# Learning Synthesis Schemes in Intelligent Systems

**Lech Polkowski**
Institute of Mathematics
Warsaw University of Technology
Pl. Politechniki 1, 00-650 Warsaw, Poland
polk@mimuw.edu.pl

**Andrzej Skowron**
Institute of Mathematics
Warsaw University
Banacha 2, 02-097 Warsaw, Poland
skowron@mimuw.edu.pl

## Abstract

We present a settting in which one can discuss problems of design, synthesis, analysis and control of complex systems by adaptive teams of intelligent distributed agents. We point to learning problems of this approach related to the necesity of extracting from the empirical data of constructs allowing the agents to negotiate their co - operative actions. We put our analysis into the framework of multistrategy learning (Michalski 1994) which combines empirical induction, abduction and reasoning by analogy in a hierarchical setting (Michalski 1994).

## Introduction

We propose a discussion of synthesis schemes for synthesis of complex objects defined by uncertain or incompletely understood requirements. Our synthesis schemes are constructed over sets of intelligent autonomous agents which co - operate and negotiate their goals towards synthesis of an appropriate artifact from some simple prescribed inventory objects. A requirement which approximately specifies a complex object comes to the agents from the external world and can be expressed in a language not fully understandable by the agents. See in this respect Example 1 below. The main result of the uncertainty caused by the language is that the agents are able to describe the complex object in their internal language approximately only; however, they strive to produce a complex artifact which satisfies the external requirement in satisfactory degree i.e. the external customer will accept this artifact as satisfying the requirement. To achieve this end, the agents form teams which co - operate towards creating the final artifact. The links among local teams of agents are formed as a result of negotiations among agents which result in a synthesis scheme; the scheme is locally robust i.e. each team of agents in the scheme has certain bounds of uncertainty within which to operate. The adaptiveness of the scheme is the result of its forming technique: when the external conditions change, the agents may re - negotiate their local uncertainty bounds as well as links among local teams which may lead to a new scheme with new boundary conditions. One may also admit the possibility of re -negotiating local goals which may lead to a new class of schemes.

Our principal aim is to give a formal model for accounting for uncertainty and approximative character of our schemes. We propose to this end to apply our recently developed (Polkowski & Skowron 1994a), (Polkowski & Skowron 1994b), (Skowron & Polkowski 1995), (Polkowski & Skowron 1995), (Polkowski & Skowron 1996a), (Polkowski & Skowron 1996b), theory of rough mereology. Rough mereology is a theory of a relation of being a part in a degree which can be traced back to mereology of Leśniewski (Leśniewski 1992). This theory allows for introducing a logic of approximative reasoning about complex objects (Komorowski, Polkowski, & Skowron 1996) in which, in particular, we are able to express the fact that a given artifact satisfies a given property in certain degree. Let us stress also that rough mereology encompasses fuzzy set theory: the notions of an element and the subset coincide in rough mereology (Polkowski & Skowron 1994), (Polkowski & Skowron 1996a) hence the relation of being a part in a degree can be regarded as a fuzzy membership function.

In our approach rough mereologies are assigned to individual agents in a distributed way: to each agent a rough inclusion (see below) is assigned which permits the agent to measure degrees of similarity among objects manipulated by it, and local mereologies of co - operating agents are related by means of propagating functors (mereological connectives) extracted (learned) from data.

Forming a synthesis scheme proceeds in two steps; first - the external requirement is absorbed as a constraint in a language of agents and it is decomposed into simpler constraints; any constraint is associated with an agent or a local team of agents which are able to fulfill it. The process of decomposition involves negotiations among agents which result in assignment to the agents of local constraints, assembling operations (decomposition rules) as well as uncertainty bounds. Constraints may be expressed in an informal language and they are rendered in logical languages of agents as approximate formulas in the form of a pair $(\Phi, \varepsilon)$

where $\Phi$ is a predicate in the logic of the agent and $\varepsilon$ is an uncertainty bound. In positive cases constraints reach the level of inventory (primitive) agents which are able to convert their approximate formulas into objects satisfying them.

The second step consists in assembling a complex artifact from primitive objects by means of negotiated operations.

Our approach is analytic: all constructs are learned from data tables which constitute the agents knowledge.

We present in what follows: preliminaries on mereology and rough mereology, logic for approximative reasoning (satisfactory fulfillment of a requirement), design and synthesis schemes, and the learning aspects of our approach. We conclude with a discussion of potential applications to problems of design, synthesis and control of complex systems.

## Preliminaries: Mereology of Leśniewski

We recall here the basic notions of the mereological system of Leśniewski (Leśniewski 1992); in the next section the mereological system of Leśniewski will be extended to the system of approximate mereological calculus called *rough mereology*.

We consider a finite non - empty set $U$. A binary relation $\pi$ on the set $U$ will be called the *relation of being a (proper) part* in the case when the following conditions are fulfilled

(P1) (*irreflexivity*) for any $x \in U$, it is not true that $x\pi x$;

(P2) (*transitivity*) for any triple $x, y, z \in U$, if $x\pi y$ and $y\pi z$, then $x\pi z$.

It follows obviously from (P1) and (P2) that the following property holds

(P3) for any pair $x, y \in U$, if $x\pi y$ then it is not true that $y\pi x$.

In the case when $x\pi y$ we say that the object $x$ is a (*proper*) *part* of the object $y$. The notion of being (possibly) an improper part is rendered by the notion of an ingredient; for objects $x, y \in U$, we say that the object $x$ is a *$\pi$-ingredient* of the object $y$ when either $x\pi y$ or $x = y$. We denote the relation of being a $\pi$-ingredient by the symbol $ingr(\pi)$; hence we can write

(I1) for $x, y \in U$, $x$ $ingr(\pi)$ $y$ iff $x\pi y$ or $x = y$.

It follows immediately from the definition that the relation of being an ingredient has the following properties:

(I2) (*reflexivity*) for any $x \in U$, we have $x$ $ingr(\pi)$ $x$;

(I3) (*weak antisymmetry*) for any pair $x, y \in U$, if $x$ $ingr(\pi)$ $y$ and $y$ $ingr(\pi)$ $x$ then $x = y$;

(I4) (*transitivity*) for any triple $x, y, z \in U$, if $x$ $ingr(\pi)$ $y$ and $y$ $ingr(\pi)$ $z$ then $x$ $ingr(\pi)$ $z$.

We will call any pair $(U, \pi)$ where $U$ is a finite set and $\pi$ a binary relation on the set $U$ which satisfies the conditions (P1) and (P2) a *pre-model of mereology*.

We now recall the notions of a set of objects and of a class of objects. For a given pre-model $(U, \pi)$ of mereology and a property $m$ which can be attributed to objects in $U$, we will say that an object $x$ is an object $m$ ($x$ *object* $m$, for short) when the object $x$ has the property $m$. The property $m$ will be said to be non-void when there exists an object $x \in U$ such that $x$ *object* $m$. Consider a non-void property $m$ of objects in a set $U$ where $(U, \pi)$ is a pre-model of mereology.

An object $x \in U$ is said to be a ***set of objects with the property*** $m$ when the following condition is fulfilled:

(SET$m$) for any $y \in U$, if $y$ *object* $m$ and $y$ $ingr(\pi)$ $x$ then there exist $z, t \in U$ with the properties: $z$ $ingr(\pi)$ $y$, $z$ $ingr(\pi)$ $t$, $t$ $ingr(\pi)$ $x$ and $t$ *object* $m$.

We will use the symbol $x$ *set* $m$ to denote the fact that an object $x$ is a set of objects with the property $m$.

Assume that $x$ *set* $m$; if, in addition, the object $x$ satisfies the condition

(CL$m$) for any $y \in U$, if $y$ *object* $m$ then $y$ $ingr(\pi)$ $x$ then we say that the object $x$ is ***a class of objects with the property*** $m$ and we denote this fact by the symbol $x$ *class* $m$. We will say that a pair $(U, \pi)$ is a model of mereology when the pair $(U, \pi)$ is a pre-model of mereology and the condition

(EUC) for any non-void property $m$ of objects in the set $U$, there exists a unique object $x$ such that $x$ *class* $m$ holds.

The notions of a set and a class permit to regard collections of objects as objects; the intuitive, mathematical idea related to them is that of a set theoretical union. Our application of these notions is explained below, e.g., in design theory.

## Rough mereology

An approximate mereological calculus called rough mereology has been proposed (Polkowski & Skowron 1994), (Polkowski & Skowron 1996a) as a formal treatment of the hierarchy of relations of being a part in a degree. We begin with an exposition of rough mereological calculus in the form of a logic $L_{rm}$.

### Syntax of $L_{rm}$

It will be the standard syntax of the predicate calculus in which we will have the following basic ingredients:

**Variables:** $x, x_1, x_2, \ldots,$ $y, y_1, y_2, \ldots,$ $z, z_1, z_2, \ldots$ of type *set_element* and $r, r_1, r_2, \ldots, s, s_1, s_2, \ldots$ of type *lattice_element*;

**Constants:** $\omega$ of type *lattice_element*;

**Predicate symbols, function symbols:** $\leq$ of type (*lattice_element, lattice_element*) and $\mu$ of type (*set_element, set_element, lattice_element*);

**Auxiliary symbols:** propositional connectives: $\vee$, $\wedge$, $\Longrightarrow$, $\neg$, quantifier symbols: $\forall$, $\exists$ and commas, parentheses.

**Formulae:** atomic formulae are of the form $\mu(x,y,r)$, $s \leq r$ and formulae are built from atomic formulae as in the predicate calculus.

**Axioms:** the following are axioms of $L_{rm}$

(A1) $\forall x.\mu(x,x,\omega)$;

(A2) $\forall x.\forall y.\{\mu(x,y,\omega) \Longrightarrow \forall s.\forall r.\forall z.[\mu(z,x,s) \wedge \mu(z,y,r) \Longrightarrow (s \leq r)]\}$;

(A3) $\forall x.\forall y.\{\mu(x,y,\omega) \wedge \mu(y,x,\omega) \Longrightarrow$

$$\forall s.\forall r.\forall z.[\mu(x,z,s) \wedge \mu(y,z,r) \Longrightarrow (s \leq r)]\};$$

(A4) $\exists x.\forall y.\mu(x,y,\omega)$;

(A5) $\forall x.\forall y.\{[\forall z.[[\exists u.\neg(\mu(z,u,\omega)) \wedge \mu(z,x,\omega)] \Longrightarrow$

$$\exists t.(\exists w.(\neg\mu(t,w,\omega)) \wedge \mu(t,z,\omega) \wedge \mu(t,y,\omega)] \Longrightarrow \mu(x,y,\omega)\};$$

and the axiom schemata $(A6)_n$ for n = 2,3,.... where

$(A6)_n$

$$\forall x_1.\forall x_2....\forall x_n.\exists y.(\alpha_n(x_1,x_2,...,x_n,y) \wedge \beta_n(x_1,x_2,...,x_n,y) \wedge \gamma_n(x_1,x_2,...,x_n,y))$$

where

$\alpha_n(x_1,x_2,..,x_n,y)$ :

$$\forall z.\{[\exists t.(\neg\mu(z,t,\omega)) \wedge \mu(z,y,\omega)] \Longrightarrow$$

$$\exists x_i.\exists w.[(\exists u.(\neg\mu(w,u,\omega))) \wedge \mu(w,z,\omega) \wedge \mu(w,x_i,\omega)]\};$$

$\beta_n(x_1,x_2,..,x_n,y)$:

$$\mu(x_1,y,\omega) \wedge \mu(x_2,y,\omega) \wedge ... \wedge \mu(x_n,y,\omega);$$

$\gamma_n(x_1,x_2,..,x_n,y)$ :

$$\forall z.\{[\alpha_n(x_1,x_2,..,x_n,z) \wedge \beta_n(x_1,x_2,..,x_n,z)] \Longrightarrow \mu(y,z,\omega)\}.$$

The formal introduction to rough mereology is expanded below when we discuss rough inclusions.

## Semantics of $L_{rm}$

We will call an *interpretation* of $L_{rm}$ a triple $M = (U^M, L^M, F^M)$ where $U^M$ is a finite set, $L^M$ is a (complete) lattice with the lattice partial order $\leq^M$ and with the greatest element $\Omega^M$ and $F^M$ is a mapping which assigns to constants and predicate symbols of $L_{rm}$ their denotations in $M$ in the following manner: $F^M(\omega) = \Omega^M$, $F^M(\leq) = \leq^M$ and $F^M(\mu) = \mu^M \subseteq U^M \times U^M \times L^M$, where the relation $\mu^M \subseteq U^M \times U^M \times L^M$ is a function i.e. $\mu^M : U^M \times U^M \longrightarrow L^M$.

An $M$-value assignment $g$ is a mapping which assigns to any variable $x$ of $L_{rm}$ of type *set_element* the element $g(x) \in U^M$ and to any variable $r$ of $L_{rm}$ of type *lattice_element* the element $g(r) \in L^M$. For an $M$-value assignment $g$, a variable $x$ of $L_{rm}$ of type *set_element* and an element $u \in U^M$, we denote by the symbol $g[u/x]$ the $M$-value assignment defined by the conditions: $g[u/x](v) = g(v)$ in case $v \neq x$ and $g[u/x](x) = u$; the same convention will define $g[p/r]$ in case of a variable $r$ of type *lattice_element* and $p \in L^M$.

For a formula $\alpha$ of $L_{rm}$, we denote by the symbol $[\alpha]^{M,g}$ the meaning of the formula $\alpha$ in the model $M$ relative to an $M$-value assignment $g$ by the following conditions

(M1) $[\mu(x,y,r)]^{M,g} = true$ iff $\mu^M(g(x),g(y)) = p$ for some $p \geq^M g(r)$;

(M2) $[s \leq r]^{M,g} = true$ iff $g(s) \leq^M g(r)$;

(M3) $[\alpha \vee \beta]^{M,g} = true$ iff $[\alpha]^{M,g} = true$ or $[\beta]^{M,g} = true$;

(M4) $[\neg\alpha]^{M,g} = true$ iff $[\alpha]^{M,g} = false$;

(M5) $[\exists x.\alpha]^{M,g} = true$ iff there exists $u \in U^M$ such that $[\alpha]^{M,g[u/x]} = true$;

(M6) $[\exists r.\alpha]^{M,g} = true$ iff there exists $p \in L^M$ such that $[\alpha]^{M,g[p/r]} = true$.

It follows that the intended meaning of a formula $\mu(x,y,r)$ is that "*the object $x$ is a part of the object $y$ in degree at least $r$*".

A formula $\alpha$ is *true in an interpretation $M$* iff $\alpha$ is $M,g$-true (i.e. $[\alpha]^{M,g} = true$) for any $M$-value assignment $g$. An interpretation $M$ is a *model* of $L_{rm}$ iff all axioms (A1)-(A6) are true in $M$.

**Rough inclusions.** A real function $\mu^M(X,Y)$ on a universe of objects $U^M$ with values in the interval $[0,1]$ is called *a rough inclusion*. It satisfies the following conditions which are translations of respective axioms $(A1)-(A6)_n$.

(A) $\mu^M(X,X) = 1$ for any $X$ ;

(B) $\mu^M(X,Y) = 1$ implies that $\mu^M(Z,Y) \geq \mu^M(Z,X)$ for any triple $X,Y,Z$;

(C) there is $N$ such that $\mu^M(N,X) = 1$ for any $X$. An object $N$ satisfying (C) is a $\mu$-null object: such objects are excluded in mereology of Leśniewski.

We let $X =_\mu Y$ iff $\mu^M(X,Y) = 1 = \mu^M(Y,X)$ and $X \neq_\mu Y$ iff $non(X =_\mu Y)$.

We have other conditions for rough inclusion:

(D) if objects $X,Y$ have the property :

if $Z \neq_\mu N$ and $\mu^M(Z,X) = 1$

then there is $T \neq_\mu N$

with $\mu^M(T,Z) = 1 = \mu^M(T,Y)$

then it follows that: $\mu^M(X,Y) = 1$.

(D) is an inference rule: it is applied to infer the relation of being a part from the relation of being a subpart.

(E) For any collection $\Gamma$ of objects there is an object $X$ with the properties:

(i) if $Z \neq_\mu N$ and $\mu^M(Z,X) = 1$ then there are $T \neq_\mu N, W \in \Gamma$ such that

$$\mu^M(T,Z) = \mu^M(T,W) = \mu^M(W,X) = 1;$$

(ii) if $W \in \Gamma$ then $\mu^M(W,X) = 1$;

(iii) if $Y$ satisfies the above two conditions then $\mu^M(X,Y) = 1$.

(E) will be applied below to show the existence and uniqueness of classes of objects.

We now outline the way in which mereology of Leśniewski follows out of rough mereology.

## Rough inclusions: reduced models and Leśniewski's mereology

Given a model $M$ of $L_{rm}$, $M = (U^M, L^M, F^M)$, we will call the function $\mu^M : U^M \times U^M \longrightarrow L^M$ the $M$-rough inclusion. We define a relation $congr(\mu^M)$ on the set $U^M$ by letting for $u, w \in U^M$ : $u\ congr(\mu^M)\ w$ iff $\mu^M(u,w) = \Omega^M = \mu^M(w,u)$ . The following proposition, whose proof follows immediately by (A2) and (A3) and is therefore omitted, establishes the basic properties of the relation $congr(\mu^M)$ and demonstrates it to be a $\mu^M$—congruence.

**Proposition 1** *The relation $congr(\mu^M)$ is an equivalence relation on the set $U^M$ and we have*
*(i) if $u\ congr(\mu^M)\ w$ then $\mu^M(v,w) = \mu^M(v,u)$;*
*(ii) if $u\ congr(\mu^M)\ w$ then $\mu^M(u,v) = \mu^M(w,v)$ for any triple $u, v, w \in U^M$.*

□

We denote by $u_\mu$ the class of $congr(\mu^M)$ which contains $u$. It follows that the rough inclusion can be factored throughout the relation $congr(\mu^M)$ i.e. we define the quotient set $U^M_\mu = U^M / congr(\mu^M)$ and the quotient function

$$\mu^M_\sim : U^M_\mu \times U^M_\mu \longrightarrow L^M$$

by letting $\mu^M_\mu(u_\mu, w_\mu) = \mu^M(u,w)$; clearly, the pair $(U^M_\mu, \mu^M_\sim)$ introduces a model $M_\sim$ of $L_{rm}$. In the sequel we will always work with a fixed reduced model $M_\sim$. We denote by the symbol $n_\mu$ the *null object* i.e. the object existing in virtue of (A4) and such that $\mu^M_\sim(n_\mu, w_\mu) = \Omega^M$ for any $w_\mu \in U^M_\mu$. We will write $u_\mu \neq_\mu n_\mu$ to denote the fact that the object $u_\mu$ is not the null object. Let us recall that the existence of a null object in a model of mereology of Leśniewski reduces the model to a singleton, as observed by Tarski. In the sequel, for simplicity of notation, we will write $\mu$ in place of $\mu^M_\sim$, $U$ in place of $U^M_\mu$, $u$ in place of $u_\mu$ etc. We will call the rough inclusion $\mu$ a *strict rough inclusion* when it satisfies the condition $\mu(x,n) = 0$ for any non-null object $x$; we observe that any standard rough inclusion is strict.

We now show how the rough inclusion $\mu$ introduces in $U$ a model of mereology of Leśniewski. To this end, we define a binary relation $part(\mu)$ on the set $U$ by letting

$u\ part(\mu)\ w$ iff $\mu(u,w) = \Omega^M$ and it is not true that $\mu(w,u) = \Omega^M$.

Then we have the following proposition whose straightforward proof is omitted.

**Proposition 2.** (i) the relation $part(\mu)$ satisfies the conditions (P1) and (P2);

(ii) the relation $ingr(\ part(\mu)\ )$ satisfies the following for any pair $u, w \in U$:

$$u\ ingr(part(\mu))\ w \text{ iff } \mu(u,w) = \Omega^M.$$

□

We now define in the model $M_\sim$ for any collection $\Psi$ of objects in $U$, the notions of a set of objects in $\Psi$ and of a class of objects in $\Psi$. We will say then that $u \in U$ is a *set of objects in* $\Psi$, *u set* $\Psi$ for short, when

(S1) for any $w \neq_\mu n$ such that $w\ ingr(part(\mu)\ )\ u$ there exist $v \neq_\mu n$ and $t \in \Psi$ such that $v\ ingr(part(\mu)\ )\ w$, $v\ ingr(part(\mu))\ t$, $t\ ingr(part(\mu)\ )\ u$;

if in addition, we have

(S2) $t\ ingr(part(\mu)\ )\ u$ for any $t \in \Psi$ ;

(S3) for any $t$, if $t$ satisfies (S1) and (S2) with $\Psi$ then $u\ ingr(part(\mu)\ )\ t$

then we say that $u$ is a *class of objects in* $\Psi$, *u class* $\Psi$ for short. It follows from (A6) that for any collection $\Psi$ there exists a unique object $u$ such that $u$ *class* $\Psi$ and there exists objects of the form *set* $\Psi$. We have therefore

**Proposition 3.** The pair $(\ U - \{n\}, part(\mu) \upharpoonright (U - \{n\}) \times (U - \{n\})$ is a model of mereology.

□

We apply the above theoretical scheme to the task of formalization of design, synthesis and control of complex systems on the basis of knowledge learned from data tables.

## Approximate logic of design and synthesis

**Design agents. Requirements.** The designer operates on the set *Des_Ag* of design agents; any design agent *dag* is equipped with an information system $\mathbf{A}_{dag} = (U_{dag}, A_{dag})$ and a rough inclusion $\mu_{dag}$. The table $\mathbf{A}_{dag}$ describes objects (possibly complex) from a universe $U_{dag}$ in the language of attributes in $A_{dag}$. The variable $b_{dag}$ runs over objects in $U_{dag}$. A valuation $v_X$ where $X$ is a set of design agents is a function which assigns to any $b_{dag}$ for $dag \in X$ an element $v_X(b_{dag}) \in U_{dag}$. A *(designer) requirement* $\Psi(ag)$ at $ag$ is a formula in the logic $C(A_{dag}, V)$ of conditional attributes (Skowron 1995); the symbol $\Psi$ will denote a requirement at some design agent. The symbol $x \models_d \Psi_d$ will denote that $x$ satisfies $\Psi$.

**Synthesis agents.** Any synthesis agent *ag* is assigned a *label*

$$lab(ag) = \{U(ag), \pi(ag, dag), D(ag), St(ag), L(ag), \mu_o(ag), F(ag)\}$$

where

$U(ag)$ is the universe of objects at $ag$;

for any synthesis agent $ag$ there exists a design agent $dag$ and a mapping $\pi(ag, dag) : U(ag) \longmapsto U_{dag}$;

$D(ag) = \{U(ag), A(ag), d(ag)\}$ is the decision system (Pawlak 1991) of $ag$; $A(ag)$ is the set of conditional attributes of $ag$. The value $d(ag)(x)$ is a designer requirement $\Psi$ satisfied by $\pi(ag, dag)(x)$;

$St(ag) \subseteq U(ag)$ is the set of standard objects (standards) at $ag$;

$L(ag)$ is a set of unary predicates at $ag$ (specifying properties of objects in $U(ag)$). Predicates of $L(ag)$ are constructed as formulas in $C(A(ag), V)$ ;

$\mu_o(ag) \subseteq U(ag) \times U(ag) \times [0, 1]$ is a pre-rough inclusion at $ag$ (Polkowski & Skowron 1996a); usually it is defined from the information system of the agent $ag$ by the formula

$$\mu^o(x, y) = \frac{cardinality\{a \in A(ag) : a(x) \neq a(y)\}}{cardinality A(ag)};$$

$F(ag)$ is a set of *mereological connectives at ag* (Polkowski & Skowron 1995), (Skowron & Polkowski 1995), (Polkowski & Skowron 1996b).

**Mereology $\mu_{DS}$.** Synthesis agents classify their objects by means of pre - rough inclusions; any pre - rough inclusion $\mu_o(ag)$ can be extended (Polkowski & Skowron 1996a) to a rough inclusion $\mu(ag)$ on the set $2^{U(ag)}$ of subsets of $U(ag)$. The $D\_S-$ *mereology* $\mu_{DS}$ (the *design - synthesis mereology*) is a family $\{\mu(ag) : ag \in Ag\}$ where $\mu(ag)$ is a fixed extension of $\mu_o(ag)$ for any $ag \in Ag$. Mereology $\mu_{DS}$ defines design objects (categories of real objects): a *design object* $x$ is a $class(\mu(ag))\Gamma$ where $\Gamma \subseteq U(ag)$. Thus, *design objects* are *mereological classes of synthesis objects*. On these classes the mereologies of design agents act decomposing them into simpler classes.

The communication between synthesis spaces and design spaces is provided by mappings $\pi(ag, dag)$ : for $x \in U(ag)$, the value $\pi(ag, dag)(x) \in U(dag)$ is a design object . Let us observe that while $\pi(ag, dag)(x)$ is unique, the object $x$ may belong to more than one design object. This causes the ambiguity in communication among design agents and synthesis agents: while a synthesis agent $ag$ classifies $x$ as $cat(x) = \pi(ag, dag)(x)$, the design agent $dag$ can regard $x$ as an element of a category $cat'(x)$ distinct from $\pi(ag, dag)(x)$. However, categories $cat(x)$, $cat'(x)$ containing $x$ should be regarded as similar. We express this similarity on the higher level of requirements by means of a tolerance relation $Des\_sat$.

**Satisfactory satisfiability of requirements.** The requirements of the designer specify classes of ideal objects but the ultimate purpose of design is a real object whose category would satisfy $\Psi$. It can happen that an object whose category satisfies a requirement $\Psi' \neq \Psi$ is accepted as satisfying $\Psi$ in a degree satisfactory to the designer. We denote by $Des\_req$ the set of designer requirements; the vagueness of designer requirements will be formalized by means of a tolerance relation $des\_sat$ on the set $Des\_req$; for $\Psi, \Psi' \in Des\_req$, $\Psi$ $des\_sat$ $\Psi'$ will read "any object satisfying $\Psi'$ (resp. $\Psi$) satisfies $\Psi$ (resp. $\Psi'$) in degree satisfactory to the designer ". We denote by the symbol $[\Psi]$ the tolerance class $des\_sat(\Psi)$. We denote by the symbol $\Psi^\vee$ the disjunction of formulas in $[\Psi]$ i.e. $\Psi^\vee$ is $\vee\{\Psi' : \Psi' \in [\Psi]\}$. Clearly, if $x \models_d \Psi^\vee$ then $x$ satisfies $\Psi$ in the satisfactory degree.

**Mereological compatibility of requirements.** We assume that $\mu_{dag}$ is compatible with $des\_sat$ i.e. if $x = class(\mu_{dag})\ \{x_1, x_2, ..., x_k\}$, $x_i \models_d \Psi_i$, $x \models_d \Psi$, $y_i \models_d \Psi_i^\vee$, $y = class(\mu_{dag})\ \{y_1, y_2, .., y_k\}$ then $y \models_d \Psi^\vee$. The compatibility condition means that the designer schemes are designed as insensitive to local communication ambiguities.

**Approximate logic of synthesis.** The symbol $b_{ag}$ will denote the variable which runs over objects in $U_{ag}$. A valuation $v_X$ where $X$ is a set of synthesis agents is a function which assigns to any $b_{ag}$ for $ag \in X$ an element $v_X(b_{ag}) \in U_{ag}$. The symbol $v_{ag}^x$ denotes $v_{\{ag\}}$ with $v_{\{ag\}}(b_{ag}) = x$.

We now define syntax of a logic of approximate formulas $L$ (Komorowski, Polkowski, & Skowron 1996). The *atomic formulas* of $L$ are of the form $< st(ag), \Phi(ag), \varepsilon(ag) >$ where $st(ag) \in St(ag), \Phi(ag) \in L(ag), \varepsilon(ag) \in [0, 1]$ for $ag \in Ag$. The *formulas* of $L$ are built from atomic formulas by means of classical propositional connectives $\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow$ and of modal unary propositional connective $\diamond$ (cf. Sect.4 for semantics of $\diamond$). The *semantics* of $L$ is defined as follows. For $ag = Root(C)$, $v \in V_C$, we say that $v$ satisfies a formula $\alpha =< st(ag), \Phi(ag), \varepsilon(ag) >$, symbolically $v \models \alpha$, in case $E_{\mu(ag)}(v(b_{ag}), st(ag)) \geq \varepsilon$ and $v(b_{ag}) \in \Phi(ag)_{D(ag)}$ (see Appendix). We let $v \models \alpha \wedge \beta$ in case $v \models \alpha$ and $v \models \beta$; $v \models \neg\alpha$ in case $non(v \models \alpha)$. For a formula $< st(ag), \Phi(ag), \varepsilon(ag) >$ of $L$, we write $x \models< st(ag), \Phi(ag), \varepsilon(ag) >$ iff $v_{ag}^x \models$ $< st(ag), \Phi(ag), \varepsilon(ag) >$.

**Decision rules of synthesis agents.** From the triple $(D(ag), L(ag), \mu(ag))$, the agent $ag$ generates by the standard techniques (Skowron 1995) the decision rules of the form

$$< st(ag), \Phi(ag), \varepsilon(ag) > \Longrightarrow \Psi^\vee.$$

The meaning of this rule is:

$$(< st(ag), \Phi(ag), \varepsilon(ag) > \Longrightarrow \Psi^\vee) \text{ is } true$$
$$\text{iff}$$
$$(\pi(ag, dag)(st(ag)) \models_d \Psi) \wedge [\, x \models$$
$$< st(ag), \Phi(ag), \varepsilon(ag) > \Rightarrow \pi(ag, dag)(x) \models_d \Psi^\vee].$$

Informally, this means that the design class of the standard $st(ag)$ satisfies the requirement $\Psi$ and design classes of objects satisfactory close to $st(ag)$ (closeness measured by $\varepsilon(ag)$ satisfy $\Psi^{\vee}$ i.e. they satisfy $\Psi$ in a satisfactory degree.

## Synthesis schemes

**Local decomposition schemes.** By means of the rough inclusion $\mu_{dag}$ the agent $dag$ induces on the universe $U_{dag}$ the mereological relation $part_{dag}$ in the sense of Leśniewski (Leśniewski 1992). The relation $part_{dag}$ establishes a *local decomposition scheme* of some complex objects in the universe $U_{dag}$ into simpler objects i.e. the relation $x\ class(\mu_{dag})\ \{x_1, x_2, .., x_k\}$ means that $x$ is designed (built) from parts $x_1, x_2, .., x_k$. In this way the designer establishes a hierarchy $\{part_{dag}\}$ of local decomposition schemes on the set of possible complex objects. Local decomposition schemes can be composed in the sense that whenever the relation $part_{dag}$ expresses an object $x$ as built from parts $x_1, x_2, .., x_k$ and the relation $part_{dag'}$ expresses, say, $x_1$ as built from parts $x_{11}, x_{12}, .., x_{1m}$ then the composition $part_{dag} \circ part_{dag'}$ expresses $x$ as built from parts $x_{11}, x_{12}, ..., x_{1m}, x_2, .., x_k$.

**The designer language $Link_d$.** The designer task is now to establish for a given complex object $x$, specified by a requirement $\Psi$, a *scheme of design agents* (for simplicity we assume this scheme to be a tree) such that the object $x$ can be decomposed over the scheme by means of a composition of local decomposition schemes into primitive objects which belong to the universes of leaf agents of the scheme. We define a language $Link_d \subseteq Des\_Ag^{+}$ where $Des\_Ag^{+}$ is the set of all finite non-empty strings over $Des\_Ag$. For a string $\mathbf{dag} = dag_1 dag_2 ... dag_k dag$, where $dag$ is the *root agent* and $dag_1, dag_2, .., dag_k$ are *leaf agents,* we have $\mathbf{dag} = dag_1 dag_2 ... dag_k dag \in Link_d$ iff there exist $x, x_1, x_2, .., x_k$, $x$ such that $x\ class(part_{dag})\ \{x_1, x_2, .., x_k\}$ where $x_i \in U_{dag_i}$ for $i \le k$, $x \in U_{dag}$. We let $set(\mathbf{dag}) = \{dag_1, dag_2 ..., dag_k, dag\}$. For $L \subseteq Link_d$, we define $Des\_Ag(L) = \cup\{set(\mathbf{dag}) : \mathbf{dag} \in L\}$ and we denote by $\le$ a relation on $Des\_Ag(L)$ defined by: $dag \le dag'$ if and only if there exists $\mathbf{dag} \in L$ such that $dag, dag' \in set(\mathbf{dag})$ and $dag'$ is the $\mathbf{dag}$-target. A set $L \subseteq Link_d$ is *a construction support* in case $(Des\_Ag(L), \le)$ is a tree. From now on $L$ will denote a construction support.

**Constructibility mapping and design operations.** For $\mathbf{dag} = dag_1 dag_2 . ..dag_k dag \in L$, we define the $\mathbf{dag}$-*constructibility relation* $\rho(\mathbf{dag}) \subseteq U(dag_1) \times U(dag_2) \times ... \times U(dag_k) \times U(dag)$ by letting $(x_1, x_2, ..., x_k, x) \in \rho(\mathbf{dag})$ iff $x\ class(part_{dag})\{x_1, x_2, .., x_k\}$.

The *constructibility mapping* $con(\mathbf{dag})$, associated with the relation $\rho(\mathbf{dag})$, will be defined by $con(\mathbf{dag})(x_1, x_2, ..., x_k) = x$ iff $(x_1, x_2, ..., x_k, x) \in \rho(\mathbf{dag})$.

The existence of the mapping $con(\mathbf{dag})$ follows from the uniqueness of classes in mereology generated by the relation $part_{dag}$. The constructibility mapping $con(\mathbf{dag})$ is in general a many - to -one mapping. We select from $con(\mathbf{dag})$ one -to -one mappings called design operations defined as follows: a *design operation* $o(\mathbf{dag})$ associated with $\mathbf{dag}$ is a one - to -one partial mapping such that if $o(\mathbf{dag})(x_1, x_2, .., x_k) = x$ then $con(\mathbf{dag})(x_1, x_2, .., x_k) = x$ and *range* $o(\mathbf{dag})$ = *range* $con(\mathbf{dag})$.

We denote by the symbol $O(\mathbf{dag})$ the collection of design operations associated with $\mathbf{dag}$. We will write $o(dag)$ instead of $o(\mathbf{dag})$, where $dag = root(\mathbf{dag})$.

We observe that the relation $\rho(\mathbf{dag})$, the mapping $con(\mathbf{dag})$ and design operations $o(\mathbf{dag})$ are compatible with requirements with respect to the tolerance $des\_sat$ viz. the condition

$(x_1, x_2, ..., x_k, x) \in \rho(\mathbf{dag})$, $x_i \models_d \Psi_i$, $x \models_d \Psi$, $y_i \models_d \Psi_i^{\vee}$ and $(y_1, y_2, ..., y_k, y) \in \rho(\mathbf{dag})$

implies $y \models_d \Psi^{\vee}$ and similar conditions hold for $con(\mathbf{dag})$ and $o(\mathbf{dag})$.

**Design schemes.** Now, given a requirement $\Psi$ (a formula in the conditional logic of the designer), the designer initiates the communication and negotiation process among design agents. This process results eventually in a design scheme which is able to design an object satisfying $\Psi$ in the satisfactory degree. We describe this initializing process in the following steps.

**Step 1.** The designer selects a design agent $dag_o$ such that there exists an object $x_o \in U_{dag_o}$ which satisfies the requirement $\Psi^{\vee}$ and there exists $\mathbf{dag}_o = dag_{o,1} dag_{o,2} ... dag_{o,k} dag_o \in Link_d$ such that the condition

$$con(\mathbf{dag}_o)(x_1, x_2, ..., x_k) = x$$

holds for some vector argument $(x_1, x_2, ..., x_k)$.

**Step 2.** Agents $dag_o, dag_{o,1}, dag_{o,2}, .., dag_{o,k}$ negotiate the choice of requirements $\Psi_{o,1}^{\vee}, \Psi_{o,2}^{\vee}, \cdot\ .., \Psi_{o,k}^{\vee}$ and a vector argument $(x_1, x_2, ..., x_k)$ such that any $x_i$ satisfies the requirement $\Psi_{o,i}^{\vee}$; this choice determines the design operation $o(\mathbf{dag}_o)$ with the property that $o(\mathbf{dag}_o)(x_1, x_2, .., x_k) = x$. The negotiated condition is the following: if $(y_1, y_2, .., y_k) \in$ *domain* $o(\mathbf{dag}_o)$ and any $y_i$ satisfies the requirement $\Psi_{o,i}^{\vee}$ then $o(\mathbf{dag}_o)(y_1, y_2, .., y_k)$ satisfies the requirement $\Psi^{\vee}$.

Factors involved in negotiations and influencing them may be e.g. the complexity of $o(\mathbf{dag})$, the cost of assembling via $o(\mathbf{dag})$, the accessibility of $x_1, .., x_k$ etc.

Next, the designer repeats steps 1 and 2 with the agents $dag_{o,1}, dag_{o,2}, .., dag_{o,k}$ and the respective requirements $\Psi_{o,1}^{\vee}, ..., \Psi_{o,k}^{\vee}$ .

The negotiation process continues with new requirements. The process stops when all branches terminate with strings in $Link_d$ whose all leaf agents can satisfy the chosen requirements with inventory objects.

We observe that the result of negotiations for any non-inventory agent $dag$ in $L$ can be described by means of the pair $lab(dag) = (\Psi^{\vee}(dag), o(dag))$; the singleton $lab(dag) = (\Psi^{\vee}(dag))$ summarizes the negotiation outcome for any inventory agent $dag$ in $L$. We will call the tuple $lab(dag)$ the *design label* of the agent $dag$. The construction support $L$ along with the set $\{lab(dag)\}$ of design labels of its agents will be called the *design scheme* and denoted by $D\_s = (L, \{lab(dag) : dag \in L\})$. The agent $dag_o$ will be called the *root agent* of $D\_s$; $dag(1), .., dag(m)$ are leaf agents of $D\_s$. We write $D\_s(\mathbf{dag}_o, \mathbf{dag}_1, .., \mathbf{dag}_m)$ when $\mathbf{dag}_o, \mathbf{dag}_1, .., \mathbf{dag}_m$ are all strings used to construct $D\_s$. $D\_s(\mathbf{dag}_i)$ is called *an elementary design scheme.*

The *operation term* of $D\_s$, denoted by the symbol $T(D\_s)$ will be defined by induction on the number of strings from *Link* in $L$:

(i) if $L = \{\mathbf{dag}_o\}$ then we let $T(D\_s) = o(dag_o)(b_{dag_{o,1}}, b_{dag_{o,2}}, ..., b_{dag_{o,k}})$;

(ii) if $L = \{\mathbf{dag}_o, \mathbf{dag}_1, ..., \mathbf{dag}_k\}$,
an agent $dag$ is a leaf agent of $D\_s$,
$\mathbf{dag}_{k+1} = dag_1 dag_2 ... dag_n dag$,
$T(D\_s) = o(dag_o)(...(...(..., b_{dag}, ...)...)$
**then** $D\_s'$ results from $D\_s$ by attaching $\mathbf{ag}_{k+1}$ to $D\_s$ at $dag$ i.e.

$$T(D\_s)' = o(dag_o)(...(...(..., o(dag)(b_{dag_1}, b_{dag_2}, ..., b_{dag_n}), ...)...).$$

The term $T(D\_s)$ represents the composition of operations along the construction support $L$. This composition is a global operation which assigns to argument vectors of objects at leaf agents of $D\_s$ the value which is an object in the universe of the root agent of $D\_s$.

The following proposition resumes the upshot of the negotiation process.

**Proposition 4.** For any valuation $v_X$ on the set $X$ of leaf agents $dag(1), .., dag(m)$ of the design scheme $D\_s$ where $v(b_{dag(i)})$ satisfies $\Psi^{\vee}(dag(i))$ and $lab(ag))$ $= (\Psi^{\vee}(dag_i))$ for $i = 1, 2, ..., m$, the uniquely defined object

$$T(D\_s)(v(b_{dag(1)}), v(b_{dag(2)}), ..., v(b_{dag(k)})) \in U_{dag_o}$$
satisfies $\Psi^{\vee}(dag_o)$

where $lab(dag_o) = (\Psi^{\vee}(dag_o), o(dag_o))$.

□

The projection

$$Dg = (\Psi^{\vee}(ag_1), \Psi^{\vee}(ag_2), .., \Psi^{\vee}(ag_k), \Psi^{\vee}(ag_o))$$

of the design scheme $D\_s$ onto the requirement space will be called a *designer goal*; the designer goal $Dg$ expresses the existence of a design scheme which can design an object satisfying $\Psi^{\vee}(ag_o)$ from objects satisfying $\Psi^{\vee}(ag_i), i = 1, 2, .., k$, respectively.

In the synthesis stage the synthesis agents are organized into a synthesis scheme modelled on a given design scheme.

## Synthesis (of approximate reasoning scheme)

We begin with a set $Ag$ of *synthesis agents* (called simply *agents* in what follows) and a set $I$ of primitive objects (the *inventory*). Elements of $lab(ag)$ constitute the basic knowledge of a synthesis agent $ag$; other elements are worked out in negotiations among synthesis agents and in their interactions with the design agents. We now describe this process.

**The synthesis language** *Link*. We recall that the designer creates a language $Link_d \subseteq Des\_Ag^+$; the language $Link_d$ is a pattern according to which a language $Link \subseteq Ag^+$ is defined in the process of communication among synthesis agents. To this end, the agents $ag_1, ag_2, .., ag_k, ag_o$ form the string $\mathbf{ag} = ag_1 ag_2 . .. ag_k ag_o \in Link$ if and only if there exist design agents $dag_1, dag_2, . .., dag_k, dag_o$ such that

$$\mathbf{dag} = dag_1 dag_2 ... dag_k dag_o \in Link_d,$$

and $\pi(ag_i, dag_i)$ is defined for each $i$,
and there exist objects $x_1 \in U(ag_1), ..., x_k \in U(ag_k)$, $x \in U(ag_o)$
such that

$$(\pi(ag_1, dag_1)(x_1), .., \pi(ag_k, dag_k)(x_k), \pi(ag_o, dag_o)(x)) \in \rho(\mathbf{dag}).$$

For $\mathbf{ag} = ag_1 ag_2 ... ag_k ag_o \in Link$, an *operation* $o(\mathbf{ag})$ is a partial one-to-one mapping with *domain* $o(\mathbf{ag}) \subseteq U(ag_1) \times U(ag_2) \times ... \times U(ag_k)$ and *range* $o(\mathbf{ag}) \subseteq U(ag_o)$ such that
**if** $o(ag_o)(x_1, x_2, .., x_k) = x$ **then**

$$o_d(\mathbf{dag})(\pi(ag_1, dag_1)(x_1), .., \pi(ag_k, dag_k)(x_k)) = \pi(ag_o, dag_o)(x)$$

for some unique design operation $o_d(\mathbf{dag})$; let us observe that to a design operation $o_d(\mathbf{dag})$ there may correspond more than one operation $o(\mathbf{ag})$. We let $\mathbf{dag} = \Pi(\mathbf{ag})$ and $o_d(\mathbf{dag}) = \Pi(o(\mathbf{ag}))$. The operation $\Pi$ extends in the natural way to compositions of operations.

**Elementary constructions.** We define *an elementary construction c*: if $\mathbf{ag} = ag_1 ag_2 .. ag_k ag_o \in Link$, then an expression

$$c = (\mathbf{ag}, p\_sign(ag_1), p\_sign(ag_2), . ., p\_sign(ag_k), p\_sign(ag_o))$$

will be called an *elementary construction* with *pre-signatures*

$$p\_sign(ag_o) = (st(ag_o), \Phi(ag_o), \varepsilon(ag_o), o(ag_o))$$

and $p\_sign(ag_i) = (st(ag_i), \Phi(ag_i), \varepsilon(ag_i))$ if there exists a $D\_s(\mathbf{dag})$ with
$\mathbf{dag} = dag_1 dag_2 ... dag_k dag_o = \Pi(\mathbf{ag})$,
$o_d(dag_o) = \Pi(o(\mathbf{ag}))$,
$lab(dag_o) = (\Psi^{\vee}(dag_o), o_d(dag_o))$,
$lab(dag_i) = (\Psi^{\vee}(dag_i))$

such that $< st(ag_i), \Phi(ag_i), \varepsilon(ag_i) > \Longrightarrow \Psi^{\vee}(ag_i)$ is true for $i = 0, 1, 2, .., k$.

We will say that $c$ *projects onto* $D\_s$. To stress the relationship between **ag** and $c$ we will write $c(\mathbf{ag})$ instead of $c$ and $\mathbf{ag}(c)$ instead of **ag**. We let $ag_o = Root(c), \{ag_1, ag_2, .., ag_k\} = Leaf(c)$, $\{ag_1, ag_2, .., ag_k, ag\} = Ag(c)$.

**Constructions.** For elementary constructions $c$, $c'$ with $Ag(c) \cap Ag(c') = \{ag\}$ where $ag = Root(c) \in Leaf(c')$, we define *the ag-composition* $c \star_{ag} c'$ of $c$ and $c'$ with $Root(c \star_{ag} c') = Root(c')$, $Leaf(c \star_{ag} c') = (Leaf(c) - \{ag\}) \cup (Leaf(c')$, $Ag(c \star_{ag} c') = Ag(c) \cup Ag(c')$. The composition $c \star_{ag} c'$ is called a *construction* if there exists a $D\_s(\mathbf{dag}_o, \mathbf{dag}_1)$ such that $c$ projects onto $\mathbf{dag}_o$ and $c'$ projects onto $\mathbf{dag}_1$; we say then that $c \star_{ag} c'$ *projects onto* $D\_s(\mathbf{dag}_o, \mathbf{dag}_1)$. A *construction* is any expression $C$ obtained from a set of elementary constructions by applying the composition operation a finite number of times. By $V_C$ we denote the set of partial valuations $v_{Ag(C)}$. By $T(C)$ we denote the unique term composed of operations $o(ag)$ such that $\Pi(T(C)) = T(D\_s)$ where $C$ projects onto $D\_s$.

**$(C, \Phi, \varepsilon)$–schemes.** For an elementary construction $c = c(\mathbf{ag})$ as above, with $p\_sign(ag) = (st(ag), \Phi(ag), \varepsilon(ag), o(ag))$, we define a $(c, \Phi, \varepsilon) - scheme$ as a pair $(c(\mathbf{ag}), sign(ag))$ where $sign(ag) = p\_sign(ag) \cup \{f(ag)\}, \Phi = \Phi(ag), \varepsilon = \varepsilon(ag)$ and $f(ag) \in F(ag)$ satisfies the condition:

(Unc(c))if $\mu_o(ag_i)(x_i, st(ag_i)) \geq \varepsilon(ag_i)$ for $i = 1, 2, .., k$
then $\mu_o(ag)(o(ag)(x_1, x_2, .., x_k), st(ag)) \geq f(\varepsilon(ag_1), \varepsilon(ag_2), .., \varepsilon(ag_k)) \geq \varepsilon(ag)$.

The construction $c$ is said to be the *support* of the $(c, \Phi, \varepsilon) - scheme$.

A construction $C$ composed of elementary constructions $c_1, .., c_m, c_o$ with $Root(C) = Root(c_o) = ag_o$ is the support of a $(C, \Phi, \varepsilon)$–*scheme* when each $c_i$ is the support of a $(c_i, \Phi_i, \varepsilon_i)$–*scheme*, where $\Phi_i = \Phi(Root(c_i))$, $\varepsilon_i = \varepsilon(Root(c_i))$, $\Phi = \Phi(ag_o)$ and $\varepsilon = \varepsilon(ag_o)$. For valuations $v = v_{Leaf(C)}$, $v' = v_{Root(C)} \in V_C$, we write $v \longrightarrow_C v'$ iff $T(C)(v) = v'$. We have a proposition.

**Proposition 5.** *Approximate synthesis theorem* For any valuation $v_X$ on the set $X$ of leaf agents $ag(1), . ., ag(m)$ of the $(C, \Phi, \varepsilon)$–scheme with $ag_o = Root(C)$ such that

$$v(b_{ag(i)}) \text{ satisfies } < st(ag(i)), \Phi(ag(i)), \varepsilon(ag(i)) > \text{ for } i = 1, 2, ..., m$$

where $p\_sign(ag(i)) = (st(ag(i)), \Phi(ag(i)), \varepsilon(ag(i))$, the uniquely defined object $x \in U(ag_o)$ such that $v_X \longrightarrow_C v^x_{ag_o}$ satisfies

$$< st(ag_o), \Phi(ag_o), \varepsilon(ag_o) >$$

where $p\_sign(ag_o) = (st(ag_o), \Phi(ag_o), \varepsilon(ag_o), o(ag_o))$.

□

**Projections of $(C, \Phi, \varepsilon)$-schemes onto design schemes.** We say that a $(C, \Phi, \varepsilon) - scheme$ *projects onto a design scheme* $D\_s$ when the support $C$ projects onto $D\_s$.

**Negotiation (top-down) of a scheme for satisfying a requirement.** Consider now a designer goal $Dg = (\Psi^{\vee}(dag_1), \Psi^{\vee}(dag_2), ..., \Psi^{\vee}(dag_k), \Psi^{\vee}(dag_o))$ realized by a design scheme $D\_s(\mathbf{dag}_o, \mathbf{dag}_1, ..., \mathbf{dag}_m)$. The realization by synthesis agents of the designer goal $Dg$ must begin with finding a $(C, \Phi, \varepsilon)$-scheme $S$ such that $S$ projects onto $D\_s$. We describe this process.

**Stage 1.** A string $\mathbf{ag}_o = ag_1 ag_2 .. ag_k ag_o \in Ag$ is chosen such that a construction $c(\mathbf{ag}_o)$ projects onto $D\_s(\mathbf{dag}_o)$; let the uncertainty coefficients of agents be $\varepsilon'(ag_1), ..., \varepsilon'(ag_k), \varepsilon'(ag_o)$.

**Stage 2.** Agents $ag_1, ag_2, .., ag_k, ag_o$ negotiate a connective $f(ag_o) \in F(ag_o)$ and uncertainty bounds

$$\varepsilon(ag_1) \geq \varepsilon'(ag_1), .., \varepsilon(ag_k) \geq \varepsilon'(ag_k) .., \varepsilon(ag_o) \geq \varepsilon'(ag_o)$$

such that $(Unc(ag_o))$ is satisfied with $f(ag_o)$ and $\varepsilon(ag_1), .., \varepsilon(ag_k), \varepsilon(ag_o)$.

Stages 1, 2 give, when successful, a $(c(ag_o), \Phi, \varepsilon)$ - scheme which projects onto $D\_s(\mathbf{dag}_o)$.

**Stages 3 and following.** Agents $ag_1, ag_2, .., ag_k$ repeat stages 1,2 with new coefficients $\varepsilon(ag_i)$, and so on.

The successful result of negotiations means that a $(C, \Phi(ag_o), \varepsilon(ag_o))$ - scheme is constructed which projects onto $D\_s(\mathbf{dag}_o, \mathbf{dag}_1, ..., \mathbf{dag}_m)$. We state a theorem which follows from Propositions 4 and 5.

**Theorem 6.** *(the sufficiency criterium of correctness)*

Assume that a $(C, \Phi, \varepsilon)$-scheme projects onto a design scheme $D\_s$ realizing a designer goal $Dg = (\Psi^{\vee}(dag_1), \Psi^{\vee}(dag_2), .., \Psi^{\vee}(dag_k), \Psi^{\vee}(dag_o))$. Then for any valuation $v_X$ on the set $X$ of leaf agents $ag(1), .., ag(m)$ of the $(C, \Phi, \varepsilon)$–scheme with

$$ag_o = Root(C)$$

such that $v(b_{ag(i)})$ satisfies

$$< st(ag(i)), \Phi(ag(i)), \varepsilon(ag(i)) >$$

where $p\_sign(ag(i)) =$

$$(st(ag(i)), \Phi(ag(i)), \varepsilon(ag(i), o(ag_o))$$

for $i = 1, 2, ..., m$,
the uniquely defined object $x \in U(ag_o)$ such that $v_X \longrightarrow_C v^x_{ag_o}$ satisfies the condition

$$\pi(ag_o, dag_o)(x) \models_d \Psi^{\vee}(dag_o)$$

i.e. $x$ satisfies the designer requirement $\Psi$ in the satisfactory degree.

□

We extend the satisfiability relation $\models$ to the connective $\diamond$ by $v \models \diamond < st(ag), \Phi(ag), \varepsilon(ag) >$ if there exists $C$ such that

$$v_{\{Root(C)\}} \models < st(ag), \Phi(ag), \varepsilon(ag) >$$

where $v \rightarrow_C v_{\{Root(C)\}}$.

The connective $\diamond$ expresses the existence of a scheme which can satisfy $(\Phi, \varepsilon)$ over a given input $v$. In particular, Theorem 6 gives a sufficient condition for finding such $C$.

Let us emphasize the fact that the functions $f(ag)$, called mereological connectives above, are expected to be extracted from experiments with samples of objects. The above property allows for an easy to justify correctness criterium of a given $(C, \Phi, \varepsilon)$-scheme provided that all parameters in this scheme have been chosen properly. The searching process for these parameters and synthesis of an uncertainty propagation scheme satisfying the formulated conditions constitutes the main and not easy part of design and synthesis.

**Bottom-up communication.** The bottom-up communication process is started by the leaf agents of $C$; the leaf agents select a valuation $v_X$ compatible with $C$ and any leaf agent $ag$ sends to its parent agent $ag_o$ the object $v_X(b_{ag})$ and the value $E_{\mu(ag)}(v_X(b_{ag}), st(ag))$. This process is repeated with non-leaf agents: any non-leaf agent $ag$ applies the operation $o(ag)$ to the objects $x_1, x_2, ..., x_k$ sent by its children agents, synthesizes the object $x = o(ag)(x_1, x_2, ..., x_k)$ and finds the value $E_{\mu(ag)}(x, st(ag))$. This communication process ends at the root agent of $C$ with the object $x_C$. In the case when the assembling process proceeds correctly, the object $x_C$ satisfies the requirement $\Psi$ in the satisfactory degree; the correctness of the assembling process is checked by means of the negotiated connectives $f(ag)$ and the found values $E_{\mu(ag)}(x, st(ag))$.

## Learning

In synthesis stage discussed above, the crucial factor is presented by mereological connectives (*rmc*) of uncertainty rules. In practical cases, we can infer from the given data tables an approximation $f$ to the appropriate *rmc* $F$. We include an example which illustrates our approach.

**Example 1.** *Learning mereological connectives from data tables.* We present a tiny fragment of a distributed system consisting of agents $B, L, M$. We may imagine thet $B, L, M$ form a part of the assembly line for assembling lego-like toys. The agent $B$ submits bodies and $L$ submits a leg (right) which $M$ fits together to be sent further up. The information system of the agent $B$ is given in Table 1.

| | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|
| B1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| B2 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| B3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| B4 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

where a, b, c, d, e, f, g, h stand for pis, cut, bel, kni, par, co, crl, crr, respectively (these boolean attributes represent here, respectively presence of pistols, a cutlass, a belt, a knife, a parot, a coat, a crutch on left, a crutch on right).

Table 1

Information system of agent $L$ is given in Table 2.

| | a | b | c | d | e | f | g | h | i | j |
|---|---|---|---|---|---|---|---|---|---|---|
| L1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| L2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| L3 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

where a, b, c, d, e, f, g, h, i, j stand for fobj, fpa, fsof, mesl, wou, kck, fcol, fwet, frsl, flsl, respectively. The attributes discerning a wooden artificial leg from a "natural" one are like fobj (*feel objects*), fpa (*feel pain*), frsl (*fit the right side*), flsl (*fit the left side*).

Table 2

Information system of agent $M$ is given in Table 3.

| | har | mar | lar | crl | crr | par | co |
|---|---|---|---|---|---|---|---|
| B1L2 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| B2L2 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| B2L3 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| B3L2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| B3L3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| B4L2 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| B4L3 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |

Attributes here describe the complex object obtained by attaching a leg submitted by $L$ to a body submitted by $B$. New attributes: har, mar lar, mean rspectively *heavily (medium, lightly) armed.*

Table 3

In the three following tables we present values of similarity functions based on initial rough inclusion

$$\mu^o(x, y) = \frac{cardinality\{a \in A : a(x) \neq a(y)\}}{cardinality(A)}$$

where $A$ is the set of all attributes.

Values of the function $\mu_B^0$ of objects at $B$ are given in Table 4.

| | B1 | B2 | B3 | B4 |
|---|---|---|---|---|
| B1 | 1 | 0.5 | 0.25 | 0.62 |
| B2 | 0.5 | 1 | 0.5 | 0.87 |
| B3 | 0.25 | 0.5 | 1 | 0.37 |
| B4 | 0.62 | 0.87 | 0.37 | 1 |

Table 4

Values of the function $\mu_L^0$ at agent $L$ are collected in Table 5.

| | L1 | L2 | L3 |
|---|---|---|---|
| L1 | 1 | 0.8 | 0.4 |
| L2 | 0.8 | 1 | 0.4 |
| L3 | 0.4 | 0.4 | 1 |

Table 5

In Table 6, we collect values of $\mu_M^0$ at the agent $M$.

| | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|
| a | 1 | 0.57 | 0.43 | 0.28 | 0.14 | 0.71 | 0.57 |
| b | 0.57 | 1 | 0.86 | 0.43 | 0.14 | 0.57 | 0.43 |
| c | 0.43 | 0.86 | 1 | 0.28 | 0.43 | 0.43 | 0.57 |
| d | 0.28 | 0.43 | 0.28 | 1 | 0.86 | 0.57 | 0.43 |
| e | 0.14 | 0.86 | 0.43 | 0.86 | 1 | 0.43 | 0.57 |
| f | 0.71 | 0.57 | 0.43 | 0.57 | 0.43 | 1 | 0.86 |
| g | 0.57 | 0.43 | 0.57 | 0.43 | 0.57 | 0.86 | 1 |

where a, b, c, d, e, f, g stand for B1L2, B2L2, B2L3, B3L2, B3L3, B4L2, B4L3, respectively.

Table 6

Now, the agent $M$ synthesizes complex objects of the form $XY$ from objects $X$ sent by $B$ and objects $Y$ sent by $L$. The mereological connective $F$ proper for this case should satisfy the condition : if $\mu_B^0(X', X) \geq \varepsilon_1$ and $\mu_L^0(Y', Y) \geq \varepsilon_2$ then $\mu_M^0(X'Y', XY) \geq F(\varepsilon_1, \varepsilon_2)$. However, we cannot know $F$ exactly as we do not know all possible objects at agents (at least, in principle, we cannot state so). What we may learn from the data is then a local approximation to $F$; it turns out in practice that $F$ can be highly inhomogeneous and may merely be a relation which however can be locally approximated by functions. We say therefore that a function $f$ is *an approximation of F at the object* $X^0Y^0$ when the condition holds: if $\mu_B^0(X', X^0) \geq \varepsilon_1$ and $\mu_L^0(Y', Y^0) \geq \varepsilon_2$ then $\mu_M^0(X'Y', X^0Y^0) \geq f(\varepsilon_1, \varepsilon_2)$. It is not necessary to learn the whole function $f$: we call a set $T$ of vectors of the form $[\varepsilon_1, \varepsilon_2, \varepsilon]$ the *threshold set of vectors for f at* $X^0Y^0$ whenever $T$ is a minimal set of vectors such that: $f(\varepsilon_1, \varepsilon_2) \geq \varepsilon$ for any $[\varepsilon_1, \varepsilon_2, \varepsilon] \in T$ implies that $f$ is an approximation to $F$ at $X^0Y^0$.

The following table shows a threshold vector set for an approximation at B1L2.

Table 7

| | $\varepsilon_1$ | $\varepsilon_2$ | $\varepsilon$ |
|---|---|---|---|
| 1 | 0.25 | 1 | 0.28 |
| 2 | 0.25 | 0.4 | 0.14 |
| 3 | 0.62 | 1 | 0.71 |
| 4 | 0.62 | 0.4 | 0.57 |

Imagine now that $M$ receives a requirement (in informal, quasi-natural language) for instance: $\Psi$: *"an artifact is needed which will serve for a toy model of an heavily armed invalid pirate"*. Taking $B1L2$ as a standard, $M$ transforms $\Psi$ into the approximate formula: $(B1L2, (a = 1) \wedge (b = 1) \wedge (c = 1) \wedge (d = 1) \wedge (e = 1) \wedge (f = 1) \wedge (g = 1) \wedge (h = 0), 0.7)$. From Table 7 it follows that $B1L2$ satisfies $\Psi$. Thus $L$ submits $L2$ and $B$ submits either $B1$ or $B4$.

The learning process is concerned primarily with learning threshold sets of vectors at complex objects. The complexity of this process will be studied elsewhere.

Learning processes on higher level are concerned with learning the dynamics of synthesis schemes. This leads to knowledge concerning the necessity of changing the topology of the scheme i.e. links among local teams or changing the goals of local teams.

## Applications: Learning rules of a mereological controller

The approximate specification $(\Phi, \varepsilon)$ can be regarded as the invariant to be kept over the universe of global states (complex objects) of the distributed system.

**The basic problems.** Now we can formulate some basic problems related to control in terms of properties of a construction $C$. The control problems can be divided into several classes depending on the model of controlled object. In this work we deal with the simplest case. In this case, the model of a controlled object is the $(C, \Phi, \varepsilon)$ -scheme **c** which can be treated as a model of the unperturbed by noise controlled object whose states are satisfying the approximate specification $(\Phi, \varepsilon)$.

The $(C, \Phi, \varepsilon)$ -scheme defines **c** a function $F_{\mathbf{c}}$ called *the output function* of the $(C, \Phi, \varepsilon)$ -scheme **c** given by $F_{\mathbf{c}}(v) = x$ iff $v \longrightarrow_C v^x_{ag_{\mathbf{C}}}$ where $ag_{\mathbf{C}} = Root(C)$. Let $ag_1, ..., ag_r$ be leaf agents of $C$. Any object $x$ from the set $F_{\mathbf{c}}(U(ag_1) \times ... \times U(ag_r)) \cap \{x \in U(ag) : x \models (st(ag_{\mathbf{C}}), \Phi, \varepsilon)\}$ is called *the* $(\Phi, \varepsilon)$*-invariant object* of **c**.

We assume the leaf agents of the $(C, \Phi, \varepsilon)$ -scheme **c** are partitioned into two disjoint sets, namely the set $Un\_control(\mathbf{c})$ of *uncontrollable (noise) agents* and the set $Control(\mathbf{c})$ of controllable *agents.*

We present now two examples of a control problem for a given $(C, \Phi, \varepsilon)$ -scheme.

**(OCP) OPTIMAL CONTROL PROBLEM:**

**Input:** $(C, \Phi, \varepsilon)$ -scheme **c**; information about actual valuation $v$ of leaf agents i.e. the values $v(b_{ag})$ for any $ag \in Control(\mathbf{c})$ and a value $\varepsilon'$ such that $F_{\mathbf{c}}(v) \models (st(ag_{\mathbf{C}}), \Phi, \varepsilon')$.

**Output:** A new valuation $v'$ such that $v'(b_{ag}) = v'(b_{ag})$ for $ag \in Un\_control(\mathbf{c})$ and $F_{\mathbf{c}}(v') \models (st(ag_{\mathbf{C}}), \Phi, \varepsilon_o)$

where

$\varepsilon_o = \sup\{\delta : F_{\mathbf{c}}(w) \models (st(ag_{\mathbf{C}}), \Phi, \delta)$

for some $w$ such that

$w(b_{ag}) = v(b_{ag})$ for $ag \in Un\_control(\mathbf{c})\}$.

These requirements on the output can be to hard to satisfy directly. One can search for changes of a given uncertainty scheme allowing to construct an object not closest to a given specification $\Phi$ but for an object satisfying a given specification in a degree higher than the sum of a given threshold and the degree defined by the current object. In this way we obtain

**(CP) $\nabla$-CONTROL PROBLEM**

**Input:** $(C, \Phi, \varepsilon)$ -scheme **c**; information about actual valuation $v$ of leaf agents (i.e. the values $v(b_{ag})$

for any $ag \in Control(\mathbf{c})$) and a value $\varepsilon'$ such that $F_{\mathbf{C}}(v) \models (st(ag_{\mathbf{C}}), \Phi, \varepsilon')$.

**Output:** A new valuation $v'$ such that $v'(b_{ag}) = v(b_{ag})$ for $ag \in Un\_control(\mathbf{c})$ and $F_{\mathbf{C}}(v') \models (st(ag_{\mathbf{C}}), \Phi, \varepsilon_o)$ where $\varepsilon_o > \varepsilon' + \nabla$ for some given threshold $\nabla$.

**The controller.** We will describe now the basic idea on which our controllers of complex dynamic objects represented by distributed systems of intelligent agents are built. The main component of controllers are $\Delta$-*rules* describing how local changes (i.e. changes at agents of a given distributed system) $\Delta\varepsilon(ag)$ of uncertainty coefficients $\varepsilon(ag)$ can be compensated by local changes $\Delta\varepsilon(ag_1), ..., \Delta\varepsilon(ag_k)$ of uncertainty coefficients at children $ag_1, ..., ag_k$ of $ag$ for agents in a given scheme. By composing $\Delta$-rules one can calculate the necessary changes of uncertainty coefficients for all agents $ag \in Control(\mathbf{c})$ and in this way to predict possible changes of controllable parameters (i.e. elementary objects being values of $b_{ag}$ for $ag \in Control(\mathbf{c})$).

The $\Delta$-*rules* have the following structure:

$$(\Delta\varepsilon(ag_{i_1}), ..., \Delta\varepsilon(ag_{i_r})) = h(\varepsilon(ag), -\Delta\varepsilon(ag), \varepsilon(ag_1), ..., \varepsilon(ag_k))$$

where $ag_{i_1}, ..., ag_{i_r}$ are all controllable children of $ag$ (i.e. children of $ag$ having descendents in $Control(\mathbf{c})$), $h : R^{k+2} \to R^r$ and $R$ is the set of reals.

The description of the function $h$ is extracted from experimental data. In the process of extracting $h$ from data rough set and boolean reasoning methods (Polkowski & Skowron 1996b) can be applied.

The semantics of the $\Delta$-rule for $\mathbf{ag} = ag_1 ag_2 \ldots ag_k ag_o \in Link$ in $\mathbf{c}$ is defined by uncertainty coefficients $\varepsilon(ag), \varepsilon(ag_1), ..., \varepsilon(ag_k)$ attached to agents in $\mathbf{c}$ in the following way: if an object $x'$ issued by the agent $ag$ is satisfying $x' \models (st(ag), \Phi(ag), \varepsilon'(ag))$ where $\varepsilon'(ag) = \varepsilon(ag) + \Delta\varepsilon(ag)$ then if the controllable children $ag_{i_1}, ..., ag_{i_r}$ of $ag$ will issue objects $y_{i_1}, ..., y_{i_r}$ satisfying $y_{i_j} \models (st(ag_{i_j}), \Phi(ag_{i_j}), \varepsilon(ag_{i_j}) + \Delta\varepsilon(ag_{i_j}))$ for $j = 1, ..., r$ where $(\Delta\varepsilon(ag_{i_1}), ..., \Delta\varepsilon(ag_{i_r})) = h(\varepsilon(ag), -\Delta\varepsilon(ag), \varepsilon(ag_1), ..., \varepsilon(ag_k))$ then the agent $ag$ will construct an object $y$ such that $y \models (st(ag), \Phi(ag), \varepsilon)$ where $\varepsilon \geq \varepsilon(ag)$.

In the above formula we assume $\Delta\varepsilon(ag) \leq 0$ and $\Delta\varepsilon(ag_{i_1}) \geq 0, ..., \Delta\varepsilon(ag_{i_r}) \geq 0$. The above semantics covers the case when $\Delta$-rules allow to compensate in one step the influence of a noise. Other cases will be treated in our next paper.

If $ag_1 ag_2 ... ag_k ag_o$, $ag'_1 ag'_2 ... ag'_r ag'_o \in Link$, $\{ag_{i_1}, ..., ag_{i_r}\}$, $\{ag_{j_1}, ..., ag_{j_s}\}$ are disjoint sets of controllable children of $ag_o$, $ag'_o$, respectively, then two $\Delta$-rules

$$(\Delta\varepsilon(ag_{i_1}), ..., \Delta\varepsilon(ag_{i_r})) = h_1(\varepsilon(ag_o), \Delta\varepsilon(ag_o), \varepsilon(ag_1), ..., \varepsilon(ag_k))$$
$$(\Delta\varepsilon(ag_{j_1}), ..., \Delta\varepsilon(ag_{j_s})) = h_2(\varepsilon(ag'_o), \Delta\varepsilon(ag'_o), \varepsilon(ag_{j_1}), ..., \varepsilon(ag_{j_s}))$$

can be composed at $ag'_o$ if $ag'_o = ag_{i_t}$ for some $t$. The result of the composition is defined by

$$(\Delta\varepsilon(ag_{i_1}), ..., \Delta\varepsilon(ag_{i_{t-1}}), \Delta\varepsilon(ag_{j_1}), ..., \Delta\varepsilon(ag_{j_s}), \Delta\varepsilon(ag_{i_{t+1}}), ..., \Delta\varepsilon(ag_{i_r})).$$

Hence the composition of $\Delta$-rules leads to the distribution of changes of uncertainty coefficients among agents.

It is easy to observe that the composition of the $\Delta$-rules leads to a new labelling $\{\varepsilon_{new}(ag)\}$ where for all non-root controllable agents of $\mathbf{c}$ we have $\varepsilon_{new}(ag) = \varepsilon_{new}(ag) + \Delta\varepsilon_{new}(ag)$ and $\Delta\varepsilon(ag)$ is obtained as the result of negotiations among agents about possible changes defined by compositions of $\Delta$-rules. For the root agent of $\mathbf{c}$ and all non-controllable agents in $\mathbf{c}$ we assume $\varepsilon_{new}(ag) = \varepsilon_{new}(ag)$.

We obtain the following proposition establishing a basic property related to the correctness of mereological controllers.

**Proposition 7.** (*the sufficiency criterium of correctness of the controller*)

Let $F_{\mathbf{C}}(v) \models (st(ag_{\mathbf{C}}), \Phi, \varepsilon(ag_{\mathbf{C}}))$ where $v$ is the valuation of leaf agents of the $(C, \Phi, \varepsilon)$ -scheme $\mathbf{c}$ and let $F_{\mathbf{C}}(v') \models (st(ag_{\mathbf{C}}), \Phi, \varepsilon'(ag_{\mathbf{C}}))$ where $v'$ is a valuation of leaf agents of $\mathbf{c}$ such that $v'(b_{ag}) = v(b_{ag})$ for $ag \in Control(\mathbf{c})$, $\varepsilon'(ag_{\mathbf{C}}) < \varepsilon(ag_{\mathbf{C}})$. If $\{\varepsilon_{new}(ag)\}$ is a new labelling of agents defined by composition of some $\Delta$-rules such that $\varepsilon_{new}(ag) = \varepsilon(ag)$ for $ag \in Un\_control(\mathbf{c})$, $\varepsilon_{new}(ag_{\mathbf{C}}) = \varepsilon(ag_{\mathbf{C}}) = \varepsilon$ and $\{x_{ag} : ag \in Control(\mathbf{c})\}$ is the set of control parameters (inventory objects) satisfying $x_{ag} \models (st(ag), \Phi(ag), \varepsilon_{new}(ag))$ for $ag \in Control(\mathbf{c})$ then for the object $x_{new} = F_{\mathbf{C}}(v_1)$ constructed over the valuation $v_1$ of leaf agents in $\mathbf{c}$ such that $v_1(b_{ag}) = v'(b_{ag})$ for $ag \in Un\_control(\mathbf{c})$ and $v_1(b_{ag}) = x_{ag}$ for $ag \in Control(\mathbf{c})$ holds $x_{new} \models (st(ag_{\mathbf{C}}), \Phi, \varepsilon(ag_{\mathbf{C}}))$.

□

It is not difficult to extract from Tables 1,2,3 of Example 1 the $\Delta$ - rules for controlling the system of complex objects at $M$.

## Multistrategy learning in our approach

It has been emphasised (Jenkins 1993), (Michalski 1994) that a multistrategy learning is achieved whenever knowledge sources (agents) are turned into independent learning units (agents). In our approach this effect is achieved by learning appropriate constructs from data tables of independent agents. In the learning process the agents use empirical induction (in order to extract rules from data tables), abduction (to fit objects satisfying the conclusions of the rules when premises are known and selected) and reasoning by analogy (when carrying the synthesis along routes outlined by standards at consecutive agents).

The above approach can be treated as a first step towards modelling complex distributed dynamical systems. We expect that it can be extended to solve control problem for complex dynamical systems i.e. dynamical systems which are distributed, highly nonlinear, with vague concepts involved in their description. It is hardly to expect that the classical methods of control theory can be successfully applied for such complex systems.

## Analysis

The process of analysis is split into the design and synthesis spaces. Given an object $x$, its analysis can be realized in the following sequence of steps.

**Step 1.** An agent $ag \in Ag$ such that $x \in U(ag)$ is chosen

**Step 2.** An agent dag such that $\pi(ag, dag)(x)$ is defined chooses its decomposition.

**Step 3.** The decomposition continues, resulting in the *analysis tree* for $\pi(ag, dag)(x)$.

**Step 4.** A design scheme for $\pi(ag, dag)(x)$ is designed.

**Step 5.** A $(C, \Phi, \varepsilon)$-scheme for synthesis of $x$ is found.

This constitutes the analysis of constructibility of $x$.

Further analysis includes e.g. the analysis of stability as well as robustness of $(C, \Phi, \varepsilon)$.

## Acknowledgements

The authors appreciate financial help of the Polish - Japanese Institute of Computer Techniques.

## References

Amarel, S. 1991. PANEL on AI and Design. In Proceedings of IJCAI-91; Twelfth Joint International Conference on Artificial Intelligence, 563-565. San Mateo: Morgan Kaufmann.

Dubois, D., Prade, H., and Yager R.R. 1993. *Readings in Fuzzy Sets for Intelligent Systems.* San Mateo: Morgan Kaufmann.

Jenkins, W.T. 1993. INTELOG: A Framework for Multistrategy Learning. In Proceedings MSL-93; The Second International Workshop on Multistrategy Learning, 58-68. Fairfax, VA: Center for Artificial Intelligence, George Masson University.

Komorowski, J.; Polkowski, L.; and Skowron, A. 1996. Towards a rough mereological logic for approximate solution synthesis. *Studia Logica.* Forthcoming.

Leśniewski, S. 1992. Foundations of the general theory of sets. In *Stanislaw Leśniewski, Collected Works,* Surma,S. J., Srzednicki, J. T., Barnett, D. I., and Rickey, V.F., eds., 128-173. Dordrecht: Kluwer.

Michalski, R.S. 1994. Inferential Theory of Learning. In *Machine Learning A Multistrategy Approach.* Vol.IV, 3-61. San Francisco: Morgan Kaufmann.

Pawlak, Z. 1991. *Rough sets: Theoretical Aspects of Reasoning about Data.* Dordrecht: Kluwer.

Pawlak, Z., and Skowron, A. 1994. Rough membership functions. In *Advances in The Dempster - Shafer Theory of Evidence,* Yager, R.R., Fedrizzi, M., and Kacprzyk, J., eds., 251-271. New York: Wiley.

Polkowski L., and Skowron A. 1994a. Rough mereology. In Proceedings ISMIS -94; International Symposium on Methodologies for Intelligent Systems, 85-94. Lecture Notes in Artificial Intelligence 869. Berlin: Springer-Verlag.

Polkowski, L., and Skowron, A. 1994b. Introducing rough mereological controllers: Rough quality control. In Proceedings RSSC-94; The Third International Workshop on Rough Sets and Soft Computing, 78-95. San Jose State University, CA.

Polkowski, L.; Skowron, A. 1996a. Rough mereology: A new paradigm for approximate reasoning. *International Journal of Approximate Reasoning.* Forthcoming.

Polkowski, L., Skowron, A. 1995. Rough mereology and analytical morphology: New developments in rough set theory. In Proceedings of WOCFAI-95; Second World Conference on Fundamentals of Artificial Intelligence, 343-354. Paris: Angkor.

Polkowski, L., Skowron, A. 1996b. Rough mereological approach to knowledge-based distributed AI. In Proceedings of WCES-3; Third World Congress on Expert Systems. New York: Cognizant Communication Corporation.

Skowron, A., and Polkowski, L. 1995. Rough mereological foundations for design, analysis, synthesis and control in distributed systems. In Proceedings of the Second Annual Joint Conference on Information Sciences, Wrightsville Beach, NC.

Skowron, A. 1995. A synthesis of adaptive decision systems from experimental data. In Proceedings of Fifth Scandinavian Conference on AI, SCAI-95, 220-238. Amsterdam: IOS Press.