=** $\langle \mathbf{ORG_0, \Phi_1, \Phi_2} \rangle$ where:

- ORG_0 is an organization
- Φ_1 corresponds to an organization homomorphism from ORG_0 to ORG_1
- Φ_2 corresponds to an organization homomorphism from ORG_0 to ORG_2

Definition 22: Composition of Organizations

The *composition* of two organizations ORG_1, ORG_2 over a configuration of organization config = $\langle ORG_0, \Phi_1, \Phi_2 \rangle$ is the organization resulting from the pushout of Φ_1 and Φ_2 in category ORG_MODEL.

Notation:

This composition is noted $\vdash (\mathbf{ORG_1, ORG_2, config}) = \mathbf{ORG_1} \vdash^{\mathbf{config}} \mathbf{ORG_2}$.

The general intuition behind the pushout construction is that it aggregates the unrelated organization structures together without adding anything new and merges the shared structure defined in the configuration. It results in a composite organization that has all elements of both organizations while eliminating duplicates identified in the shared part. In fact, we are interested in composing two organizations that have some elements in common. Actually, composing two completely unrelated organizations is possible but uninteresting.

Example 7

Figure 12 shows an example of composition of organization ORG_1 and ORG_2 over the configuration $\langle ORG_0, \Phi_1, \Phi_2 \rangle$. This composition results in the pushout organization ORG_3 as depicted in Figure 12. The goal models and role models from the organizations shown here have been studied in Example 5 and Example 6. Therefore, we will not go into the details of the mapping for the goal models and roles models. We will just give the details for the *achieves* mappings.

Organization ORG_0:

GM_0: defined in Example 5.

RM_0: defined in Example 6.

$achieves_0 = \{|r_4, g_5|, |r_6, g_7|, |r_2, g_8|\}$

Organization ORG_1:

GM_1: defined in Example 5.

RM_1: defined in Example 6.

$achieves_1 = \{|r_1, g_2|, |r_3, g_3|, |r_4, g_5|, |r_6, g_7|, |r_2, g_8|\}$

Organization ORG_2:

GM_2: defined in Example 5.

RM_2: defined in Example 6.

$achieves_2 = \{|r_4, g_5|, |r_6, g_7|, |r_2, g_8|, |r_5, g_9|\}$

Organization Homomorphism Φ_1: (mappings k_1 is shown in Figure 11)

$\Phi_1 = \langle \Gamma_1, \Delta_1, k_1 \rangle$ where Γ_1 and Δ_1 have been defined in Example 5 and Example 6 and k_1: $achieves_0 \rightarrow achieves_1$. We have:

- $k_1 = \{\langle |r_4, g_5|, |r_4, g_5| \rangle, \langle |r_6, g_7|, |r_6, g_7| \rangle, \langle |r_2, g_8|, |r_2, g_8| \rangle\}$;

Organization Homomorphism Φ_2: (mappings of k_2 is shown in Figure 11)

$\Phi_2 = \langle \Gamma_2, \Delta_2, k_2 \rangle$ where Γ_2 and Δ_2 have been defined in Example 5 and Example 6 and k_2: $achieves_0 \rightarrow achieves_2$. We have:

- $k_2 = \{\langle |r_4, g_5|, |r_4, g_5| \rangle, \langle |r_6, g_7|, |r_6, g_7| \rangle, \langle |r_2, g_8|, |r_2, g_8| \rangle\}$;

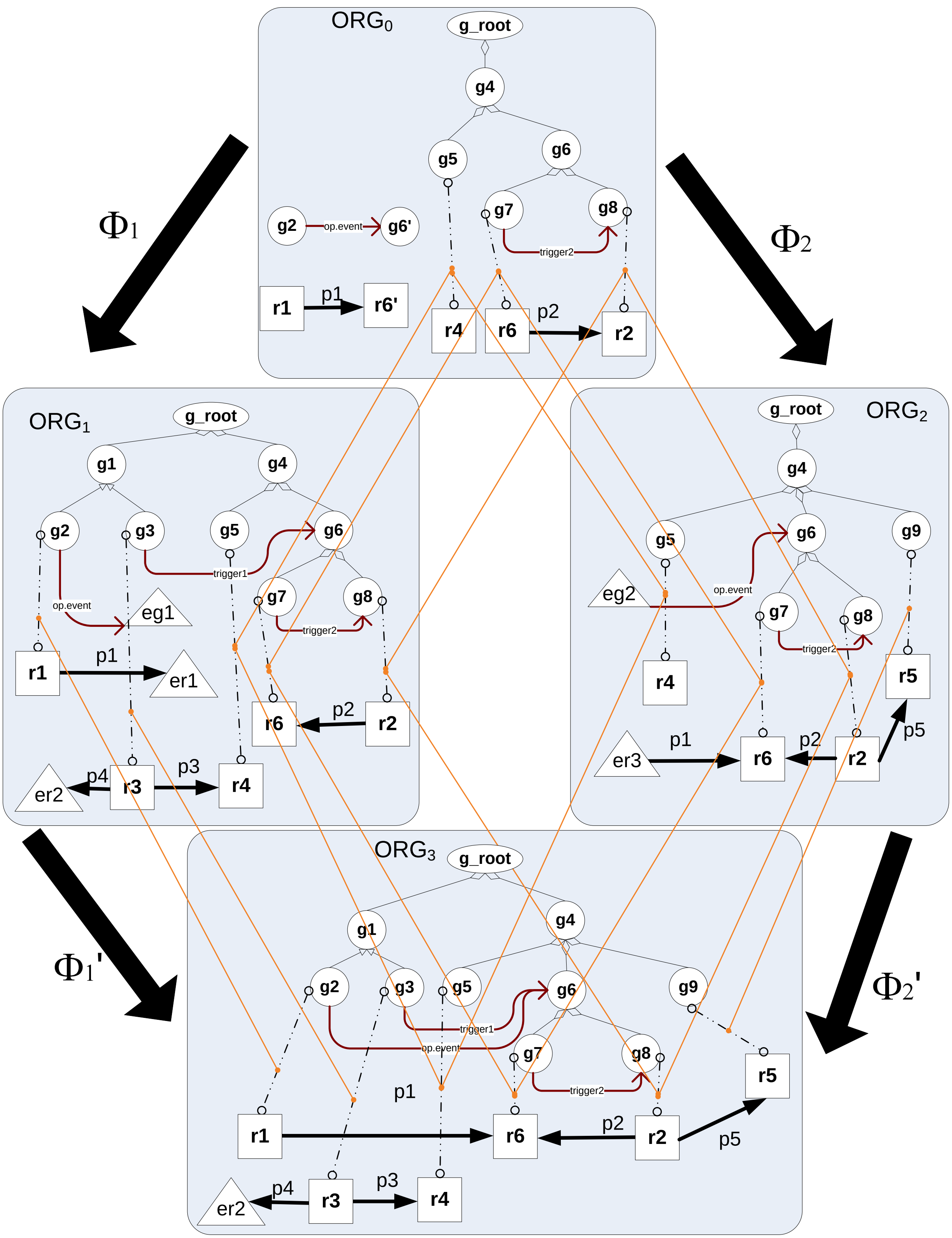

Figure 12. Pushout of Organizations. Only *achieves edges* mappings (k_i) are shown.

Organization Homomorphism Φ_1': (mappings k_1' is shown in Figure 11)

$\Phi_1' = \langle \Gamma_1', \Delta_1', k_1' \rangle$ where Γ_1' and Δ_1' have been defined in Example 5 and Example 6 and k_1': $achieves_1 \rightarrow achieves_3$. We have:

$k_1' = \{\langle |r_1, g_2|, |r_1, g_2| \rangle, \langle |r_3, g_3|, |r_3, g_3| \rangle, \langle |r_4, g_5|, |r_4, g_5| \rangle, \langle |r_6, g_7|, |r_6, g_7| \rangle, \langle |r_2, g_8|, |r_2, g_8| \rangle\}$;

Organization Homomorphism Φ_2': (mappings k_2' is shown in Figure 11)

$\Phi_2' = \langle \Gamma_2', \Delta_2', k_2' \rangle$ where Γ_2' and Δ_2' have been defined in Example 5 and Example 6 and k_2': $achieves_2 \rightarrow achieves_3$. We have:

$k_2' = \{\langle |r_5, g_9|, |r_5, g_9| \rangle, \langle |r_4, g_5|, |r_4, g_5| \rangle, \langle |r_6, g_7|, |r_6, g_7| \rangle, \langle |r_2, g_8|, |r_2, g_8| \rangle\}$;

ORG_3 along with homomorphism Φ_1' and Φ_2' represent the pushout of ORG_0 with homomorphism Φ_1 and Φ_2 . In fact, we have:

- $k_1' \circ k_1 = \{\langle |r_4, g_5|, |r_4, g_5| \rangle, \langle |r_6, g_7|, |r_6, g_7| \rangle, \langle |r_2, g_8|, |r_2, g_8| \rangle\}$;
- $k_2' \circ k_2 = \{\langle |r_4, g_5|, |r_4, g_5| \rangle, \langle |r_6, g_7|, |r_6, g_7| \rangle, \langle |r_2, g_8|, |r_2, g_8| \rangle\}$;

Hence, we have $k_1' \circ k_1 = k_2' \circ k_2$. Moreover, we have shown that $\Gamma_1' \circ \Gamma_1 = \Gamma_2' \circ \Gamma_2$ (Example 5) and $\Delta_1' \circ \Delta_1 = \Delta_2' \circ \Delta_2$ (Example 6). As $\Phi_1' \circ \Phi_1 = \langle \Gamma_1' \circ \Gamma_1, \Delta_1' \circ \Delta_1, k_1' \circ k_1 \rangle$ and $\Phi_2' \circ \Phi_2 = \langle \Gamma_2' \circ \Gamma_2, \Delta_2' \circ \Delta_2, k_2' \circ k_2 \rangle$, we have $\boldsymbol{\Phi_1' \circ \Phi_1 = \Phi_2' \circ \Phi_2}$. □

7. Related Works

The problem of composing models has been studied in various domains [4] and many approaches have proposed the use of the notion of colimit in category theory as a formalism to compose various types of models. For instance, some works have been done to compose UML models [3, 16], requirement models [28, 30], statechart models [25], database schemas [5, 27], ontologies [6, 17] and programs [26].

In the multiagent systems community, there are very few works unifying category theory and multiagent systems. Most of those types of research are done at the implementation level. For instance, Johnson et al. [20] use category theory to formalize the composition multiagent dialogue protocols while Soboll [31] proposes to model multiagent cooperation patterns as categories. However, none of those approaches explicitly considers organizational designs. In this report, we proposed a formal approach to compose a set of interrelated models that compose a multiagent organization design.

8. Acknowledgements

This work was supported by grants from the US National Science Foundation (0347545) and the US Air Force Office of Scientific Research (FA9550-06-1-0058).

9. Conclusion

We have presented a framework for the compositional design of Organization-based Multiagent Systems (OMAS). This framework uses category theory to formalize organization designs consisting of goal and role models. The main contribution of this report is to provide an abstract mechanism for merging OMAS designs. We have shown that the composition of multiagent organizations can be formulated using the pushout notion in category theory. We defined three main categories, GOAL_MODEL, ROLE_MODEL and ORG_MODEL, as the category of goal models, role models and organization models respectively. Then, we have defined the notion of *organization homomorphisms* and specified the composition of organization as the pushout object of organization homomorphisms. Nevertheless, finding suitable organization homomorphisms is not an easy task. Moreover, arbitrary homomorphisms could potentially lead to semantically incorrect composite organizations that cannot be implemented into a coherent system. In future work, we are investigating how we can provide a specific approach that will guide designers to decide what organizations to compose. Such approach will guarantee the construction of correct homomorphisms that can be used in the composition by pushout.

10. References

[1] S. Awodey, *Category theory*. 2006: Oxford University Press, USA.

[2] M. Barr and C. Wells, *Category theory for computing science*. Prentice Hall international series in computer science. 1990, New York: Prentice Hall.

[3] A. Boronat, et al., *Formal Model Merging Applied to Class Diagram Integration.* Electronic Notes in Theoretical Computer Science, 2007. **166**: p. 5-26.

[4] G. Brunet, et al., *A manifesto for model merging*, in *Proceedings of the 2006 international workshop on Global integrated model management*. 2006, ACM: Shanghai, China.

[5] P. Buneman, S. Davidson, and A. Kosky, *Theoretical aspects of schema merging*, in *Advances in Database Technology — EDBT '92*. 1992. p. 152-167.

[6] I. Cafezeiro and E.H. Haeusler, *Semantic interoperability via category theory*, in *26th international conference on Conceptual modeling*. 2007: Auckland, New Zealand.

[7] S.A. DeLoach, *OMACS: a Framework for Adaptive, Complex Systems*, in *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, V. Dignum, Editor. 2009, IGI Global: Hershey, PA.

[8] S.A. DeLoach and M. Miller, *A Goal Model for Adaptive Complex Systems.* International Journal of Computational Intelligence: Theory and Practice, 2010. **5**(2).

[9] S.A. DeLoach, W.H. Oyenan, and E. Matson, *A capabilities-based model for adaptive organizations.* Autonomous Agents and Multi-Agent Systems, 2008. **16**(1): p. 13-56.

[10] A. Estefania, J. Vicente, and B. Vicente, *Multi-Agent System Development Based on Organizations.* Electronic Notes in Theoretical Computer Science, 2006. **150**(3): p. 55-71.

[11] J. Ferber, O. Gutknecht, and F. Michel, *From Agents to Organizations: An Organizational View of Multi-agent Systems*, in *Agent-Oriented Software Engineering IV*. 2004. p. 443-459.

[12] J.L. Fiadeiro, *Categories for software engineering*. 2005: Springer.

[13] J.C. Garcia-Ojeda, et al. *O-MaSE: A Customizable Approach to Developing Multiagent Development Processes*. in *8th International Workshop on Agent Oriented Software Engineering* 2007.

[14] J.A. Goguen, *A categorical manifesto.* Mathematical Structures in Computer Science, 1991.

[15] P. Hell and J. Nešetril, *Graphs and homomorphisms*. 2004: Oxford University Press, USA.

[16] C. Herrmann, et al., *An Algebraic View on the Semantics of Model Composition*, in *Model Driven Architecture- Foundations and Applications*. 2007. p. 99-113.

[17] P. Hitzler, et al. *What Is Ontology Merging?* 2005.

[18] M.P. Huget, *Representing Goals in Multiagent Systems*, in *Proc. 4th Int'l Symp. Agent Theory to Agent Implementation*. 2004. p. 588–593.

[19] M.P. Huget and J. Odell. *Representing agent interaction protocols with agent UML*. in *Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004. Proceedings of the Third International Joint Conference on*. 2004.

[20] M.W. Johnson, P. McBurney, and S. Parsons, *A Mathematical Model of Dialog.* Electronic Notes in Theoretical Computer Science, 2005. **141**(5): p. 33-48.

[21] T. Juan, A. Pearce, and L. Sterling, *ROADMAP: extending the gaia methodology for complex open systems*, in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*. 2002, ACM: Bologna, Italy.

[22] H.J. Keisler, *Elementary calculus*. Bull. Amer. Math. Soc. 83 (1977), 205-208. DOI: 10.1090/S0002-9904-1977-14264-X PII: S 0002-9904 (1977) 14264-X, 1977.

[23] M. Kolp, P. Giorgini, and J. Mylopoulos, *Multi-Agent Architectures as Organizational Structures*. Autonomous Agents and Multi-Agent Systems, 2006. **13**(1): p. 3-25.

[24] S. Mac Lane, *Categories for the working mathematician*. 1998: Springer verlag.

[25] S. Nejati, et al., *Matching and Merging of Statecharts Specifications*, in *Proceedings of the 29th international conference on Software Engineering*. 2007, IEEE Computer Society.

[26] N. Niu, S. Easterbrook, and M. Sabetzadeh, *A Category-theoretic Approach to Syntactic Software Merging*, in *Proceedings of the 21st IEEE International Conference on Software Maintenance*. 2005, IEEE Computer Society.

[27] R.A. Pottinger and P.A. Bernstein, *Merging models based on given correspondences*, in *Proceedings of the 29th international conference on Very large data bases - Volume 29*. 2003, VLDB Endowment: Berlin, Germany.

[28] D. Richards, *Merging individual conceptual models of requirements*. Requirements Engineering, 2003. **8**(4): p. 195-205.

[29] D.E. Rydeheard and R.M. Burstall, *Computational category theory*. Prentice Hall International (UK) Ltd. Hertfordshire, UK, UK. 1988: Prentice Hall, 1988. 257.

[30] M. Sabetzadeh and S. Easterbrook. *An algebraic framework for merging incomplete and inconsistent views*. in *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*. 2005.

[31] T. Soboll, *On the Construction of Transformation Steps in the Category of Multiagent Systems*, in *Intelligent Computer Mathematics*. p. 184-190.

[32] A. van Lamsweerde, et al., *The KAOS Project: Knowledge Acquisition in Automated Specification of Software*. Proceedings AAAI Spring Symposium Series, 1991: p. 59-62.

[33] M.F. Wood and S.A. DeLoach, *An overview of the multiagent systems engineering methodology*, in *First international workshop, AOSE 2000 on Agent-oriented software engineering*. 2001, Springer-Verlag New York, Inc.: Limerick, Ireland.

[34] F. Zambonelli, N.R. Jennings, and M. Wooldridge, *Developing multiagent systems: The Gaia methodology*. ACM Trans. Softw. Eng. Methodol., 2003. **12**(3): p. 317-370.