Electronic Communications of the EASST
Volume 12 (2008)

Formal Modeling of Adaptive and Mobile Processes

On a General Notion of Transformation for Multiagent Systems and its Implementation

Jochen Pfalzgraf, Thomas Soboll

21 pages

Guest Editors: Julia Padberg, Kathrin Hoffmann
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

On a General Notion of Transformation for Multiagent Systems and its Implementation

Jochen Pfalzgraf, Thomas Soboll

University of Salzburg
Department of Computer Sciences
A-5020 Salzburg, Austria
{jpfalz, tsoboll}@cosy.sbg.ac.at

Abstract: The focus of this contribution is on the construction of a transformation system for Multiagent Systems (MAS) based on categorical notions. Based on former work on the categorical modeling of MAS the category $\mathbb{MAS}$ of all Multiagent Systems is introduced. A transformation system over this category is established using the Double Pushout Approach. For illustration we present a simple example. First steps of implementational work are described.

Keywords: Multiagent Systems (MAS), Graph Transformation, Category Theory (CAT), Transformation Systems, Double Pushout Approach (DPO), Implementations

1 Introduction

In recent work a generic categorical model for MAS has been introduced leading to the category MAS (cf. [Pfa05] for a first step). In that category the objects are agents of various types and the morphisms represent all kinds of relations between the agents, we call it general communication and cooperation arrows. This general communication and cooperation structure is represented by a corresponding arrow diagram, called Base Diagram of a MAS.

Of basic importance for our work is the observation that every arrow diagram (i.e. directed graph, possibly with labels on the arrows) can be interpreted as a category named PATH - c.f. [Pfa94] - morphisms are sequences (paths) of arrows. This viewpoint leads to a general categorical semantics for relational structures. Vice versa, every category is a graphical structure (with nodes and arrows). We point out that the identity morphisms can always be assumed to exist, artificially.

The very idea of MAS is to solve problems decentralized by autonomous actors (the agents), this leads to a wide field of applications which resort to MAS techniques. Until now there is no general, unique definition of agent and Multiagent System. We are convinced that there is a strong need for a formalization of MAS. It is our goal to develop a toolbox for MAS modeling using categorical notions. Typical characteristics of Multiagent Systems can be summarized by the following statement: Each agent has only local information and a limited ”sphere of influence”, there is no global system control, information is available only in a decentralized manner and the processes are asynchronous [Woo02].

A Multiagent System can be modeled with categorical notions by typed categories which we introduce in this paper. The objects are the agents and the typed morphisms represent the

relations between the agents. To each MAS we associate a Base Diagram $\mathbb{D}$ which represents the complete relational structure (i.e. communication in the general sense). The nodes of this arrow diagram represent agents, the arrows (and paths of arrows) are the morphisms of this category.

A MAS is a dynamical system which means that relations between agents can change. This fact gives rise to the definition of the category $\mathbb{MAS}$ of all MAS where the objects are Multiagent Systems and the morphisms are MAS-Morphisms. Based on this category $\mathbb{MAS}$ a transformation system for Multiagent Systems is introduced by applying the double pushout approach ([EPS73, EEPT06]) to Multiagent Systems. We introduce it with the aim to develop a generic method for the formal manipulation of a Base Diagram of a MAS. This new activity has links to the extensive work of H. Ehrig and his group.

2 Some Introductory Notions and Notation

Category Theory (CAT) is a general, unifying mathematical modeling language providing many universal construction principles. In the sequel we repeat some basic definitions for the convenience of the reader, for more details we refer to the literature, c.f. e.g. [Lan98, Gol84, AHS90].

2.1 Some basic Categorical Notions

Definition 1 A category $\mathbf{C}$ consists of a class of objects denoted by $A, B, C, \ldots \in Obj(\mathbf{C})$. For each pair of objects A, B there is a set of morphisms, $Mor(A,B)$, also denoted by $\mathbf{C}(A,B)$ (the "arrows" between A and B). $\mathbf{C}(A_1,B_1)$ and $\mathbf{C}(A_2,B_2)$ are disjoint unless $A_1 = A_2$ and $B_1 = B_2$. (Note that $Mor(A,B)$ can be empty). There are two operations assigning to each $\mathbf{C}$ -arrow f a $\mathbf{C}$ -object $dom(f)$ and a $\mathbf{C}$ -object $codom(f)$. If $f \in Mor(A,B)$ then $A = dom(f)$ and $B = codom(f)$ and we display this as $f : A \to B$ or $A \xrightarrow{f} B$. There is a composition operation on morphisms: if $f : A \to B$ and $g : B \to C$ are morphisms, then there is a morphism $g \circ f : A \to C$, the composition of f and g. In a category the following axioms have to hold.

- The composition of morphisms is associative, that is for morphisms $f : A \to B, g : B \to C$ and $h : C \to D$ it holds: $h \circ (g \circ f) = (h \circ g) \circ f$.
- For every object $A \in Obj(\mathbf{C})$ there is the identity morphism id_A with the properties $f \circ id_A = f$ and $id_B \circ f = f$ for all $f : A \to B$.

"Popular" Examples of categories are: **SET** the category of sets and set mappings, **GROUP** the category of groups and group homomorphisms, **TOP**,etc... "Special " examples are: Partially ordered set $(M, \leq)$, generally an arrow diagram X can be interpreted as a category with the nodes as objects and the sequences (paths) of arrows [Pfa05] as morphisms.

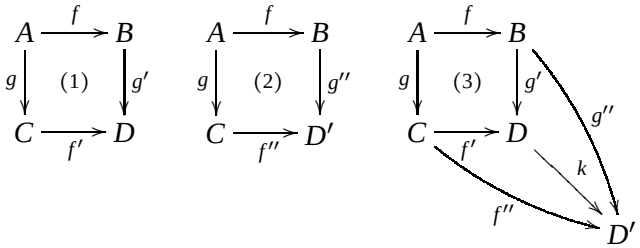
Definition 2 Let $\mathbf{X}$ and $\mathbf{Y}$ denote two categories. Then a functor $F : \mathbf{X} \to \mathbf{Y}$ assigns to every object $A \in Obj(\mathbf{X})$ an object $F(A) \in Obj(\mathbf{Y})$ and to every morphism $f : A \to B$ in $\mathbf{X}$ a morphism $F(f) : F(A) \to F(B)$ in $\mathbf{Y}$ such that the following holds for morphisms $f : A \to B$, $g : B \to C$ and id_A in $\mathbf{X}$

(1) $F(g \circ f) = F(g) \circ F(f)$
(2) $F(id_A) = id_{F(A)}$

Definition 3 Such a functor is called covariant. A functor is called contravariant if it reverses arrows.

Remark 1 *A category* **C** *is said to be small if its class of objects* $(Obj(\mathbf{C}))$ *is a set. The category of all small categories denoted by* **Cat** *, has small categories as objects and for* $\mathbf{X}, \mathbf{Y} \in Obj(\mathbf{Cat})$ $Mor(\mathbf{X}, \mathbf{Y})$ *consists of the functors from* **X** *to* **Y** *.*

Definition 4 The diagram (1) is called a pushout (or fibred coproduct) square if it commutes (i.e. $g' \circ f = f' \circ g$) and for any commuting square (i.e. $g'' \circ f = f'' \circ g$) of the form (2) there exists a unique morphism $k : D \to D'$ such that the diagram (3) commutes (i.e. $g'' = k \circ g'$ and $f'' = k \circ f'$).



As an example of a pushout situation we consider two morphisms in the category **SET** $f : A \to B$ and $g : A \to C$, a pushout in **SET** is obtained by forming the disjoint union $B \amalg C$ and then identifying $f(x)$ with $g(x)$ for all $x \in A$.

2.2 The category PATH(X): Categorical Semantics for Relations

Let $R \subset X \times X$ denote a general relation. One can create the corresponding arrow diagram by drawing an arrow from x to y $(x, y \in X)$ whenever $(x, y) \in R$. We associate with it the category denoted by PATH(X,R), PATH(X) or just PATH. The objects are the elements $x \in X$ and the morphisms are all sequences (paths) of adjacent arrows. This naturally defines a *composition of arrows*.

Recall that there is a morphism $x \longrightarrow y$, iff xRy. In general, for arrows $x \to y$ and $y \to z$, we do not have a "direct arrow" $x \to z$ (the relation can be not transitive) - this causes no problem. We can always form a sequence (path) of consecutive arrows, like $x \to y \to z$. This is a morphism of a more general type between x and z. More generally, we can have (finite) sequences, for example $x_0 \to x_1 \to \ldots \to x_n$ (a path), this is a morphism in Mor(x_0, x_n) in the new sense of our definition. It can also be interpreted as the composition of other morphisms being represented by adjacent parts of the long sequence.

Thus, PATH becomes a category. The existence of the identity arrow for each object will always be assumed by definition, we interpret the identity arrows as sequences of length zero.

An arbitrary binary relation R on X induces a corresponding arrow diagram $\mathbb{D}$, "visualizing" the given relations between objects by corresponding arrows. Vice versa, a given arrow diagram $\mathbb{D}$ induces (or defines) a corresponding binary relation R on the set of elements (nodes) of $\mathbb{D}$ in the

obvious, natural way, i.e. a specific arrow $x \longrightarrow y$ in $\mathbb{D}$ defines xRy. This leads to a categorical semantics for general relational structures. For more details we refer to PATH(X) in [Pfa94]. Summarizing we point out that every arrow diagram can be interpreted as a category - this aspect is of basic importance.

3 Categorical Modeling of MAS

In [Pfa05] a first formulation of the idea is given. The general communication and cooperation structure of a MAS is represented by a corresponding arrow diagram, called Base Diagram of the MAS. Based on the categorical modeling of relations via PATH(X,R) a natural description of the Base Diagram of a MAS in categorical notions arises. The idea is to define a category with typed objects representing the agents (having individual properties) and to define types of morphisms covering the relational structures between agents - including usual communication. We will speak of 'general communication' arrows comprising all kinds of relations among agents. This means that in the corresponding arrow diagrams (Base Diagrams $\mathbb{D}$) that visualize the MAS there can be more than one arrow between agents (directed multigraph).

Remark 2 *This is an external (global) modeling of MAS. The modeling approach focuses on general structures not on the internal (technical) modeling or implementation of specific agents.*

3.1 The Base Diagram of a MAS

Of basic importance for our subsequent considerations is the observation that every arrow diagram can be interpreted as a category where the nodes are the objects and every sequence of consecutive arrows, i.e. every path of arrows (not only a single arrow) in the diagram is a morphism. It leads to an associated category PATH, as previously discussed. This categorical interpretation is very useful and has a broad spectrum of concrete applications, in particular for establishing a categorical semantics for general relational structures.

We recall that in our categorical definition of a MAS objects are the agents and morphisms (arrows) are all kinds of relations (of specific types) between agents. This includes "classical agent communication" like KQML (see [FWW+93]) , KIF (see [GFB+92]), etc., and specific comparisons - agent to agent - concerning strength, power, capability, skills, availability, etc., each one represented by a corresponding arrow of specific type (it can be interpreted as a relation). In a very general way, this models the Communication Structure of the MAS - every arrow represents a particular type of general communication - we speak of "communication arrows".

This categorical structure model is of general nature and provides a formal basis to apply existing and new approaches from other areas with the objective to establish a detailed and concise structure description and system classification of the MAS. An interesting and basic topic of future work, extending these aspects, deals with a corresponding fibered structure (fibering, fiber bundle) associated with a given MAS where the base space of such a MAS-fibering is the Base Diagram of the MAS. A local fiber is attached to every agent representing all relevant data and information items characterizing the agent. This approach will be a generalization of the concept of Logical Fiberings - systems of distributed logics for logical modeling in MAS [Pfa05] [Pfa91].

We can observe that a Base Diagram is a network in the sense that every arrow (directed edge)

has a type, i.e. an abstract weight in the sense of a general net. This motivates to consider notions from neural network structure modeling that we have introduced (the category of Geometric Nets, GeoNet). Among others, it deals with the simplicial structure of a geometric net based on the notions of simplex and simplex configurations in topology and noncommutative geometry [Pfa03].

Summarizing, the communication between agents is modeled via typed morphisms between agents, specifying corresponding **"communication types"**. A subgroup of cooperating agents or a subsystem of a MAS is modeled as a subcategory. A mapping between two MAS is described by the notion of a functor between the two categories.

3.2 Typed Categories

The dependencies in a Multiagent System are only in special cases covered by a single relation. The need for a set of relations arises quickly. Consider as examples logical constraints and specific dependencies concerning agent cooperation and specific order relations as basis for comparisons (e. g. "winner takes all", "dominant agents", "power criteria", "skills/qualification") (compare [Pfa06]). Thus in general we need a method to handle more than one relation.

This leads to typed morphisms and in the sequel to typed categories. With typed morphisms we mean morphisms with additional information attached, holding the type information. Due to the type information in typed categories some differences from the definition of a classical category arise.

Definition 5 A typed category $\mathbf{T}$ consists of a collection of objects $Obj(\mathbf{T})$, a set of arrow types denoted by $ArrTypes(\mathbf{T})$, a set of object types denoted by $ObjTypes(\mathbf{T})$, a map $\tau_{\mathbf{T}} : Obj(\mathbf{T}) \to \mathfrak{P}(ObjTypes(\mathbf{T}))$ assigning to each object in $Obj(\mathbf{T})$ a set of object types where $\mathfrak{P}(ObjTypes(\mathbf{T}))$ denotes the power set of the set of object types, and for each triple (A,B,t) with $A,B \in Obj(\mathbf{T})$ and $t \in ArrTypes(\mathbf{T})$ a set of $\mathbf{T}$-morphisms $Mor_t(A,B)$. We call $f \in Mor_t(A,B)$ a typed morphism from A to B and write $f : A \to_t B$, also $A \overset{f}{\to}_t B$. We use the convention that for $t,s \in ArrTypes(\mathbf{T})$ it holds: $Mor_t(A,B)$, $Mor_s(A,B)$ are disjoint sets unless $t = s$.
The set of all arrows between A and B is given by the coproduct $\coprod_{t \in ArrTypes(\mathbf{T})} Mor_t(A,B)$ in SET (i.e. the disjoint union of sets). There is a typewise composition operation on morphisms such that for all objects $A,B,C \in Obj(\mathbf{T})$ and all $t \in ArrTypes(\mathbf{T})$ it holds:
If $f : A \to_t B$ and $g : B \to_t C$ then there is a unique morphism $g \circ f : A \to_t C$, the composition of f and g. This means in our definition of typed category only arrows of the same type can be composed.

In a typed category the following axioms have to hold:

- The composition of morphisms is associative, i. e. $h \circ (g \circ f) = (h \circ g) \circ f$.
- For every object A and arrow type $t \in ArrTypes(\mathbf{T})$ there is the corresponding identity morphism denoted by id_{tA} such that for every morphism $f \in Mor_t(A,B)$ holds $f \circ id_{tA} = f$ and $id_{tB} \circ f = f$.

If $ArrTypes(\mathbf{T})$ is a singleton it follows immediately that **T** behaves like a classical category. This is the situation if we model a single relation.

Definition 6 For $t \in ArrTypes(\mathbf{T})$ the category $\mathbf{S}_t$ is called a typed subcategory of **T** if the following holds:

- Every $\mathbf{S}_t$ object is a **T**-object.
- For A $\in Obj(\mathbf{S}_t)$ the set of object types in $\mathbf{S}_t$, $\tau_{\mathbf{S}_t}(A)$ equals the set of object types $\tau_{\mathbf{T}}(A)$ in **T** .
- For A,B $\in Obj(\mathbf{S}_t)$ the set of morphisms from A to B in $\mathbf{S}_t$ is a subset of the set of **T**-morphisms of type t denoted by $Mor_t(A,B)$.

$\mathbf{S}_t$ is called a full typed subcategory of **T** if for all $\mathbf{S}_t$ objects A, B it holds: the set of morphisms in $\mathbf{S}_t$ from A to B equals the set of **T**-morphisms of type t, denoted by $Mor_t(A,B)$.

In a typed category **T** it holds: **T** has $|ArrTypes(\mathbf{T})|$ full typed subcategories, with the set of objects in these subcategories being the set $Obj(\mathbf{T})$ such that every of these subcategories behaves like a category in the classical sense, and vice versa, each collection of categories having the same collection of objects yields a typed category.

In general a Multiagent System can be modeled as a typed category. The class of objects is the set of agents X, the object types $o_1, o_2, ..., o_m \in O$ represent the different properties of the agents, the arrow types $t_0, ..., t_n$ are the identifiers for the different relations $R_0, ..., R_n$, and the morphisms are the paths that arise for each relation in PATH. We can construct this Multiagent System Category MAS(X) by taking the collection of all PATH(X,R_i) categories which model the relations and defining the map $\tau : X \to \mathfrak{P}(O)$ that assigns to each agent its set of properties. In other words for the relations $R_i, i \in \{0,1,...,n\}$ PATH(X,R_i) together with a suitable map τ, is a full typed subcategory of the typed category MAS(X).

Remark 3 *Such a category is a small category for its class of objects is a set namely the set of agents.*

In the following the notion of a typed functor is introduced. Such a typed functor F constitutes the concept of ”map” between typed categories.

Definition 7 Let MAS_i and MAS_j be two typed Categories. A typed functor $F : MAS_i \to MAS_j$ assigns to every object $A \in Obj(MAS_i)$ an object $F(A) \in Obj(MAS_j)$, to every arrow type $t \in ArrTypes(MAS_i)$ an arrow type $F(t) \in ArrTypes(MAS_j)$, to every object type $o \in ObjTypes(MAS_i)$ an object type $F(o) \in ObjTypes(MAS_j)$ and to every typed morphism $f : A \to_t B$ of type t a morphism $F(f) : F(A) \to_{F(t)} F(B)$ such that for morphisms $f : A \to_t B$, $g : B \to_t C$, id_A and $A \in Obj(MAS_i)$ it holds:

- $F(g \circ f) = F(g) \circ F(f)$
- $F(id_A) = id_{F(A)}$
- $F(\tau_{MAS_i}(A)) \subseteq \tau_{MAS_j}(F(A))$

Summarizing, we have a tool to interpret the Base Diagram of a MAS as a category and we can establish subcategories and mappings between two such categories. Note that the typewise definition of the composition operation is essential.

We give a simple example with four agents a,b,c,d two object types 1,2, two arrow types that arise for the binary relations $domi := \{(b,a),(a,c),(d,c)\}$ which models the dominance hierarchy, and $comm := \{(a,b),(a,c),(c,d)\}$ that models the communication structure, the identity morphisms (paths of length zero) are not displayed.

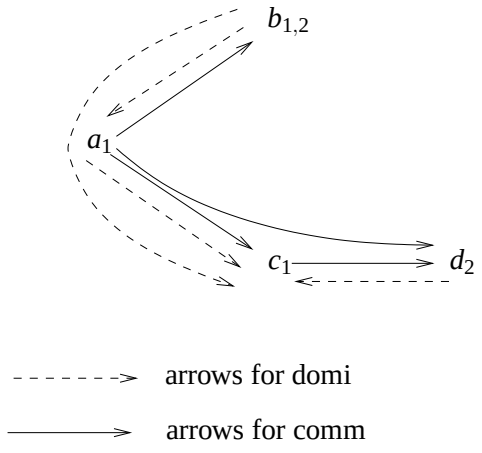


Figure 1: Simple Example

Note that agent b has type 1 and 2, i.e. agent b has both properties. The arrow from a to d is not a "direct arrow but the path (or sequence) $a \to c \to d$. The arrow from b to c of type domi is also no direct arrow but a path of length 2.

For reasons of readability in the sequel only sequences (paths) of length one are displayed. In particular the identity arrows and the compositions are not visualized in the remainder of this contribution.

4 The Category $\mathbb{MAS}$

Now we introduce the category of all Multiagent Systems $\mathbb{MAS}$. The objects of $\mathbb{MAS}$ are Multiagent Systems and the morphisms are $\mathbb{MAS}$ morphisms. The previous definition of a MAS is sufficient for situations with static relations, what we mean by this is that the relations between the agents remain the same and no agents are added or deleted. Nevertheless there exist many MAS where dynamics take a great impact on the system i.e. the relations between the agents change. This is of great importance for practical applications describing realistic scenarios. For instance take a relation modeling the communication possibilities of the agents. Let A be the set of agents and $C \subseteq A \times A$ be a binary relation. Two agents $a,b \in A$ are in relation C, i.e. aCb, if agent a can send messages to agent b. There are many possibilities why the communication relation C changes, consider as an example mobility of agents or absence of agents. What we need is a transformation system that transforms one MAS into another MAS, by changing the relations

between the agents and possibly add or delete agents. To be able to model this in categorical notions we define the category of all MAS where the objects are typed categories representing the Multiagent Systems in the above sense and the morphisms are $\mathbb{MAS}$ morphisms i.e. covariant typed functors between the typed categories.

Remark 4 *Note that every category can be visualized by a graph* $\Gamma(V,E)$*, where* V *the set of vertices represents the objects and* E *the set of edges represents the morphisms. We can visualize a general relational structure or a typed category as a labeled graph, having vertices representing the elements of a Universe* X*, in the case of MAS the agents, and an edge with label* R_i *whenever two elements* $x,y \in X$ *are in relation* R_i*.*

A $\mathbb{MAS}$ morphism F is a structure preserving map between two $\mathbb{MAS}$ objects, having the property to respect the type information of the arrows and their direction as well as the object types. This is necessary to preserve the relational information of the Multiagent Systems. We can observe that this is done by covariant typed functors.

Remark 5 $\mathbb{MAS}$ *is a category. Its composition operation is given by the composition of typed functors, the identity arrows are the identity functors.*

4.1 $\mathbb{MAS}$ Morphisms

In the sequel we will discuss $\mathbb{MAS}$ Morphisms, which are essentially typed functors. The purpose of this section is to introduce $\mathbb{MAS}$ Morphisms componentwise and thereby motivate the implementation of the concept that is based on the category SET.

Let MAS_i be a $\mathbb{MAS}$ object, this implies that the class of objects of the typed category MAS_i is a set namely the set of agents, as a consequence the class of arrows of MAS_i is a set too. In the sequel we will denote the set of all arrows within a Base Diagram MAS_i as $Arr(MAS_i)$. The set of objects, the set of arrows, the set of object types, the set of arrow types, the domain and codomain maps as well as the map τ_{MAS_i} and a map assigning to each arrow an arrow type $\pi_{MAS_i} : Arr(MAS_i) \to ArrTypes(MAS_i)$ allow us to analyze the category $\mathbb{MAS}$ by means of sets and maps.

Given two Multiagent Systems MAS_i and MAS_j a $\mathbb{MAS}$ morphism $F : MAS_i \to MAS_j$ is a quadruple $F = (F_O, F_A, F_{OT}, F_{AT})$ of maps

- $F_O : Obj(MAS_i) \to Obj(MAS_j)$
- $F_A : Arr(MAS_i) \to Arr(MAS_j)$
- $F_{AT} : ArrTypes(MAS_i) \to ArrTypes(MAS_j)$
- $F_{OT} : ObjTypes(MAS_i) \to ObjTypes(MAS_j)$
  obviously this map induces a map $F_{\mathfrak{P}} : \mathfrak{P}(ObjTypes(MAS_i)) \to \mathfrak{P}(ObjTypes(MAS_j))$ assigning to each subset of $ObjTypes(MAS_i)$ its image via the corresponding image operator.

Remark 6 *The introduction of sets of object types for each object (agent) and the corresponding power sets as well as the induced map* $F_{\mathfrak{P}}$*, allows to define agents that have a set of properties, which is important in realistic scenarios.*

This quadruple is a $\mathbb{MAS}$ morphism provided, that for the following diagram it holds that (1), (2) for domain and (2) for codomain maps are commutative squares and for (3) it holds (because this square is not commutative): $\forall A \in Obj(MAS_i) : F_{\mathfrak{P}} \circ \tau_{MAS_i}(A) \subseteq \tau_{MAS_j} \circ F_O(A)$.

$$\begin{array}{ccc}
ArrTypes(MAS_i) & \xrightarrow{F_{AT}} & ArrTypes(MAS_j) \\
\uparrow{\scriptstyle \pi_{MAS_i}} & (1) & \uparrow{\scriptstyle \pi_{MAS_j}} \\
Arr(MAS_i) & \xrightarrow{F_A} & Arr(MAS_j) \\
\downarrow{\scriptstyle codom_{MAS_i}}\ \downarrow{\scriptstyle dom_{MAS_i}} & (2) & \downarrow{\scriptstyle codom_{MAS_j}}\ \downarrow{\scriptstyle dom_{MAS_j}} \\
Obj(MAS_i) & \xrightarrow{F_O} & Obj(MAS_j) \\
\downarrow{\scriptstyle \tau_{MAS_i}} & (3) & \downarrow{\scriptstyle \tau_{MAS_j}} \\
\mathfrak{P}(ObjTypes(MAS_i)) & \xrightarrow{F_{\mathfrak{P}}} & \mathfrak{P}(ObjTypes(MAS_i))
\end{array}$$

The diagram above visualizes a $\mathbb{MAS}$ morphism in the category SET, the left hand side represents the $\mathbb{MAS}$ object MAS_i, the right hand side represents the $\mathbb{MAS}$ object MAS_j, and the four horizontal arrows represent the $\mathbb{MAS}$ morphism $F = (F_O, F_A, F_{OT}, F_{AT})$.

Remark 7 *We take a closer look at the MAS semantic for the maps F_{AT} and $F_{\mathfrak{P}}$.*
F_{AT} is interpreted as a translation map for arrow types, in addition it allows to merge relations. Due to the fact that there are in general no constraints to the map, it can be non-monic this leads to the possibility to merge relations by mapping different arrow types in MAS_i to a single arrow type in MAS_j.
In a similar way we interpret $F_{\mathfrak{P}}$ as a translation map. Important is the fact that a $\mathbb{MAS}$ morphism preserves object types, in the sense that after the application of the morphism the translated object types of an agent are at least a subset of the object types of the translated agent. This means we do not "lose" properties.

Consider as an example Figure 2 visualizing a $\mathbb{MAS}$ morphism F. Note that the MAS_i arrow from a to d is mapped to the path $a \to c \to d$ in MAS_j.

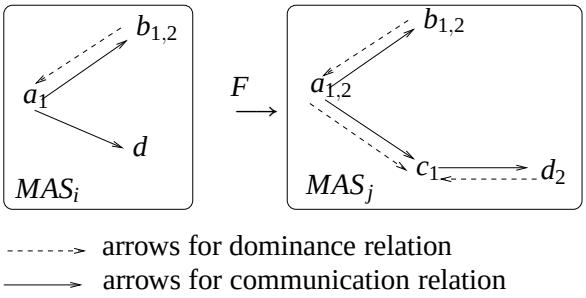


Figure 2: Simple Morphism

4.2 Pushout construction in $\mathbb{MAS}$

Previously we observed that a $\mathbb{MAS}$ morphism as well as $\mathbb{MAS}$ objects can be described via sets and maps. In category SET there exist pushouts, this suggests to create pushouts in $\mathbb{MAS}$ componentwise in SET for the objects, the morphisms, the arrow types and the object types.

Let I, J and K be $\mathbb{MAS}$ objects and $F : I \to J$, $G : I \to K$ be $\mathbb{MAS}$ morphisms (typed functors), in this part we show how the pushout PO is constructed. Consider the diagram:

$$\begin{array}{ccc} I & \xrightarrow{G} & K \\ {\scriptstyle F}\downarrow & & \downarrow{\scriptstyle F'} \\ J & \xrightarrow[G']{} & PO \end{array}$$

We construct the pushout componentwise, this leads to the sets

$$Obj(PO), Arr(PO), ArrTypes(PO), \text{ and a set } ObjTypes(PO)$$

together with the corresponding maps in the obvious way. Next we need to compute the domain and codomain map dom_{PO}, $codom_{PO}$ as well as the map π_{PO} assigning a morphism type to each morphism. The last step is to generate the power set over the object types $\mathfrak{P}(ObjTypes(PO))$ and a map $\tau_{PO} : Obj(PO) \to \mathfrak{P}(ObjTypes(PO))$ assigning to each object (agent) its set of types (properties).

Figure 3 displays the componentwise pushout construction in SET. The uniqueness of the arrows dom_{PO}, $codom_{PO}$, and the map π_{PO} is a consequence of the fact that F and G are $\mathbb{MAS}$ Morphisms i.e. the left and back faces of the two upper cubes are commutative squares and that the horizontal squares are pushouts in SET.

Consider the situation for the map π_{PO}: $ArrTypes(PO)$ is the pushout object for the arrow types, it holds: $F'_{AT} \circ G_{AT} = G'_{AT} \circ F_{AT}$. As a consequence $F'_{AT} \circ G_{AT} \circ \pi_I = G'_{AT} \circ F_{AT} \circ \pi_I$. Recall that for the $\mathbb{MAS}$ morphisms F and G it holds $F_{AT} \circ \pi_I = \pi_J \circ F_A$ and $G_{AT} \circ \pi_I = \pi_K \circ G_A$. This implies $(F'_{AT} \circ \pi_K) \circ G_A = (G'_{AT} \circ \pi_J) \circ F_A$. Since the square $G'_A \circ F_A = F'_A \circ G_A$ is a pushout square, due to the universal property of pushouts it follows that there is exactly one map $\pi_{PO} : Arr(PO) \to ArrTypes(PO)$.
To proof the uniqueness of the maps dom_{PO} and $codom_{PO}$ we argue analogously.

The last step is to compute the map τ_{PO} such that F' and G' are $\mathbb{MAS}$ morphisms. Recall, for F' and G' the following property has to hold: $\forall A_k \in Obj(K) : F'_{\mathfrak{P}} \circ \tau_K(A_k) \subseteq \tau_{PO} \circ F'_O(A_k)$ and $\forall A_j \in Obj(J) : G'_{\mathfrak{P}} \circ \tau_J(A_j) \subseteq \tau_{PO} \circ G'_O(A_j)$.
We define the map in the following way. To each object A_{po} in $Obj(PO)$ we set $\tau_{PO}(A_{po}) = F'_{\mathfrak{P}} \circ \tau_K(F'^{-1}_O(\{A_{po}\})) \cup G'_{\mathfrak{P}} \circ \tau_J(G'^{-1}_O(\{A_{po}\}))$. Via τ_{PO} we assign to each object A_{po} the union of the "translated" object type sets of its preimages $F'^{-1}_O(\{A_{po}\})$ and $G'^{-1}_O(\{A_{po}\})$.

Note that for every object $A_{PO} \in Obj(PO)$ there exits a nonempty preimage set in $Obj(J)$ or in $Obj(K)$, due to the fact that $Obj(PO)$ is the pushout of the maps F_O, G_O.

Thus τ_{PO} is a well defined map and G' and F' are $\mathbb{MAS}$ morphisms.

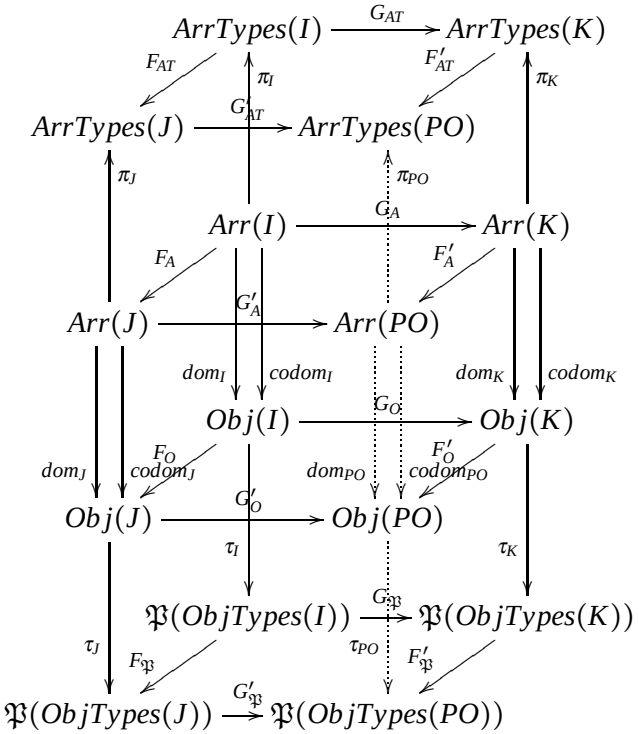


Figure 3: Componentwise Pushout Visualization

4.3 MAS Transformations

In this part we formalize what we mean by a transformation system and derivation steps for Multiagent Systems. A transformation system is defined by a concept introduced by Ehrig, Pfender, and Schneider [EPS73] the so called double pushout (DPO) approach, which is a far developed concept in the field of algebraic graph transformations [EEPT06] and was generalized to adhesive HLR categories which are a suitable categorical framework for graph transformation in a more general sense. Since Category Theory is our common unifying linguistic formal bases there is a natural way to apply these approaches and methods to our MAS modeling problem areas. We give a MAS semantic to the DPO approach. In the sequel we give a brief sketch of the concepts production, derivability, DPO, and pushout complement (c.f. [EEPT06]).

In the sequel we restrict the class of $\mathbb{MAS}$ morphisms in the productions to typed embeddings.

Definition 8 A functor $F : \mathbf{X} \to \mathbf{Y}$ is called an embedding provided that F is injective on morphisms. Note that F is an embedding if and only if it is injective on objects and injective on the hom-set restrictions $F : Mor_{\mathbf{X}}(A,B) \to Mor_{\mathbf{Y}}(F(A),F(B))$. [AHS90]

We define a typed embedding straightforward as a typed functor that acts injectively on morphisms, injectively on arrow types and injectively on object types.

Definition 9 In $\mathbb{MAS}$ a $\mathbb{MAS}$ -production $p = (p^l, p^r)$ is defined as a pair of $\mathbb{MAS}$ morphisms

with common domain. Given a $\mathbb{MAS}$ -production p, a $\mathbb{MAS}$ object MAS_j and a $\mathbb{MAS}$ morphism $m : codom(p^l) \to MAS_j$, called match, defines a direct transformation step as follows:

A Object MAS^r is called direct derivable from an object MAS^l in $\mathbb{MAS}$, $MAS^l \Rightarrow MAS^r$, iff there exists a $\mathbb{MAS}$ -production $p = (p^l : MAS^I \to L, p^r : MAS^I \to R)$ and a context Object MAS^C with corresponding MAS-Morphism $g : MAS^I \to MAS^C$, such that MAS^l and MAS^r are pushout objects in the following diagram.

$$\begin{array}{ccccc} L & \xleftarrow{p^l} & MAS^I & \xrightarrow{p^r} & R \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle g} & & \downarrow{\scriptstyle g^r} \\ MAS^l & \xleftarrow{p^{-l}} & MAS^C & \xrightarrow{p^{-r}} & MAS^r \end{array}$$

Remark 8 *This diagram illustrates a Double Pushout, for more details we refer to the book [EEPT06].*

MAS' is called derivable from MAS^l if there exists a sequence MAS_0, MAS_1, ..., MAS_n such that $MAS^l = MAS_0 \wedge (\forall 1 \le i \le n)\ (MAS_{i-1} \Rightarrow MAS_i) \wedge MAS_n = MAS'$.

Given objects L, MAS^I, R and a context object MAS^C and the associated morphisms, the pushout objects MAS^l and MAS^r exist. But the usual situation is the following: We have a production $p = (p^l, p^r)$ and a morphism $m : L \to MAS^l$ called match. The task is to find a so called pushout complement PC and two morphisms g, p^{-l} such that the square

$$\begin{array}{ccc} L & \xleftarrow{p^l} & MAS^I \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle g} \\ MAS^l & \xleftarrow{p^{-l}} & PC \end{array}$$

is a pushout square. Considerations concerning existence and uniqueness of such pushout complements in $\mathbb{MAS}$ will be content of future work. For now we point out that the pushout complements that arise in the following examples exist.

For MAS the semantic of the match $m : L \to MAS^l$ is that we search for an occurrence of the relational structure of L in the system MAS^l. The match checks if the preconditions required by the left hand side of the production p are fulfilled in MAS^l, if this is the case the production can be applied, otherwise not.

4.4 Application of MAS Transformations

We present an example of a transformation system for MAS. Let $A = \{a, b, c, d, e\}$ be the set of agents. The task is to assemble a workpiece which consists of two parts (one of type X and and one of type Y). There are three object types $ObjTypes = \{1, 2, 3\}$ describing the properties of the agents, 1 stands for: the corresponding agent can perform an assembly action, 2 stands for: the corresponding agent has a gripper and can deliver parts and 3 stands for: the corresponding agent is not attached to a task. (Note that it is possible that an agent has all or none of the properties 1,2,3). Three agents will cooperate, two of object type 2 grabbing parts of type X and Y and positioning them on the worktable of an agent of type 1 which assembles the parts. Note that despite of these three acting agents there can be other agents that simply pass messages from one

agent to another.

Next we define the relevant relations that represent the cooperation and communication information of our example.

4.4.1 Definition of the Relevant Relations

We define:

- A symmetric communication relation $C \subseteq A \times A$: Two agents *a*,*b* are in relation *C* if *a* is able to communicate with *b* i.e. *aCb* (it follows *bCa* due to the symmetry of *C*)
- A relation *Dx*: two agents *a*, *b* are in relation *Dx*, i.e. $(a,b) \in Dx$ if *a* delivers a part of type *X* to *b*
- A relation *Dy*: two agents *a*, *b* are in relation *Dy*, i.e. $(a,b) \in Dy$ if *a* delivers a part of type *Y* to *b*
- A relation $W \subseteq A \times A$: $(a,a) \in W$ if *a* is assembling the parts.

In the sequel the arrows from Figure 4 are used to display the relational structure, for the convenience of the reader the different arrows indicate the different morphism types (instead of indices at the tip of the arrows).

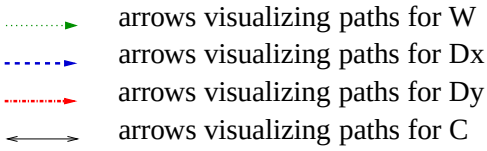


Figure 4: Arrow Types

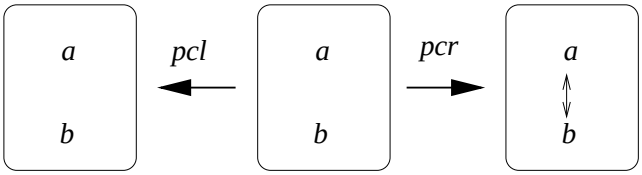
4.4.2 Definition of Productions

Next we define the productions which describe actions together with their application conditions. These productions implement the changes that take place in the MAS. For the convenience of the reader the types of the objects are indicated by indices of the objects.

Remark 9 *In the given example the productions are applied from left to right. Obviously a production is symmetric i.e. given a suitable match m defined from the righthand side of the production to a* $\mathbb{MAS}$ *object gives a valid transformation step, too.*

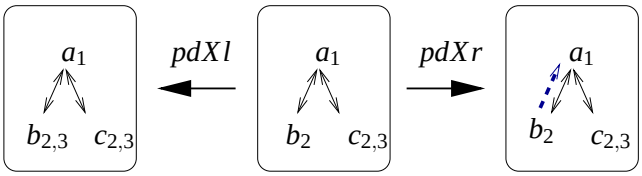
In the sequel we give a visualization of the five productions used to describe the systems dynamics.

- *pc* (production communication) is the production that modifies the communication relation, if two agents ”meet” the production is applied (from left to right) such that after the derivation step the two agents can communicate.



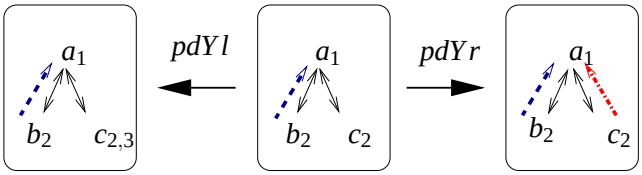
production $pc = (pcl, pcr)$

- *pdX* (production deliver part of type X) is applied if two agents having property 2 and 3 can communicate with an agent that has property 1, after the derivation step one of the agents delivers a piece of type X to the assembly agent.



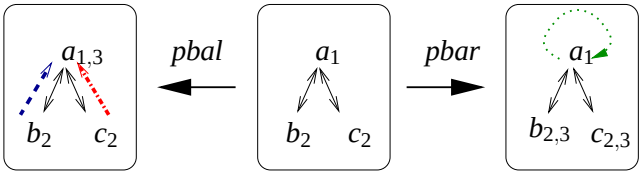
production $pdX = (pdXl, pdXr)$

- *pdY* (production deliver part of type Y) is applied if two agents having property 2 can communicate with an agent that has property 1 and one of the agents is already delivering a piece of type X and the other agent is able to perform a task(indicated by the property 3), after the derivation step one of the agents delivers a piece of type Y to the assembling agent.



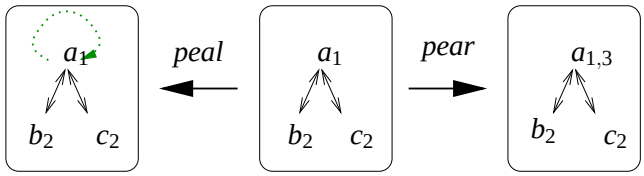
production $pdY = (pdYl, pdYr)$

- *pba* (production begin assembling) is applied if two agents with property 2 can communicate with an agent that has property 1 and parts of type X and Y are delivered to the assembling agent which has the additional property 3, after the derivation step the assembly agent received the parts and starts to assembly them.



production $pba = (pbal, pbar)$

- *pea* (production end assembling) is applied when an agent stops assembling. This means the task is finished.



4.4.3 Application of Productions

Figure 5 shows the application of production *pdX* to a given MAS object, the diagram shows a double pushout situation. We search for three agents that can communicate with each other, one of them (*a*) can perform the assembly task (1) the others (*b*), (*c*) have to be able to perform the delivery task (2), and both of them are free to act, or are not assigned to another task yet (3). Such a situation is given in the downleft object of the diagram, this means we can apply the production. Observe that the communication arrow $a \leftrightarrow b$ in the production is mapped to the sequence of arrows $a \leftrightarrow e \leftrightarrow b$. After the transformation step agent *b* delivers part X to agent *a* and is assigned to a task (*b* lost property 3).

There are in general more possibilities for a production to be applied to a given MAS object. In some situations there will exist more than one suitable match whereas in others there might exist no suitable match at all. Future research will try to determine criteria on how to choose the ”best” or most suitable match to address the needs of an efficient transformation system for Multiagent Systems.

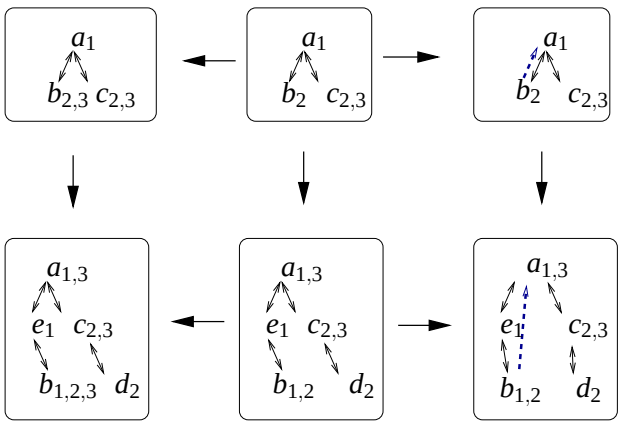


Figure 5: Application of production pdX to a given MAS.

4.4.4 System Run

Figure 6 shows a sequence of $\mathbb{MAS}$ -transformations, with initial $\mathbb{MAS}$ object S0, that fulfills the assembly task. The productions are applied together with suitable matches.

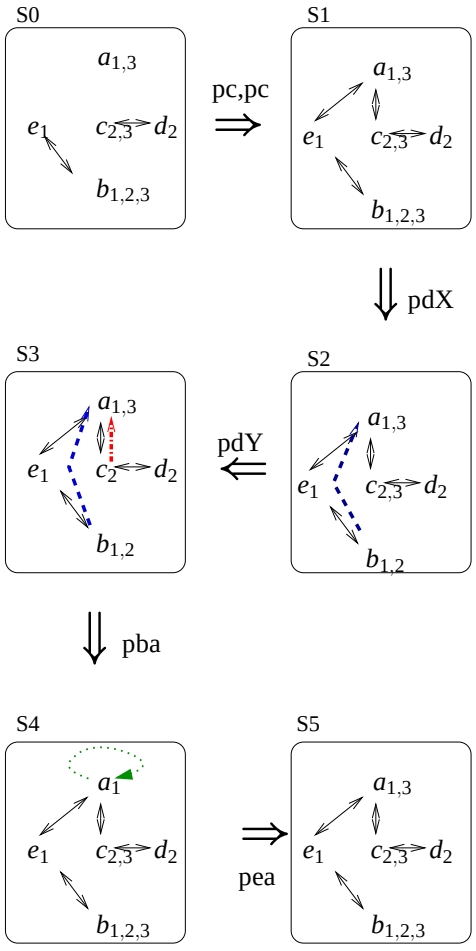


Figure 6: A sequence of transformation steps

The production pc is applied twice, first with match m_0, the object part of the morphism m_0 is $[m_0(a) = a$ and $m_0(b) = e]$, and next with match m_1 the object part of the morphism is $[m_1(a) = a$ and $m_1(b) = c]$.

Now the application condition for *pdX* is fulfilled and *pdX* is applied to S1 with a suitable match m_2 resulting in S2. In the sequel *pdY*, *pba* and *pea* are applied in the obvious way together with suitable matches. In S5 we reach the end of the task, a workpiece has been assembled. We observe that production *pdX* can again be applied to S5, this would restart the assembly process.

5 Implementation

In the working group two new tools concerning MAS Transformations are designed and programmed. The first one ”MASTRANSF” is based on category theory and is implemented as a module that provides support for set like categories and the category $\mathbb{MAS}$. It supports construction of pullbacks, pushouts, equalizers, coequalizers, products, coproducts, among others. As a next step the construction of pushout complements will be implemented.

Due to its fundamental importance the category SET was implemented with the associated set theoretic notions (e.g. equivalence relations, image operator and inverse image operator). Implementation of new results, such as the notion of a quasi coproduct complement [Sob08] as a basis for pushout complement constructions, and modeling with fibered structures for local global modeling, is part of ongoing work, among others. A GUI is under developement that allows the fast input of Base Diagrams, as well as morphisms between them. It will be possible to define a set of $\mathbb{MAS}$ productions and apply them to the Base Diagrams. This results in a rule based controller for Multiagent Systems.

5.1 Classes

The implementation is organized around 3 abstract classes:

- Category - With objects, morphisms, a composition operation and a name.
- Object - Has on the abstract level only a name.
- Morphism - Has a codomain, a domain and a name.

General results can be implemented in abstract classes, that are extended from the above ones. See for example the next section dealing with the canonical construction of pushouts.

Additionally, there are classes for special limit and colimit objects and their associated morphisms, e.g. a class Product, that has two Morphisms the projections and one object the product object. SET and $\mathbb{MAS}$ implement the abstract class CategoryWithCoproductsAndCoequalizers. This is due to the fact that the main focus of the implementation is on Pushouts. Generally, the implementation of limit and colimit constructions in $\mathbb{MAS}$ follows the constructive proofs that category theory provides (see e.g. the proofs in [Sob08]). For an early general treatment using ML implementations we refer to [RB88].

Remark 10 *Further work will be necessary to define fitting inheritance schemes. For example abstract classes for finitely cocomplete and complete categories, skeletal categories, etc.. But as mentioned, the focus of this paper is not on a general implementation of category theory, but on the implementation of the categorical notions and results that are applicable within the Category* $\mathbb{MAS}$ *.*

5.2 Implementation of Pushout Construction

The construction of pushouts is implemented within an abstract class (CategoryWithCoproductsAndCoEqualizers) which extends the class Category.

Given the existence of Coproducts and Coequalizers there is no need for an implementation of the pushout construction within the specific category because this can be done abstractly.

See the following Java code:

```
public Pushout createPushout(Morphism f, Morphism g) {
        Pushout pushout = new Pushout();
        Coproduct coproduct = new Coproduct();
        coproduct = createCoproduct(g.codomain, f.codomain);

        Coequalizer coequalizer = new Coequalizer();
        coequalizer = createCoequalizer(compose(coproduct.i1,g),
                        compose(coproduct.i2, f));

        pushout.morA = compose(coequalizer.mor, coproduct.i1);
        pushout.morB = compose(coequalizer.mor, coproduct.i2);
        pushout.pushoutObject = coequalizer.coequalizerObject;
        return pushout;
}
```

5.3 Pushout Construction within $\mathbb{MAS}$

The category $\mathbb{MAS}$ extends the Class CategoryWithCoproductsAndCoequalizer. The screenshots (Figures 7 and 8) show a pushout construction.

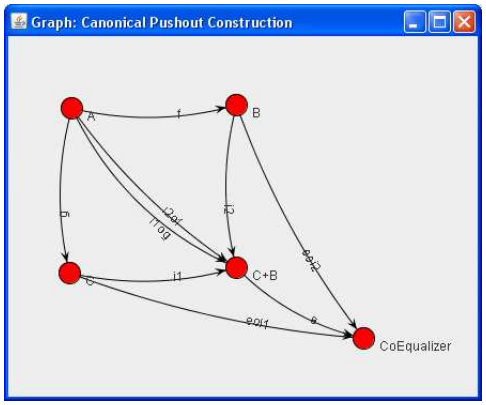


Figure 7: Pushout Construction, Objects and Morphisms

Given 3 $\mathbb{MAS}$ -objects A, B, C and two $\mathbb{MAS}$ -morphisms $f : A \to B$, $g : A \to C$, we construct the coproduct of B and C, this results in the $\mathbb{MAS}$ -object $B+C$ and two injections $i_1 : B \to B+C$ and $i_2 : C \to B+C$. The next step is to coequalize along the morphisms $i_1 \circ f$ and $i_2 \circ g$. The result is the $\mathbb{MAS}$ -object named "CoEqualizer" which is the pushout of f and g.

The second tool under construction is a 3D robot simulator that is also implemented in Java with the aim to visualize the processes realistically. As we have seen above cooperating robots can be intuitively interpreted as a MAS. It is intended to control the robots via "MASTRANSF". The simulator works as a demonstrator, where on one side the robots movements and on the other side the changes in the Base Diagram of the MAS are displayed. See Figure 9, where the upper part of the figure shows the base diagrams, the lower part depicts the three cooperating robots.

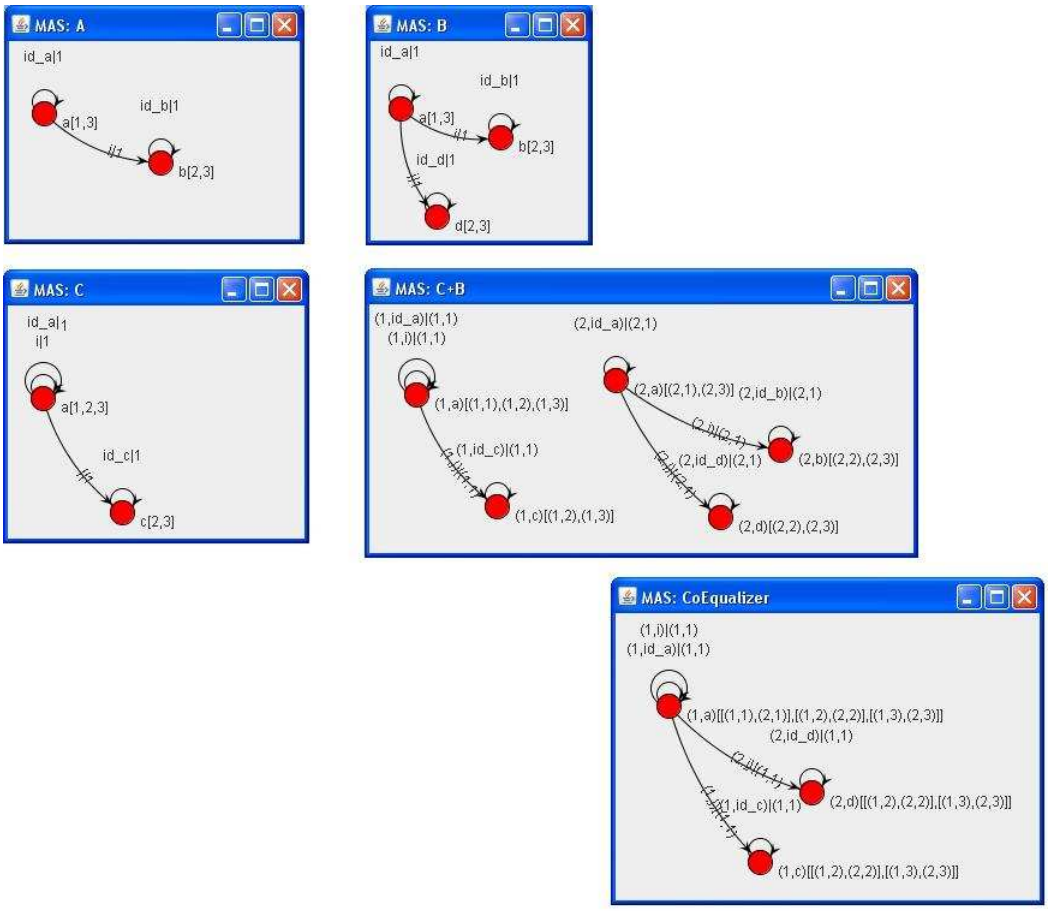


Figure 8: Pushout Contstruction, Base Diagrams

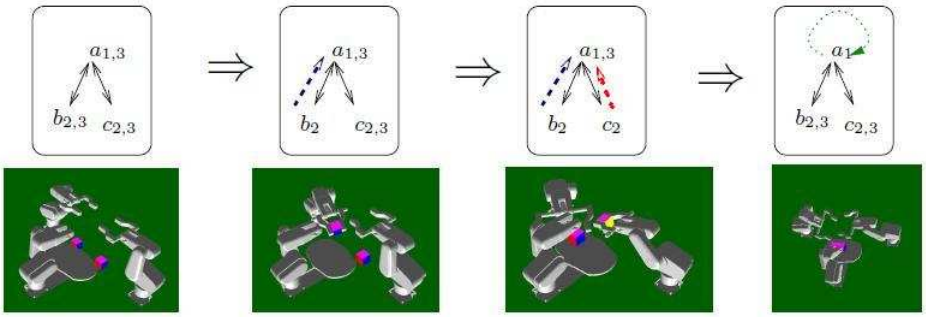


Figure 9: Sequence of Transformation Steps

6 Conclusion and Future work

The concept of $\mathbb{MAS}$ transformations, which is an adaption of graph transformations [EEPT06] to typed categories, is a natural way to describe changes in the Base Diagram of Multiagent Systems. Due to the categorical modeling the proposed approach is independent of the implementation of the agents (agents are analyzed via their external properties). This allows to analyze MAS on the basis of their cooperation and communication structures, which represent significant information. We can define actions and their corresponding preconditions in a MAS by productions in $\mathbb{MAS}$. Now we can ask the question: Which transformation steps lead to a desired result? For example: Which transformation steps have to take place that the resulting Base Diagram of the MAS has a limit or colimit object for one or more types? This can be interpreted as a universal communicator, mediator, or steering agent [Pfa06].

Another aspect of future work will be the investigation of interdependencies between object types and arrow types. In some cases in a $\mathbb{MAS}$ there clearly are interdependencies between the properties of agents and the relations (arrows) between the agents.

Another part of work in MAS modeling concerns logical modeling aspects. It turned out that logical fiberings [Pfa91] provide a concept to assign a system of distributed logics to a MAS in a natural way. The basic idea is to assign a logical fiber to every agent, this fiber models the local logical state space of an agent, the entire logical fiber bundle forms the global logical state space of the whole MAS. For more details we refer to [Pfa04]. This motivates the introduction of a 'Relational Fibering' with the aim to model local global interactions in the relational structure of a MAS. We assign a relational fiber to every agent, the fiber models the relational information attached to the agent. A first application of this approach is to compute subcategories of a MAS on demand, by taking the collection of the fibers over a defined set of agents as a starting point.

The notion of Activity Networks, as used in operations research, can be deployed for modeling certain constraints in communication flow in Base Diagrams.

A further aspect of intended future work concerns construction principles from Category Theory like limit and co-lomit constructions that can be deployed, e.g. to extend a given MAS (using the Base Diagram) by a kind of universal communicator (coordinator) agent. Simple scenarios of cooperating robot agents provided first motivating examples ("experiments") and first steps. [Pfa06].

We thank the referees for some helpful remarks.

Bibliography

[AHS90] J. Adámek, H. Herrlich, G. Strecker. *Abstract and Concrete Categories*. John Wiley & Sons, 1990.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Volume 1. Springer, 2006.

[EPS73] H. Ehrig, M. Pfender, H. J. Schneider. Graph Grammars: an Algebraic Approach. In *Proceedings of FOCS*. Pp. 167–180. 1973.

[FWW+93] T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, D. Mckay, J. McGuire, R. Pelavin, S. Shapiro, C. Beck. DRAFT Specification of the KQML Agent-Communication Language. 1993.

[GFB+92] M. Genesereth, R. E. Fikes, R. Brachman, T. Gruber, P. Hayes, R. Letsinger, V. Lifschitz, R. Macgregor, J. Mccarthy, P. Norvig, R. Patil. Knowledge Interchange Format Version 3.0 Reference Manual. 1992.

[Gol84] R. Goldblatt. *TOPOI, The Categorial Analysis of Logic*. Studies in Logic and The Foundations of Mathematics; v. 98. Elsevier Science Publishers B.V., revised edition, 1984.

[Lan98] S. M. Lane. *Categories for the Working Mathematician*. Springer Verlag, Graduate Texts in Mathematics **5**, 2nd ed., 1998.

[Pfa91] J. Pfalzgraf. Logical Fiberings and Polycontextural Systems. In *Fundamentals of Artificial Intelligence Research, Ph.Jorrand, J.Kelemen (eds.). Lecture Notes in Computer Science 535, Subseries in AI, Springer Verlag*. 1991.

[Pfa94] J. Pfalzgraf. On a general notion of a hull. In *Automated Practical Reasoning, J.Pfalzgraf and D.Wang (eds.). Texts and Monographs in Symbolic Computation, Springer-Verlag Wien New York*. 1994.

[Pfa03] J. Pfalzgraf. Modeling Connectionist Networks: Categorical, Geometric Aspects (Towards Homomorphic Learning). In Dubois (ed.), *Proceedings Computing Anticipatroy Systems: CASYS 2003*. Volume 718. AIP Conference Proceedings, 2003. Recieved a Best Paper Award.

[Pfa04] J. Pfalzgraf. On logical fiberings and automated deduction in many-valued logics using Gröbner bases. *RACSAM, Rev. Real. Acad. Ciencias, Ser. A. Mat., Vol. 98(1)*, pp. 213–227, 2004.

[Pfa05] J. Pfalzgraf. On Categorical and Logical Modeling in Multiagent Systems. In Lasker and Dubois (eds.), *Anticipative and Predictive Models in Systems Science*. Volume 1. 2005.

[Pfa06] J. Pfalzgraf. On an Idea for Constructing Multiagent Systems (MAS) Scenarios. In Lasker and Pfalzgraf (eds.), *Advances in Multiagent Systems, Robotics and Cybernetics: Theory and Practice.* Volume 1. International Institute for Advanced Studies in Systems Research and Cybernetics, 2006.

[RB88] D. E. Rydeheard, R. M. Burstall. *Computational Category Theory*. Prentice Hall, 1988.

[Sob08] T. Soboll. *Categorical Modeling of Multiagent Systems*. PhD thesis, University of Salzburg, 2008.

[Woo02] M. J. Wooldrige. *An Introduction to Multiagent Systems*. John Wiley and Sons LTD, 2002.