UNIVERSITÀ DEGLI STUDI
DI MILANO

SCUOLA DI DOTTORATO IN INFORMATICA

Tesi di Scuola di Dottorato

HIERARCHICAL-GRANULARITY HOLONIC MODELLING

Marco Calabrese

Relatore: Prof. Vincenzo Piuri
Correlatore: Prof. Vincenzo Di Lecce

Coordinatore del Dottorato di Ricerca: Prof. Ernesto Damiani

Anno Accademico 2009/2010

ACKNOWLEDGMENTS

I would like to express all my gratitude to those who have shared this three-year experience with me.

First of all, I'm indebted to the "Three Graces" of my life: Manuela, my sister Claudia, and my mother Nella.

Then, my thanks go to my advisors, my colleagues and friends, and all who have permitted me to grow spiritually and professionally during these years.

A special thought goes to my father, who instilled in me the passion for Computer Science.

Last but not least, I would like to recall the importance (to me) of Faith as a perpetual aid both in good and bad times. I'm sure that without it, things would have been more difficult and meaningless.

By the way, as an auspice for this research voyage, I intend to commence with a omen that should cheer up anyone in search of something (and which I strongly believe in), that: "… *you will know the truth, and the truth will set you free*" (John, 8:32).

Enjoy the reading.

CONTENTS

ABSTRACT

This thesis aims to introduce an agent-based system engineering approach, named Hierarchical-Granularity Holonic Modelling, to support intelligent information processing at multiple granularity levels. The focus is especially on complex hierarchical systems.

Nowadays, due to ever growing complexity of information systems and processes, there is an increasing need of a simple self-modular computational model able to manage data and perform information granulation at different resolutions (i.e., both spatial and temporal). The current literature lacks to provide such a methodology. To cite a relevant example, the object-oriented paradigm is suitable for describing a system at a given representation level; notwithstanding, further design effort is needed if a more synthetical of more analytical view of the same system is required.

In the literature, the agent paradigm represents a viable solution in complex systems modelling; in particular, Multi-Agent Systems have been applied with success in a countless variety of distributed intelligence settings. Current agent-oriented implementations however suffer from an apparent dichotomy between agents as intelligent entities and agents' structures as superimposed hierarchies of roles within a given organization. The agents' architectures are often rigid and require intense re-engineering when the underpinning ontology is updated to cast new design criteria.

The latest stage in the evolution of modelling frameworks is represented by Holonic Systems, based on the notion of 'holon' and 'holarchy' (i.e., hierarchy of holons). A holon, just like an agent, is an intelligent entity able to interact with the environment and to take decisions to solve a specific problem. Contrarily to agent, holon has the noteworthy property of playing the role of a whole and a part at the same time. This reflects at the organizational level: holarchy functions first as autonomous wholes in supra-ordination to their parts, secondly as dependent parts in sub-ordination to controls on higher levels, and thirdly in coordination with their local environment.

These ideas were originally devised by Arthur Koestler in 1967. Since then, Holonic Systems have gained more and more credit in various fields such as Biology, Ecology, Theory of Emergence and Intelligent Manufacturing. Notwithstanding, with respect to these disciplines, fewer works on Holonic Systems can be found in the general framework of Artificial and Computational Intelligence. Moreover, the distance between theoretic models and actual implementation is still wide open.

In this thesis, starting from the Koestler's original idea, we devise a novel agent-inspired model that merges intelligence with the holonic structure at multiple hierarchical-granularity levels. This is made possible thanks to a rule-based knowledge recursive representation, which allows the holonic agent to carry out both operating and learning tasks in a hierarchy of granularity levels.

The proposed model can be directly used in terms of hardware/software applications. This endows systems and software engineers with a modular and scalable approach when dealing with complex hierarchical systems. In order to support our claims, exemplar experiments of our proposal are shown and prospective implications are commented.

1. INTRODUCTION

Modularity, scalability, self-organization, have always been desirable traits in complex systems development: a simple evidence for this claim comes from a lookout at the evolution of software engineering methodologies in the latest decades.

From the 70's on, structured programs (Bhom & Jacopini, 1966) progressively replaced 'spaghetti code' that made unwise use of the criticised GOTO statement (Dijikstra, 1968) in favour of more abstract programming methodologies. Later, the object-oriented paradigm enlarged the programmer's ability to figure out complex software by introducing the abstract notion of class. In the middle 90s, agent-based technologies (Wooldridge & Jennings, 1995) injected Artificial Intelligence (AI) inside software entities providing them with highly specific behaviours such as autonomy, coordination, goal-oriented behaviour and so on.

Along this line, each new achievement embraced the previous one in a climax of powerful theoretical and operational methodologies to gain control and design ability on systems of increasing complexity.

With reference to the latest developments in the engineering of complex systems, two methodologies that account for different solutions to hierarchical organizations should mentioned, namely: Multi-Agent Systems and Holonic Systems. These approaches represent two pillars of our proposal, i.e., providing a novel holonic-based methodology to support intelligent information processing at various granularity levels.

1.1 Multi-Agent Systems and Hierarchical Organizations

Nowadays, the leading paradigm, especially in the engineering of large distributed systems, is represented by the concept of Multi-Agent Systems (MAS). MAS are goal-driven organizations of intelligent agents that interact with one other on behalf of users (Wooldridge, 2009). The impressive growth of computer networks, such as the Internet, has made it possible to think to countless applications for MAS considered as machines talking to other machines for human benefit. To make an example, as envisioned by Tim Berners Lee with the prospect of the Semantic Web (Berners Lee et al., 2001), in the (near?) future an Internet user will let a group of agents searching the Web on his/her behalf for relevant documents, avoiding tedious hours spent in unfruitful search. Such a task is extremely coherent with the idea of MAS.

One key aspect of MAS is that, in order for agents to interact successfully, they require the ability to cooperate, coordinate, and negotiate with each other, much as people do. As it happens in the real world, this engagement requires some social organization model. Consequently, MAS provide a novel new framework for simulating societies, which may help shed some light on various kinds of social processes (Wooldridge, 2009).

The two extremes of MAS organizations range from 'horizontal' architectures (using, for example, a blackboard architectural model for agent communication (Corkill, 1991)) to 'vertical' architectures that arrange agent

management and control according to a mechanism of delegation in a hierarchical fashion (Sycara, 1998). Generally, hierarchical MAS are preferable as long as the system grows in complexity: a horizontal organization model in fact soon becomes unwieldy when the number of agent interactions increases.

Any vertical architecture employs some kind of hierarchical organization. According to Mesarović et al. (Mesarović et al, 1970), a hierarchical organization can be defined as a collection of decision makers each of whom acts as a controller of the decision level below him and, at the same time, as a dispatcher of information to the level above him; the bottom level of controlles operates in the given process.

For example, in the Theory of Information Systems, the classification of business activities within organizations (strategic, tactical and operational) is described in hierarchical terms by means of the so-called *Anthony's pyramid* (Anthony, 1965).

A hierarchical MAS-based adaptation of the Anthony's pyramid is depicted in Figure 1.1. In this kind of hierarchical representation, the members of the organization are clustered into layers. The boundaries between layers account for well-defined, distinct and strict roles within the organization. Depending on the role they play, members have to fulfil specific duties with a certain degree of freedom and carry out some activities in compliance with the upper level strategy. In this sense, they can be considered agents in the proper sense.

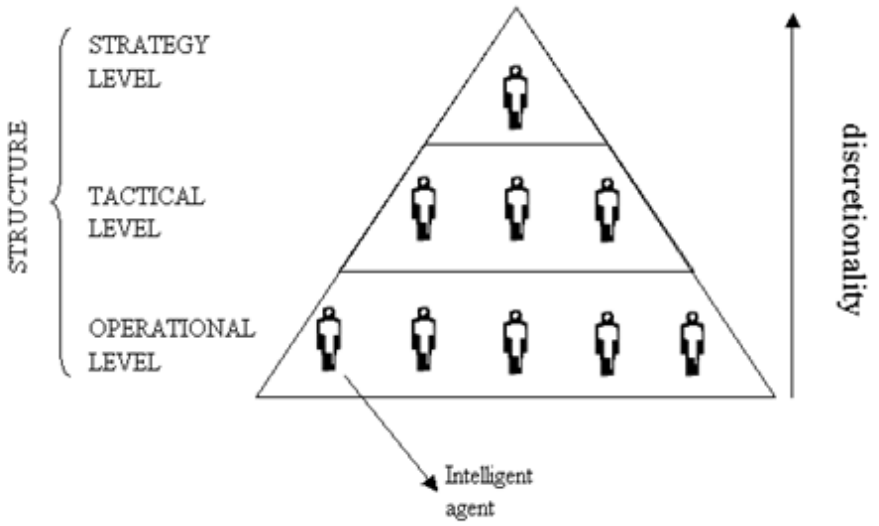

Figure 1.1: A representation of the Anthony's Pyramid in terms of a hierarchical MAS.

In the Anthony's pyramid, discretionality (hence autonomy) progressively increases towards the top, thus agents at higher levels are supposed to be more autonomous than their subordinates. This aspect can be viewed as caused by a special arrangement of the system knowledge. Knowledge at each layer has different qualities. At each level of the pyramid, knowledge is more long-termed, more wide covering and more general than that of subordinate layers. Furthermore, it is generally not observable by subordinate layers. With a

qualitative assessment, we can judge the knowledge at the strategy layer to be of higher quality with respect to other layers, at least from the system goal point of view. It is noteworthy that the quality of knowledge follows the same pattern as autonomy. Hence, agents at each level are biased somehow in their social behaviour by the quality of knowledge/autonomy they handle.

Vertical MAS architectures are very desirable from the engineering perspective since they guarantee a rationale partition of functional tasks within agents' organization. Nevertheless, they inevitably suffer from the stiffness of the agent roles superimposed during the design phase.

1.2 Holistic and Reductionist Approaches in Complex Systems Design

Making an agent change its role is not so straightforward. This would require endowing the agent with the ability to understand the context in which it operates coupled with the ability to perceive its specificity within the wholeness of the surrounding environment. Such requirements are generally pursued by a *holistic* approach, while MAS are typically engineered according to a *reductionist* vision.

Holism (from ὅλος - holos, a Greek word meaning all, entire, total) is the idea that all the properties of a given system (biological, chemical, social, economic, mental, linguistic, etc.) cannot be determined or explained by its component parts alone. Instead, the system as a whole determines in an important way how the parts behave. Aristotle in the *Metaphysics* concisely summarized the general principle of holism: "*The whole is more than the sum of its parts*".

Reductionism is sometimes seen as the opposite of holism. Reductionism in science says that a complex system can be explained by reduction to its fundamental parts. Reductionism essentially claims that psychology and sociology are reducible to biology, biology is reducible to chemistry, and finally chemistry is reducible to physics. Some other proponents of reductionism, however, think that holism is the opposite only of greedy reductionism.

1.3 Holonic Systems and Hierarchically-Nested Structures

From an engineering perspective, the holistic vision is well suited to the paradigm Holonic Systems. With respect to MAS, Holonic Systems show the major difference that the former are based on the notion of 'holon' rather than that of agent.

The term holon first appeared in 1967, in a work authored by Arthur Koestler (Koestler, 1967) meaning an entity capable of playing the role of a whole and a part at the same time.

In order to have a better insight into Koestler's thought, it is worthwhile to refer to his own words, as reported in the Alpbach Symposium in 1968 (Koestler, 1969), where he presented his conceptual framework as an attempt to overcome the dichotomy between reductionism and holism.

Koestler enlisted several aspects regarding the general properties of what he called Self-regulating Open Hierarchic Order (SOHO) and successively took the name of holarchies. In particular, the claims in the following are devoted to introducing the concept of holon and its Janus effect.

Claim 1.1: *The organism in its structural aspect is not an aggregation of elementary parts, and in its functional aspects not a chain of elementary units of behaviour.*

Claim 1.2: *The organism is to be regarded as a multi-levelled hierarchy of semi-autonomous sub-wholes, branching into sub-wholes of a lower order, and so on. Sub-wholes on any level of the hierarchy are referred to as holons.*

Claim 1.3: *Parts and wholes in an absolute sense do not exist in the domains of life. The concept of the holon is intended to reconcile the atomistic and holistic approaches.*

Claim 1.4: *Biological holons are self-regulating open systems which display both the autonomous properties of wholes and the dependent properties of parts. This dichotomy is present on every level of every type of hierarchic organization, and is referred to as the "Janus phenomenon".*

Claim 1.5: *More generally, the term "holon" may be applied to any stable biological or social sub-whole which displays rule-governed behaviour and/or structural Gestalt-constancy. Thus organelles and homologous organs are evolutionary holons; morphogenetic fields are ontogenetic holons; the ethologist's "fixed action-patterns" and the sub-routines of acquired skills are behavioural holons; phonemes, morphemes, words, phrases are linguistic holons; individuals, families, tribes, nations are social holons.*

Koestler's own words move from the concept of organism as a systemic whole (Claim 1.1) to introduce that of multi-level hierarchy (Claim 1.2) conceived as a self-regulating structure of sub-wholes. The latter, as parts of a greater whole, have to be considered as functional to the system they are hosted in; however, at the same time, they also show autonomous characteristics, which make them being a system as well. Alternatively speaking, holons account for a recursive interpretation of the concept of system where part and wholes are not considered as separate entities. This is easily observable in the domain of life (Claim 1.3). This dichotomy reflects on every level of the hierarchy (Claim 1.4) and can be extended to any biological or social sub-whole based on rules (Claim 1.5).

In summary, Koestler's holon is a basic model component suitable for building self-regulating hierarchical organizations. The chief distinction between holon and other model-based entities is the appearance of the so-called *Janus phenomenon*. Janus was the ancient roman god who reigned over the realm of doors, passages, beginnings and endings.

The holon actually shows a Janus face since it contemplates within a unique entity two distinct but complementary perspectives: top-down and bottom-up.

- **Top-down**: one side looks "down" and acts as an autonomous system following its own goals and rules, also giving directions to lower-level components (sub-holons);

- **Bottom-up**: the other side looks "up" and serves as a part obeying to a higher-level component (super-holons).

This double-face nature reflects in the part/whole relationship that is observed in living and social organisms and can be extended to any complex hierarchical system as well. According to this view, it turns clear how the concept of holon can be relevant in the general framework of Systems Theory.

Holon part/whole duality could be represented mathematically by means of the recursive notion of subsets and in fact, as shown further in the text, recursion is essential to characterize part-whole relationships within Holonic Systems. However, since Computational Intelligence (CI) is the scope of the thesis, we do not engage with mathematics and set theory, instead we deal with architectures for supporting knowledge extraction, representation and management, hence, ultimately, intelligent information processing.

A holon, as intelligent entity, accomplishes the following two tasks:

- acts as an autonomous system following its own goals and rules, also giving directions to lower-level components (sub-holons);
- serves as a part obeying to a higher-level component (super-holons)

Consequently, a hierarchical organization of holons, also known as 'holarchy', should leverage MAS architectures at the maximum extent by overcoming the rigid distinction between intelligent entities and the hierarchical structure that enrols them.

Thanks to the concept of holarchies, Holonic Systems instantiate a different organizational paradigm where the roles of parts and whole coincide. Holarchy can be described as multi-strata hierarchy (C. Ferber, 1999), i.e., a hierarchical ordered system where every level is a domain specific abstract version of the overall complex system under scope.

It is important to point out that, while in common lexicon words such as 'stratum', 'layer', 'level', are all synonyms meaning "an abstract place usually conceived as having depth", in the context of this paper, they account for very different senses.

Multi-level (or multi-strata) hierarchy, in contrast with multi-layered hierarchy, is characterized by the complete (physical or conceptual) nesting of each level into the higher adjacent one. This representation should be borne in mind throughout the thesis when referring to holarchies. Except from the most abstract level (which is only a whole and not a part of something bigger), at any given level, groups of holons, as parts, completely defines their super-holons, as wholes. In this case, holons are the components for modelling parts of the system at different granularity levels. If holarchy is explored from whole to parts, i.e., towards more detailed granularity levels, then a process-oriented decomposition is followed (refer to (Clegg and Shaw, 2008) for an example); otherwise, if parts are used to aggregate into wholes, i.e., towards more abstract granularity level, then an emergent behaviour (see, for example, (Ulieru and Este, 2004)) is observed.

A pictorial representation of holarchy as multi-strata hierarchy is displayed in Figure 1.2: holons at level *n* are grouped into organisations that can be considered at level *n*+1 as an individual entity. Inversely, individual entities at level *n*+1 can be seen at level *n* as organisations. The process can be repeated on any number of levels until a *ground-level* representation is reached out.

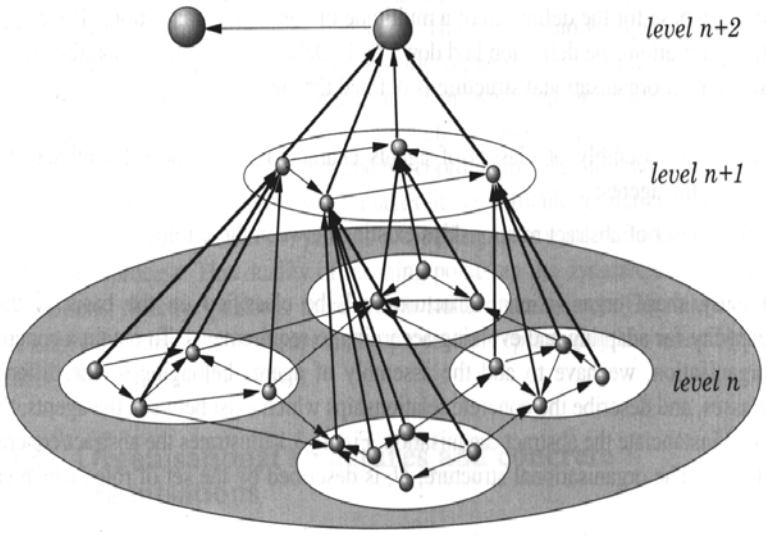

Figure 1.2: A representation of holarchy as a multi-strata hierarchy, from (C. Ferber, 1999).

1.4 Thesis Objective

Holons, in the theoretical frame provided by Koestler, are an abstract inspiration for hierarchical systems with intelligent behaviours; furthermore, they allow for modelling complex phenomena in a non-reductionist way (Pichler, 2000). In this sense, they paved the way for studies in AI, since autonomy and self-organization are two distinctive properties of intelligent agents and MAS respectively (Russell and Norvig, 2003). Furthermore, since holon is intrinsically a modular object, it can be used to empower class-based design and development with interesting features from the Software Engineering viewpoint.

Unfortunately, the utility of reconsidering holons and holarchies from the AI perspective has not been perceived so far at the right extent. Possibly, this is due to the genesis and further development of holonic theories which are native of non-AI (but, we daresay, close-to-AI) fields such as manufacturing or business organizations.

In this thesis, our effort is devoted to examine holon and holarchy models from the standpoint of AI. The aim is to introduce a holonic computational representation for leveraging the analysis, design and implementation of complex multi-level organizations. Metaphorically, we pursue the goal of *merging intelligence with agents' hierarchical organizations*, which is quite a novel engagement in the literature.

An artwork drafting our intention is depicted in Figure 1.3. Holarchies are holons at a wider granularity level: this way the part/whole distinction between the container and the contained entity vanishes turning into a granularity level problem based on a unique conceptual entity.

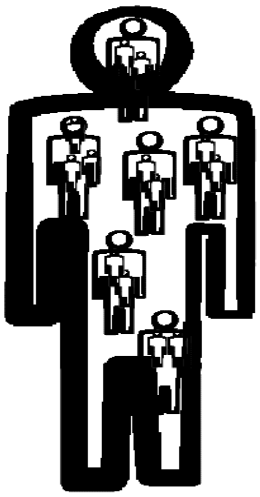

Figure 1.3: Intelligence inside the holarchy. An artwork.

1.5 Thesis Structure

In order to achieve the objective of this thesis, the rest of the text is organized into these parts:

- Chapter 2 reviews some basic findings of the AI literature concerning agent modelling. Afterwards, holons and holarchies are introduced and put in comparison with agents and MAS respectively to highlight differences and similarities between the two approaches. This confrontation is carried out mainly at the architectural level. Then, the focus is moved towards information granulation viewed as a suitable conceptual and operational link to connect the realm of agents with the realm of holons. The survey is carried out under the light of prospective CI theories such as Granular Computing and Computing With Words;
- Chapter 3 is aimed at introducing the proposed hierarchical-granularity holonic model (HGHM) as a novel agent-inspired computational machinery that fulfils the goal of the thesis. The presentation of HGHM is quite elaborated since it involves the reader in the task of assembling oddly shaped, interlocking and tessellating ideas, as in a jigsaw puzzle. First, the duality between parts and wholes is faced at the knowledge representation level. In particular, a granularity-level-independent description based on the object-oriented notation is used for re-defining the two concepts of holon and holarchy in a computational way. This multi-level representation supplies for a

recursive decomposition of both the system knowledge and computation at the same time by introducing the novel concept of 'holonic granule'. The focus is mainly on "crisp" holonic granules: their extension to fuzzy logic, although of a great interest, would yield another specific work, which goes further beyond the scope of the thesis. As next step, holonic granules are considered as basic entities of a 'holonic grammar' as the automaton suitable for describing systems at different granularity levels. An archetype algorithm for parsing holonic grammar-generated descriptions is hence designed. To endow HGHM with unsupervised knowledge acquisition ability, a heuristics is introduced to allow automated extraction of holonic grammars from observational data. This completes the picture: the jigsaw puzzle is finally assembled and the overall model is presented also in a formal notation;

- Chapter 4 shows some multi-topics applications springing from the proposed model: string parsing, signal self-description, time-series prediction and intelligent system modelling;
- Chapter 5 concludes the thesis by highlighting prospective implications and future works of HGHM in AI and Software Engineering.

2. RELATED WORK

This chapter surveys in a row the basic concepts underpinning our proposal, namely agents, holons and granular computing, as they have appeared in the literature. We start with a brief introduction to the concept of intelligence with particular reference to the ideas of Minsky, one of the forerunners of the agent theory. Next we go in depth through an overview of agent theory and agent societies (MAS) focussing on their architectural limits with reference to complex systems modelling; then we move towards holonic modelling and the paradigm of holistic thinking which seems to be facing a great renewal in these last years. The current limits of Holonic Systems methodologies conclude the section on holonic-based approaches. Finally, we explore Granular Computing and Computing With Words as computational approaches that appear to be useful to reduce the gap between the agent and holonic realms.

2.1 Intelligence and Computational Models

No doubt, the concept of 'intelligence' is a controversial one. Engaging with it is a cumbersome task that will be deliberately skipped here. A complete analysis would imply considering other areas such as human sciences, epistemology, philosophy, all well beyond the scope of the thesis.

What is certain indeed, is that AI nowadays strives for a shared definition. As a matter of course, it is further more pragmatic understand which (seeming) intelligent things can be done with artificial machines and to what extent the so-called 'Machine Intelligent Quotient' (MIQ) can be increased.

A troublesome suggestion comes from the famous article *Steps toward Artificial Intelligence* dated back 1961 (Minsky, 1961) where Marvin Minsky asserted ironically but in a rightful way: "*A computer can do, in a sense, only what it is told to do*". At first glance, the claim does not leave much space for AI-enthusiasts. However, it diverts the attention from the problem of intelligence focussing on another aspect: what machines can do.

The true power of a machine is in the range and type of input manipulations it can perform, which ultimately depends on the computational model empowered. A computational model is any (numerical or symbolic) model that allows for studying the behaviour of complex systems. It is fair to imagine that the more complex the system, the more sophisticated the computational model employed.

By restricting the view to only symbol manipulation based systems, i.e., a framework where input and outputs are strings of symbols of a given alphabet and input/output matching is performed by some algorithmic procedure (Minsky, 1967), computational models corresponds to automata. An automaton, interpreted as a string recognizer, is an abstract machine able to recognize languages written in some formal grammar (Chomsky, 1956), (Chomsky, 195) (see Figure 2.1, taken from (Ausiello et al., 2008)).

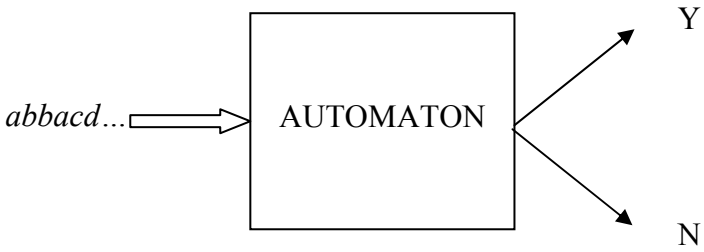

Figure 2.1: Automaton as a string recognizer.

Automata theory is a well-established discipline with a solid theoretical background. To have a clear mind of this, it suffices recalling the names of some of its forefathers such as Turing, Markov, Church, Kleene and Post. They all contributed to the cause of newborn computer science from the perspective of Mathematical Logic.

An overview of their achievements, especially for Turing and Markov, can be found in Mendelson (Mendelson, 1964). Almost at the same time when Chomsky was working on his seminal theory of generative grammars for the description of natural languages, Markov proposed a string rewriting system based on the formal assessment of the notion of algorithm. He was, indeed, mainly interested in investigating how to determine the effective computability of a given function, dealing with pure mathematics from a symbolic logic perspective. It is worth saying that the theory of Markov algorithms is proved to produce a Turing-equivalent computational model. Kleene, Church and Post had devised similar computational models some years before.

Notwithstanding, the primary perplexity remains on how pure computation, which is conceptually very close to theorem proving, string parsing and, in general, formal theories, i.e., apparently semantic-less actions, could drive out intelligent, i.e., semantic-full, consequences (an interesting essay on this point from the perspective of cognitive science can be found in (Hutchins, 1995)).

A solution to this dilemma was proposed by Minsky, several years after his informative article on AI, by conjecturing that intelligence, as a complex process, is the manifest appearance of a number of simpler phenomena taking place at a lower observation level.

In the book Society of Mind (Minsky, 1986), Minsky gathered his intuitions by saying that the complex jigsaw puzzle of intelligence can be ultimately recomposed only by unveiling its atomic intertwining mindless parts. The name he gave to these particles was that of *agents*. Using the same Minsky's words:

> Each mental agent by itself can only do some simple things that need no mind or thought at all. Yet when we join these agents in societies – in certain very special ways – this lead to true intelligence.

Ahead in the text, he quoted:

> […] whenever we find that an agent has to do anything complicated, we'll replace it with a sub-society of agents that do simpler things. […] When we break things down to their smaller parts, they'll each seem dry as dust at first, as though some essence has been lost.

In Minsky's view, agents represent a computational model for describing the complex processes that happen within our own mind in terms of simpler entities, arranged as if they constituted a society. Furthermore, his description contains elements of 'granular thinking', a viewpoint enlightened by the more recent studies of Zadeh (Zadeh, 1998), the founder of Fuzzy Logic (FL) and one of the main contributors to the CI field.

After Misky's intuitions, the relevant work of Wooldridge and Jennings (Wooldridge & Jennings, 1995) helped the computational paradigm of agents attract the interest of scholars over the successive years. Although (as for intelligence) a shared definition for agent does not exist, today, agents represent a widely used design and development framework that supplies a trade off between formal theories and engineering applications. In particular, due to ever-growing networked resources, such as the Internet, and the availability of computational nodes at relatively low-cost, the attention of scholars and practitioners has progressively moved from single agent design to agents' society, i.e., MAS.

Central to MAS modelling is communication as a direct consequence of the design process. Typically, the followed approach when building MAS consists in a top-down assignment of the global problem to functional sub-units each of which endowed with a specific role and a specific task to accomplish. MAS are then arranged in a hierarchy of agents: agents at a higher level delegate more operational functions to agents at the lower levels through the institute of communication (see (Fornara et al, 2008)). The drawback of this approach is the stiffness of the overall architecture, which requires heavy reengineering when some unexpected change is needed at the knowledge representation and consequently at the communication level (Calabrese et al., 2010).

On the opposite side, when a bottom-up methodology is pursued, the design approach is completely different, requiring minimal (or no) communication (Crespi et al., 2008): the design target changes from the global system to the sole agent's behaviour. In this case, single autonomous entities, if properly programmed, should be able to make a collective behaviour emerge out of interaction within the environment. The difficulty lays in providing a sufficiently powerful autonomous behaviour capable also of taking into account requirements of a global and social vision.

Efforts in this direction come from holistic theories and in particular from Holonic System theories. Unfortunately, due to their genesis in the field of intelligent manufacturing, Holonic Systems are currently quite far from the AI and CI standpoint. At the same time, they account for interesting properties such as the conceptual predisposition to handle, natively, multiple granularity levels, which may represent an interesting point of contact with CI sub-domains like Granular Computing and Computing With Words.

2.2 Agents

Nature is the most significant source of inspiration for CI models.

Among nature-inspired computational paradigms, the agent-oriented one plays a primary and ambitious role since it does not only aim at imitating some specific processes that we commonly ascribe to intelligence (cognition, learning, communication, etc…) but tries to encode an entire intelligent entity within a single computational entity.

2.2.1 Agents and CI

Agent-based methodologies have accompanied CI evolution, especially in the last decade. An early agent-based approach to CI is that of Poole et al. (Poole et al., 1998) who claimed that CI "*is the study of the design of intelligent agents*". This assumption may appear quite restrictive for many scholars in CI since other computational models have at least the same dignity as agent-based solutions. Nevertheless, for the scope of this thesis, CI will be mainly conceived from the agent-modelling viewpoint although not certainly exhaustive of the entire CI field.

A confirmation of the amplitude of the CI literature and its relationships with the concept of agent can be found in the work of van Eck et al. (van Eck et al., 2006). They adopted concept maps trained over the abstracts of the papers presented at the IEEE World Congress on Computational Intelligence (WCCI) in 2002 and 2006 to visualize the most used concepts employed by authors. The two maps are shown in Figure 2.2.

Like after a "big bang", it seems that CI is moving from a chaotic dense mass of concepts toward some clutters of similar research domains. Certainly enough, we are facing only the very beginning of this CI-universe-forming process and it is hard to predict what we may expect these maps would be, say, the next decade.

As far as now, three main clusters are being formed in the CI literature. They coarsely correspond to the three principal constituents of soft computing technology, namely: genetic algorithms (GA), fuzzy logic (FL) and neural networks (NN). Zadeh had envisaged such a tripartition already several years before (Zadeh, 1994) with the only relevant difference that, instead of GA, he had identified the cluster of probabilistic reasoning as the abstract framework to deal with uncertainty.

The main reason for the tripartition is that each master theory works on a specific aspect of CI: FL is primarily concerned with imprecision, NN with learning and GA with search. Generally, any intelligent system needs all the three aspects covered during the engineering process. This often implies some sort of mesh-up of different techniques, thus requiring a composite system design (Alippi et al., 1999). Agents indeed are sufficiently undifferentiated with respect to the previous tripartition since they encompass natively, as shown further in the text, all these aspects.

Turning back to the figures, in both 2002 and 2006 concept maps, we note that the agent label is present but in different positions. While in 2002 it was laying in the overlapping area between evolutionary and fuzzy systems, in

2006 it had moved to the outskirt of the graph, perhaps denoting a more independent assessment with respect to traditional CI techniques.

The ultimate goal of the thesis is to add a new concept in proximity of that of agent, the concept of holon.

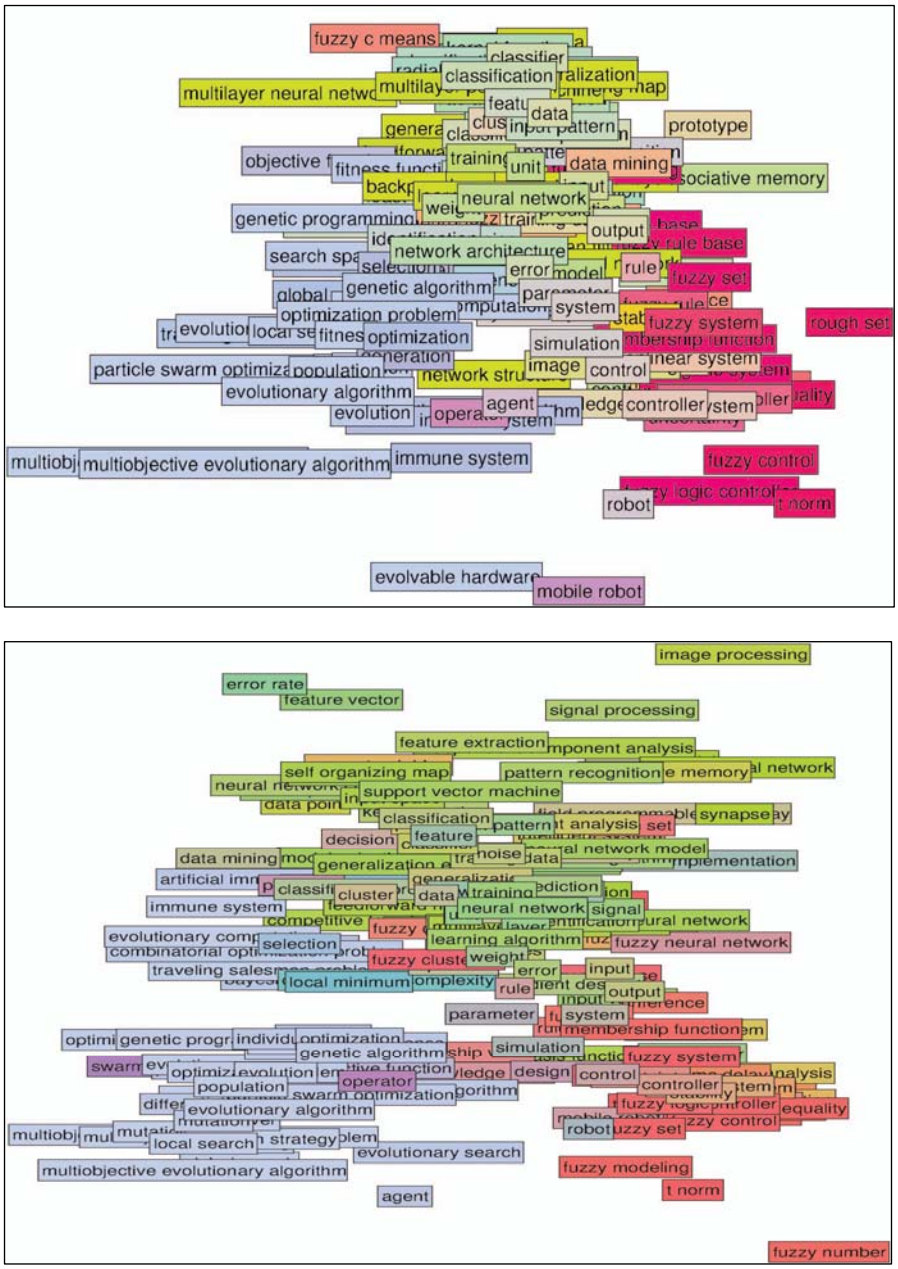

Figure 2.2: concept maps representing the CI field as of 2002 (uppermost figure) and as of 2006 (lowermost figure). Images taken from (van Eck et al., 2006).

2.2.2 Agent basic definitions and foreground aspects

In order to manage the concept of agent on a more objective basis, it is useful to refer to some widely accepted definitions.

Agent "black-box" definitions

***Def.* 2.1**: *An agent is something that acts in an environment*.

***Def.* 2.2**: *An intelligent agent performs at least the following tasks:*

1. *it builds models of the environment;*
2. *given observations, it determines, through reasoning, what the world is and will be like;*
3. *it learns from past experiences;*
4. *given a model of the world and a goal, it decides what should be done.*

Def. 2.1 and *Def. 2.2* allow for drafting a first coarse picture of the agent concept and its relationship with the environment (see Figure 2.3, slightly adopted from (Poole et al., 1998)). Agent is viewed as a dynamic box-model having observations, prior knowledge, goals in input and having actions in output. Observations represent agent's perceptions of the surrounding environment. Prior knowledge and goals are built-in elements set by the agent designer. Learning allows for coding new behaviours hence actions by training on past experiences.

Def. 2.1 introduces the important concept of environment; *Def. 2.2* enlists the minimum set of tasks an agent is supposed to accomplish to gain the quality of being defined as intelligent. Namely they are:

- Knowledge Representation;
- Reasoning;
- Learning;
- Decision-making.

Since these notions account for crucial aspects of agent design, they are treated briefly in the next subsections.

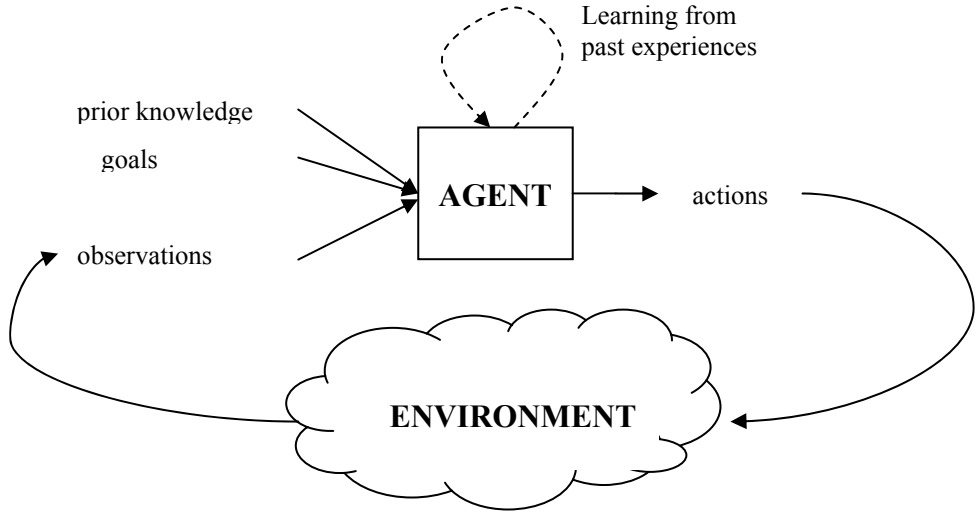

Figure 2.3: Agent abstract model.

2.2.2.1 The role of environment

Agent definition introduces the notion of 'environment' as something that *must* be taken into consideration when defining an agent. As reported in (Odell et al, 2003):

> An environment provides the conditions under which an entity (agent or object) can exist. It defines the properties of the world in which an agent will function.

The environment is then an unavoidable part of the agent definition because it defines the boundary conditions of the agent design process. Furthermore, its importance is evident from the fact that, in a closed-loop architecture as the one depicted in Figure 2.3, it represents the block that governs the feed-back process; hence, as we can infer from Control Theory, it is fundamental in the system (agent) dynamic behaviour.

It is fair to imagine that environment is generally a complex system, i.e., composed by different elements, which change in time and interact in a nonlinear way. A vast literature exists on the topic, especially in the field of complex systems science (CSS) and cellular automata. Despite the fact that that CSS and AI have many points in common, the interaction between the two communities is however still very limited (Bandini & Serra, 2006).

The attempt to model the environment as a complex system from an agent-based perspective leads however to an awkward conclusion, being it only a way to divert from the original problem of environment definition. In fact (unless we claim an agent-based environment to be like a monad) for each agent employed to characterize the environment, we should define, in accordance with agent definition, the environment in which it operates. The result is that we have a circular definition with no apparent escape.

As shown in more detail further in the text, holistic thinking handles the notion of environment in a more naturally way since it makes no such dichotomic distinction between the whole (environment) and a part of the whole (agent) as it happens instead for previous agent definitions.

2.2.2.2 Aspects in Knowledge Representation

As it often happens in AI, also Knowledge Representation (KR) skips a precise definition. In (Davis et al. 1993) the authors pointed out that, at that time, the question about what KR actually is had not been answered directly yet. Since then, a great effort has been engaged by scholars in different research fields to provide at least an operational definition for KR. One of the major outcomes of such an engagement is the notion of 'ontology'.

Ontologies: a brief overview

Nowadays, when computer scientists refer to KR, they commonly employ the name of ontology, but with a meaning different from the one that can be usually found in a dictionary. The online Oxford dictionary for example provides for ontology the following definition: "*the branch of metaphysics concerned with the nature of being*".

In Computer Science, ontology is defined indeed as "*a specification of a representational vocabulary for a shared domain of discourse -- definitions of classes, relations, functions, and other objects*" (Gruber, 1993). Throughout the text, we will refer only to the latter sense.

The importance of ontologies is at least twofold, being related to the possibility of drawing inference in automated way but also to the mechanism of knowledge sharing among different applications, which allows for interoperability and knowledge reuse, two desirable properties for a system engineer.

Ontologies are knowledge structures based on two parts, the definition of concepts and the relationships among them (Gruber, 1995) (Uschold et al., 1998). Put this way, also an Entity-Relationship Model (Chen, 1976) could be defined as a type of ontology. However, in common practice, ontologies are not specifically engineered to produce a database; instead, they play the role of knowledge bases by supplying some kind of machinery for the knowledge representation and reasoning (KRR) task.

A simple way to think about an ontology implementation is to employ a *hierarchy of concepts*. Hierarchies can have a number of purposes: from classification to control or system description. When dealing with classification in KR tasks, hierarchies often assume the form of taxonomies.

A relevant formalization of ontologies according to taxonomical structures can be found in (Dellshaft & Staab, 2006). As suggested in (Velardi et al. 2007), taxonomy can be regarded as a form of business intelligence, to integrate information, reduce semantic heterogeneity, facilitate the communication between information systems.

Taxonomies are generally implemented by means of graph-based structures, in particular directed acyclic graphs (DAGs) or trees (Di Lecce & Calabrese 2008). Similar considerations have been drawn by Ning and Shihan (Ning & Shihan, 2006) who suggest that the structure of an ontology should satisfy the structure of its referring domain knowledge, i.e., the quality of the ontology strictly depends on the way its knowledge is structured. In particular, the authors consider ontology as an undirected graph G = <V,E>. Each concept is a vertex in the graph. If a concept has an object property whose value is an instance of another concept, an edge is drawn between these two concepts.

Almost the same assumptions can be found in (Chakrabarti et al., 1999). By the way, representing ontology knowledge in form of a graph is a widely accepted paradigm.

An example in this sense is given by the standard Resource Description Framework (RDF) data model which consists in a collection of statements (each made of the triplet Subject-Verb-Object) representing a labelled directed graph.

An RDF statement (W3C, 2004) is a triplet having the structure pattern <subject, predicate, object> as in the serialized XML notation that follows.

```
<?xml version="1.0" ?>
    <rdf:RDF  xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
            xmlns:au="http://mydictionary.org/schema/">
            <rdf:Description about="http://www.myself.it/Phd_Thesis/">
                <au:author>myself</au:author>
            </rdf:Description>
    </rdf:RDF>
```

The subject of RDF statements must actually be a resource defined by a unique identifier as a Uniform Resource Locator, so the above statement could be turned into an RDF statement illustrated in Figure 2.4.

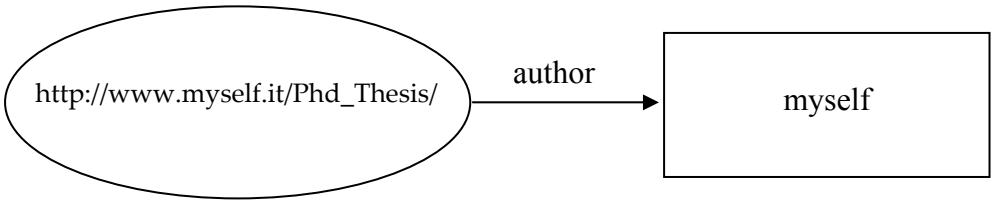

Figure 2.4: RDF statement example expressed in a graph notation.

Despite its flexibility, RDF representation does not include important features that a real ontology should have such as the management of modal or fuzzy assertions, uncertainty, inconsistence and so on. Generally, it can be said that an ontology structure may be reduced to a graph; however, its real model is actually more complex than it (Di Lecce & Calabrese, 2008).

An extension of RDF is the W3C standard Web Ontology language (OWL), (W3C, 2009) which comes with increasingly-expressive sublanguages (OWL Lite, OWL DL, OWL Full) for supporting different inference capabilities and adds more vocabulary for describing properties and classes: among others, relations between classes (e.g., disjointness), cardinality (e.g., "exactly one"), equality, richer typing of properties, characteristics of properties (e.g., symmetry), and enumerated classes.

One of the main problems related to building ontologies is that this is a very time consuming activity. In the last years, ontology engineers have devised several semi-automatic ontology-building approaches, especially for the Semantic Web (Maedche & Staab 2001). Nevertheless, the human factor is still determinant. An awkward aspect of this is that multiple ontologies describing the same or narrow domains may be hardly mapped each other. The same concept may be in fact lexicalised in different ways; furthermore, some relations comprised in a given ontology may not be present in another.

Ontology for representing language: semantic lexicon

As shown before, ontologies are employed for leveraging reusable KR and automated inference: hence, a critical point for ontology engineers is how to express specifications of concepts in a symbolic machine-readable way. Concepts and relations characterizing the ontological layer in fact need to be somehow lexicalised in order to be used and shared by a computational model. This is the reason why any KRR implementation requires both the ontological layer, considered as the semantic representation of the world of interest, and the lexical layer, considered as the symbolic transposition of the semantic representation.

A very particular subset of ontology studies is the one considering language itself as the target domain of conceptualization. In this special case, concepts represent the meanings of lexical entries of a dictionary (words). One of the most widely used names for this kind of ontology is *semantic lexicon*.

In a semantic lexicon, a one-to-one relationship can be drawn between the concept (semantic layer) and its word form (lexical layer) giving rise to the notion of word sense. Then, the collection of all word senses can be represented by a matrix whose rows and columns are respectively the lexical set and the semantic set. The sense matrix element can be expressed as a binary relation between a word form and a concept. A sense between lexical entry l_i and concept entity c_j occurs if the (i, j) element of the matrix comes with unary value.

This matrix is generally referred to as lexical matrix in the literature (Magnini et al., 1994) (Komarovaa & Nowak, 2001). Such a naming convention however does not take care explicitly of the semantic aspect. For this reason, we prefer to use for this structure the name of sense matrix. Table 2.1 depicts an example of a sense matrix.

Notice that, depending on the entry points (by row or by column) to the matrix, two possible patterns can be identified: lexical entries with more than one concept associated accounts for *polysemy*, concepts with more than one lexical entry associated accounts for *synonymy*.

Table 2.1. Sense matrix example.

Sense Matrix		**Concepts**			
		c_1	c_2	c_3	c_4
Lexicon	l_1	0	1	0	**1**
	l_2	0	0	0	**1**
	l_3	1	1	0	**0**
	$l4$	1	0	1	**1**
	l_5	**0**	**0**	**1**	**0**

In the domain of real-world applications, significant outcomes have been obtained with WordNet (Fellbaum, 1998) semantic lexicon. WordNet, an open project of the Princeton University, is referred to in the literature in several ways: lexical knowledge base (Basili et al., 2002) (Inkpen, 2001), lexical taxonomy (Jiang & Conrath, 1997) (Gangemi et al., 2001), lexical database (Miller, 1995) (Ordan & Wintner, 2005), machine readable dictionary (Kegl,

1995) (Hayashi & Ishida, 2006), ontology (Snasel et al., 2005) (Qian et al., 2005), semantic lexicon (Di Lecce et al., 2009) (Quian et al., 2009). The latter definition is here preferred for the reason commented above.

Originally based on the concept of synset (i.e., groups of synonyms) as atomic information granule, the recent release of WordNet 3.0 data model instead has been developed around the concept of sense, which is a more fine-grained one. The underpinning KR model is represented by lexico-semantic chains over the domain of senses organized according to given structural patterns. For example hypernymy taxonomy arranges all WordNet senses according to a hierarchy that spans from the root concept of 'entity' (most general concept node with the meaning of '*that which is perceived or known or inferred to have its own distinct existence*') to very specific leaf concepts such as 'thesis' (with the sense gloss of '*usually a requirement for an advanced academic degree*').

WordNet's structure makes it a useful tool for computational linguistics and natural language processing (see Figure 2.5 for an excerpt); for example it has been employed with promising results in the Semantic Web and automatic sense disambiguation (Navigli & Velardi, 2005)(Di Lecce et al., 2009).

- S: (n) **entity** (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))
  - *direct hyponym* / *full hyponym*
    - S: (n) physical entity (an entity that has physical existence)
      - *direct hyponym* / *full hyponym*
        - S: (n) thing (a separate and self-contained entity)
        - S: (n) object, physical object (a tangible and visible entity; an entity that can cast a shadow) *"it was full of rackets, balls and other objects"*
        - S: (n) causal agent, cause, causal agency (any entity that produces an effect or is responsible for events or results)
        - S: (n) matter (that which has mass and occupies space) *"physicists study both the nature of matter and the forces which govern it"*
        - S: (n) process, physical process (a sustained phenomenon or one marked by gradual changes through a series of states) *"events now in process"; "the process of calcification begins later for boys than for girls"*
        - S: (n) substance (material of a particular kind or constitution) *"the immune response recognizes invading substances"*
      - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) abstraction, abstract entity (a general concept

Figure 2.5: Excerpt of the WordNet taxonomy at the root level.

2.2.2.3 Aspects in Automated Reasoning

In the attempt to mimic human abilities, one key aspect of intelligent agents is automated reasoning. It broadly consists in determining whether a conjecture φ can be proved relying on Γ assumptions. In other words, the central problem of automated reasoning is theorem proving.

Basically, theorem proving can be made according to two possible directions (Bonacina & Martelli, 2006):

- from general to particular with deductive theorem proving, which is concerned precisely with the entailment problem (in symbols: $\Gamma \models \varphi$);
- from particular to general with inductive theorem proving, where the problem is to determine whether Γ entails all ground instances σ of φ (in symbols: $\Gamma \models \varphi\sigma$, for all ground substitutions σ).

In both cases, the problem is to find an effective computation that leads to the desired proof. In (fully) automated theorem proving, the machine is expected to find a proof alone basing on its own build-in algorithms.

Limits of classical approaches

In classical logic settings, many proof techniques have been studied and implemented. Generally, the main limitation of these techniques is the so-called *logical omniscience problem*: it implies the agent being a perfect reasoner. Given a Knowledge Base Γ, the agent should be capable of inferring all possible consequences of its axioms.

In case of partial knowledge, the hypothesis of logic omniscience forces the agent to think of all possible scenarios (called *possible worlds*), thus leading to unfeasible or extreme resource-consuming situations even to model a card play. Possible world semantics is most commonly formulated by means of the Kripke's formalism of modal logic.

A thorough dissertation about all these issues can be found in (Wooldridge & Jennings, 1995).

Modern approaches

Traditional formalisms, although supported by strong theoretical bases, often fails to provide a valuable solution to real world problems. In many occasions in fact, an intelligent agent is required to infer and take decisions even when there is not enough information to prove that an action will work. This case is better known to as *reasoning under uncertainty*.

A wide comprehensive historical coverage on this topic has been performed by the work of Dubois and Prade (Dubois & Prade, 2009). The authors consider that an agent believes a piece of information to be uncertain when it is not able to state if the piece of information is true of false. To this end, they exploit the mathematical definition of disjunctive set as the formal foreground for handling more-than-Boolean semantics.

The mathematical notion of uncertainty perhaps represents one of the best achievements in modern theory of formal logic reasoning. It encompasses in a unique framework all previous probabilistic theories (from Boole to De Finetti) along with more recent fuzzy set theory (Zadeh, 1965) (which is, however, basically conceived for dealing with imprecision rather than uncertainty) and possibilistic logic (Shafer, 1976) theories.

2.2.2.4 Aspects in Machine Learning

A precise definition for Machine Learning (ML) does not exist, however the Langley's assumption (Langley, 2000), can be considered as a commonly accepted one: he considers learning as any "*mechanisms through which intelligent agents improve their behaviour over time*".

In more operational terms, ML can be described as the process of discovering recurrent structures from a set of available data (examples) and extrapolating these regularities to new data (Esposito et al., 2006). This process can be undertaken in several ways, but in general all the available ML techniques fall within the two categories of supervised and unsupervised approaches (Jain et al., 1999).

Fostered by the need of unveiling hidden relationships among variables in large repositories, a relevant research direction in ML is that of Data Mining and Knowledge Discovery in databases. At the same time, it is useful mentioning the growing awareness among researchers about identifying ML as a classification problem. These two topics, data mining and classification, are treated in the following.

Data mining and Knowledge Discovery in data

Much interest seems nowadays to be arousing on the themes of Data Mining (DM) and Knowledge Discovery in Data (KDD). A thorough description of the two fields goes beyond the scope of the thesis. Here it follows a brief summary.

DM is the discipline concerned with the search for *"useful nuggets of information among huge amounts of data*" (Jain, 1999). Spurred from the early work of Agrawal on databases (Agrawal et al., 1993) at the beginning of 90s, DM has become now a popular technique in several application domains.

DM has traditionally concentrated on the analysis of a static world, in which data instances are collected, stored and analyzed to derive models and take decisions according to them. In recent times, the focus has moved to on-the-fly data with the attempt of detecting dynamic behaviours and extracting spatio-temporal patterns (Mennis & Liu, 2005).

KDD is a branch of DM aimed at turning data into (structured) knowledge. This is achieved by combining cross-domain expertise such as AI, databases, statistics and cognitive psychology (Pazzani, 2000).The outcome of a KDD process is generally a (formal) model for structuring and representing knowledge.

To cite a relevant example, Object Management Group (OMG) developed a specification called Knowledge Discovery Metamodel (KDM) (OMG, 2007)

which defines an ontology for the software assets and their relationships for performing knowledge discovery of existing code.

Inductive Learning

According to the studies of Vapnik (Vapnikin, 1979), (inductive) learning can be viewed as a classification problem. Here, we mainly refer to the formalism and arguments used by Alippi and Braione (Alippi & Braione, 2006) to provide a summary of the learning problem along with some basic definitions that will be helpful ahead in the text.

Consider a stationary random pair, $z = \langle \mathbf{x}, y \rangle$, $\mathbf{x} \in X \subset \Re^{n_x}$, $y \in \{0,1\}$ and a set $F = \{ f(\mathbf{x}, \boldsymbol{\alpha}) | \boldsymbol{\alpha} \in \Lambda \subset \Re^{n_\Lambda} \}$ of real valued functions called the hypothesis space. $f(\mathbf{x}, \boldsymbol{\alpha}) : X \times \Lambda \to \{0,1\}$ represents an hypothesis or classifier made on $\mathbf{x}$ (the input vector of the acquired data) with controllable parameters $\boldsymbol{\alpha}$. As weights and biases in a neural network, $\boldsymbol{\alpha}$ array accounts for the parameters needed to tune the machinery used to verify the hypotheses.

$L(y, f(\mathbf{x}, \boldsymbol{\alpha}))$ denotes the loss function expressing the cost of observing y instead of $f(\mathbf{x}, \boldsymbol{\alpha})$. A risk is then associated to L this way: $R(\alpha) = E[L(y, f(\mathbf{x}, \boldsymbol{\alpha}))]$. This way, learning is turned into the process of finding an $\alpha_0 \in \Lambda$ such that the risk function is minimized given a set of N observational measurements $S = \{\langle x_1, y_1 \rangle, \ldots, \langle x_N, y_N \rangle\}$.

Such a learning process is generally referred to as *training*. The ultimate goal of training is hence to find a classifier with minimum risk. It is a common practice to assign a unit cost to the event of incorrect classification $y \neq f(x, a)$. In this case, risk reduces to the error probability $Err(\alpha) = P\{y \neq f(x, a)\}$.

It has been proved (Devroye et al., 1996) that no classifier can be more precise that the Bayes one (a.k.a maximum *a-posteriori* classifier), whose knowledge is associated with the knowledge of the conditional probability distribution of y with respect to x. Since, in the general case, we are unable to state whether the chosen hypothesis space contains the Bayes classifier or not then the following holds $Err(\alpha_0) \geq Err_B$.

The two errors are also known as the language-intrinsic and inherent error, respectively. The first ($Err(\alpha_0)$) is concerned with the way the problem is described in terms of hypotheses by means of the computational model with parameters $\boldsymbol{\alpha}$ adopted: the closer the description to the actual process, the less its value. The latter (Err_B) depends on the learning problem itself and can only be improved by improving the problem itself.

An inductive learning principle defines how data are used to select a classifier from a given hypothesis space. Inductive principles define relationships between α and S such either in functional terms $\alpha = \alpha(S)$ or in probability terms $P^{\alpha|S}$. The two cases correspond to deterministic and stochastic inductive learning principles respectively. Finally, a learning algorithm is a procedural implementation of an inductive principle.

2.2.2.5 Aspects in decision making

When agent is required to take decisions, it should accomplish the task at the best of its available possibilities. Especially in uncertain environment (partially observed processes, limited information, time constraints etc…) such a commitment is hard to put into practice since decisions have to be based on minimizing or maximizing some stochastic variable or mediating among different alternatives.

In the literature, a great deal of proposals has aroused to handle such noisy and imprecise environments. As an example, in the case of mobile robots acting in outdoor scenarios, behavioural FL-based rules can be adaptively learnt by the robotic agent by means of specifically engineered GAs (Hagras et al., 2001).

As a matter of course, agent decisions and consequently agent behaviour can be related to the way agent is engineered. The next paragraph is devoted to present this aspect in more detail.

2.2.3 Agent basic design principles

Previous definitions Def 2.1 and Def. 2.2 characterize intelligent agents at the black box level. In this paragraph, we move toward the design level. For this reason, it is useful referring to another definition, featured by Russel and Norvig in their leading book *Artificial Intelligence: a Modern Approach* (Russel & Norvig, 2003):

Agent "gray-box" definition

Def. 2.3: *An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors*

Def 2.3. Adds to the glossary of the agent definition two other important terms: namely 'sensors' and 'effectors' which allow for defining respectively the input and the output agent interface with the external world (Figure 2.6).

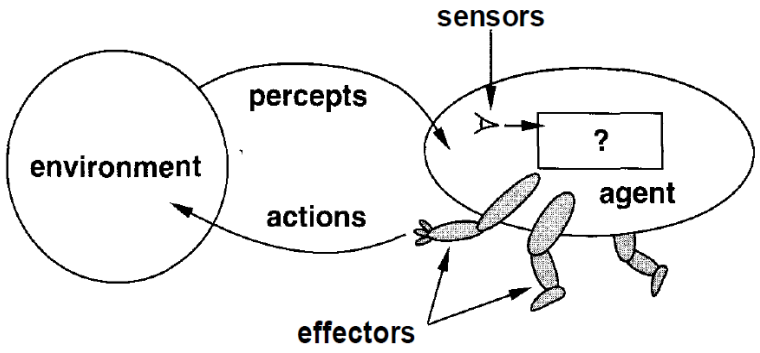

Figure 2.6: Agent pictorial representation emphasizing the role of sensors and effectors as interfaces with the environment, from (Russel&Norvig, 2003).

At the design level, agent intelligence has to be something related to sensors-effectors mapping. Following a mathematical formalism, agent model can be represented as a function defined in the domain P^N of percept sequences of length N with values in the co-domain A of actions:

$$f : P^N \rightarrow A$$

However, this representation is too inflexible and largely impractical in real-world situations. In fact, function f is supposed to be non-linear in complex settings. For example, if we had to model an agent playing chess, we should consider an extraordinary number of percept sequences thus requiring an immense look-up table to implement all possible percept-action pairs defined by the f function. To overcome this limitation, other solutions have to be pursued.

2.2.3.1 Agent grey-box models

Agent models can be arranged in a climax of complexity starting from very trivial perception/action mapping to more sophisticated ones. In general, as far as agent models gain in complexity, the corresponding agent behaviour seems to be more human-like and intelligent. In the following, a list of different models is briefly presented and commented:

- A SIMPLE REFLEX AGENT is the simplest conceivable agent endowed with a static set of condition-action rules. Percept sequence, once decoded, is matched against a table of action rules. When the match is found, the corresponding action is triggered. At the software level, the simple reflex agent barely corresponds to a "switch case" construct; hence, it acts as a selector of the input. At the hardware level, a simple reflex agent corresponds to a switchboard or similar machinery.
- A REFLEX AGENT WITH STATE yields the concept of internal state. A perception sequence, given an internal state, may determine a change in it. In this case, the action is triggered by the agent internal state. This allows designers to setup finite-state-automata (FSA) agents. FSA alone account for a myriad of applications especially in the field of automation engineering from vending machines to robotic arms in production lines. FSA can be employed to drive the logic of a software program as well. As for simple reflex agent however, the limit of this approach is in the fact that agent has no autonomous learning ability and shows only regular behavioural patterns.
- A GOAL-BASED AGENT is purposely engineered to accomplish a given task. This accounts for a germinal autonomous behaviour, although driven by a well-established and pre-defined goal. Furthermore, in presence of a goal, some target-following strategy has to be applied. This can be done in several ways. In principle, we could imagine a cost function to minimize, such that when its value is below a certain threshold the agent considers that the goal has been reached. Applications of this kind are currently widespread: for example, GPS car navigation systems or web crawlers.

- A UTILITY-BASED AGENT has its own utility functions to decide what is better for itself in presence of multiple contrasting choices. Utility provides a way in which the likelihood of success can be weighed up against the importance of the goals. This allows for a great flexibility in terms of adaptive behaviours. Implementations for this kind of agents are progressively appearing in the market in the latest times. Consider, for example, autonomous robotic vacuum cleaners: these robots have to deal with both a goal-oriented behaviour (clean one or more rooms) and a utility-based behaviour (come back to the re-charge station when battery is low).

2.2.3.2 Multiple possible views in agent design: the AI panorama

Agents represent quite a well-established topic within the vast field of CI. Since the latter is only a subset of the AI domain, what about agents in AI?

AI is a cross-fertile domain where different schools of thoughts confront each other sometimes with harsh and implacable positions.

Weak and strong AI

A first coarse-grained distinction is between weak-AI and strong-AI supporters (Wooldridge & Jennings, 1995). According to the first group, agents are hardware-centred (*robots*) or software-centred computer systems (*softbots*) showing a minimal set of features commonly considered as being a sign of an intelligent behaviour. Namely, these are:

- *Autonomy*: ability to have their own self-control mechanism;
- *Reactivity*: ability to react to environment changes (physical for robots or software for softbots);
- *Pro-activeness*: ability to self-activate also in absence of external stimuli, generally due to a goal-oriented or utility-oriented behaviour;
- *Social-ability*: ability to communicate with peers using a given language (such as FIPA Agent Communication Language).

Actually, the weak notion of AI can be easily conceptualized in terms of software programs implementing the four previous abilities up to a certain extent. Probably weak AI is the mainstream at present for the span of academic and industrial applications that are leveraged by this kind of research perspective.

Strong-AI supporters are indeed convinced that agents, maybe in the next future, will be endowed with typical human-like abilities such as pure abstraction, easy context-switching, unsupervised learning and even consciousness.

If this last ambitious research trend were really successful, machines would probably replace mankind in most of activities currently accomplished by humans, thus having a dramatic impact in common man everyday life. It is fair to say that, up to now, strong AI seems to be more a matter of theorists rather than of application engineers.

A crisp classification for AI approaches

Enlarging the previous analysis, Russel and Norvig (Russel & Norvig, 2003) proposed to classify all conceivable AI approaches basing on two features we refer to as: frame of reference and reference model.

- *Frame of reference* relies on two possible values corresponding to antithetical perspectives: the mentalist viewpoint and the behaviourist viewpoint. The first one can be considered as an inside-the-box approach, the other as an outside-the-box approach.
- *Reference model* can be either the human being or a generic rational entity.

Hence, four crisp categories are possible; they are reported in Table 2.2 (slightly adapted from (Russel & Norvig, 2003)).

Table 2.2 Classification of different approaches to AI.

AI approaches	Human	Rationalist
Mentalist	Thinking humanly	Thinking rationally
Behaviourist	Acting humanly	Acting rationally

- **Thinking humanly**: this approach requires, by definition, a cognitive model of the human mind. Cognitive Science unbinds a number of human-centred sciences (like Psychology), thus it necessitates an interdisciplinary panel of experts to be dealt with appropriately. Nevertheless, there can be found some authors in the AI field facing the issue of designing agent mental states. Some of them focus on Knowledge Belief Intention (a.k.a. KBI) or Belief Desire Intentions (BDI) models (Long & Esterline, 2000); others consider Obligations, thus giving rise to BOID (van der Torre, 2003) and BDO models (Ma & Shi, 2000). Moreover, other authors investigate emotional agent models (Camurri & Coglio, 1998).
- **Acting humanly**: spurred by the famous Turing Test proposal (Turing, 1950), it is based on the idea of measuring agent intelligence through a fair competition with human one. In order for the test to be passed, i.e., fooling a human observer, computer machine should be endowed with knowledge representation, natural language processing, automated reasoning and learning ability at the maximum extent. Up until now, the gold medal of the Loebner Prize that would award the designer of the first machine passing the Test still remains unassigned. Despite the initial enthusiasm strived by the Turing's proposal, much criticism has been raised about the effectiveness of the Test. The philosopher of mind John Searle for example opposed to Turing's view the hypothetical experiment of the "Chinese Room" (Searle, 1980). The gist of Searle's argument is that any symbol manipulation machine cannot be considered as having an intentional mind in the proper sense. In other words,

according to Searl's thought, computational intelligence cannot resemble human one even for ontological reasons. The Turing test is a measure of the complexity of a problem from the perspective of AI. It is noteworthy, for example, that the general problem of Word Sense Disambiguation has been shown to be at least as much difficult as the Turing test (as quoted in (Navigli, 2009)). In recent times, almost sixty years after the Turing proposal, conversational-oriented applications called *chatterbots*, such as A.L.I.C.E. by Richard Wallace, winner of several Loebner prizes, are attracting the interest of scholars and investors. Chatbots are computer programs able to simulate intelligent behaviour in textual conversation with humans in very restricted domains. For example, they can be used as virtual assistant in Web content presentation. Although they are not certainly able to undertake an open-domain conversation for long time, they seem to be effective in accomplishing task-oriented dialogue-based activities (Carberry & L. Lambert, 1999) (DeVault et al., 2009).

- **Thinking rationally**: it is probably the oldest approach *ante litteram* to AI, since it can be traced back to ancient Greek philosophers such as Plato and Aristotle (who first formalized in the IV century B.C. the notion of logic through his famous *syllogism*). In the last two centuries, the logicist tradition has been dominated by mathematicians that produced several contributes to formal knowledge representation and reasoning like first-order-logic. Looking at AI from a theorem-proving perspective is appealing for the rigorousness of the approach, but theoretical and practical obstacles arise from the computational perspective. Some critical points, which are barely distinguishable since they present different facets of the same epistemic issue, are for example: informal and empirical knowledge representation (to what extent observed facts comply with learned and/or formal rules?) and tractability (is a sub-optimal solution to the problem achievable in reasonable time?).
- **Acting rationally**: it likely represents the predominant current approach to AI and is centred on designing *rationale agents*. A rational agent abdicates from the achievement of correct inference at any cost in favour of a more pragmatic action that also accepts sub-optimal choices. The outstanding development of multi-agent systems in the literature can be ascribed to this approach. Russel and Norvig advocate the superiority of rationale agent by saying: *"First, it is more general than the 'laws of thought' approach, because correct inference is only a useful mechanism for achieving rationality, and not a necessary one. Second, it is more amenable to scientific development than approaches based on human behaviour or human thought, because the standard of rationality is clearly defined and completely general."*

Most of the current trends in agent and MAS modelling and implementation seem to be following the latter paradigm.

2.2.4 Agents from the literature to real-world applications: the MAS paradigm

Fostered by a rapid advance in hardware and software technologies and by the increasing need for the management of complex distributed systems, the attention of researchers has progressively moved the focus on MAS, i.e., societies of agents aimed at accomplishing (often in a completely automated way) human user-centred task.

In the recent past, the search for a shared theoretical model for MAS has produced a long debate within the research community (Flores-Mendez, 1999). Three points of discussions mainly following the survey work of Omicini and Poggi (Omicini & Poggi, 2006) are:

- Standardized MAS design;
- Adopted communication languages;
- Real-world application domains.

2.2.4.1 Standardized MAS design

The work toward standards for agents interoperability was mainly carried out from the middle of 90s by FIPA – Foundation for Intelligent Physical Agent - that boosted the study and development of MAS applications: now, as a result, a large number of both open source and commercial agent development environments and toolkits are available (see, for example, JADE – Java Agent Development framework – (Bellifemine et al., 2001) and JACK (Winikoff, 2005)). In particular, it is here worth noting that JADE is today the most used agent-oriented platform worldwide.

In the last years, research mainly focused on enhancing the most widely used development tools with new features, aimed at simplifying software development, as well as to extend their use in other application domains. In particular, a number of researchers are working in: (i) the development of tools for bridging agent technologies with both Web services and Semantic Web technologies (Motta et al., 2003), (ii) the definition of agent programming layer on the top of the most known peer-to-peer middleware (Bertolini et al., 2003), and (iii) the introduction of the most sophisticated security techniques in the MAS architectures (Poggi et al., 2005).

2.2.4.2 Adopted communication language

One key element in MAS is communication. In fact, agents need to be able to communicate with users, with system resources, and with each other if they are to cooperate, collaborate, negotiate and so on. Therefore, a number of researchers focussed on communication components for MAS and, in particular, on the definition of a language for the communication between agents.

Agent languages rely on speech act theory (Searle, 1969) and are based on a separation between the communicative acts and the content language. Currently the most used and studied agent communication language is the FIPA ACL (FIPA, 2002), whose main features are the possibility of using

different content languages and the management of conversations through predefined interaction protocols.

However, some researchers proved the limits of this languages and are working on the improvement to provide alternative semantics, new ontological supports, new content languages (Di Stefano et al., 2004) and even new generalized theory of communication acts as in (Fornara et al., 2008).

2.2.4.3 Real-world application domain

The current trend in MAS studies seems to put the focus more on applications than on theoretical issues.

In the latest years, ranging from comparatively small systems for personal assistance to open, complex, mission-critical systems for industrial applications (Jennings & Wooldridge, 1998) (Shen & Norrie, 1999) (Pechoucek & Marik, 2008), MAS-based approaches have spawned a countless variety of engineering applications.

Industrial applications are very important for MAS because they represent the field where the MAS techniques were first experimented, and where they first showed their huge potential. Today, MAS are used for a number of different industrial applications: in particular, they are employed in application scenarios like process control (Jennings, 1994), system diagnostics (Albert et al., 2003), manufacturing (Parunak, 1987) and network management (Bieszczad et al., 1998), whose distributed nature easily falls within the reach of MAS techniques.

One of the first and most important application fields for MAS is information management (Decker & K.Sycara, 1997). In particular, the Internet has been described as an ideal domain for MAS, given its distributed nature and the sheer volume of information available that make the use of agents of great interest for searching and filtering the information (Klusch, 2001).

Internet has also pushed the use of MAS technologies in the fields of commerce and business process management. Today, electronic commerce and automated business processes have increasingly assumed a pivotal role in many organizations because they offers opportunities to significantly improve the way in which the many entities involved in the business process interact. In this scenario, MAS have been shown both to be suitable for the modelling and the design of business process management systems (Camarinha-Matos & Afsarmanesh, 2001), and to be amenable to work as key components for the automation of some or all the steps of these processes (Jennings et al., 1996). Moreover, the metaphor of the electronic marketplace has suggested buyer-supplier or producer-consumer strategy models, often based on FL criteria (Minghua He et al., 2003) (Chi-Bin et al., 2005) (Lagorse et al., 2009).

The distributed nature of traffic and transport processes, along with the strong independence among the entities involved in such processes, have made MAS a key solution for the engineering of effective, real-world applications for both traffic management and transport logistics (Davidsson et al., 2005). Different applications have been already realized; in particular, one

of them OASIS (Ljungberg & Lucas, 1992) can be considered as the proof that MAS are the ideal means for building open, complex, mission-critical systems.

Another setting where MAS technology comes into effect is the management of pervasive and ubiquitous applications (see, for example, (Doctor et al., 2005)). The implementation of wide and dense sensor networks able to monitor various parameters, such as the air quality in environmental applications, is nowadays feasible even with off-the-shelf technologies. Since these networks are generally composed of many low-cost nodes, they allow for monitoring wide areas with a high level of spatial detail. On the other hand, they acquire a huge quantity of data, thus requiring advanced approaches to be handled (Abilemona et al. 2010). Another specific aspect of these sensor networks is the significance of the sampled data. Indeed, data sampled by a given node can be considered as detailed observations of a local phenomenon. The integrated analysis of data simultaneously sampled by various neighbourhood nodes gives information about a phenomenon interesting a wider area. By working on the dimension of the neighbourhood, it is possible to have a multi-level vision of the observed phenomenon by changing the observation scale. This requires however, flexible and scalable architectures endowed with sufficient autonomous intelligence in order to solve specific problems without human intervention.

The effort to minimize the semantic distance between smart devices and the final human user is a major point of concern (Acampora and Loia, 2008). In fact, employing a large amount of low-cost general-purpose devices puts forth the need for managing local intelligence in an effective and efficient way where the use of agents and MAS can be relevant.

2.2.4.4 MAS architectural limits

Although MAS provide unquestionable advantages in the field of distributed systems, a number of challenges arise in their design and implementation (Sycara, 1998). Namely, the most important are: problem decomposition, communication, global coordination, technology issues, decision-making.

In (Di Lecce et al., 2004) a MAS-based multi-layer communication architecture facing these points from the perspective of an ontology-driven design was introduced. The key intuition is that, by using an adequate ontological approach, it is possible to define a system having the ability of performing knowledge extraction and providing, at the same time, information to unskilled users too.

Applied in a number of different application domains (Di Lecce et al., 2008) (Di Lecce et al., 2009), the MAS architecture is based on layers that represent the functional steps virtually adopted in any intelligent information processing task, namely: interface layer (managed by the an interface agent), brokerage layer (hosting the broker and the coach agent) and the analysis layer (employing multiple validation and forecast agents). The employed agents are reported in the following:

- Interface agent translates the message from human/natural language into an ACL;

- Broker agent decides which agents can satisfy the requirement. Broker analyzes a local database in which services offered by MAS are stored and, starting by one query, it produces as many messages as the request needs. The language that broker uses to communicate is generated using understandable and common language for all agents;
- Validation agents are concerned with providing a quality assessment for data;
- Forecast agents have their own knowledge, based on a forecasting model. Each forecast agent applies its analysis method, in this way it is possible to define which is the best forecast method among those ones that are used;
- Coach is a complementary agent of the broker, because it is able to assemble the information contained in the messages that the validation and forecast agents have sent. In the end of the process, the Coach sends a message to the Interface agent, which translates the answer from an ACL into a human understandable language.

Communication flow starts at the top-level, in consequence of the user query submission, and propagates down the hierarchy toward database level, were data are structured, confronted and used to make predictions. Then, information flow is pulled back to the high level of the hierarchy to provide a suitable response to the user.

The main critical aspect of this architecture is its dependence on the application-specific ontology. Changes either in the problem semantics or in the granularity level description have a significant impact on the overall system re-engineering process. In other words, this architecture is flat with respect to the problem description: if ontology is granulated in a different manner, this requires rewriting some or every single agent of the architecture.

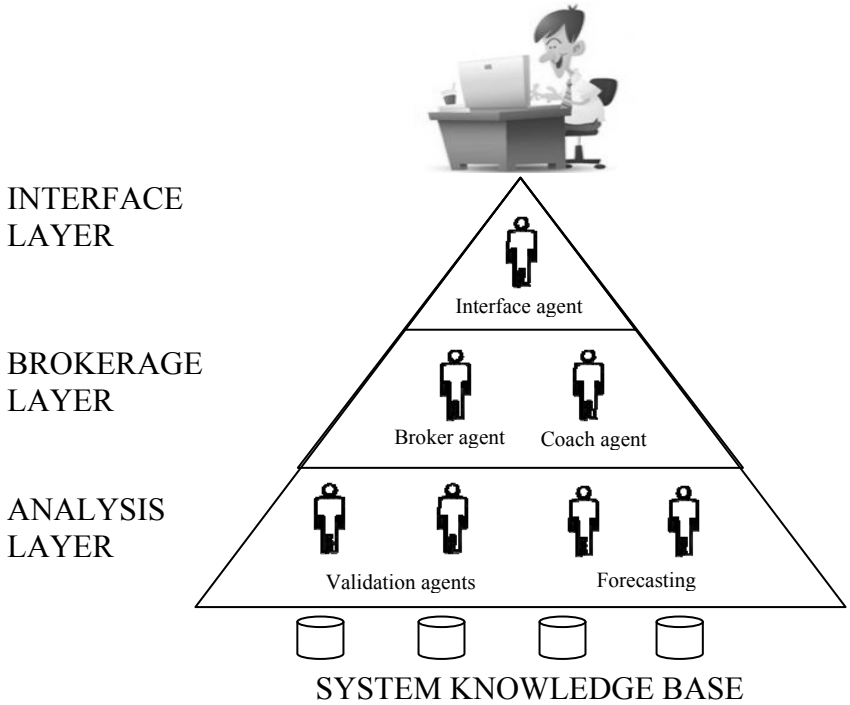

Figure 2.7: Hierarchical MAS derived from (Di Lecce et al., 2004).

2.3 Holons

2.3.1 What is a holon?

The first recorded use of the term 'holon' in the literature is ascribed to Koestler (Kostler, 1967) who devised, in late 60's, the concept of an entity being a whole (from the Greek 'hol') and a part (from the Greek 'on') at the same time. The definition is intrinsically *recursive* since it accounts for describing holons in terms of other holons. As it will be shown in the next chapter, this particular property is essential for characterizing the computational model proposed in this thesis.

The Koestler's intuition was led by comparison with the biological world where multiple entities participate *at different granularity levels* to the goal of the living creature that host them. Examples are cells arranged into more complex structures as organs in a living multi-cell (prokaryotic) organism. These cells are in fact autonomous entities with respect to their own goals; notwithstanding they cooperate for sustaining the organism (with his own goal) they are part of. The search for self-similar, autonomous, and cooperative building blocks to employ in the management of complex systems has taken great benefit from the holon concept.

This is particularly evident in the field of Intelligent Manufactory Systems (IMS) where the new paradigm has given birth to the so-called Holonic Manufacturing Systems (van Brussel et al., 1998) (Kopacek, 1999) (Gruver et al., 2003) (Brennan et al., 2005). In fact, the complexity of manufacturing systems integration, ranging from enterprise resource planning (ERP) to supervisory control and data acquisition (SCADA), combined with the increasing demand for agile and reconfigurable production lines, seems to be particularly tailored to the holonic philosophy.

Since the offspring, several holon-based systems have been presented in the literature, especially in the last decade (Adam et al., 2000) (Fletcher et al., 2000) (Kremer & Norrie, 2000) (Fujita, 2001) (Cheng et al., 2001) (Fleetwood et al. 2003). However, the contribution of the holon paradigm in the scientific literature goes further beyond applications. A late research trend seems to support the idea that the basic of holonic methodology is a way to conceive process description as a system of systems, hence being useful for system modelling and ultimately system thinking theories (Jackson & Keys, 1984).

In a recent work (Simão et al., 2009), a classification of system architecture approaches is framed according to both theoretical and modelling aspects (Table 2.3). The authors identify holonic thinking as an extension of ontology-based theories. This can be considered an evolution of the heterarchical approaches to adaptable and agile systems. Furthermore, the authors consider MAS to be the natural implementation of holonic modelling. Under this perspective, MAS technologies represent an efficient way to support holonic-based systems design, provided that ontology paradigm is revised according to a holistic representation.

Table 2.2 Hierarchy of systems architectures (Simão et al. 2009).

Approach		Paradigms	
		Theoretical	**Modelling**
5	Adaptable or Agile	Fractals, Bionics, and **Holonics**	MULTI AGENT SYSTEMS (MAS)
4	Heterarchical or Interoperable	Ontologies and Cognitics	Uncoupled System (Objects/Agents)
3	Hierarchical Integrated or Visible	Systemics and System Engineering	Computer Integrated Manufacturing (CIM)
2	Hierarchical or Rigid	System Theory	Automatic Control
1	Isolated or Fragmented	Empiricism	*Ad hoc* approaches

2.3.2 Holonic modelling in the literature

Early considerations about the use of the holon paradigm in the framework of system modelling can be traced back at least to late 90's. In 1998, Thompson and Hughes (Thompson & Hughes, 1998) introduce an object-oriented theoretical model to describe (human and computer) activities within a given organization. The work was led by the aim of finding an improved solution to the design of computer integrated manufacturing systems. The basic building block characterizing the holon formal description was adapted from the object-oriented notation to represent IT support of a business process at any given granularity level.

The authors start their analysis from a simple observation: the most significant initiatives in defining CIM architecture approaches and enterprise modelling show that these approaches are both difficult to use and comprehend, and significantly lacking in scope. They claim that the existing approaches do not attend adequately to people and organizational aspects, and their relationship with computer-based systems: traditional hierarchical perspectives are predominant, while business-process views based on the flow through an enterprise towards the customer are lacking.

According to the authors, a manufacturing enterprise can be represented as a network of semi-autonomous cells, "*alike and fractal in nature*", with the common purpose to satisfy the 'supply=demand' equation. Interesting enough, the cells have a dynamic existence: they exist as long as they have a role to play; their specialization depends on the process involved.

In this view, organizational structure *"is provided by the system-subsystem relationship and the classification structure"*. Interaction between subsystems does not imply subsystems loosing their autonomy. There is no superimposed hierarchy of command and control and no hierarchy of decision making. As it happens in biological systems, *"subsystems work autonomously but broadly to the same agenda*".

This kind of 'cooperation in autonomy' capitalizes on the property of emergence: some complex system behaviours are evident only at a higher

echelon as it happens in biological systems. Granularity levels are then properly accounted for without the need of an external top-down decomposition imposed by a hierarchy of commands/control, but only referring to a system-subsystem part-whole decomposition. The use of this paradigm as a conceptual means for describing complex systems is properly called 'holonic modelling'.

Holonic modelling has been successively endorsed and further formalized in a recent work focusing on Process-Oriented Holonic (PrOH) modelling (Clegg & Shaw, 2008), a methodology that uses holistic thinking and a the holon concept to build business process descriptions at different granularity levels. In particular, it has been conceived as a useful support for modellers whishing to analyzing business processes in organizations characterized by high complexity, low volume and high variety.

One key element of PrOH modelling is the concept of granularity. According to the authors' view, some modellers use the notions of scope (i.e., the range of activities modelled) and/or level (the detail/depth of that modelling) with the intent to frame a model's content (Robinson, 2003; Greasley, 2004; Valacich et al., 2006). In PrOH terms, this approach is considered as an oversimplification: a more sophisticated notion of granularity is needed. Put in simple terms, it means deciding what goes in and what stays out of the process model (Gardiner & Gregory, 1996). The modeller must decide on the size of each piece of the model (whether that piece is an entity within a model, an entire model or a set of models). It is plain to say that this approach is intrinsically recursive (Jackson & Keys, 1984) and can be applied at any level of modelling.

Without entering the PrOH methodology in depth (since our thesis does not specifically deals with business process analysis), it suffices saying that the model's scope and the number of levels are defined according to a triplet of parameters, namely: pitch, width and length. Pitch value ranges across organizational levels of the Anthony's pyramid (operational, tactics, strategy); width represents the degree of relationship to core process statement (higher values account for supporting processes): length indicates where to start and finish modelling a process. Given a certain granularity level, the modeller has to ask himself: "does the inclusion of any particular elements, relationships, inputs, outputs or feedback loops in the model, help to describe the behaviour of the core business process and its critical success factors within these dimensions?" The coarser the description used the closer to the strategic level.

PrOH modelling allows for overcoming the traditional 'inside-the-box' task breakdown approach that employs aggregation/reduction patterns in favour of a new 'outside-the-box' methodology based on abstraction/enrichment criteria. Using the authors' own words:

> Aggregation assumes that the truthfulness of activity relationships in the lower pitched models is absolute. In contrast, abstraction does not assume this but builds upon the premise that through developing higher-level models, one can identify new properties, which reshape existing process descriptions and lower level models. In reverse, enrichment is built upon the premise that lower level models can also possess new properties requiring new process descriptions.

In other words, holonic modelling as an offspring of holistic thinking is about providing enrichment/abstraction patterns in a holarchical way. Models explicitly aim to show properties that occur relative to the chosen granularity level. Traditional hierarchical thinking instead aims to define systems in absolute terms. The contrast between the two approaches is depicted in Figure 2.8.

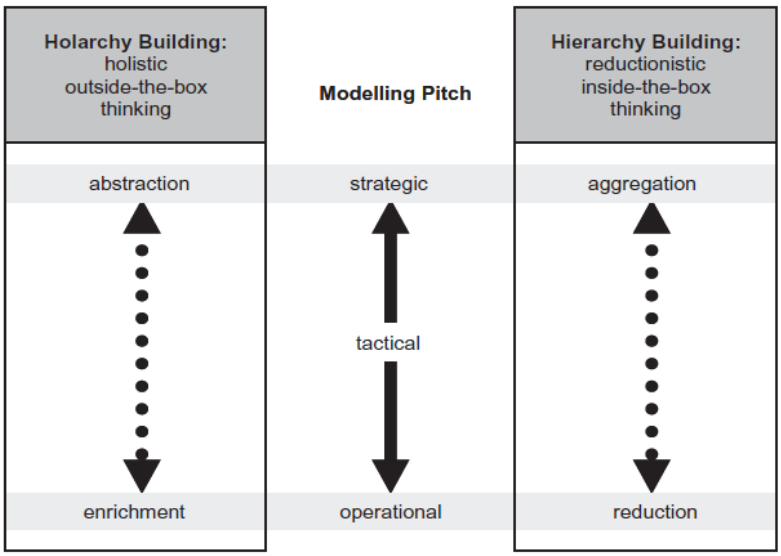

Figure 2.8: Holarchy vs Hierarchy building (Clegg & Shaw, 2008).

Contrarily to hierarchical MAS where agent position is determined by its role in the hierarchy (driven by functional system decomposition during the design process), holons assemble into holarchies depending on the 'emergent knowledge' that, at any granularity level, is necessary to accomplish the system goal within a given process description.

The principle of emerging knowledge has been formally postulated in (Ulieru & Este. 2004) with the intent of describing holarchy as a coordinated system aiming at minimizing system entropy. The authors explicit that optimal knowledge at the holarchy highest level of resolution (inter-enterprise level) corresponds to an optimal level of information organization and distribution among the agents within all levels of the holarchy. Moreover, they use entropy as a measure of the degree of order in the information spread across the multi-agent system modelling the holarchy.

In a very recent work (Ulieru & Doursat, 2010), Ulieru and Doursat emphasise the role of emergent engineering as a radical paradigm shift with respect to traditional top-down hierarchical analysis. The authors contrast traditional engineering approaches where designer imposes order exogenously to the modelled system as a supreme architect of the whole design process with the new one where the design actually becomes a facilitator of the self-assembling process. The difference is evident since, in the latter case, the supervision is demanded to an implicit fitness evaluation function that

depends on the environmental constraints in which the basic blocks of the entire architecture have to deal with.

Consequently, the role of system engineer changes significantly: instead of defining the system along with its constraints in advance, following a top-down hierarchical thinking, he/she supports and guide the complex system through its process of "self-design". Organisational structure then arises from the bottom-up, through interactions among elementary components. The change in role from "dictator" of a system's blueprint to "facilitator" of the self-organization process allows the system "*to adapt its development and evolve to meet dynamic goals and unexpected situations in an anticipative manner—an impossible feat under the traditional approach."* The two approaches are confronted in Figure 2.9.

The design-by-emergence paradigm has inspired our proposal, as we will see in the next chapter.

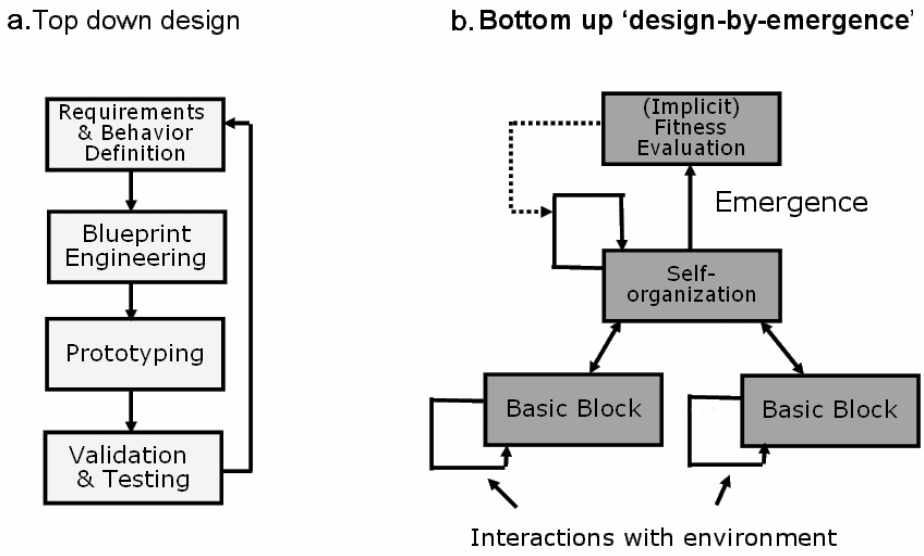

Figure 2.9: Top-down vs. bottom-up 'design by emergence' (Ulieru & Doursat, 2010).

2.3.3 Contrasting Aspects between Holonic and Agent-based Systems

Following the work of Marik and Pechoucek (Marik & Pechoucek, 2002), a comprehensive comparison between holon and agent was presented in (Giret and Botti, 2004). Confrontation is carried out on a number of features. The result of the authors' analysis is summarized by the table in Figure 2.10. Three interesting points that mark the difference between the two models are:

- Information and physical processing: both elements are present in holons while agents are generally considered only as software entities;
- Recursiveness: which is characteristic for holons but not for agents;

- Organization: holons organize themselves according to holarchies, generally represented as dynamic hierarchic structures (Xiaokun & Norrie, 1999), while agent architectures are fixed and can range from horizontal to vertical organizations (Sycara, 1998)(Okamoto et al., 2008).

These points are considered in more detail hereinafter.

Property	*Holon*	*Agent*
Autonomy	Yes	Yes
Reactivity	Yes	Yes
Pro-activity	Yes	Yes
Social Ability	Yes. Human Interface is specific to of each holon.	Yes. Human Interface is generally implemented by one or several specialized agents.
Cooperation	Yes. Holons never deliberately reject cooperation with another holon.	Yes. It may compete and cooperate.
Organization, openness	Yes. Holarchies.	Yes. Hierarchies, horizontal organizations, heterarchies, etc. Holarchies can be implemented using several MAS architecture approaches for federations such as facilitators, brokers, or mediators.
Rationality	Yes	Yes
Learning	Yes	Yes
Benevolence	Yes	Yes
Mobility	Holons rarely need mobility for the execution of their tasks.	Yes
Recursiveness	Yes	There is no recursive architecture as such, but some techniques could be used to define federations that could simulate different recursive levels.
Information processing and physical processing	Yes. The separation is explicit, although the Physical Processing part is optional.	There is no explicit separation.
Mental attitudes	Yes. They do not need to reason on their own mental attitudes or those of other control units.	Yes

Figure 2.10: Holon vs Agent (Giret & Botti, 2004).

2.3.3.1 Information and physical processing

As for information and physical processing are concerned, the commonly accepted architecture to take as a source of inspiration is the one proposed in (Christensen, 1994) and reported in Fig. 2.11.

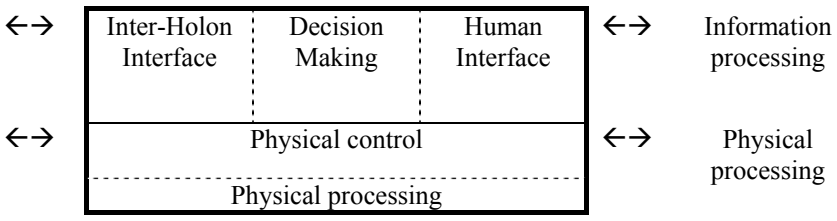

Fig. 2.11: Multi-layer intra-holon architecture according to (Christensen, 1994).

The interesting behind this representation is that holon is an indivisible composition of HW and SW, along with its functional constituent layers. It is therefore impressive how this three-layered architecture can be mapped onto the three levels (bare machine, firmware and operating system) of a multi-

level Von Neumann architecture (Tannenbaum, 2006) equipped with Operating System (OS). A similar tripartition can be also found in other works (Colombo et al., 2006) in the field of Intelligent Manufacturing Systems.

In (Fletcher & Deen, 2001) functional blocks are proposed to manage real-time control for low-level process-machine interaction. In the authors' view, each autonomous holon is composed of a hierarchy of large-grain functional components where interaction is carried out by user-defined cooperation strategies. It is useful mentioning that the authors apply IEC 61499 as a standard-based implementation of their model.

In industrial process management and control systems, function blocks are considered to be computational elements (Fig. 2.12A) of distributed application in a decentralized control system (Fig. 2.12B). Since applications map into devices over the communication network, any application model can be viewed as the composition of event-driven functional blocks exchanging data to manage process control (Fig. 2.12C). A more detailed overview on IEC 61499 can be found in (Christensen, 2007).

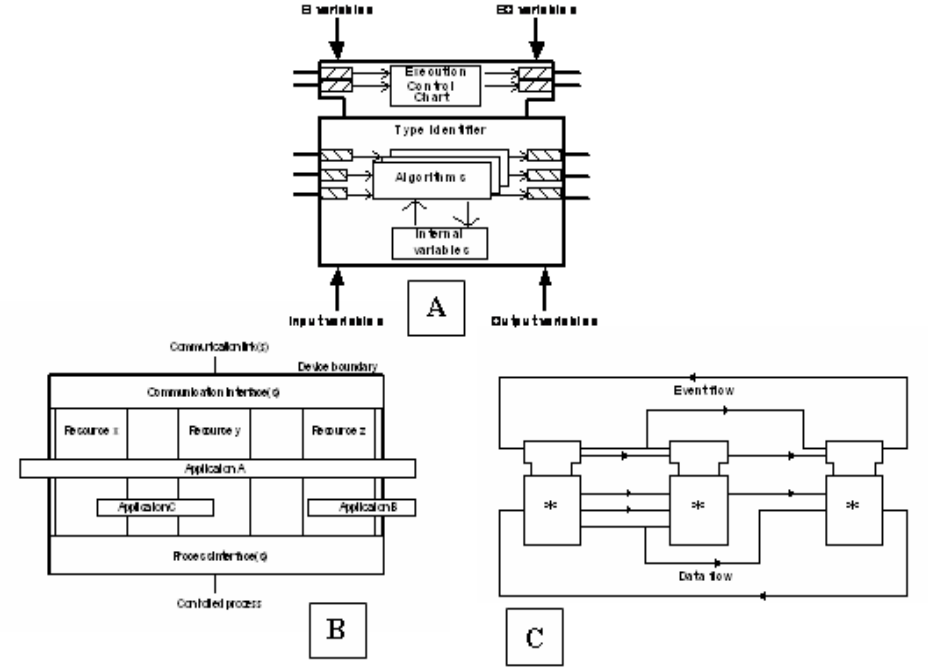

Figure 2.12: IEC 61499 standard. Three basic views are displayed: function block [A], system model [B], application model [C].

2.3.3.2 Recursiveness

Recursiveness is a special property of a function to call itself in a nested fashion; therefore, it is evident that in order for a recursive function to be properly executed, an OS layer is necessary to handle the stack of nested calls. From the point of view of recursiveness, any (decomposable) holon can be described by a recursive agency according to the model presented in 2002 (Parunak & Odell, 2002). The authors extend the Unified Modelling Language to support the distinctive requirements of MAS through an object-based description. They state: "[…] *agent systems are a specialization of*

object-based systems, in which individual objects have their own threads of control and their own goals or sense of purpose".

The holonic (recursive) object-based representation is depicted in Fig. 2.13; with minor adaptations, it is confirmed by more recent works concerning Holonic Manufacturing Systems (Walker et al., 2005). With reference to Fig. 2.13, MAS are made of a collection of agents and is an agent itself; the atomic agent corresponds to an agent that cannot be decomposed (hence, it is not another MAS). Since MAS appear to be a significant component in holonic systems implementation, some authors explicitly refer to 'Holonic Multiagent Systems' (Schillo & Fischer, 2003) (Fischer et al., 2004).

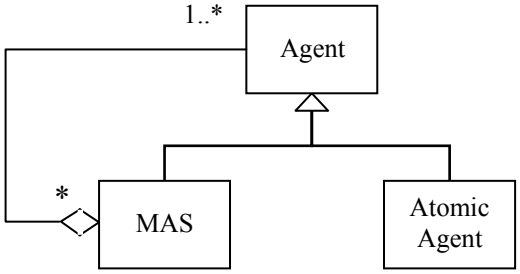

Fig. 2.13: Agent recursive architecture adapted from (Parunak and Odell, 2002).

2.3.3.3 Organization

Holarchy is a specific organization of holons across different levels (compliant with either process-oriented or functional-oriented paradigms). Theoretically, any (MAS) holon could be described recursively by a holarchy until the desired granularity level description is reached. For these reasons, when referring to a holarchy, the generally accepted abstract underlying structure is a hierarchical aggregation of holons like the one in Fig. 2.14. Holons correspond to nodes, while relationships correspond to edges.

Holons groups into small clusters (sub-holarchies) at each layer. External relationships allow the holarchy for communicating with the external world.

Some authors (Shafaei & Aghaee, 2008) attempt to provide a behavioural description of the holarchy. They assume that, for an external observer, those simple and reactive acts take place at the base of the holarchy while complex activities and behaviours are observable at the top of the holarchy. In other words, lower levels are more reactive and upper level holons are more proactive. It is useful noticing that this layered viewpoint is the same described in (Sycara, 1998) to MAS.

Building holarchies is an essential stage in holonic modelling. Nevertheless, in the literature of systems engineering, automatic holarchy building has received little attention so far (Clegg, 2007).

In some recent works (Hsieh, 2008a), (Hsieh, 2008b), collaborative algorithms for holarchy forming are developed as a solution to an optimization problem. Inter-holon communication is achieved using FIPA standard contract net protocol (CNP). The employed formalism for holonic

process modelling is Petri nets, although modified for handling self-reconfiguring situations (Hsieh, 2009). Broadly speaking, this approach can be considered the ultimate evolution of task-driven scheduling algorithms towards holonic-based control architectures.

A benchmark of several task-graph scheduling algorithms was assessed in (Kwok & Ahmad, 1998), while example of reactive scheduling holonic techniques adapting to dynamic real-time constraints can be found in (Chen et al., 2005). Generally, all these techniques, along with their formalisms, develop in the framework of Automation and Operational Research.

In this thesis, a different direction is followed. More specifically, the concept of "holonic granule" is introduced as a basic building block for dealing with granular systems. In particular, emphasis is given to holarchy formation from a KR perspective, hence moving from the field of Artificial Intelligence. In this sense, some bridging works are those of Ulieru (Ulieru & Cobzaru, 2005) (Ulieru & Doursat, 2010).

2.3.4. Holonic Systems: what is still missing

From an engineering perspective, holon behaves as an intelligent agent at the interface level and, at the same time, is decomposable into other holons from the inside. This property makes holon a suitable conceptual model for handling different granularity levels (Calabrese et al., 2010). However, real world implementations of Holonic Manufacturing Systems are still few (Tichy et al., 2005) (Leitão & Restivo, 2008), although they are expected to increase in the near future (Brennan et al, 2011).

In the author's view, a major breakthrough in holonic applications may come from the adoption of a suitable model capable of handling different Holonic Systems properties such as self-organization, self-similarity, capability of handling hierarchically-nested granularity levels and even self-description (as we will see further) within a single computational model. As we saw, attempts in this direction have already been made, but more on a theoretical base and certainly not with reference to the CI field.

In this thesis instead, we setup a CI-based approach to holonic modelling which builds upon the concept of granularity in a more practical and software-oriented way. Since this attempt is quite ambitious, it is useful to bank upon some pre-existing hooks in the CI literature. For our purposes, it is useful in fact to talk about granularity with almost the same language of CI researchers.

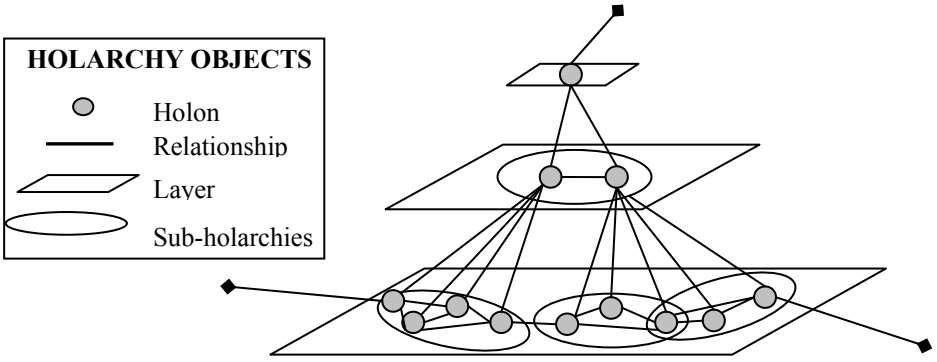

Fig. 2.14: Holarchy layered architecture expressed in a graph-based notation.

2.4 Granular Computing and Computing with Words

2.4.1 The notion of granularity

Granularity is, by lexicon, the property of resembling or consisting of granules.

A granule, metaphorically, is conceived as any atomic element that is not distinguishable from its peers for manifest features but only for the fact that it represents a singleton (eventually embracing a whole) among other singletons. Under this interpretation, set theory partially grasps the essence of granule with the notion of subsets and elements of a set (Hobbs, 1985) (Pawlack, 1982). Following this direction, a brand new theory called Granular Rough Theory (GRT) has been formalized in recent times (Chen et al., 2009). GRT stems from an ongoing work with the ambitious goal of a redefinition of classical set theory by investigating its granular nature with the sole notion of part-whole relation. In this sense, there could be drawn interesting connections between GRT and holonic modelling approaches. However, due to the recentness of the proposal, no specific work in the literature can be found on the topic.

Any abstract reasoning process requires a certain level of understanding, in the sense that abstract entities can be arranged into ontological relations, without zooming in their inner nature (Giunchiglia & Walsh, 1982).

In philosophy, granules can be objects, or ideas; in general, they are abstract entities that are self-consistent, at least at the level of granularity at which they are considered.

In the framework of KR, granules become concepts of a given ontology (Gruber, 1993). An early overview on KR can be traced back to early 80s (Mylopoulos, 1980), where the ideas of ‘aggregation’, ‘generalization’ and ‘context’ were already present.

Leaving aside the epistemic aspect of what ontology actually is, ontology engineers have preferred to search for how to express knowledge (Uschold & Gruninger, 1996). This has progressively emphasized the importance of hierarchically structured systems. Especially in the last decade, with the advent of research on Semantic Web (Berners-Lee, 2001), this shift has given rise, on one hand, to formal ontology languages (like OWL (Antoniou & van Harmelen, 2004)), on the other, to taxonomies and machine-readable dictionaries (like WordNet (Fellbaum, 1998)) semantic lexicon.

In the field of CI, Lofti Zadeh, has been addressing the computational aspects behind the notion of granularity for more than two decades. He first introduced the notion of information granulation (Zadeh, 1979); then, he formalized that concept in the more general theory of FL (Zadeh, 1996) (Zadeh, 1997).

Led by the observation of the human reasoning process, which is inevitably built upon some machinery for handling approximate and imprecise logical inference, Zadeh considers information granulation to be a key aspect of both human concept formation and intelligent information systems. According to his view, granules (whether crisp or fuzzy) are (Zadeh, 1998):

Granule definition

Def. 2.4: "*clump of objects (points) drawn together by indistinguishability, similarity, proximity of functionality*"

The process of forming information granules is called *information granulation*. In the light of FL, information granulation is the basic process of Granular Computing (GrC). Ultimately, it provides a basic framework for Computing with Words (CWW) methodology, i.e., expressing knowledge of observed phenomena in terms of linguistic propositions rather than numerical equations. The two concepts of GrC and CWW are presented hereinafter in more detail.

2.4.2 What is Granular Computing (GrC)?

The term "Granular Computing" is a relatively new one. It was first used in 1997 (Lin, 1997) to provide a unique label for a number of models, ideas, applications sprouted from different domains such as machine learning, data mining, high-performance computing and so on.

We employ the definition quoted by Pedrycz along with some logical passages from his introduction to GrC (Pedrycz, 2001):

Granular Computing (GrC) definition

Def. 2.5: *" GrC deals with representing information in the form of some aggregates (that embrace a number of individual entities) and their ensuing processing."*

According to Pedrycz's view, GrC as opposed to numeric computing (which is data-oriented), is knowledge-oriented and accounts for a new way of dealing with information processing in a unified way. Since knowledge is basically made of information granules, information granulation operates on the granule scale thus defining a sort of pyramid of information processing where low levels account for ground data and higher level for symbolic abstraction (see Figure 2.15).

The problem of traversing different granularity levels according to both enrichment or abstraction criteria becomes then a relevant issue from the system engineer's point of view. Several points need to be addressed. For example, are knowledge structures developed with the use of "large" information granules useful when more specific results are required? Is the identity of the granule forming elements lost when granulation is carried out: i.e., is granulation a non-recoverable process? What are the limits of abstracting and enriching information structures? As Pedrycz says, "*these aspects boil down to the mechanisms of encoding and decoding granular information.*"

Notice that, given an information granule X, the encoding/decoding mechanism at each level should be such that:

$$||COD(ENC(X)) - X|| \rightarrow \min$$

If the absolute difference in the left of the equation were zero, the algorithm would be information loss-less with respect to different granularity-level representations. Efforts to develop algorithms in this direction have been devised by the same Pedrycz in recent times using FCM in a collaborative agent environment (Pedrycz & Rai, 2008).

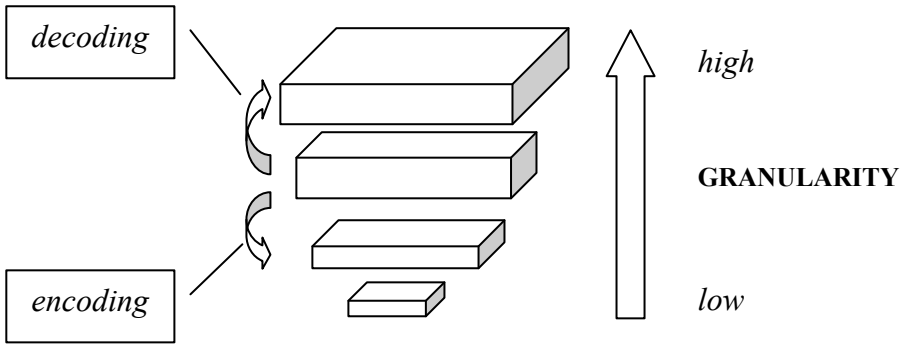

Figure 2.15: Information pyramid according to Pedrycz (Pedrycz, 2001).

2.4.3 Hierarchies in GrC

In GrC, the importance of hierarchies has been recently addressed by Yao (Yao, 2005) who provides an overview of the methodological, computational and philosophical aspects of GrC by means of the unifying element of granular structure. He argues that granular systems self-manifest their properties through (often multi-level) hierarchical patterns.

The following subsections synthesize part of Yao's considerations.

2.3.3.1 Architectural aspects of hierarchies in GrC

A hierarchy characterizing a granular system is made of two basic elements:

- Granules: representing the system building blocks;
- Relationships among granules: defining the system structural properties.

Granules arrange into levels. A level is populated of granules whose properties characterize the behaviour of the level. Generally, granules at a particular level can be recursively described in terms of internal hierarchies at a different system granularity level. Due to such taxonomic pattern, a partial order can be employed to define precedence within the structure.

From an architectural point of view, two kinds of relationships can be drawn, namely: inter-level and intra-level. The former accounts for system multi-level expansion, the latter is indeed useful for describing a whole in terms of its parts. Figure 2.16 depicts these elements (granules, levels, and relationships) in a single abstract frame.

Granules correspond to nodes. Relationships are of two kinds: intra-level relationships are expressed as nodes containing further granules; inter-level relationships correspond to edges among nodes at the same level. In other words, granulation seems to produce holarchies rather then hierarchies. This is a key point for our discussion in the next chapter.

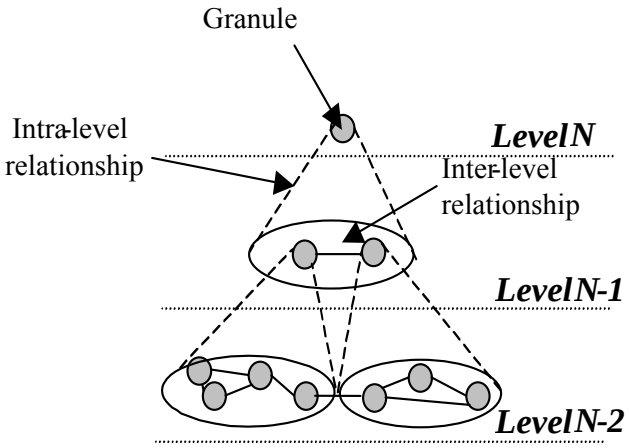

Figure 2.16: Multi-level granular taxonomy expressed in a graph-based notation.

2.4.3.2 Semantic aspects of taxonomies in GrC

Granular taxonomic descriptions can assume different meanings depending on the chosen application domain. Yao suggests, for example, that partial order in hierarchy levels can have, among others, these interpretations: levels of abstraction, levels of reduction, levels of control and levels of detail.

The list of items can be easily extended as well by looking at other contexts, namely: system theory, system modelling, system thinking, logics, philosophy etc. It is noteworthy that, depending on the direction by which we transverse the taxonomy, we obtain, for each context, opposite approaches. Table 2.3 summarizes the stances obtained in the two cases of either going from the root towards the leaves of the taxonomy or vice-versa.

Table2.3 Different interpretations of taxonomy in granular systems.

	System theory	**System modelling**	**System thinking**	**Logics**	**Philosophy**
from root to leaves	from whole to parts	top-down approach	analysis	deduction	reductionism
from leaves to root	from parts to whole	bottom-up approach	synthesis	induction	holism

2.4.3.3 Critical aspects in GrC hierarchies

As a result of his inspection, Yao admits that several questions related to hierarchies in GrC cannot be answered unless a particular system and domain

specific knowledge is adopted. Namely, these questions are: What generates levels? How many levels are needed? Why are the levels discrete? What separates the level? What ties the level together? How do levels interact with each other?

As we will see, our proposed methodology provides an answer to these points according to a CWW-oriented methodology.

2.4.4 From GrC to CWW

GrC and CWW are strictly related. At the core of the CWW methodology lays in fact the concept of granule due to the inner fuzziness of linguistic expressions. In the Zadeh's view, a word w is considered as a label of a granule (Zadeh, 1996). Under this perspective, the use of words becomes *de facto* a form of granulation. For example, saying that Mary is young equals to granulate the concept of age of Mary.

In Zadeh's own words:

CWW definition

Def. 2.6: "*Computing with Words is a methodology for reasoning, computing and decision-making with information described in natural language"*

The reason for studying CWW is therefore simple since "c*onventional systems of computation do not have the capability to deal with linguistic valuations*" (L. Zadeh)

This consideration puts forth the need for a computational model having words in input and words in output. As envisaged by Mendel (Mendel, 2007), the model should be activated by words, which would be encoded into a mathematical representation using fuzzy sets (or other equivalent theories), processed through a CWW engine and finally decoded back into a word (see Figure 2.17). It is fair to say that such a model is an automaton in accordance to what has been discussed at the beginning of the chapter.

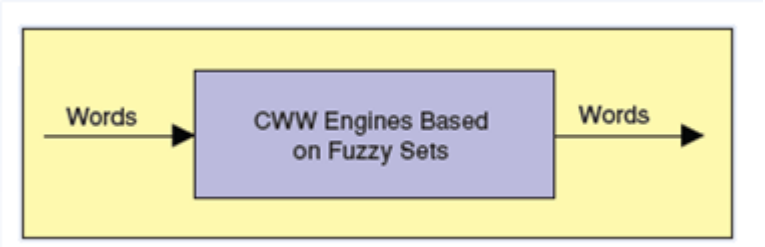

Figure 2.17: A CWW machine according to (Mendel, 2007).

2.4.5 Computing With Words: open questions

How CWW can be put into practice in an effective way is still a point of debate. In a recently published discussion forum (Mendel et al., 2010), researchers from the CWW task Force of the Fuzzy Systems Technical Committee of the IEEE Computational Intelligence Society exchange their opinions about CWW. Their quotes on the critical aspects of CWW can be considered a good starting point for our following argumentation.

> CWW is a broad overarching methodology, which makes it very rich because it is open to interpretations and different instantiations (J. Mendel)
>
> A critical point for the CWW paradigms is to develop reasoning mechanisms that are able to map inputs words, perceptions and propositions to words, decisions, etc. (H. Hagras)
>
> What has been largely missing from the CWW literature is the connection between data and fuzzy set model. This connection should be made at the start of works about CWW because those works need to incorporate the uncertainties about words (J. Mendel)

Starting from previous claims, in the next chapter we will provide a novel instantiation of CWW from the perspective of Holonic Systems. In particular, we bring Holonic Systems and MAS theories within the boundary of GrC by exploiting the notion of multi-level hierarchy (a core aspect of GrC) and its counterpart in Holonic Systems (Figure 2.18).

To achieve this aim, a novel agent-based holonic computational methodology called Hierarchical-Granularity Holonic Modelling is introduced. The proposed methodology will be capable of mapping input words to perceptions, representing knowledge, learning and taking decisions with respect to the given problem domain. Finally, it will be shown how uncertainty can be dealt within the proposed model.

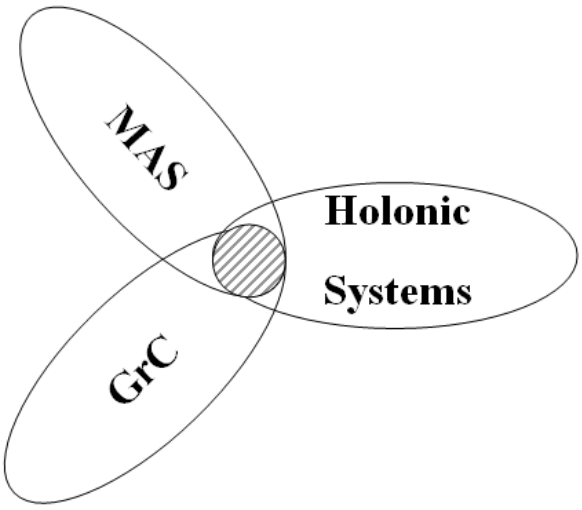

Figure 2.18: Multi-disciplinary approach employed in the thesis.

3. HIERARCHICAL-GRANULARITY HOLONIC MODEL (HGHM)

As shown in the previous chapter, agents and holons account for well-established paradigms in the arena of complex systems engineering. The two approaches both deal with the same issue, i.e., intelligent information processing (especially on large scale), but from different viewpoints.

The realm of agent-based systems is intimately related to the AI perspective sprouted from the Minsky's idea of intelligence as a society of agents. On the other hand, the realm of Holonic Systems grounds on the holistic intuition proposed by Koestler who considered biological systems as a coordinated multi-level structure of beings conciliating the part/whole duality in a unique entity that he called holon. In between, we set GrC and, in particular, CWW as promising approaches to deal with complex systems from a linguistic-oriented direction in opposition to traditional data-oriented techniques.

The offspring from AI towards linguistic approaches is apparent and well documented, passing through CI, FL, GrC, CWW (to cite probably the most relevant branch of this evolution); the other way that starts from Holonic Systems is indeed rather accidental at the moment, if existing at all. Our work attempts to reduce such a misbalance with the ultimate intention of importing aspects of holistic thinking into AI-inspired linguistic-oriented computational models and vice-versa.

To fulfil this commitment, we consider it useful to restart from the original Koestler's definition of holon with the aim of finding a minimal computational concept to use as an atomic entity for our modelling technique; once defined, it will be used as a basic building block to setup the whole theoretical and operational framework. Because of the previous considerations, the concept we search for has to encompass both holonic- and CI-oriented perspectives.

For this reason, we begin our discussion in this chapter by presenting the notion of *holonic granule*, i.e., a formalization of the notion of granule independent from the chosen granularity level. Holonic granule re-defines the two concepts of holon and holarchy by devising a unique computational entity based on a recursive structure. As following step, holonic granules are used as basic entities of a 'holonic grammar': the machinery employed for describing linguistically complex hierarchical systems at different granularity levels. An archetype algorithm for managing holonic grammar-generated descriptions is hence designed. Next, a heuristics is introduced to allow automated extraction of holonic grammars from observational data. The overall computational model, endowed with this unsupervised learning ability, completes the picture and gives birth to the proposed hierarchical-granularity holonic model (HGHM).

For the sake of clarity, the chapter is divided into two parts: Part I introduces the novel concept of holonic granule using an object-oriented notation along with its compositional and generative nature; Part II shows how to compute with holonic granule considered as an basic block for building agent-oriented systems.

PART I – HOLONIC GRANULES

3.1 The Holonic Granule

According to Koestler's original ideas, holon is an entity playing the role of a part and a whole at the same time. This is a bit weird at the operational level since it requires the same holonic entity having both the properties of a singleton and a community: how to do that?

Our proposed solution stems from the observation that holons are agents able to show an architectural recursiveness (Giret & Botti, 2004). Already, we know that a holon which can be recursively decomposed at a lower granularity level into a community of other holons is said to produce a holarchy. *If we also considered a holarchy to behave intelligently as if it were a holon*, then the apparent dichotomy between parts and whole would vanish in favour on a new computational entity being a holon and a holarchy at the same time. These two roles are thus interleaved and one does not exist without the other. In particular, viewed from the extern, both holons and holarchies should appear as intelligent agents.

Now, there follow the two underpinning assumptions that help us fixing our proposal: they are based on the explicit notion of granularity level considered from two complementary viewpoints.

Claim. 3.1: *holon is an agent of a holarchy at a given granularity level*

Claim. 3.2: *given a certain granularity level, holarchy is an agent*

The first claim is compliant with the traditional holonic literature, i.e., holon as an autonomous whole (agent) being also a part of the holarchy at a certain granularity level; the second claim accounts for a stronger notion of intelligence as a society of agents assuming any level of the holarchy to be as a whole resembling to its intelligent parts.

Notice that while Claim 3.1 defines a description pattern going 'down' the holarchy (top-down enrichment), the pattern described by Claim 3.2 allows for rising 'up' the holarchy (bottom-up abstraction).

To visualize the semantics of the two claims, it is useful to consider the holon playing the role of an entity and the holarchy playing the role of a granule. It follows that a granule behaves like an entity and groups of entities behave like a granule. This interpretation unveils the double facets of our interpretation of Holonic Systems that can be viewed both at an entity level (enrichment) and at a granular level (abstraction). Such twofold nature is in full accordance with the traditional holonic-based paradigm (Ulieru et al., 2002) and can be expressed in object-oriented notation as in Figure 3.1

While the concept of entity is quite intuitive and does not deserve further attention, the nuance of granule in our interpretation of Holonic Systems as supported by Claim 3.1 and 3.2 is indeed crucial for the following discussion**.**

A first informal identikit of the holonic granule can be drawn:

Holonic Granule definition (informal)

Def. 3.1*: *A holonic granule is a granule showing the properties of a holarchy

Note that this definition is an extension of that of Zadeh presented in the previous chapter: it adds in fact the aspect of holarchy accounting for an inherent architectural recursiveness. It is noteworthy that this aspect, in GrC, is generally not taken explicitly into account.

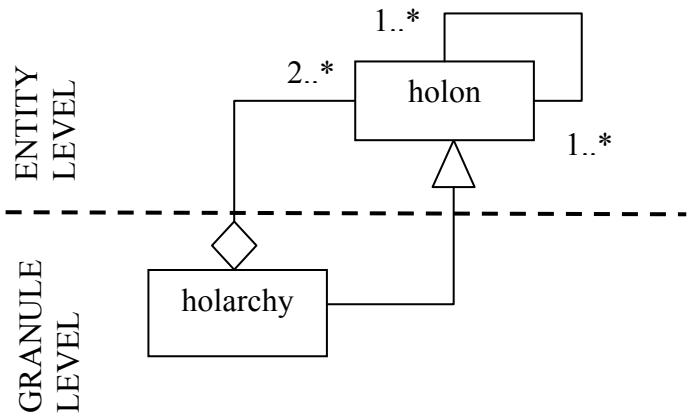

Figure 3.1: Object-oriented representation of a holonic system.

3.1.1 Providing a formal definition of holonic granule

We now introduce a more rigorous assessment of the same concept of holonic granule presented above.

Holonic Granule definition (formal)

***Def. 3.2*:** *Consider a set E of n entities, a set G_E of k-tuples whose elements determine a total cover (not necessarily a partition) of E and a set R_G of binary relationships defined over G_E such that the graph $< G_E , R_G >$ is connected*

$$E = \{e_1, e_2, \ldots, e_n\}$$

$$G_E = \left\{ g_i^k \in E^k \mid g_i = (e_{i_1}, e_{i_2} \ldots, e_{i_k}), \bigcup_i g_i = E \right\}$$

$$R_G = \left\{ r_j \in G_E \times G_E \mid \langle G_E, R_G \rangle \text{ is a connected graph} \right\}$$

A holonic granule (HG) is the graph $HG = \langle G_E, R_G \rangle$

g_i elements represent groups of similar and indistinguishable entities; consequently, they are granules in the Zadeh's sense. Mathematically, g_i, which are nodes of the graph defined by the HG, correspond to edges of the hypergraph defined over the entities of set E. We prefer however to disregard HG representation as a hypergraph since we retain it to be misleading. The reason for this position is that HG, as in Def 3.2, already incorporates the presence of entities, albeit implicitly. This because in order for a HG to be designed, it suffices defining the granules of the next neighbour sub-level (g_i) as they were nodes of the whole graph without worrying about how these nodes can be further decomposed into subsumed entities in a recursive fashion. HGs in fact are abstract categories that can account for any given

granular problem. In other words, HG concept, by itself, guarantees recursive decomposition/granulation.

It is important to observe that while a HG is always an entity (semantically, everything can be defined as to be an entity!), an entity can be either a HG or not. If an entity is a HG, we are simply defining a new decomposition where what we previously called as entities now become granules thus implying the need to find out some other more specific entity as basic building block. This case will be treated more in depth shortly hereinafter. Otherwise, if entity is not a HG, this means that, according to our problem representations, entity 'has reached the ground' and then represents an atomic concept. In this case, we refer to entity as 'primitive' or 'ground' HG.

Primitive HG definition

Def. 3.3*: *Given a problem description expressed in terms of HGs, a primitive (or ground) HG is any HG at the lowest granularity level.

Notice that HG, by definition, can be expressed as a particular type of a UML (OMG, 2007) class diagram (see Figure 3.2) where composition relationships are recursively defined over HG class in addition to the relationships defined in R_G.

One could ask why we considered composition rather than aggregation relationship. The reason is that composition is conceptually strongest than aggregation. Given a certain HG, if we could leave out a sub granule from it we would obtain a different whole, hence a different HG. In other words, composition accounts for the semantics of the phrase 'a whole is more than the sum of its parts', which is a well-known motto in the holistic thinking community.

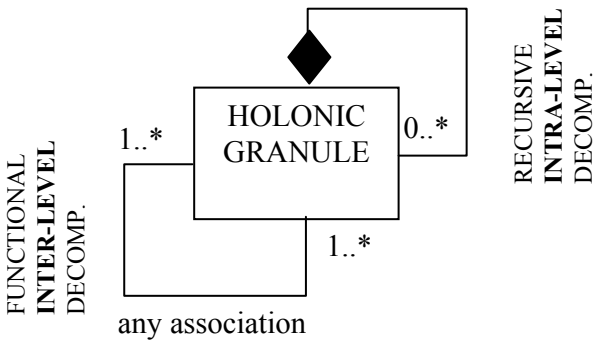

Fig. 3.2: Representation of a HG-based system as a UML class diagram.

3.1.2 HG-based system description: inside-the-box and outside-the-box aspects

According to previous considerations, a HG-based system can be represented architecturally as a holarchy of HGs arranged in a given multi-level structure which depends on the nature of the system under scope hence,

ultimately, on the system ontology. In this regard, two main granule relationships can be identified:

- *inter*-granule relationships;
- *intra*-granule relationships.

When the HG-based holarchy is described at a given (entity) level, the focus is on inter-granule relationships. In this case, the meaning of inter-granule relationships depends on the semantics of the HG. Alternatively, when the focus in on the mappings between a HG and its subsumed components at the next lower level then intra-granule recursive relationships are being considered. This because a process of conceptual refinement in granule description is being carried on.

The distinction between intra- and inter-granule relationships defines two kinds of complementary granular system description approaches:

- 'inside-the-box' (traditional);
- 'outside-the-box' (innovative).

The former relates to connections among subsystems at the same level, i.e., it describes a whole in terms of its parts; the latter relates to internal sub-system decomposition (mapping from a level to the next one). The two approaches are both needed when a complete HG-based system is studied. Using the same object-oriented notation employed above, we include this new aspect in Figure 3.3.

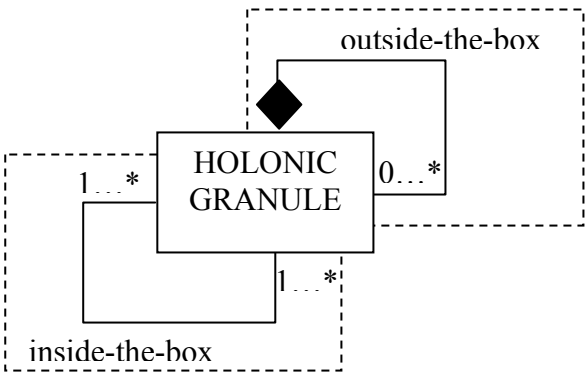

Fig. 3.3: Outside-the-box and Inside-the-box descriptions for a given HG-based system.

3.1.3 Inside-the-box vs outside-the-box views: what comes first?

A reasonable point of suspicion about the HG-based system decomposition may be the ambiguity hidden behind the fact that some granules show both PART-WHOLE and functional relationships at the same time. It is fair to ask ourselves: what comes first in complex system modelling? The answer has a dramatic impact on the credibility of the whole proposal.

Our position is that the supposed ambiguity is a false problem, since it can only solved with specific regards to the given scenario.

For the sake of clarity, we prefer not to scatter problem discussion. Instead, we deal with this point by means of an intuitive example.

Suppose we have two concept granules: a car and an oil station. The car is composed of a number of parts (accounting for part-whole relationships): wheels, front and rear glasses, engine, etc… Among various parts, there is also the tank. Suppose that the tank is near empty and the car is at a certain distance, say 10 miles, from the next station (this situation accounts for a functional relationship between the car granule and the oil station granule). Car granule has to perform a decision about refuelling, i.e., exploiting the service of the oil station granule or not. This decision is in the mind of the car driver (an intelligent agent – not necessarily a human being) and may depend on a number of factors. Likely, the more the tank is near to be empty, the more certain the stop to the station. In this way, the problem is modelled in a fuzzy-logic fashion with a simple rule. Otherwise, driver's decision can be influenced by his/her own attitudes or even external condition (in heavy rain the driver may wish to stop in a covered place) and hence could be described by a very complex inference model. In this worst-case scenario, how many times do we (or an intelligent program) use abstraction and enrichment patterns and functional relationships among components and in which order? There seems to be no optimal answer for this.

This short story is to support the idea that ambiguity resolution is solved by the decision model, which cannot be a-priori defined. In other words, it is the system that drives out the correct sequence of inter and intra-granule 'calls' during system functioning.

3.2 Multi-Level HG-Based Systems

By similarity with the assumption of Claim 3.2 that holarchy is also a holon, we hypothesize from Def. 3.2 that any given HG is itself an entity: a fair hypothesis since everything can be considered an entity at the most abstract level of representation. Thus, the following remarks also hold:

Rem. 3.1: *any HG can be recursively decomposed depending on the granularity levels one may want to reach*

This decomposition is actually a type of enrichment, as we showed in Chapter 2. *Rem. 3.1* provides slightly a better insight into granularity level understanding in GrC terms. If L is the level of representation of a system granule HG^L, its enrichment at a finer granularity level leads to something like this:

$$HG^L = HG_1^{L-1} \cup HG_2^{L-1} \ldots \cup HG_m^{L-1}$$

It is interesting to note that enrichment is a top-down description pattern aimed at zooming inside system structure. The result of this zoom is the discovery of new entities and relationships, hence new HGs.

We assume that, at any level, decomposition (hence enrichment) is *loseless* meaning the inverse composition process (hence abstraction) should give back exactly the original HG.

During HG-based decomposition process, there may be clumpiness among granules at level L-1. This accounts for non-crisp distinctions among HGs at

that level. For example, the idea of HGHM, the core of this doctoral thesis, is mainly presented in this chapter, but some references to it can be found in other chapters as well. We should undertake a finer granulation, for example at paragraph level, to appraise crisp boundaries for that concept.

Notice that HG-based enrichment can be iterated until a desired abstraction level is reached, i.e., primitive HGs appear. This actually produces a holarchy structured at different granularity levels. Hence, we define a HG-based system this way:

HG-based system definition

***Def. 3.4.*:** *A system is said to be HG-based iff it is representable as a HG-based holarchy, i.e., at multiples granularity levels*

Figure 3.4 helps figure out the idea of granularity levels in HG definition by means of a pictorial representation.

HG-based description as a modelling technique can be used to characterize any sort of complex system or process. For example purposes, we apply the notion of HG to the description of the well-known bubblesort algorithm.

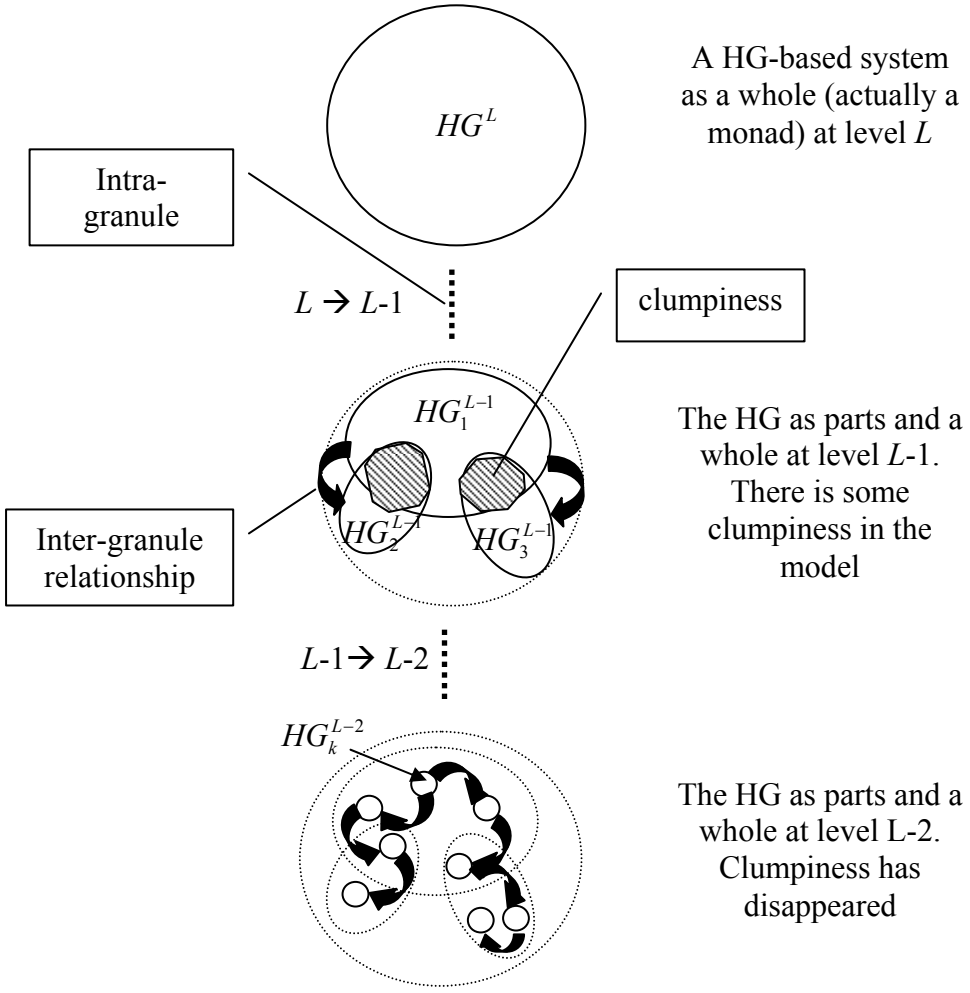

Figure 3.4: HG-based holarchical decomposition.

3.2.1 Exemplar HG-based description: the bubblesort algorithm

Bubblesort is the name of a popular simple sorting algorithm based on the metaphor of bubbles. Spanning through the entire input array length, elements with highest values bubble to the top of the array through a binary swapping mechanism based on the confrontation of adjacent value pairs. A (X, Y) value pair is swapped if X is found to be grater than Y. The maximum number of swaps is $O(n^2)$ where n is the array length; this because the swapping mechanism requires two nested cycles over indices i and j to cover all possible situations. At the end of the swapping process, the sorted array is returned.

The previous description identifies several granules of information. The engagement is now to find a HG-based holarchy that allows for describing the bubblesort algorithm at different granularity levels.

Assume we want to move in a top-down fashion, i.e., following an enrichment pattern. We then decompose the conceptual representation of the bubblesort algorithm starting from most abstract HG towards lower granularity levels. HG-based decomposition is carried on according to progressively more detailed view of the process under scope. In particular, as long as new elements contribute to enrich the description, a new level with a finer granulation is setup. Granulation is driven by new variables that allow for detailing the underpinning ontology behind the algorithm.

An exemplar HG-based decomposition is visually represented by the holarchy in Figure 3.5, which is summarized in tabular form in Table 3.1.

Table 3.1 Tabular representation of granularity levels in bubblesort HG-based decomposition.

Level	Semantics of the decomposed HG	New concept causing enrichment [variable:type]	Primitive functions employed (external HGs)
0	bubblesort routine as a black box	A: array	Scanf(&data_in): data read from external environment
			Printf(data_out): write in the external environment
-1	Core sorting process	i,j: ranges	Length(A:array): num find the length of array A
-2	Sorting value pairs	A[i], A[j]: elements of array	Select_pairs(A:array, [i,j]:ranges): list output list of adjacent value pairs
-3	Verification	b: boolean	Greater_than(X:num, Y:num): boolean verify if X is greater than Y
-4	Swapping	<ground>	Swap(X:num,Y:num, A:array): void swap two values in an array

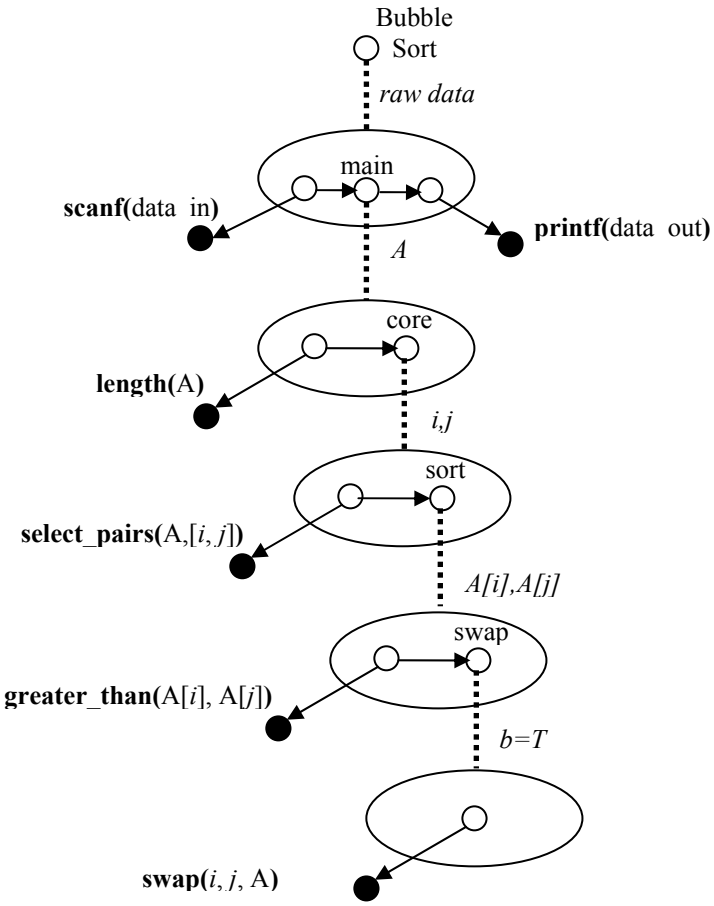

Figure 3.5: Example HG-based holarchy for representing the bubblesort algorithm.

3.3 Expressing HG Linguistically

Until now, HG-based system description has been dealt mainly on an intuitive base. We now pursue its transposition of HG-based description at the computational level by following a linguistic approach: in fact, any description requires a (possibly formal) language to be made effective. For this to be achieved a HG-based grammar made of simple rules is presented, thus providing the suitable machinery to deal with HGs in CWW terms.

3.3.1 Holonic Grammars

To better introduce the concept of HG-based grammar it is useful referring to some basic definitions and remarks.

HG proposition definition

Def. 3.5: *A HG proposition (HGp) is a HG made up of two parts: a string representing a proposition about some observed data and a predicate value ('T' or 'F') representing its semantics.*

For example, an HGp can be of this kind:

$$('\text{the temperature is } 15°C', 'F')$$

which means that, given a certain context, it is observed that the temperature is not 15°C.

Note that, according to the previous definition, the most intuitive way of representing an HGp *h* is:

$$h = (\alpha, v)$$

with α pertaining to some alphabet *A* and *v* representing a value in the set {'T', 'F'}. However, since the couple of symbols (α , v) is itself a symbol, a more compact notation can be employed.

$$(\alpha, v) = \begin{cases} \alpha & \text{if } \alpha \text{ is TRUE} \\ \overline{\alpha} & \text{if } \alpha \text{ is FALSE} \end{cases}$$

The over-line is used to indicate the logic complement to proposition α. From now on, this compact notation will be preferred.

It is important to notice that:

Rem. 3.2**:** *any proposition can be made true at the next higher (abstract) level of understanding, by simply including the falsehood inside the proposition*

For example the proposition:

'The Earth is flat'

is a measurable false hypothesis. However,

'It is false that 'the Earth is flat''

is conversely true. This logic step equals to rewriting a 'false' HGp this way:

$$\overline{\alpha} \equiv (\overline{\alpha}, \text{'T'})$$

It is useful to introduce the following definition:

Self-descriptive HGp definition

Def. 3.6: *An HGp is self-descriptive if it represents a true proposition*.

Note that *Rem. 3.2* implies that any HGp can be made self-descriptive.

We now introduce a compositional property of HGp thanks to the following theorem:

Theorem 1. *Given two self-descriptive HGps* α_i, α_j*, if the logic implication* I_{ij} *holds:*

$$I_{ij} : \alpha_i \rightarrow \alpha_j$$

then the HGp defined by the string '$\alpha_i \rightarrow \alpha_j$*' is itself a self-descriptive HGp.*

Proof. It suffices reminding that, by definition, HGps always occupy the fourth row of truth table of logic implication reported in Table 3.2.

Table 3.2. Truth table of logical implication .

A	B	$A \rightarrow B$
F	F	T
F	T	T
T	F	F
T	T	T

Theorem I can be interpreted as follows. We have two HGps, say A and B, accounting for some verifiable true assertion about some process or phenomenon. Assuming that a logic implication between the two is found through some machinery, consequently the new assertion "IF A THEN B" *considered as a whole* is another true HGp. New HGps can be then built upon other HGps thus forming a growing collection of true IF…THEN assertions.

Notice that "IF A THEN B" can account for two opposite types of logic implication:

- (inductive) Abstraction: B is an abstraction of A
- (deductive) Enrichment: B is an enrichment of A

In the abstraction case we may think to an example like: <IF 'take' THEN 'verb'>. In this example 'take' as a singleton is logically an element of the class 'verb'.

In the enrichment case, we may think to example like: <IF 'Sentence' THEN 'Noun Phrase+Verb Phrase'> meaning that, given a sentence, it consists in a composition of two syntactical parts.

In both enrichment and abstraction case, Theorem I produces something more than the sum of its parts, in particular:

- a new HG;
- a structural relation (the logical implication) connecting its components.

With reference to Theorem 1 and to the concept of abstraction pattern described above we introduce the following definition:

Abstraction rule definition

Def. 3.7**:** *a self-descriptive HGp abstraction rule or simply abstraction rule is any rule that produces a subsumer HGp at a higher granularity level*

The evidence of how abstraction rules come in effect is confirmed by observing the following tautology:

$$\alpha_i \rightarrow \alpha_j \equiv \alpha_i \rightarrow \alpha_i \wedge \alpha_j$$

Note that the right side of the equivalence above, more evidently than Theorem I, highlights the self-descriptive nature of the abstraction rule. Actually, it represents a recursive rule.

Although not manifest, recursion is implicit even in the HGp definition. This can be shown trivially, by considering that the self-implication rule A → A always holds true (because it is a tautology).

It is useful to stress that the premise of the logical implication α_i can be a single HGp or a composition (through logic AND) of more HGps without affecting the generality of *Def. 3.7*. Hence, abstraction rule contemplates both outside-the-box view (entities in relationship with each other) and inside-the-box view (mapping from parts to a new whole).

We can now introduce the following:

HGp alphabet definition

***Def. 3.8*:** *a self-descriptive HGp alphabet or simply an HGp alphabet is any set of self-descriptive HGps.*

The set grows as long as new abstraction rules are found throughout system inspection. This is described more formally in Theorem 2.

Theorem 2. *Given a self-descriptive HG alphabet HA, for any couple of self-descriptive HGps* $\alpha_1 \in \mathrm{HA}$, $\alpha_2 \in \mathrm{HA}$ *related by the abstraction rule* I_{12}*, the following logical entailment holds*:

$$HA \underset{I_{12}}{\longmapsto} HA \cup \{\alpha_1 \rightarrow \alpha_2\}$$

The proof is trivial from Theorem I.

As complementary to the abstraction rule, we can define an enrichment rule this way:

Enrichment rule definition

***Def.3.9*:** *a self-descriptive HGp enrichment rule or simply enrichment rule is any rule that, given an HGp, produces an HG enrichment at a lower granularity level as logical implication of subsumed HGs.*

In the previous example of the bubblesort algorithm, enrichment rules were applied each time description was led to a lower granularity level. Notice that also in the case of enrichment rule, in consequence of rule application, those new symbols that were not initially present in the HGp alphabet now come out.

In order to support the visual understanding of both the abstraction and the enrichment rule, Figure 3.6 helps rendering these logical passages in a pictorial way.

Finally, the core definition of holonic grammar is introduced:

Holonic Grammar definition

Def. 3.10: *Given a HG-based system, a (self-descriptive) holonic grammar (HGGr) is the set of abstraction and/or enrichment rules that supports system description*

HGGr introduces a computational assessment of the original definition provided by Koestler. By means of abstraction rule, two holons build up a new super-holon at a higher granularity level, which comprises the two parts but represents also a new whole. Similarly, by using an enrichment rule, a holon is viewed at a lower granularity level in terms of its intertwining parts. The canonical property of being a whole and a part at the same time is so preserved.

It is worthwhile mentioning that HGGrs are conceptually different from and cannot be mapped directly into any Chomsky grammar (Chomsky, 1956) (Chomsky, 1959). This would require in fact a symbol either being terminal and non-terminal contemporarily (due to HG inner recursion) or introducing a new symbol in the input alphabet because of a rewriting operation, which is not due. Nevertheless, as reported hereinafter, HGGr can be used for parsing natural language expressions as well.

Another point of major differentiation is that Chomsky grammars are not explicitly conceived for grammar-induction tasks, while HGGr can be easily derived from knowledge extraction algorithms applied over sets of data as shown further in the text.

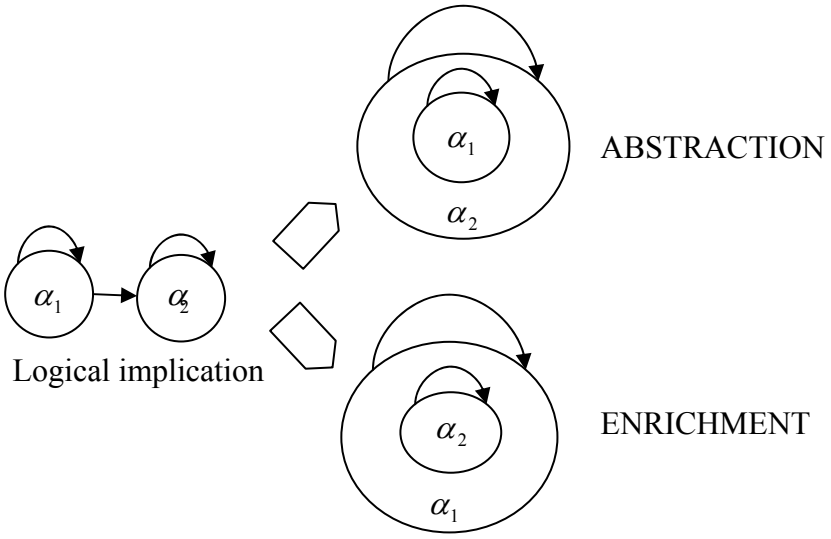

Figure 3.6: Figurative description of the two possible kinds of holonic rules, namely abstraction and enrichment.

3.3.2 HGGr rewriting rules

Abstraction and enrichment rules have been previously presented at a logical/semantic level. In this subsection, we introduce a rewriting notation to use these rules also at the 'syntax' level.

To characterize the rewriting process, the following simple notation is employed: square brackets [] are used to denote an HGp; at the pedix of the right bracket is the label that names the granulation process; strings represent primitive HGp, i.e., HGps that cannot be further enriched into other subsumed HGps. They are denoted directly by their names, in formulae:

$$[]_x = X$$

This is what happens for abstraction and enrichment rewritings respectively:

- Abstraction: A B → $[AB]_{abs}$ where *abs* is the abstraction rule, A and B are HGps, [AB] is a new HGp;
- Enrichment: [AB] → $[[]_A[]_B]_{enr}$ where *enr* is the enrichment rule, [AB] is a HG, A and B are new HGps.

Since a HG-based system is the same independently from the granulation process, assuming a lossless decomposition, it can be hypothesised that,

$$enr = abs^{-1} \text{ and } abs = enr^{-1}$$

if we apply consecutively abstraction and enrichment on the same data, we then obtain:

- A B → $[AB]_{abs}$ # abstraction rule is applied to A and B HGps
- $[AB]_{abs}$ → $[[[]A[]B]_{abs}]_{enr}$ # enrichment rule is applied
- $[[[]_A[]_B]_{abs}]_{enr}$ → $[]_A[]_B$ # $f(f^{-1}(x))=x$
- $[]_A[]_B$ → A B # []x = X

From now on, we will consider the terms 'HG' and 'HGp' as equivalent, since the latter has been introduced only to stress the linguistic representation of the former. Furthermore, 'granule' and 'HG' will be considered always synonyms.

3.3.3 Rewriting examples

To gain practice with holonic rewriting notation, two examples of enrichment (top-down) rewriting are proposed. Abstraction (bottom-up) rewriting will be dealt specifically further in the text for describing the mechanism of (inductive) HG-based knowledge extraction.

3.3.3.1 Rewriting example 1 – Describing bubblesort algorithm with words

With reference to the bubblesort example, the enrichment rewriting process is transcribed as follows:

1. BS →
2. [**scanf** A **printf**]$_{BS}$ →
3. [**scanf** [**length** i,j]$_A$ **printf**]$_{BS}$ →
4. [**scanf** [**length** [**select_pairs** A[i],A[j]]$_{i,j}$]$_A$ **printf**]$_{BS}$ →
5. [**scanf** [**length** [**select_pairs** [**greater_than** b]$_{A[i],A[j]}$]$_{i,j}$]$_A$ **printf**]$_{BS}$ →
6. [**scanf** [**length** [**select_pairs** [**greater_than** [**swap**]$_b$]$_{A[i],A[j]}$]$_{i,j}$]$_A$ **printf**]$_{BS}$

3.3.3.2 Rewriting example 2 – Describing a simple phrase

Consider the example phrase "the man took the book" provided by Chomsky for describing context-free grammars in its groundbreaking paper about the structures of languages (Chomsky, 1956).

Using context-free rules, the phrase can be parsed according to these rewritings:

S → NP VP

VP → V NP

NP → **the man** | **the book**

V → **took**

Where S, NP, VP and V are non-terminal symbols that account for: sentence, noun phrase, verb phrase and verb, respectively, and the others are terminal symbols.

Viewed in holonic terms we have:

1. S
2. $[NP\ VP]_S$ →
3. $[[\textbf{the man}]_{NP}\ [V\ NP]_{VP}]_S$ →
4. $[[\textbf{the man}]_{NP}\ [[\textbf{took}]_V\ [\textbf{the book}]_{NP}\ ]_{VP}\ ]_S$

3.4 Computing with HGs

At this point, HG has been discussed at both an abstract and a linguistic level. As we saw, HGs account for a re-interpretation of the concept of granule from the perspective of holonic-based theories. Furthermore, they represent a viable means to describe linguistically systems at different granularity levels.

In this section, we introduce three basic elements for computing with HGs, namely:

- HG-based holarchy representation;
- HG-based Holarchy management;
- Automated HG-based holarchy extraction from data.

Afterwards, these three aspects will be combined into a unique computational frame (HGHM) which is at the core of our proposal.

3.4.1 Encoding the HG-based structure in a compact KR

Holonic abstraction and enrichment criteria define a conceptual framework for handling the structure of a granular system in a holarchical way. In fact, they employ a unique concept, that of HG, to build up multi-granularity level structures. The interesting is that, in addition to hierarchical patterns, HGs encode naturally the concept of recursion that allows for nesting hierarchies into hierarchies, thus determining multi-strata holarchies.

As next step, we saw how the hierarchical granularity structure of a HG-based holarchy could be expressed linguistically by using the notation of holonic rules.

It is now useful to understand how holonic rules can be useful to support KR in complex system modelling.

A recent work (Hoang Thi Thanh Ha et al., 2009) applies type theory as a means of KR for describing complex systems that are recursively decomposable into subsystems. With respect to this kind of mathematical KR however, HGs have the advantage to be managed more easily, for example by software engineers. Granules at the lowest level in fact can be figured out as classes organized into subsuming granules, which can be considered as packages, i.e., component-based software granules. This interpretation is indeed commonly accepted in the GrC community (Han & Dong, 2007).

Reasoning in terms of abstraction/enrichment rules, the structure of a HG-based system can be encoded into a hashtable having the left part of the rules as keys and right part of the rules as values.

With reference to previous example 2, a possible encoding employing both abstraction and enrichment rules is the one presented in Table 3.3 (information granules are deliberately redundant to allow for different composition/decomposition patterns).

Table 3.3. Hashtable encoding the HG-based structure of example 2.

Keys	Record		#Rule type
	New HG	Program code	
$[\ ]_S$	$[[\ ]_{NP}[\ ]_{VP}]_S$	Program for granule *S*	#enrichment
the man	$[\text{the man}]_{NP}$	Program for granule *the man*	#abstraction
took	$[\text{took}]_V$	Program for granule *took*	#abstraction
the book	$[\text{the book}]_{NP}$	Program for granule *the book*	#abstraction
$[]_{NP}$	$[\text{the man}]_{NP}$	Program for granule NP_1	#enrichment
	$[\text{the book}]_{NP}$	Program for granule NP_2	#enrichment

In analogy with semantic lexica described in the previous chapter, we refer to different keys having the same values as 'synonymous rules' and single keys with multiple values as 'polysemous rules' (as $[]_{NP}$).

In principle, hashtable can host both abstraction and enrichment rules. The two formulations are equivalent at the logical level.

Consider the HG = $[XY]_Z$. HG can be obtained as:

- X Y→ $[XY]_Z$ if an abstraction rule is used, or
- Z → $[[]_X[]_Y]_Z$ if an enrichment rule is used

For practical reasons, it is useful to convert all rules of a type into the other one. In particular, we will use enrichment patterns by default in Holonic System structuring.

Finally, it is very important to stress that the hashtable encoding mechanism can be used to retrieve pieces of information related to the granule used as key for entering the table. For example, system KB can be split according to the constituent HGs and hence called at run-time during system processing. This concept will be clearer in the next chapter where a specific example regarding this aspect will be presented.

3.4.2 HG-based system management algorithm

Suppose to have a system entirely described in terms of HGs through the mechanism presented above: all system granularity levels are then represented by means of holonic rules. In this section, we answer (affirmatively) the question:

Is there a compact program capable of handling the HG-based granulation process granule by granule?

In other words, we are searching for an algorithm that, starting from a given HG, is able to roll and unroll the system holarchy according to its HG-based structure. The algorithm has the form of a FSA where each state of computation corresponds to a HG. We do not deliberately go deep into this equivalence with FSA since this would require a very complex mathematical and conceptual framework. It suffices saying that similar attempts in other fields are being investigated in the literature under the name of abstract state machines (Gurevich, 2000). Therefore, we divert from extreme formalization and we employ already used example to keep on with our discussion.

With reference to the simple bubblesort example, our commitment consists in finding an algorithm based capable of processing information flow granule by granule throughout the levels of the HG-based decomposition. This is somewhat similar to mimicking human granulation abilities when, for example, starting from a general concept, one goes in depth to provide a more detailed view of the ontology he/she wants to communicate.

Here it follows an archetype solution to our previous question. The whole program control structure is based on granules and is entirely defined by the hashtable table presented before where only enrichment rules are considered. Each granule corresponds to a class provided with a unique external method, which acts as a constructor for the class.

```
                ABSTRACT ARCHETYPE CLASS FOR
      (NON-POLYSEMOUS)HOLONIC GRANULE-BASED SYSTEM MANAGEMENT

CLASS Granulate{
  static WM;   //      working_memory
  WM.Granules_List ← this;

  Public Granulate(Granule_to_run, WM){
    1.  WM.Granules_List ←  Look_up(Granule_to_run)
    2.  UNTIL new_Gr in WM.Granules_List {
    3.         WM      ←      Perfom_task(Granule_to_run, WM)
    4.         WM      ←      Granulate(new_Gr, WM)
               } LOOP
    5.  RETURN WM
  }

  Private perfom_task(Gr, my_WM) implements Interface
  // do something…
}

Public static Look_up(Gr)
//      connects to system HG-based structure
```

At start, a main function is supposed to call the first granule. Then, program execution is managed entirely by the HG-based structure. Throughout program flow, a granule/class may leave the program execution to its sub-granules and/or to other granules according to rules of the KB structure. When a call is made towards a sub-granule (intra-granule method call), the program flow jumps down a logical level. In this case, when the callback occurs, program flow returns at the caller granule level. Alternatively, a call is made towards a granule at the same level (inter-granule method call).

Since the system architecture is entirely described in terms of granules, a unique recursive thread can manage execution once for all. Its structure is synthesized by the archetype class whose pseudo-code is presented hereinafter. Examples of actual implementations of the algorithm are reported in the next chapter.

It is noteworthy that the archetype structure is the same for each granule execution. This means that HG-based software is modular at the highest extent.

Two main aspects are worth mentioning:

- the archetype class supervises program unfolding with an extern call to the hashtable for the HG-based system structure (code step 1)
- only one execution thread with a loop-back at the end (code step 4) is required. If the list of granules that have to interact with the thread is not empty (code step 2), the thread performs a task specific to the granule/class (code step 3) and then instantiates recursively another granule/class (code step 4).

The first point is interesting since it conceptually separates business logic execution from business logic management. The second point is fundamental for its implication on the overall code complexity as shown shortly after.

The archetype program is conceived for managing HG-based program execution in a top-down fashion. Starting from a given HG, the holarchy gets unrolled at a lower granularity level according to the structure provided in the hashtable. Note that working memory is passed through the entire computation thread just as if it were a 'moving' tape across sequential instantiation of the granulation class. In this regard, we obtain a very weird computational metaphor. Data is progressively digested by the structure (holarchy) until computation stops and the first called HG outputs processed data. This data digestion mechanism is typical of a recursive program (such as the factorial of an integer). The interesting is that depending on the data digestion, the entire holarchy develops around unrolling and rolling patterns. When a primitive HG is called, it represents the lowermost granule of computation for that branch of the program. Once completed its task, the primitive HG call backs its subsuming HG, thus giving rise to the rolling phase.

3.4.2.1 Handling polysemous rules in HG-based system management

A particular pattern of program execution occurs when a polysemous rule is found. A polysemous rule is a decision point for the holarchy evolution process. The calling thread should be multiplexed into as many new threads of computation as there are polysemous values for the given key.

At least two solutions can be imagined. The first one consists in admitting that there is some decision function embedded in the program code able to choose one of the possible threads. In this case, the granulation process should be endowed with some thread killing mechanism (as it happens for voting agent inside MAS hierarchies). Otherwise, a holarchy cloning mechanism can be supposed.

Each possible thread would give rise to a holarchy cloning. Another possibility is the combination of the two extremes.

Let us consider the previous example about the parsing of the phrase 'the man took the book'.

As we saw, granule $[]_{NP}$ is polysemous and hence it represents a decision point for the HG-based control flow algorithm. As it will be shown in the next chapter, this state of the program execution can be dealt with a call to a fork on the state of the process (giving rise to a multi-threaded program control flow) or on the entire holarchy (giving rise to a cloned process).

3.4.2.2 Cyclomatic complexity of HG-based system management programs

Cyclomatic complexity is a popular metric for measuring software complexity by reckoning the number of basic paths encountered through a program execution (McCabe, 1976). In this regard, program code is arranged according to a graph where nodes represent code blocks and edges represent basic control paths in the program flow.

The cyclomatic number C(G) of a strongly connected graph G (i.e., a graph with the end node cycled back to the entry node) is equal to the total number of linearly independent cycles. A cycle is linearly independent if it does not contain other cycles in it.

With reference to the HG-based archetype presented before, it can be noticed that the cyclomatic number is equal to one. This because the recursive call represents by itself a closing cycle which is a call to another instance of the archetype (see Figure 3.7 for more detail).

Hence, the cyclomatic number of the entire HG-based program is the number of total granule calls (part-whole granulations plus the number of functional relationships) representing system architecture. This value is a lower bound since it depends on the system decomposition performed by the software designer.

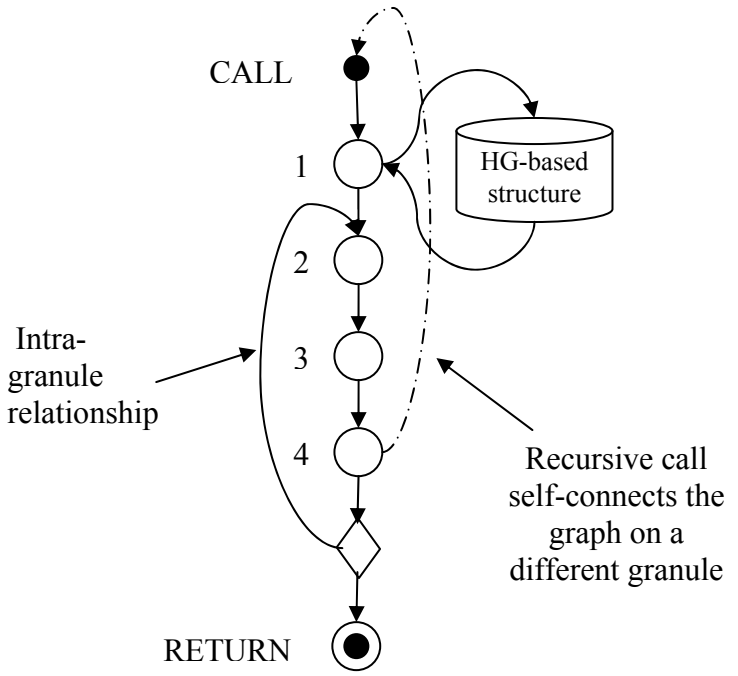

Figure 3.7: Graph control flow for the HG-based archetype program.

3.4.3 Knowledge acquisition in HG-based approach

An important and distinguishing element of intelligent systems is the way its knowledge is acquired. A desirable property of such systems is the ability to automatically extract information from input data. In HG-based setting, this equals to extracting the holonic rules that drive out the granulation process.

In this section, we present a recently published heuristics (Calabrese, 2010) that provides a simple computational process for obtaining HG-based structures from signals.

A (discrete-time) signal is any time-ordered sequence of real numbers. Mathematically, a signal can be denoted as a function such that:

$$s : Z \to \Re$$

Given a set S of signals (made of one single or more elements), the heuristic attempts to extract IF THEN rules from it by means of a two-step procedure. The two steps are respectively called *hypothsesization* and *structuring* for reasons that will be motivated hereinafter. They are preceded by two ancillary phases: signal pre-processing and buffering, whose description is worthless because it corresponds to typical application-specific phases in the signal processing activity. An overview of the proposed technique is highlighted in Figure 3.8.

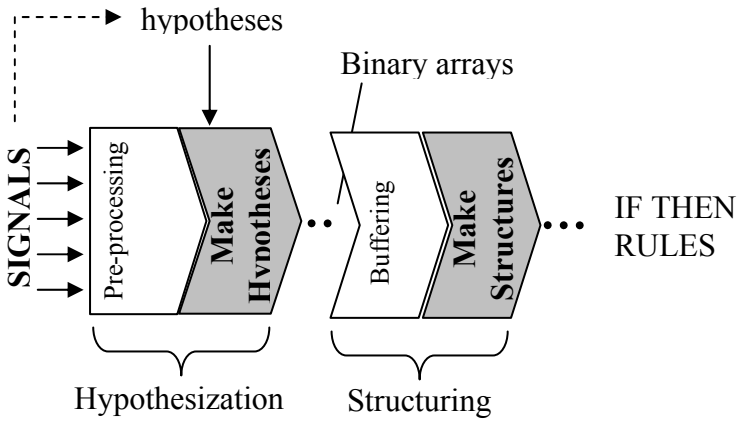

Figure 3.8: Process flow for automated HG-based structure extraction from data.

3.4.3.1 Hypothesization

As first processing step, a transformation is performed on the set of input signals *S* or on a pre-processed form of it. In particular, for each available signal, a function is defined as follows:

$$\varphi_s(h_k) = \begin{cases} 1 \text{ if } h \text{ is verified at time } k \\ 0 \text{ otherwise} \end{cases}$$

h is a numerical hypothesis made on $s \in S$. For example, an exemplar hypothesis can answer the question: is signal *s* at time *k* greater than its mean? The hypothesis can be set by an external agent, or taken from a list of predefined items.

Notice that the input signal is transformed into a binary array. The same procedure is applied to all the available input signals so that a binary matrix is actually outputted by this step.

Since the hypothesization procedure can be performed on running data, the output binary matrix is stored in a first-in-first-out buffer of a given length. Once the buffer is full, it flushes data towards the next step.

3.4.3.2 Structuring

Data coming from the hypothesization step are subjected to a structuring procedure by means of the well-known technique of the classification and regression tree (CART) (Breiman et al., 1984).

CART is a widely used non-linear regression technique that takes a set of data arrays as input and then outputs inter-signal relationships according to the following form:

$$s_k = CART_k\left(s_1, s_2, ..., s_{k-1}, s_{k+1}, ...\right)$$

where $s_i, i \neq k$ are called *predictor* variables, while s_k is called the *predicted* variable.

CART defines a decision tree structure on incoming data. Each decision node (predictor) is a binary variable corresponding to an IF-THEN algorithmic structure. Leaf nodes represent the possible values of the predicted variables. In our setting, the IF THEN structures covering each branch of the tree corresponds to series of hypotheses on the input signals. These patterns represent logical implication rules of the type:

Predictors (intermediate nodes) → *Predicted* (leaf node)

hence, they are holonic rules in the proper sense. In particular, they account for abstraction (bottom-up or inductive) patterns since they use terminal/observed symbols to build up more complex wholes. There are as many holonic rules as the number of branches in the tree.

Notice that, since we cannot know a-priori predictor-predicted relationships, an exhaustive search requires considering all the N possibilities, where N represents the number of binary arrays outputted by the hypothesization phase. Thus, N trees are obtained. They actually form the HG-based structure for the proposed signal processing system.

3.4.3.3 Extracting holonic (self-descriptive) rules: the agent knowledge acquisition problem

Consider, for example, an agent observing three binary signals s_1, s_2, s_3 such that s_3 is defined by the logic function $s_3 = (s_1 \text{ XOR } s_2)$ with s_1, s_2 generated by a random source.

The nature of the logic relations is supposed hidden to the observer agent who can only take note of the measurements he performs on the lines that comes out from the 'black box' of the circuit. We call this setting the 'agent knowledge acquisition problem' and we provide a visually representation for it in Figure 3.9.

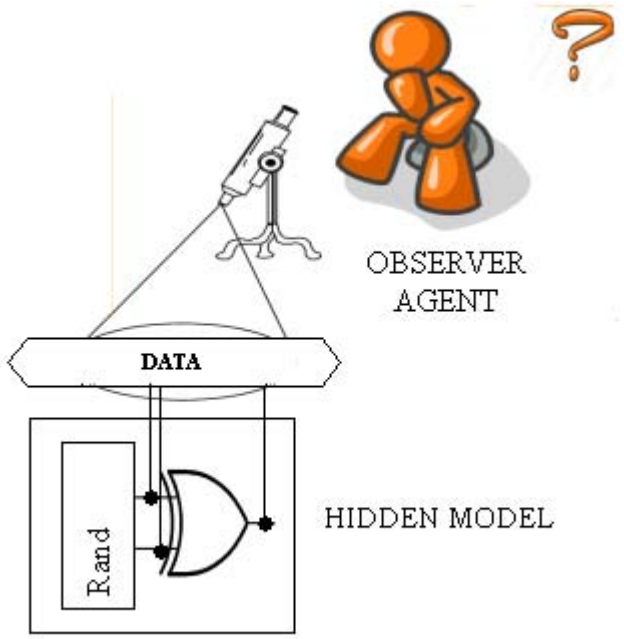

Figure 3.9: Pictorial representation of the agent knowledge extraction problem.

Suppose the agent applying the proposed holonic-based technique. In particular, the hypothesis done is the same for each signal:

$$\varphi(h) = \begin{cases} 1 & \text{if the measured value is} \geq 0.5 \\ 0 & \text{if the measured value is} < 0.5 \end{cases}$$

In order to unveil the kind of relations among signals, the observer agent collects a sufficient number of self-descriptive triplets. Here, the word 'sufficient' is awkward. It means: sufficiently large as to guarantee that all relevant hidden patterns are observed during the series of measurements. Of course, without external knowledge, there is no way to be sure that the measures taken are sufficient.

In this very simple case, the truth table of the inspected logic circuit has only four possible slots. After having observed a random succession of the four types of triplets (namely <0, 0, 0><0, 1, 1><1, 0, 1><1, 1, 0>), we may think that the agent decides to stop buffering data collection and proceeds with CART-based analysis.

Since the agent does not know a-priori which candidate to choose as predicted variables, he should try all three possibilities. The structure obtained from the CART applied over s_3 predicted variable is depicted in Figure 3.10. This structure is immediately referable to holonic rules by inspecting the branch of the obtained decision trees. In particular, four rules can be extracted, namely they are:

$$\varphi_1\varphi_2 \rightarrow \overline{\varphi}_3 \qquad \varphi_1\overline{\varphi}_2 \rightarrow \varphi_3 \qquad \overline{\varphi}_1\varphi_2 \rightarrow \varphi_3 \qquad \overline{\varphi}_1\overline{\varphi}_2 \rightarrow \overline{\varphi}_3$$

Note that the found holonic rules correspond to the truth table of the inspected digital logic circuit. Simulations were run in Matlab® R14 environment, with the libraries of the Statistics Toolbox for CART-based processing.

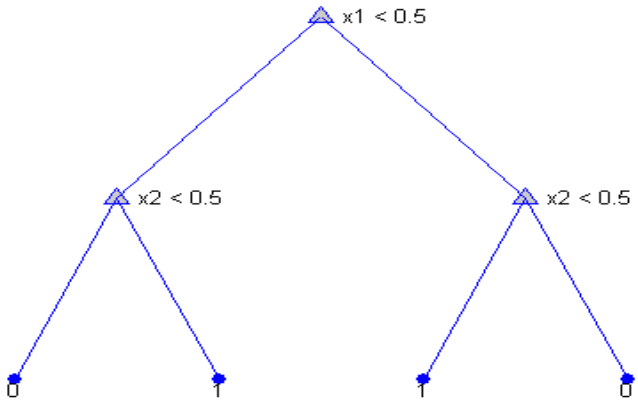

Figure 3.10. Tree structure representing s_3= (s_1 XOR s_2).

3.4.3.4 Uncertainty in holonic rules

It is important to stress that when observations are reduced to binary values, this does not implies that the CART predicted values will fall into the crisp set {0,1}. Suppose to take three random binary variables s_1, s_2, s_3 and impose, for example, s_3 being the predicted variable. The CART obtained after repeated observations is drawn in Figure 3.11. Note that all s_3 values almost equal 0.5. These situations can be interpreted as producing uncertain holonic rules, thus allowing for inferring only uncertain models (at least with respect to the given dataset).

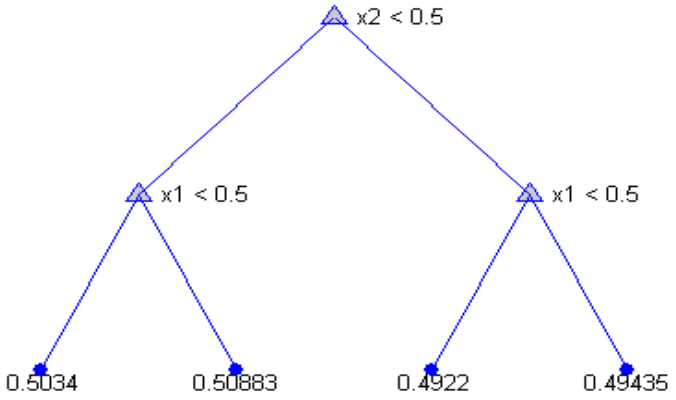

Figure 3.11: Tree structure representing $s_3 = s_1 = s_2$ = RAND.

3.5 Devising a New Kind of Holonic Computational Model: HGHM

So far, different aspects of our HG-based approach have been presented, namely: HG-based decomposition, linguistic description, holarchy structure encoding and management and, finally, extraction of holonic rules from data.

In this paragraph, we pack all previous achievements within a unique full-fledged computational model called Hierarchical Granularity Holonic Model (HGHM), actually a new kind of intelligent agent to use as building block in complex hierarchical systems modelling.

A picture of the proposed HGHM is depicted in Figure 3.12.

As an agent, this new holonic model is endowed with an interface with the external world made up of sensors to perceive the environment and actuators to produce effects in the environment. HGHM behaviour depends on the FSA governed by the HG-based structure.

The presented computational model is structured into two overarching layers:

- KR layer: it is characterized by the knowledge extraction process (implemented through the heuristics like the one presented before) that traverses this layer in order to feed the HG-based KR. Real-time data are taken from the lower computational body layer and the information structuring heuristics is applied on. If a holonic rule with certain accuracy is extracted from data, it contributes to build up the system structure in the form of a HG-based holarchy. Non-structured data are archived in order to enlarge the dataset used for future runs.
- Computational body layer: it is characterized by the holarchy rolling/unrolling mechanism in dependence of the data flow coming from the sensors. It actually represents the automaton (string recognizer) inside the computational model. The recursive HG management algorithm attempts to decode real-time data based on the rules provided by the upper KR layer. If a match is found, a corresponding action is triggered.

From a CI perspective, HGHM is both an (holonic) agent and a holarchy at the same time. This is possible thanks to the underpinning notion of HG. HGHM in fact is a computational model that computes through granulation. During computation, it can be assumed that each HG is in charge of a holon and hence the whole holarchy is nothing but a structure of nested holons working in cooperation.

From a software-oriented perspective, HGHM architecture is a holonic factory where the knowledge extraction process defines the form of the holarchy (by adding or updating holonic rules) and the HG management process is responsible for building and destroying the holons that maps input from sensors to output for actuators.

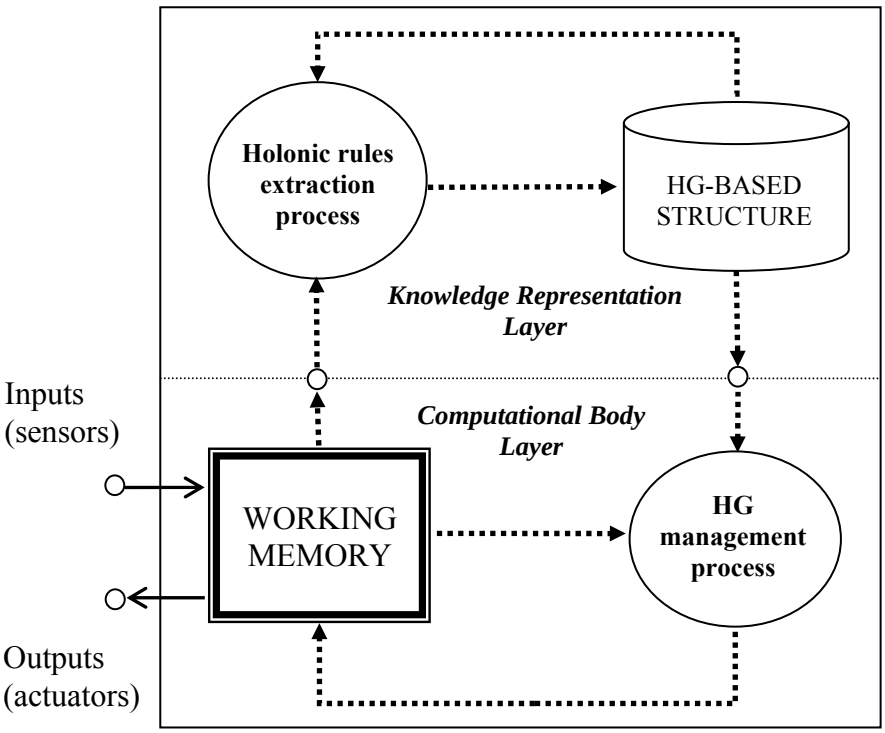

Figure 3.12: The proposed HGHM architecture.

3.5.1 Simple reflex HGHM

The computational model as presented above is mainly the holonic equivalent of the simple-reflex-agent model discussed in the previous chapter. It takes data from sensors, performs a mapping from data to some logical state (i.e., the HG under computation) and triggers the action related to that state according to its FSA model. At the current stage of development, this is a very basic setting that can be certainly improved in terms of dynamic response by adding, for example, internal states for goal-driven or utility-driven behaviours. In this thesis however, we will not go beyond this simple reflex model, leaving HGHM development to further studies on the subject.

Nevertheless, in this basic setting HGHM has some noteworthy property with respect to its homologous in the agent realm. At least there are two major enhanced characteristics:

- Automated knowledge extraction from observational data;
- Native information granulation abilities.

While in simplex-reflex-agent design, it is responsibility of the agent engineer to define behavioural rules for the agent; in the holonic counterpart some simple behaviour can emerge in an unsupervised manner. For example, as the previous section on automated extraction of holonic rules shows, we may use the HGHM to self-describe an observed process in terms of HG-based structures. Thanks to the proposed heuristics, HGHM in fact can start from a '*tabula rasa*' representation of the observed data and then, after sufficient data collection, can build up by itself the holarchical structure extracted from the observed data.

At the same time, since holarchical KR is structurally conceived as a multi-level hierarchy, the kind of data self-description is arranged at different granularity levels, which is indeed a desirable property when observing a complex unknown phenomenon.

3.6 Hierarchical Granularity Holonic Model: a More Formal Assessment

Let $H = \{L_1, L_2, ..., L_n\}$ be a collection of holonic layers composing the hierarchical framework of a HG-based system where each layer $L_i = \{h_1^i, h_2^i, ..., h_{n_i}^i\}$ contains a collection of n_i HGs; let $S = \{s_1, s_2, ..., s_t\}$ be a set of environmental sensors connected to L_1 and let $A = \{a_1, a_2, ..., a_r\}$ be a collection of actuators settable by holons in L_i with $i = 1 \ldots n$.
To assemble the communication 'backbone' of the holarchy infrastructure a binary relation with peculiar characteristics is purposely introduced.

Holonic communication constraints definition

Def. 3.11: *A binary relation* $<^R$ *is defined on the set* $H = \bigcup_{i=1}^{n} L_i$ *, such that:*

1. $h_l^p <^R h_q^p$ *iff* $q = p$ *or* $q = p + 1$*, with* $p = 1 \ldots n_i$
2. $s <^R h_l^p$ *if* $s \in S$ *and* $p = 1$
3. $h_l^p <^R a$ *if* $a \in A$ *and* $p = 1 \ldots n_i$

The relation $<^R$ defines the holonic communication constraints by means of the following subsets:
$C_H = \{(h_l^p, h_k^q) \in <^R \mid l = 1 \ldots n_p, k = 1 \ldots n_q, p = q \vee p = q + 1$ and there exists a communication channel between h_l^p and $h_k^q\}$.
With reference to the pair (h_l^p, h_k^q), two possibilities are given:

1. if $q = p$ then the channel is named *intra-level* channel,
2. otherwise, it is named *inter-level* channel.

Let $O = \{o_1, o_2, ..., o_n\}$ be a collection of ontologies that models a set of sub-contexts related to a composed application domain, and be $f : \bigcup_{i=1}^{n} L_i \to O$ a function such that $f(h_l^p) = o_q$ iff $p = q$.

Finally, let P be the program that manages the HG-based system under analysis and L the heuristics that drives the knowledge extraction process.

All previous definitions provided, it is possible to define HGHM this way:

HGHM definition

Def. 3.12: *An HGHM is any tuple of the following type:* $HGHM = \langle H, S, A, <^R, O, f, P, L \rangle$

3.6.1 HGHM to support knowledge-based modelling

The proposed HGHM, as formally described above, grounds on physical data coming from the sensors and, depending on the structure of its holonic rules evolves from granule to granule to carry out computation. This description holds true for modelling from really simple tasks to really complex ones.

In the case of the bubblesort algorithm for example, HGHM would read the array, unroll the holarchy as shown in Figure 3.5, perform computations on data at each granularity level and then roll back the holarchy at the caller HG level to output the processed result. In this regard, with respect to other modelling techniques (e.g., flow diagrams), HGHM provides in addition the ability to look through abstract/enrichment patterns straightforwardly thanks to the concept of HG.

This property of handling different granularity levels is of course more useful in complex distributed system. For example, in most pervasive monitoring applications, the use of locally distributed smart devices is not sufficient to guarantee an effective and efficient management alone. The amount of information that comes from scattered sources, actually representing different observation points of the same macro-phenomenon, needs in fact to pass through a number of phases (i.e., pre-processing, validation, elaboration, fusion etc…).

Traditional MAS approaches would face this issue by imposing an agent organization that fixedly assigns a functional role to each agent: the main drawback is the stiffness of the whole system. In this case, in fact, MAS architecture is built in consequence of a thorough inspection of the underlying ontology.

A so formed hierarchical dataset can be sliced at different levels with cuts occurring at the same distance from the root. By convention, the root (most general concept) is at the top of the hierarchy, while concepts that are more specific span toward leaves. Each level aggregates concepts with similar granularity: these levels can be themselves clustered according to their sub-contexts, actually defining sub-ontologies organized in the form of holarchies.

In the HGHM-based holarchy, all the holons at the same level (representing one or more sub-holarchies) share the same ontology. The holons at the lowest level receive data from the real world using a set of sensors. Furthermore, these holons can handle various actuators to operate in the real world. All the holons at the higher levels receive data only from the holons at the neighbouring lower level. Each holon can communicate with other holons at its same level or with a holon at the neighbouring higher level. The relationship between the holons at a given level and the holons at the neighbouring lower level is a one-to-many relationship.

Using this approach, by rising from a level to another of the holarchy, it is possible to synthesize concepts. In other words, the proposed system is able to give a telescopic vision of the model under analysis. Each level gives a specific detail about the observed system. A figurative description of the HGHM is depicted in Fig. 3.13.

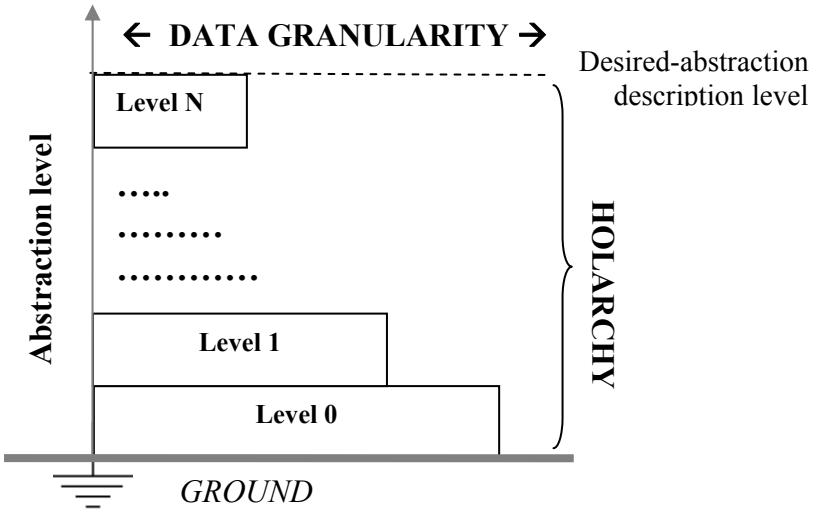

Figure 3.13: HGHM Holarchy spans across different granularity levels according to increasing semantics and decreasing data granularity until a desired description level is reached.

3.6.2. HGHM: an epistemic issue

HGHM has the noteworthy property of being applicable also almost without explicit human intervention during the design phase. In the case of the simple reflex holon, the tuple $\langle H, S, A, <^R, O, f, P, L \rangle$ is produced in part by the knowledge extraction process.

In fact, assuming that *S* represents the input channel from where the data come into the model and A is the output channel where linguistic descriptions of data come out, the other elements of the tuple emerge from the HG structuring algorithm. The holarchy H along with the partial order relation $<^R$ is the result of the knowledge extraction heuristics and evolves as long as new data feeds the process. The interesting is that, since H is defined in terms of hypotheses on observed variables data structure is self-described not needing for new symbols.

Of course, ontology remains a human matter laying in the interpretation of rules behind observed phenomena. However, the issue is quite more subtle. When rules are extracted from data, this is because there is the fair hypothesis that some deterministic phenomenon (automaton) is producing that data (if it were not so, we would have great difficulties in admitting science is a good means to interpret reality!). Hence, the core ontology, i.e., the ground truth, lays within the environment; otherwise, any heuristics (human and non-human) would be effectiveless. In this sense, environment 'self-describes' through data in coherence with the ontology that provides semantics to the whole machinery. In other words, when using HGHM for automated rule extraction from data we are referring to a subset of the previous tuple, disregarding semantic aspects, hence *O* and *f* elements.

In this thesis, we were not concerned with facing this important epistemic issue (which the author reputes to be central to AI); we hope that the proposed HGHM-based approach can be useful to promote an open debate on it.

4. HGHM-BASED APPLICATIONS

This chapter is devoted to explore some of the possible uses of the proposed holonic modelling technique by means of example applications. In particular, three main topics are covered:

1. HG-based holarchy management algorithms;
2. Automated HG-based holarchy extraction from data (both for descriptive and predictive purposes);
3. Hierarchical-granularity holonic modelling in distributed intelligence settings.

4.1 HG-based Holarchy Management Algorithms

In this section, we apply HGHM to string parsing, a typical process in automated translating and compiling tasks. The aim is not however to deploy new parsing algorithms (outside the scope of this thesis) but indeed to grasp the basic concepts that allow engineers to develop progressively more complex holarchy generation strategies by means of the proposed computational model.

4.1.1 Parsing task outline

For example purposes, we employ a very well known string as the target of our processing, the famous Shakespeare's motto *"to be or not to be that is the question*". Punctuation is deliberately left aside in order to handle with the very raw text (along with its possible ambiguous interpretations).

We consider a holonic enrichment (top-down) setting, i.e., we assume that the holonic grammar accepting the text is engineered in a holarchical fashion at progressively higher levels of detail: from the root to the leaves.

4.1.2 Setting up of the KB structure

In order to deal only with available data avoiding the use of external knowledge, in this first example, we disregard part-of-speech tags and consider our KB structure to be composed of a short list of enrichment rules. The result is that we are assuming the syntactical structure uses some words/holons of the text as super-holons for other words/holons. In other words, there are no meta-symbols superimposed. This may appear weird for computational linguists and is certain not canonical to Chomsky production, but it is useful recalling that the goal pursued here is completely different: providing some tutorial examples to understand better how the HGHM actually works when unrolling out and rolling in the holarchy. The parsing setting seems to fit quite well to this commitment.

KB structure, as seen in Chapter 3, is deployed as a hashtable with keys and values corresponding respectively to the left part and the right part of the holonic rule. In this example, each row (HG) of the KB structure is added a column with the pieces of information that define the software processing relative to that row (HG). In this way, the core of the processing algorithm is

encoded in the KB structure and retrieved on the fly during computation only when needed.

In the case of the presented example 1, the added column simply proposes the right part of the holonic rule according to the natural lexical order of the words. This means that the codes retrieved from the added column during software running do not transform run-time data but only perform swapping on them

The KB structure employed in example 1 is displayed in both graphical (Figure 4.1) and tabular notation (Table 4.1) (other structures, of course, can be imagined). The first row, by convention, has an empty entry for the initialization task. The system is supposed in fact to be triggered by the perception of the input. In our case, the input corresponds to the whole phrase, which is the target of our parsing procedure.

In the sequel, it follows a high-level description of the holarchy generation algorithm.

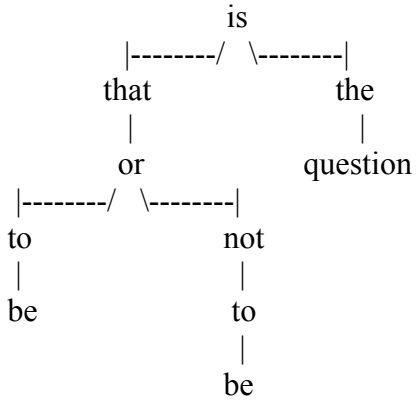

Figure 4.1: Holarchy used for example 1.

Table 4.1 KB structure used for the holonic generation task in example 1.

Left holonic rule	**Right holonic rule**	**Right holonic rule in lexical order**
$[]_{is}$	$[[]_{that}\ []_{the}]_{is}$	that is the
$[]_{that}$	$[[]_{or}]_{that}$	that or
$[]_{or}$	$[[]_{to}[]_{not}]_{or}$	to or not
$[]_{to}$	$[[]_{be}]_{to}$	to be
$[]_{the}$	$[[]_{question}]_{the}$	the question
$[]_{be}$	be	be
$[]_{question}$	question	question

4.1.3 Holarchy generation algorithm

The holarchy generation algorithm is chiefly based on:

- a core function HOLARCHY_GEN;
- a working memory structure referred to as WM.

At start, a MAIN function is supposed to be run. Its solely purpose is to initialize the WM structure; after that, it releases program control to HOLARCHY_GEN.

All computation consists in recursive calls to the function HOLARCHY_GEN. This function takes WM in input, eventually makes some processing on it (thus changing its state) and returns it back to itself. Hence, it is noteworthy that no global variable is used: all data needed for computational purposes is encoded within WM structure.

The other sub-routines are:

- KBS_CALL: which manages interaction with the KB structure;
- UPDATE_WM: which can be considered as the state transition function.

The recursive calls modifies the shape to the holarchy over time: we can think of it as a holarchy forming process that commences at the root and develops from it, growing along a branch, then, after reaching the leaf, rolling back to the last bifurcation node and taking the other branch, going on like that until holarchy is completely rolled in again at the root level.

Here it follows the whole algorithm in pseudo-code notation:

Function MAIN(input, seed): struct

```
Def struct WM                    #    Define WM
Init(&WM, input, seed)           #    Initialize WM
WM = HOLARCHY_GEN(WM)            #    Initial call to recursive structure

Return WM
```

Function HOLARCHY_GEN(WM: struct): struct

```
holonic_words = KBS_CALL(WM)                    # call to system KB structure

FOR j from 1 to length(Holonic_words)     # span through found holons
        WM = UPDATE_WM(WM, holonic_words(j))      # change WM state
        WM = HOLARCHY_GEN(WM)                     # recursive call
END

Return WM
```

Function KBS_CALL (WM:struct): list

```
WM.tmp = Lookup(WM.tmp_ parent) # lookup the KB structure (with a key)
                                        # to find the new component values of
                                  #the holonic rule
WM.tmp = Tokenize(WM.tmp)          # obtain a list of tokens
WM.tmp = mask(WM.tmp, WM.check) # consider only non-checked results

Return WM.tmp
```

```
Function UPDATE_WM(WM, holonic_word): struct
WM.tmp = find(WM.tokens, holonic_word) # find where holonic_word occurs

IF(pos)                                          # update WM state
        Check_out(WM.tokens.check, WM.tmp)       # check the holonic word

#>>>>>>>>>>>>>>>>>>perform holon-related computation>>>>>>>>>>>>>>>>#
        WM.parse_struct = Process(WM.parse_struct, holonic_word)
 #<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<#

        WM.parent = holonic_word # update current holonic word
END

Return WM
```

4.1.4 Example 1 - Step by step processing

In order to have a more precise idea of what happens during computation of the holonic generation algorithm a step by step analysis is devised. As first steps, WM structure is defined and then initialized according to the arguments passed through the MAIN() function.

WM is composed of some basic variables encoding the state of the computation. Namely they are:

- WM.tokens: to store the list of holonic words received in input;
- WM.tokens.check: to store the list of tokens which have been already processed;
- WM.tmp: acting as a buffer area where storing temporary results;
- WM.parent: to keep trace of the holonic word currently being processed;
- WM.parse_struct: to store the parsing structure at run-time.

WM is then initialized: WM.tokens is initialized to the tokenized input string, WM.Parent is initialized with the seed passed as MAIN argument, WM.parse_struct is set to the right part of the holonic rule in lexical order corresponding to the given seed. All checks are set False, except for the one corresponding to the seed, which is set True.

After the INIT() call, WM represents the initial state of the algorithm. For example, a call to MAIN like that:

MAIN((to be or not to be that is the question), (or))

Produces the structure in Table 4.2.

Table 4.2. Exemplar starting state for the holonic generation algorithm.

WM	tokens		[to]	[be]	[or]	[not]	[to]	[be]	[that]	[is]	[the]	[question]
		check	F	F	T	F	F	F	F	F	F	F
		index	0	1	2	3	4	5	6	7	8	9
	tmp		[]									
	parent		[or]									
	parse_struct		[to or not]									

Note that the state of computation can be described univocally by WM.parse_struct and WP.parent. Concisely, the initially state can be expressed using a unique notation by bolding the checked holonic words and underlining the parent node that way:

[to **or** not]

which corresponds to:

$[[]_{to}\ []_{not}]_{\underline{\mathbf{or}}}$

For readability purposes, we prefer, in this context, the former notation.

Now, let us proceed with HOLARCHY_GEN() call. The first line of code is

holonic_words = KBS_CALL(WM)

The subroutine KBS_CALL() lookups WM.parent within the KB structure hashtable. Since WM.parent is equal to [or] holonic word, this key retrieves from the KB structure the value [to not or].

Then, a mask operation is run on the tokenized text [[to][not][or]] against WM.check to find out only non-checked tokens. At this stage only [or] token is checked, hence the list of non-checked tokens [[to][not]] is returned back to the calling HOLARCHY_GEN() function.

For each token in WM.tokens list, a holarchy branch will be generated. In other words, each holonic word will be the new seed of a holarchy generation process.

Let us start with the first seed corresponding to the holonic word [to]. The UPDATE_WM() function is called. Now it comes the task of identifying the position of the holonic word [to] inside the input. This is a reverse engineering task that can be handled hypothesising WM.tokens to be an associative array and **find**() a primitive function performing such a task. In this case the token [to] is found at position 0 and 4 in WM.tokens. These values are equally distant with respect to the seed [**or**] so it turn clear how the choice between the two is a matter of 'dice rolling'. We will consider this ambiguous case further in order not to make description too unwieldy. For the sake of simplicity, we suppose to take always, the leftmost value, hence [to] at position 0. Token [to] is then checked: [to] → [**to**]

Now it comes the turn of updating WM.parse_struct. The following line of code:

WM.parse_struct = **Process(**[to **or** not]**,** [**to**]**)**

accounts for some process made at the level of the current holonic word. In this very simple case, the following transition occurs on WM.parse_struct value (bold is used to indicate checked tokens):

[to **or** not] → [**to** be **or** not]

which is equivalent to write:

$[[]_{to}\ []_{not}]_{\underline{\mathbf{or}}} \rightarrow [[[]_{be}]_{\underline{\mathbf{to}}}\ []_{not}]_{\mathbf{or}}$

What happens is that [**to**] is responsible for inserting its holonic rule inside the parsing structure according to the lexical order specified in the KB structure. [**to**] is then set as the new top of stack seed for the holarchy generation process:

WM.parent = [[**to**] [**or**]]

Now that state transition is over, a recursive call is made to function HOLARCHY_GEN() to keep up with the overall process. The list of states encountered during all computation is then:

1. [to **or** not]
2. [**to** be **or** not]
3. [**to be or** not]
4. [**to be or not** to]
5. [**to be or not to** be]
6. [**to be or not to be**]

Note that, according to the given KB structure, when the initial seed corresponds to [is], the holarchy reaches its maximum extent since it spans through the whole input phrase according to the following states:

1. [that **is** the]
2. [or **that is** the]
3. [to **or** not **that is** the]
4. [**to** be **or** not **that is** the]
5. [**to be or** not **that is** the]
6. [**to be or not** to **that is** the]
7. [**to be or not to** be **that is** the]
8. [**to be or not to be that is** the]
9. [**to be or not to be that is the** question]
10. [**to be or not to be that is the question**]

It is noteworthy that from steps 3 to 8 the computation is the same as for the holarchy generation driven by the [or] holonic word.

4.1.4.1 Handling ambiguous (polysemous) configuration

Ambiguity is a very undesirable property of computational systems. In the case of holarchy generation process, the problems raised by ambiguous configurations can be well explained by means of an example.

Consider the Chomsky example presented in the previous chapter analyzing the phrase ‘the man took the book’. Suppose the algorithm to be in this state:

$$[[]_{NP}\ []_{VP}]_{S}$$

Now, NP can drive two possible productions, namely:

$$[]_{NP} \rightarrow [\text{the man}]_{NP}$$

$$[]_{NP} \rightarrow [\text{the book}]_{NP}$$

Which one to choose? Of course, this question cannot be answered with the solely information of the KB structure. The problem has a random solution from the point of view of the computational model unless some decision strategy is adopted (e.g., based on statistical properties derived from a corpus).

The option of the decision strategy equals to having some voting mechanism implemented. From a programming point of view this can be done considering $[[]_{NP}\ []_{VP}]_S$ state in the program flow as a broker/coach point where two distinct threads split from the parent one. When the threads leave control back to the parent thread, some decision mechanism based on the processed data will select the winner.

Another possibility is duplicating the parent thread through a sort of cloning mechanism: in this case, the entire holarchy is cloned as many times as there are polysemous rules to apply.

In both cases what actually happens is a call to a **fork()** function (Figure 4.2). The difference is that, in the case of the broker/coach configuration the fork is limited to the process of the caller granule - fork($[[]_{NP}]$) - and the decision point is inside the holarchy at the level of that granule; while in the second configuration, the fork is applied to the process characterizing the entire holarchy - fork($[[]_{NP}\ []_{VP}]_S$) - and the decision point is outside the holarchy thus have to be defined by an external process.

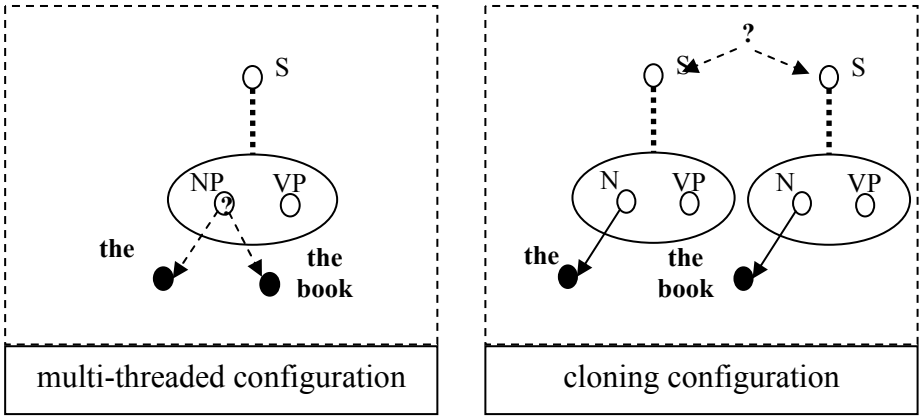

Figure 4.2: Two possible fork configurations to solve holarchy ambiguity.

4.1.5 Example 2 - Handling more complex cases

Once endowed with the ability to handle polysemy, holonic generation algorithm is suitable to manage more complex cases than example 1. It is interesting to point out that 'complex' in this context corresponds simply to assuming KB structure to be richer as for the number and form of holonic rules. In fact, thanks to the proposed holonic modelling technique, complexity is shifted from the computational level (algorithm) to the KR level (holarchy). The algorithm is designed once for all to work well independently of the holarchy extension, hence, what it really matters in computational terms, is the number of rules and decision points the algorithm has to compute.

Because of previous observation, the challenge is in building a holarchy which could represent system knowledge in a more semantic-full way. A proposal is depicted in Figure 4.3. Here we note that we make use of new granules that semantically correspond to abstract concepts like sentence [S], noun [N], subject [Subj] and so on. Once again, we are not concerned with computational linguistic aspects, nor we pretend our representation being

correct in pure syntactical terms. What is important to highlight is indeed that, by enriching the holonic rules with more sophisticated concept granules, the holarchy generation problem remains the same at the computational level. This is of a great interest, since it allows model engineer to focus on KR rather then to implementation issues.

With particular reference to the holarchy in Figure 4.3, the holonic words corresponding to the input text have been purposely considered as primitive holons, i.e., not further decomposable, considering the enrichment rules to be a matter of (meta-) symbols representing abstract concepts.

Example 2 is highly polysemous due to holonic word [S] that has different productions:

$[]_S \rightarrow [[]_{Subj}$ is $[]_{NP}]_S$
$[]_S \rightarrow [[]_S$ or $[]_S]_S$
$[]_S \rightarrow [$to $[]_V]_S$
$[]_S \rightarrow [$not $[]_S]_S$

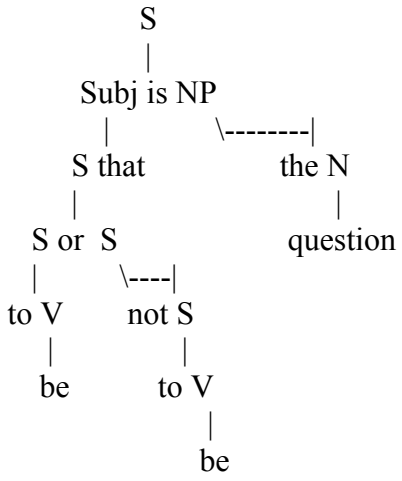

Figure 4.3: Holarchy used for the proposed example 2.

4.2 Automated Holarchy Extraction from Data

In this section, we exploit HGHM as a means for extracting holonic rules from measurement signals. Signals can be of any kind, physical or digital; the only requirement is the possibility to gather signal time series in order to have a sufficient number of observations. This makes the proposed technique sufficiently general for a wide variety of measurement settings.

4.2.1 Example 3 - Temperature time-series analysis

As for example 3 is concerned, an experiment employing the maximum temperature time-series collected in the two Italian cities of Taranto and Rome (from November 25th to January 11th 2010) with a sampling rate of three samples per day is reported. This measurement signal is set as input to the holonic IF THEN rules extraction process described in Chapter 3.

4.2.1.1 Step 1 - Hypothesisation

As first step, a pre-processing activity is made on available data in order to have multiple sources for the hypothesization task. From the sampled data, two other signals are extracted: the 1-day moving average and the 7-day moving average. The three signals are then subjected to the following cross-conditional hypotheses:

- Hypothesis 1 - $\varphi_1(h_k)$: if the sampled data at time *k* is *greater* than the 1-day moving average at the same sampling time;
- Hypothesis 2 - $\varphi_2(h_k)$: if the sampled data at time *k* is *greater* than the 7-day moving average at the same sampling time;
- Hypothesis 3 - $\varphi_3(h_k)$: if the 1-day moving average at time *k* is *greater* than the 7-day moving average at the same sampling time;

The three hypotheses generate an *n* x *m* binary matrix M where *n* is the total number of considered sampling times and m the total number of considered hypotheses. This matrix represents the dataset that feeds the computational process based on CART. Note that, for any given sampling time, hypotheses encoded in matrix M sometimes are all verified and sometimes not (Figure 4.4).

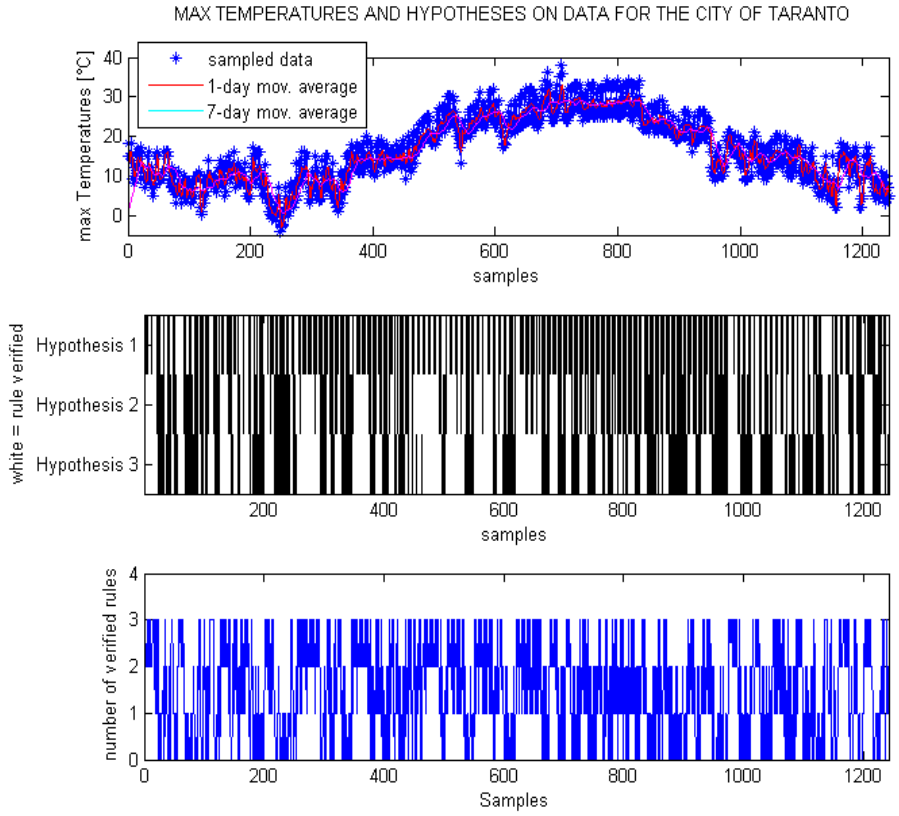

Figure 4.4: Visual representation of the hypotheses devised for the dataset of the city of Taranto.

4.2.1.2 Step 2 - Holarchy structuring

In the second processing step, in order for holonic rules to be extracted, binary matrix $M_{n,m}$ is decomposed into two subsets: $D1_{n,m-1}$ for the predictor variables (whose mutual order is not significant) and $D2_{n,1}$ for the predicted variable. Hence, *n* possible arrangements of the matrix M are obtained, one for each possible predicted variable.

In this first case, we suppose that all *n* samples in the two datasets are used to feed the rule extraction process. In the reported experiment, all available data have been used within a single run to feed the computational process. In the forth, we will consider a different setting.

After CART processing, six rules with total certainty are found: they are reported in Table 4.3. It is noteworthy that they are the same for the two considered datasets. The table lists the rules written in a formal notation as logical implication rules, then it proposes the holonic rule formalism, finally it provides the coverage of each rule with respect to the whole dataset. It is interesting to visually note that the binary activation patterns for each rule (1 if the rule is verified at the given sample, 0 otherwise) define different signatures between the two considered datasets in the rule vector space (see Figure 4.5 and 4.6). This consideration allows for future studies aimed at classifying meteorological patterns on the base of the proposed technique.

An interesting aspect is that the obtained activation patterns define a complete partition of the original dataset, i.e. only one rule is activated at each sampling time. This means that the two datasets can be completely described in terms of the IF THEN hypotheses used for the experiment. Consequently, thanks to the proposed computational technique, original numerical datasets have been converted into algorithmic structures, which allows for shifting the problem of signal analysis at a logic/linguistic level. We now go more in-depth into this aspect exploring the KR provided by the found rules.

Table 4.3. List of no-uncertainty Holonic Rules found and their distribution over the dataset.

Rule Id	Rules				
	Implication Rule	***Holonic rule*** (enrichment)		***Dataset coverage***	
				Taranto	***Rome***
1	$\overline{\varphi}_2\varphi_3 \rightarrow \overline{\varphi}_1$	$[]_{\overline{\varphi}_2}[]_{\varphi_3}$	$[[]_{\overline{\varphi}_2}[]_{\varphi_3}]_{\overline{\varphi}_1}$	16.75%	20,43%
2	$\varphi_2\overline{\varphi}_3 \rightarrow \varphi_1$	$[]_{\varphi_2}[]_{\overline{\varphi}_3}$	$[[]_{\varphi_2}[]_{\overline{\varphi}_3}]_{\varphi_1}$	10,22%	13,78%
3	$\overline{\varphi}_1\overline{\varphi}_3 \rightarrow \overline{\varphi}_2$	$[]_{\overline{\varphi}_1}[]_{\overline{\varphi}_3}$	$[[]_{\overline{\varphi}_1}[]_{\overline{\varphi}_3}]_{\overline{\varphi}_2}$	28,26%	30,02%
4	$\varphi_1\varphi_3 \rightarrow \varphi_2$	$[]_{\varphi_1}[]_{\varphi_3}$	$[[]_{\varphi_1}[]_{\varphi_3}]_{\varphi_2}$	22,06%	19,93%
5	$\overline{\varphi}_1\varphi_2 \rightarrow \overline{\varphi}_3$	$[]_{\overline{\varphi}_1}[]_{\varphi_2}$	$[[]_{\overline{\varphi}_1}[]_{\varphi_2}]_{\overline{\varphi}_3}$	8,13%	3,94%
6	$\varphi_1\overline{\varphi}_2 \rightarrow \varphi_3$	$[]_{\varphi_1}[]_{\overline{\varphi}_2}$	$[[]_{\varphi_1}[]_{\overline{\varphi}_2}]_{\varphi_3}$	14,58%	11,90%
				100%	**100%**

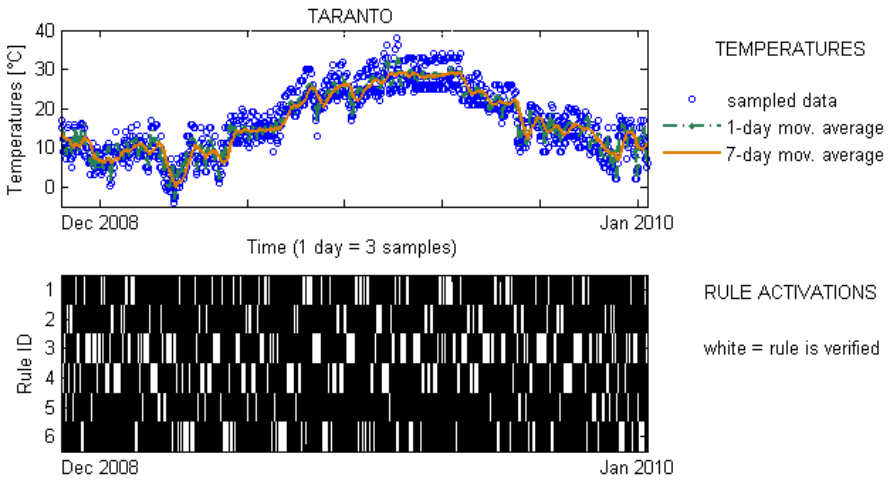

Figure 4.5: Max temperatures (diagram above) and holonic rule verification patterns (diagram below) extracted from the dataset of the city of Taranto. In the rule diagram each row accounts for the verification of the corresponding rule reported in Table 4.3.

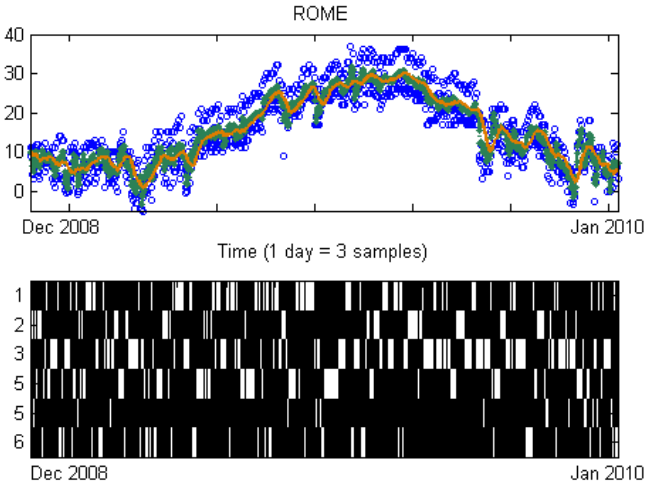

Figure 4.6: Max temperatures (diagram above) and holonic rule verification patterns (diagram below) extracted from the dataset of the city of Rome. In the rule diagram each row accounts for the verification of the corresponding rule reported in Table 4.3.

4.2.1.3 Holarchy extracted from data

Let us consider the first two holonic rules of Table 4.3. Assuming T for 'true' and F for 'false' the two rules can be rewritten this way:

- Rule 1: $[F]_{\varphi_2}[T]_{\varphi_3} \rightarrow [[[F]_{\varphi_2}[T]_{\varphi_3}]_F]_{\varphi_1}$
- Rule 2: $[T]_{\varphi_2}[F]_{\varphi_3} \rightarrow [[[T]_{\varphi_2}[F]_{\varphi_3}]_T]_{\varphi_1}$

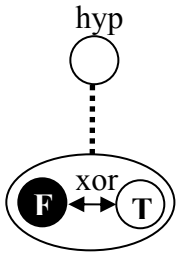

Figure 4.7: Holarchy representing a given hypothesis in Boolean logic.

Each hypothesis has been granulated into its basic components according to its semantic values, namely T or F (Figure 4.7). It is fair to consider these two values as atomic ones, so they do represent non-decomposable holons, i.e., elementary granules of information. In Boolean logic, any given hypothesis must be true or false, i.e., hypotheses can be considered as super-holons obtained through abstraction of the two primitive holons T and F related by XOR logical relationship.

Since Rule 1 and Rule 2 abstract the same granule, they can be viewed as parts of a bigger abstraction rule. The so obtained enrichment rules realizes a very particular holarchy where, starting from the bottom, each two consecutive levels the first concerns semantics (ontology layer) and the second concerns the variables hosting the semantic values (lexical layer).

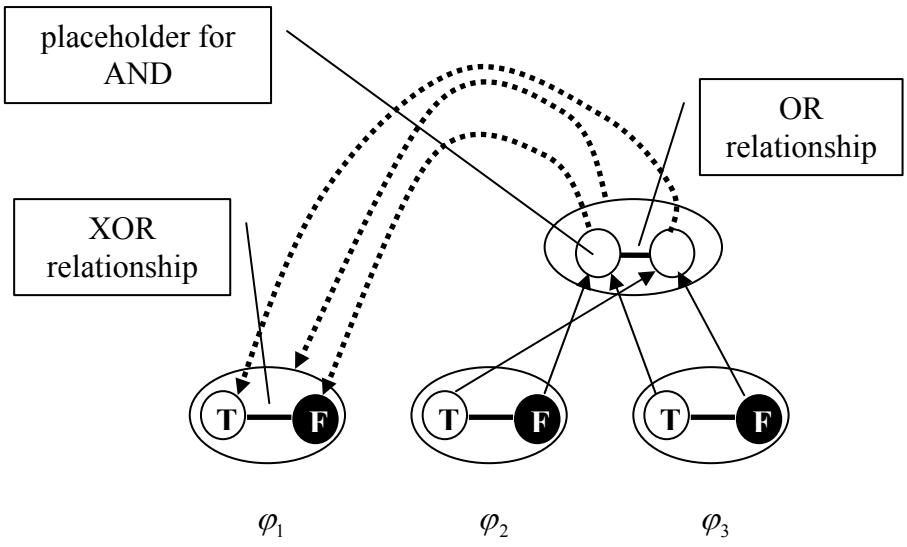

Figure 4.8: Holarchy representing the first two rules extracted from example 3. Arrows point in bottom-up direction.

Note that similar considerations can be drawn for the couples of rules (3, 4) and (5, 6). The holarchy representing the knowledge extracted from data is then highly interleaved since each granule corresponding to a hypothesis can be an abstraction of other hypotheses or it can be a part itself as an enrichment of another hypothesis. In fact, KR has the following six possible states:

$$\overline{\varphi}_1\overline{\varphi}_2\varphi_3,\ \varphi_1\varphi_2\overline{\varphi}_3,\ \overline{\varphi}_1\overline{\varphi}_2\overline{\varphi}_3,\ \varphi_1\varphi_2\varphi_3,\ \overline{\varphi}_1\varphi_2\overline{\varphi}_3,\ \varphi_1\overline{\varphi}_2\varphi_3$$

4.2.1.4 Managing uncertainty in holonic rules: some observations

The six found holonic rules of example 3 have been extracted from the whole dataset represented by the matrix M of binary observed hypotheses. In order to be sure that there were no bias in the experimental setting, hundreds of independent runs have been performed each one obtained through a random permutation of the columns of matrix M related to predictor variables.

Each run confirmed the same set of six holonic rules with no uncertainty (i.e., having a zero or one as leaf node of the CART). At the same time, six uncertain rules have always been found (i.e., having a leaf node equal to a number comprised in (0, 1)). They can be interpreted as holonic rules with a given amount of uncertainty attached.

Uncertainty could be computed by means of some function in the space of True and False hypothesis using FL (Figure 4.9). For example by means of I-type FL, we could imagine uncertainty is maximum when the leaf value is 0.5 while, as long as it approaches to 0=False or 1=True, it decreases accordingly. Alternatively, by employing II-type FL (Mendel & John, 2002) (Hagras, 2007) (Hagras et al., 2007), a so-called *Footprint of Uncertainty* can be sketched by contouring the two membership functions defining the True and False fuzzy predicates. The values of the footprint of uncertainty area can be assigned differently depending on the type of the applied uncertainty modelling technique (Wagner & Hagras, 2010).

The six uncertain rules along with their computed I-type FL uncertainty values are reported in Table 4.4.

Table 4.4. Uncertain implication rules of example 3.

Rule Id	Implication Rule	Leaf node value [0, 1]	Uncertainty value [0, 1]
7	$\overline{\varphi}_2\overline{\varphi}_3 \rightarrow \overline{\varphi}_1$	0.22345	0.4469
8	$\varphi_2\varphi_3 \rightarrow \varphi_1$	0.6022	0.7956
9	$\overline{\varphi}_1\varphi_3 \rightarrow \overline{\varphi}_2$	0.4653	0.9306
10	$\varphi_1\overline{\varphi}_3 \rightarrow \varphi_2$	0.55702	0.88596
11	$\overline{\varphi}_1\overline{\varphi}_2 \rightarrow \overline{\varphi}_3$	0.37209	0.74418
12	$\varphi_1\varphi_2 \rightarrow \varphi_3$	0.68329	0.63342

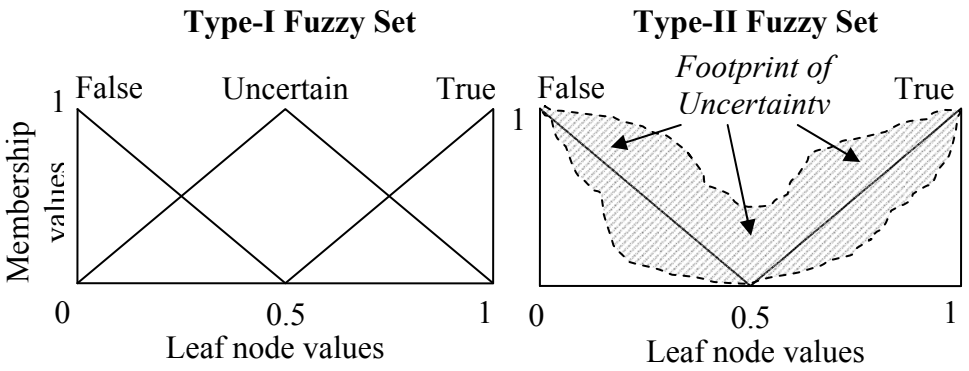

Figure 4.9: possible type-I (left) and type-II (right) fuzzy description of uncertainty in holonic rules.

4.2.1.5 Managing imprecision in holonic rules: some results

Finally, we verify another aspect related to holonic rule extraction in example 3. We could easily imagine that: the less the window size of the dataset, the more imprecise the set of holonic rules retrieved. This behaviour has actually been found in data. In particular, we used as a measure of imprecision the number of samples covered by more than one certain holonic rule. In fact, 'certain' means 'certain with respect to the observed dataset': if the dataset is a subset of the original one, some more specific rules that were not so general to be retrieved from the whole dataset may now come out. Table 4.5 summarizes obtained results.

Table 4.5 Statistics collected from example 1 regarding multiple rules distribution per sample.

considered training window length in % of the total dataset	Total number of rules found after knowledge extraction	% of the whole dataset having N rules per sample			
		N=1	N=2	N=3	N=4
Whole dataset	6	100%	0%	0%	0%
50%	6	100%	0%	0%	0%
7%	6	100%	0%	0%	0%
5%	8	22.7%	77.3%	0%	0%
4%	10	0%	49.7%	50.3%	0%
2%	11	0%	31.3%	68.7%	0%
1.5%	12	0%	0%	100%	0%
1.2%	13	0%	0%	77.9%	22.1%
0.8%	9	0%	18.4%	53.4%	28.2%
0.4%	6	0%	0%	100%	0%
0.15%	6	0%	0%	100%	0%
1/whole dataset	6	0%	0%	100%	0%

4.2.2 Example 4 – Making predictions

The previous setting of example 3 can be reused for making prediction with minor modifications. Given a signal *s* along with a number of samples available until time *k*, make a prediction on *s* means finding the value of $s_{(k+m)}$ for some $m > 0$. In general, the realm of prediction is accompanied with uncertainty: effective predictor has to be paired with some 'hit' function aiming at scoring the quality of prediction.

With reference to example 3, we introduce a simple trick at the hypothesis generation level. In particular, we reformulate the first hypothesis this way:

- Hypothesis 1 - $\pi_1(h_{k+m})$: if the sampled data at time $k+m$ is *greater* than the 1-day moving average at time *k*;

Hence, after the knowledge extraction task, if $\pi_1(h_{k+m})$ is found being logically implied as a *consequence* of the other hypotheses, then it realizes a prediction rule with an uncertainty reckoned from the leaf node value as shown above. As expected, with this new setting, no more certain rules are found but only uncertain ones.

In particular, for m=3, 12 uncertain rules have been found, four of them having $\pi_1(h_{k+m})$ as implied variable. Table 4.6 summarizes obtained results.

Table 4.6: Implication rules accounting for prediction in temperature time-series.

Rule Id	Implication Rule	Leaf node value [0, 1]	Uncertainty value [0, 1]
1	$\overline{\varphi}_2\overline{\varphi}_3 \rightarrow \overline{\varphi}_1$	0.4368	0.8736
2	$\varphi_2\overline{\varphi}_3 \rightarrow \overline{\varphi}_1$	0.10628	0.21256
3	$\overline{\varphi}_2\varphi_3 \rightarrow \varphi_1$	0,84921	0.30158
4	$\varphi_2\varphi_3 \rightarrow \varphi_1$	0.52308	0.95384

4.2.2.1 Applying prediction to other application domains: stock market

Of course, the proposed prediction technique can be applied to any kind of application domain, for example to stock market prediction. Stock market is in fact considered as a typical example of stochastic model (Wang et al., 2007). Consequently, the automated extraction of rules from time-series data is of great interest for stock price forecasting programs.

Here it follows the result of three hypotheses similar to that of the previous temperature prediction model applied to the case of Dow Jones Industrial (DJI) average index on a dataset comprising the stock prices from January 2005 to September 2010.

The three employed hypotheses are the following:

- Hypothesis 1 - $\varphi_1(h_{k+m})$: if the 5-day moving average at time *k*+*m* is *greater* than the 10-day moving average at *k* sampling time;
- Hypothesis 2 - $\varphi_2(h_k)$: if the 10-day moving average at time *k* is *greater* than the 20-day moving average at the same sampling time;
- Hypothesis 3 - $\varphi_3(h_k)$: if the 20-day moving average at time *k* is *greater* than the 30-day moving average at the same sampling time;

The next Table summarizes the found results obtained with different values of m. In general, the higher the value of m, the more the proximity of prediction to pure chance (i.e., T = 0.5 and F = 0.5).

Table 4.7. $\varphi_1(h_{k+m})$ values accounting for prediction of DJ index trend at k+m sample.

Predictors→					
$\varphi_1(h_{k+m})$	***m***	$\overline{\varphi}_2\overline{\varphi}_3$	$\varphi_2\overline{\varphi}_3$	$\overline{\varphi}_2\varphi_3$	$\varphi_2\varphi_3$
Leaf node value [0, 1]	1	0.40753	0.50935	0.62179	0.69681
	5	0,43197	0.56808	0,66667	0,66607
	10	0,5068	0,56808	0,66026	0,67921
	15	0,48299	0.62441	0.64103	0,6962
	30	0,48966	0.53774	0.59732	0.68545

4.3 Using HGHM for Complex System Management Design

Apart from structure parsing, signal analysis and signal prediction, HGHM can be also employed to support system engineer in devising an alternative solution to MAS-driven design approaches.

To supply a deeper comprehension of HGHM-based design methodology, three different test cases have been considered, namely:

1. Electric Power Distribution Management;
2. Distributed air quality monitoring system;
3. Software modelling.

The first case is aimed at accompanying the reader through a step-by-step description process, to gain insight into the technicalities of the proposed methodology. The second test case is taken from an already published work (Calabrese et al., 2010) and is reported with the only aim of corroborating the wide applicability of HGHM-based design. Finally, the last application case shows the potential impact of our proposal to Software Engineering in terms of modularity and scalability of HGHM-based software.

4.3.1 Electric Power Distribution Management

Electric power distribution network represents an evident example of a complex system arranged as a geographical network of electric producers and consumers. Basically, the aim of the system is to guarantee, at any given time, that the electric load requested by all customers (being they domestic or industrial) is supplied by, at least, an equal amount of electric power produced by electric plants connected to the network.

A lot of equipments are requested for the process to be satisfied and can be encountered during the process of electric flow from the power plant to customer's meter socket, namely: extra high (i.e., 800kV), high (i.e., 220kV) or medium (i.e., 33kV) voltage transmission lines (generally aereal), transmission towers, transformer stations and substations, low voltage distribution lines, relais, inverters and other electrical stuff.

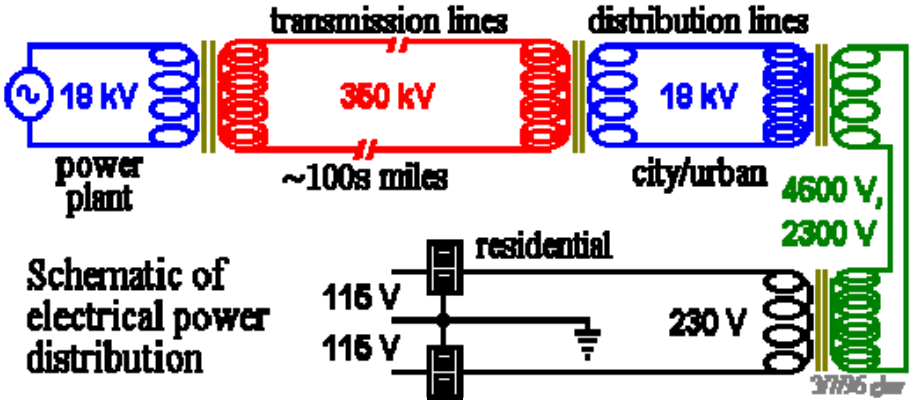

Figure 4.10: Typical electrical energy transformation steps encountered from power plant to residential load (in US). Image taken from the Internet.

4.3.1.1 Architectural aspects

Wide electric networks covering the needs of entire states and countries are all hierarchic since at least two levels can be identified:

1. Transmission level: it accounts for the bulk transmission of electrical energy from generating power plants to substations located near to population centres. This level can be further subdivided into two sub-levels: one for extra-high voltage transmission and the other for high voltage transmission. In both cases, a three phase alternating current is used to cover very long distances (hundreds or miles) minimizing the energy loss in transmission. Generally, interconnections at this level are redundant hence having the form of a grid so that electric energy has more possible routes to flow if some failures occur.
2. Distribution level**:** it covers more limited areas (urban or rural). It is made of local wiring between high voltage substations and customers and can be schematized as a bus with different kind of loads attached. It is noteworthy that small energy production sources (such as city power plants, solar or wind farms) are generally connected at this level.

Architecturally, a holarchy-based representation of the electric network can be hence drawn with simple assumptions:

1. A **holon** is any entity between two transformer stations;
2. A **primitive holon** is any entity between a transformer station and the ground;
3. Any transformer station represents the **interface** between two or more holons;
4. An overarching layer called "Electric Network" is introduced as **super-holon** subsuming the entire holarchy. This holon is connected to subsumed holons of the lower adjacent level.

It is easy to notice that holons accounting for transmission or distribution lines have the role of hubs for primitive holons at their own level. This reflects the star network topology, which is typical of such wide geographical complex systems.

4.3.1.2 Information processing aspects

At the information processing level, the system-engineering task consists in defining the archetype holonic component. This corresponds to finding an appropriate algorithm, as shown in the previous chapter, that recursively updates the amount of information that flows through the holarchy at any time instant. For this to be achieved, the first step defines which elements the archetype holon is composed of. Of course, the here provided archetype is a toy one, since it is aimed only at better explaining the proposed methodology. A very detailed and functional description of the archetype holon for electric power distribution management deserves a specific work, which is further beyond the scope of the thesis.

Consider the archetype holon H in terms of a class made of the following variables and methods:

- H.primitive: it is a Boolean value indicating whether H is a primitive holon or not;
- H.power: it represents the current electric power measured at the primary circuit;
- H.powercost: it is a value representing the cost of producing or consuming one Kwh of electric energy;
- H.tolerance: it is a value to use in the system management strategy described forth;
- H.forecast(k): it forecast H.power values in the future k instants;
- H.alter(amount): alter energy consumption/production (if allowed by H) of a given amount.

The holarchy building mechanism is entirely described by an archetype enrichment rule to apply to non-primitive holons by means of the following formula:

$$\text{H.power} = \sum_{i=1}^{N} \text{H}^{i}.\text{power}$$

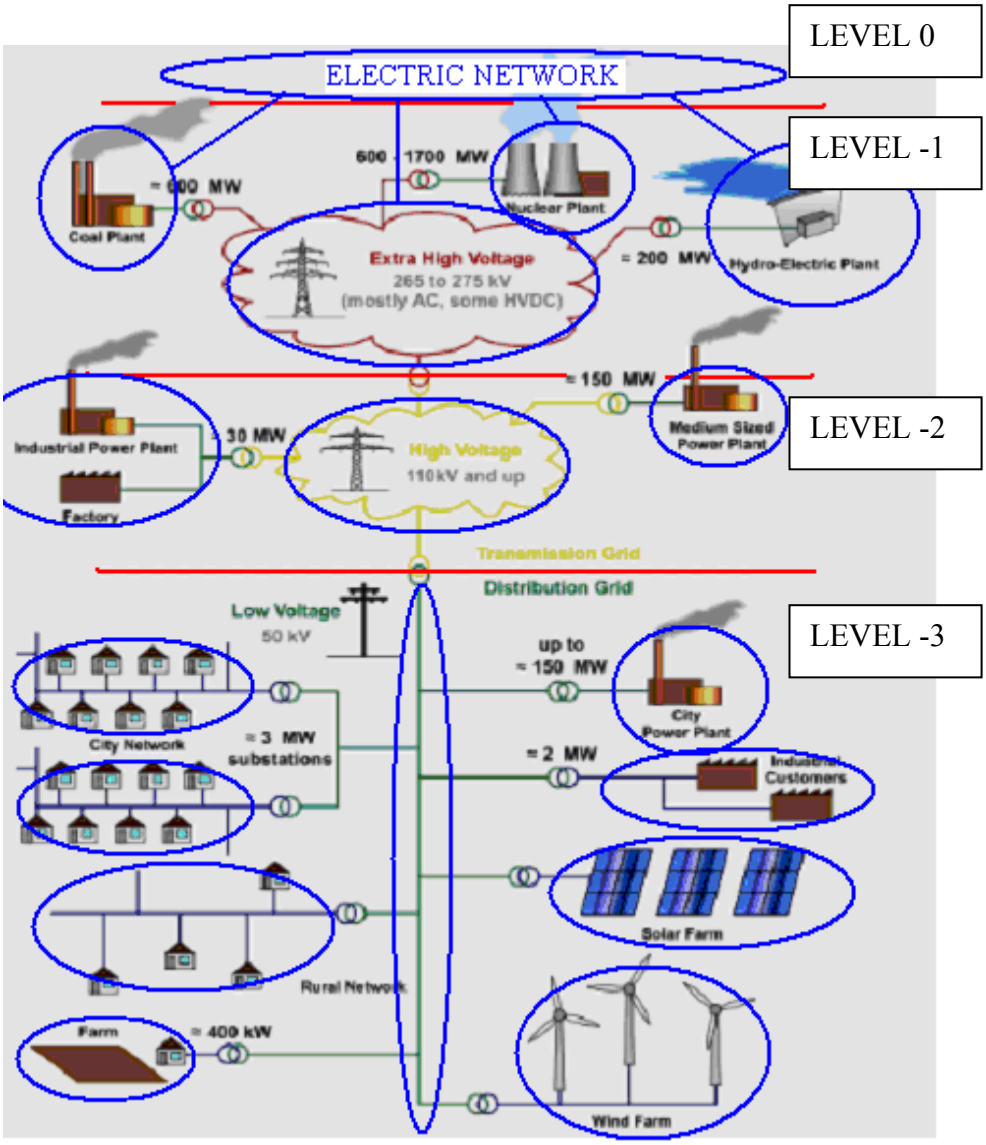

Figure 4.11: Electric power distribution network viewed in holarchical terms. Background image is taken from the Internet.

where H^i is a holon connected to H at the same level of H or at the lower adjacent level. The meaning of the enrichment rule is straightforward: at any time instant, the electric power measured in any point of the network is equal to the algebraic sum of the load and the produced power. When the sum is negative then a drop in current affects the area and a blackout occurs. Of course, the system has to be engineered to avoid any cause of power failure.

The description of H is now almost complete. The only thing missing is the algorithm for characterizing holon behaviour. To this end, we start by hypothesizing a very simple strategy that can be realized repeatedly over time according to these steps:

1. IF H.power > 0 THEN
2. energy_gap = H.power-H.tolerance
3. H.restore(energy_gap)
4. END

The key point of the holon management program is the restore() function whose aim is to adjust dynamically a given tolerance threshold to avoid the risk of sudden increase in load request. Actually, this mechanism is a target-following one, with the target being the given threshold.

The restore() function can be engineered in several ways. The simplest one is to implement a function like that:

Function restore(energy_gap)

1. IF energy_gap > 0 THEN
2. decrease energy production in subsumed holarchy calling H.alter() functions
3. ELSE
4. increase energy production in subsumed holarchy through H.alter() functions
5. END

It is apparent that the proposed strategy is the more robust to variability in load request or local failures as higher the tolerance threshold is. This results in a very stiff and expensive risk-avoidance strategy.

The same restore() function can be however made more adaptable to network dynamics by introducing more knowledge in the management process. This can be done employing the H.forecats() function and building a more developed strategy on top of it. For example, if the holon forecasts that a certain increase in energy request will affect the network in the next few hours, particular energy increasing production patterns can be activated among energy producers in order to minimize the cost of the extra-amount of energy to be produced.

4.3.1.3 System Reengineering: adding new levels

Thanks to the hierarchically nested structure of the holarchy it is possible to add a new granularity level to the network description, thus allowing for energy balancing strategy also at lower levels of the networks. This can be useful since, in general, the stock of extra-energy produced at the highest levels of the holarchy (nuclear plants, coal plants etc…) is more expensive and less manageable than the stocks produced at lowest levels. The algorithm that determines the holon behaviour can be even maintained the same, thus accounting for a modular structure.

In Figure 4.12, the lowest level of the holarchy (level -3) is further enriched by adding a new level The so defined new holons supervise single house energy load or city distribution network. It is interesting that, this level can be even further decomposed thus having very small-scale energy management systems like Demand-Side Management systems, which are progressively attracting the interest of both industrial and academic applications (Calabrese et al., 2007).

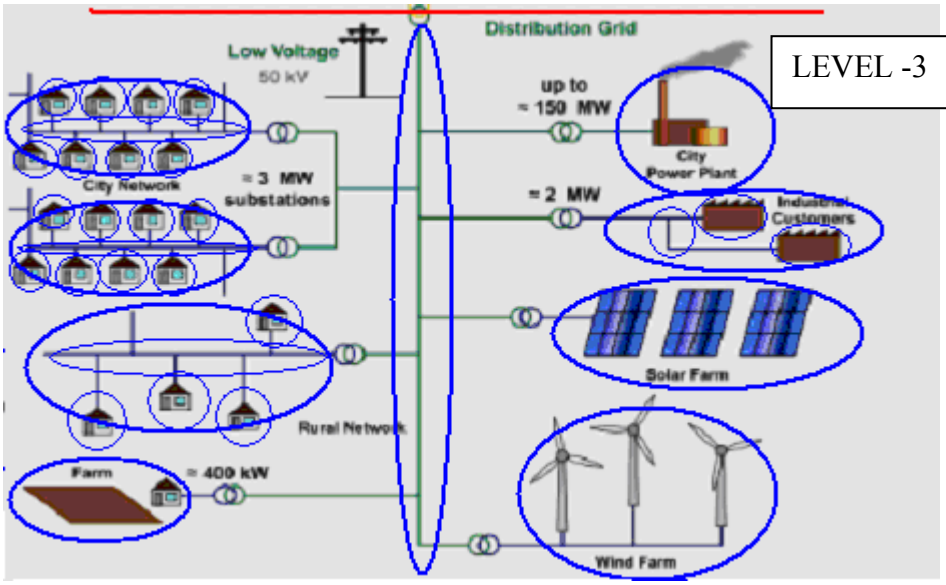

Figure 4.12: Previous holarchy at level -3 is enriched at the next lower level.

4.3.2 Distributed air quality monitoring system

The first studies about the air pollution, undertaken by industry, started in the 1950s. In various countries, the obtained results allowed the introduction of specific laws to protect the environment and consequently the human health. The air quality analysis was extended to indoor environments later, when, in the 1970s, there were some cases of pulmonary diseases into air-conditioned buildings. The air quality monitoring for indoor environments is becoming an interesting issue in the most economically developed countries. Indeed, in these countries people show strong tendency to spend their time in indoor environments. According to (Bocchio and Masoero, 1992), already in the early 90s, people spent until the 90% of their time in indoor environments, 30-40% of this time is spent in working environments. Probably these figures have not changed significantly since then, as most people can experience. From these data, it appears clear how the necessity of new monitoring systems designed to work in various indoor and outdoor environments is extremely relevant.

The latest improvements both in the field of sensors and in ICT technology are opening the way to the development of innovative pervasive distributed sensor networks composed of many low-cost nodes. In (Di Lecce et al. 2010) the nodes are composed of two main modules as shown in Fig. 4.13. The sensors module is made up of a set of analogical sensors whose number and type vary according to the application. The processing module is based on a programmable unit handling various aspects of the acquisition and data management process (sampling, compressing, sending and, possibly, setting actuators). A key element of these nodes is their ability to send data through a network using various connections (such as wired and wireless network, GPRS and UMTS) according to the specific application. These processing units are characterized by reduced size and good computational power allowing them to execute complex tasks.

Typical examples of indoor air quality monitoring applications are big civilian buildings and industrial environments. These environments are composed of various areas characterized by strong micro-climatic heterogeneities (i.e., different rooms, area around different machineries, etc.). In these conditions, the proposed holonic monitoring system, based on a dense sensor network, is able to analyze each micro-climatic area. On the other hand, the proposed architecture allows changing the observation scale in order to have various levels of detail about the monitored environment.

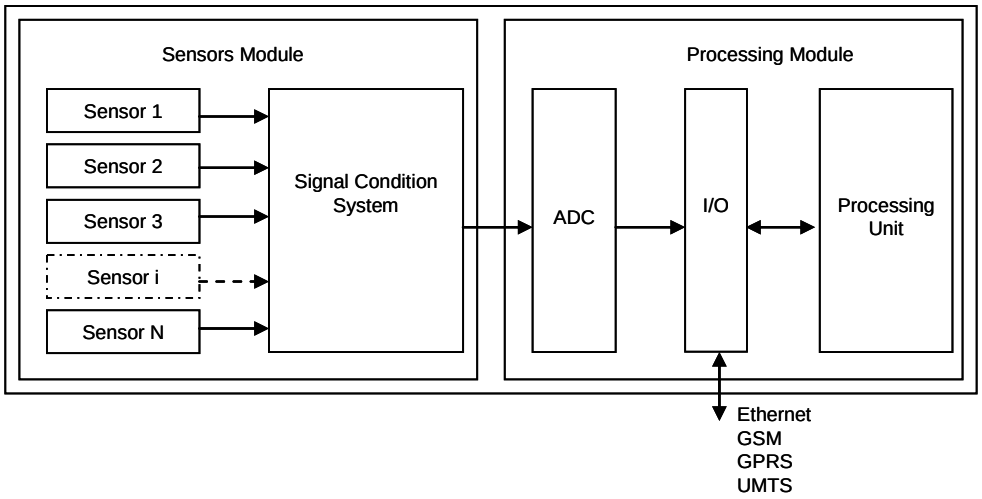

Fig. 4.13: A schematic overview of the proposed acquisition system.

4.3.2.1 Proposed architecture in more detail

One of the advantages of the HGHM-based architecture is that it is possible to add as many levels as one wants without affecting the modularity of the basic holon. Here it follows a more detailed description for each layer.

Level 1

The lowest level is composed of all the sensors in the network. Here information is local, namely, it is referred to specific monitored location. The main tasks of the holons at this level are:

- Sampling data: each holon samples data at a given sampling rate according to the specific application;
- Data validation: each holon implements various validation algorithms in order to avoid the well known problem of incomplete data series (V. Di Lecce et al. 2008) These algorithms work only on local sampled data;
- User interface: each holon at this level has a simple web based user interface. Using this interface a user can have local information about the monitored environment.

Level 2

The second level of the architecture works at a higher semantic level. Here the monitored building is considered as divided in various regions. A region is

composed of several neighbouring locations (i.e., a set of neighbouring rooms). In the proposed HGHM-based architecture, at this level there is one holon for each region. Here each holon:

- Works with several neighbouring holons standing in the previous level;
- Handles information referred to a whole region of the monitored environment. The size of this region is in inverse proportion to the heterogeneity level among the various areas of the monitored environment.

At this level, each holon has the following tasks:

- Area modelling: for each monitored region, a holon builds a model about the daily evolution of the monitored parameters. This model plays a significant role in the next task;
- Spatial validation: this is a further level of validation implemented in this system. Data sampled from various nodes are compared among them. When a significant discordance is found between two or more nodes (namely between two ore more neighbouring monitored areas), the sampled data are compared to those computed by the model. If the actual situation is compatible with the model then data are labelled with a high reliability coefficient else the reliability coefficient is reduced (proportionally to the divergence);
- Alarm management: when critical conditions are detected, the system is able to raise various levels of alarm, according to the criticality of the event. Using area modelling and spatial validation, it is possible to infer if a given situation is due to a sporadic local event or to a phenomenon that is interesting a wider area;
- User interface: at this level, user can obtain qualitative and quantitative information about the trend of the various monitored parameters. The interface shows average information about the whole region. When a local critical condition is detected the interface passes the control to the holon at the lower level in order to show local detailed data about the event under analysis.

Level 3

The third level of the HGHM architecture implements the same functions of the second layer but works at a higher abstraction level. Indeed, here the holon works on the features extracted by the holons at the second level. The area-modelling task is referred to the entire monitored structure. The alarm management function is used to handle critical events involving more than one regions of the monitored structure. Likewise, the user interface shows the same kind of information but referred to the whole monitored environment.

4.3.3 HGHM as a software modelling paradigm

The proposed HGHM resembles human reasoning approach to complex problems where hierarchical abstraction/enrichment patterns are continuously employed.

As a matter of course, HGHM is an ideal one to deal with software engineering, a typical knowledge-intensive application domain where abstraction and enrichment are required to build up ever more complex and detailed models at different logical resolutions. In recent times, there is in fact a growing need of a balance between pure abstraction (e.g., refer to (Wang, 2008)) and practical software design (Wu, 2005) (Auprasert & Limpiyakorn, 2009) due to increasing software systems complexity.

The importance of computing with granules in Software Engineering is quite a recent finding in the literature of CI (Pedrycz, 2002). HGHM, which is actually a GrC methodology, has the advantage of undertaking abstraction/enrichment modelling tasks in a natural way. Now we deepen this aspect by highlighting HGHM as a software paradigm especially for complex system modelling.

4.3.3.1 Handling data and program organization

Software programs can be imagined as a collection of data processing tasks that implement a business logic to satisfy some application-dependent user need. Data are generally structured in a database to encode business logic elements both at an abstract and at an implementation level.

This entire conceptual frame continues to hold when using HGHM-based approach to software modelling, however a major enhancement in data representation is required with respect to traditional Software Engineering approaches. Since HGHM is entirely based on the concept of HGs, also program data have to be organized accordingly. In particular, each class characterizing system description has to be re-mapped into a HG class, as shown in Figure 4.14.

This is something that can be always done since system description contemplates, at an abstract level, only two kinds of classes we can refer to as part classes and whole classes. As a result, the (hierarchical) target system is modelled as a recursive composition of HGs classes.

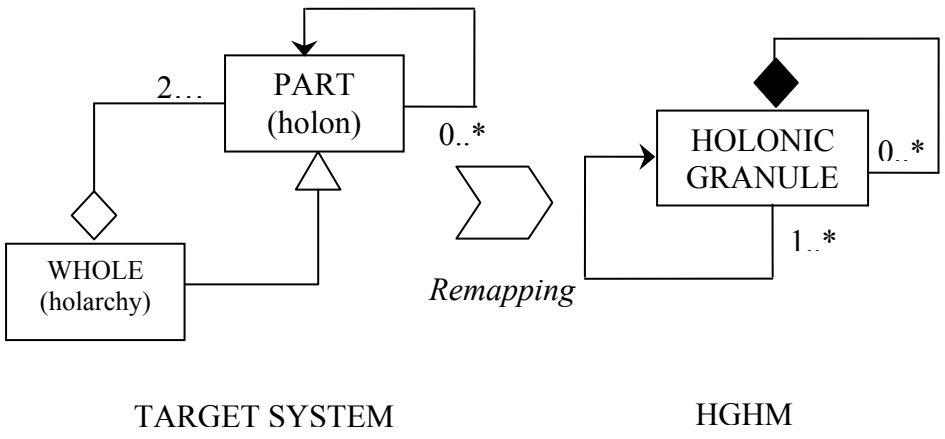

Figure 4.14: Object-oriented view of the HGHM impact on KR: target system (on the left) is represented in terms of HGs (on the right).

After transforming business logic data into HG-based KR, we can organize the elements of the HGHM into groups using UML package diagram notation (OMG, 2007b). To this end, we identify four main packages drawn in Figure 4.15:

- **Holonic factory:** this package is responsible for grouping all the archetype components called when a new holon is instantiated in the HGHM. It is made of three sub-packages, namely:
    - **Interface** for handling data input/output from sensors to actuators;
    - **Holarchy structuring** for implementing knowledge extraction functions that feed HG-based KR;
    - **Holarchy management** for holarchy control at runtime.

  Note that this sub-packages supervises the processes that characterise HGHM (I/O, structuring, management)

- **Knowledge management:** it consists of two sub-packages:
    - **Holarchy structure**
    - **Knowledge base**

  These are two DBMS working in cooperation. The Holarchy structure package supervises the hashtable representing HG-based system decomposition; the Knowledge base package stores the granulation process to be assigned to each holonic rule. This process is recalled at runtime by the holarchy granulation package to characterize the dynamics of the HGHM.

- **Holarchy granulation:** this package is responsible for the dynamics of the HGHM. It supervises granulation (hence state transition) during system program running Every object working in this package is built at runtime from the holonic factory package for the class instantiation and from the knowledge management package for importing the knowledge necessary to carry out the granulation process

- **Primitive holons:** this package includes all that functions which have to be considered primitive with reference to KR. These functions actually perform as black-box components already available 'off-the-shelf'.

The software engineer is atop of the HGHM. It is responsible for designing system KR. This process consists in the same steps we devised with the example of the bubblesort algorithm but on a greater scale. It is interesting to note that the role of the software engineer is greatly reduced when primitive holons are already at his/her disposal. This means that HGHM will resemble an assembly of parts at the KR level.

The holonic factory package can be thought as realized once for all while holarchy granulation entirely depends on the holarchy structure and the KB. This means that the true effort of the software engineer is moved to the KR level, which is a desirable property for complex systems design.

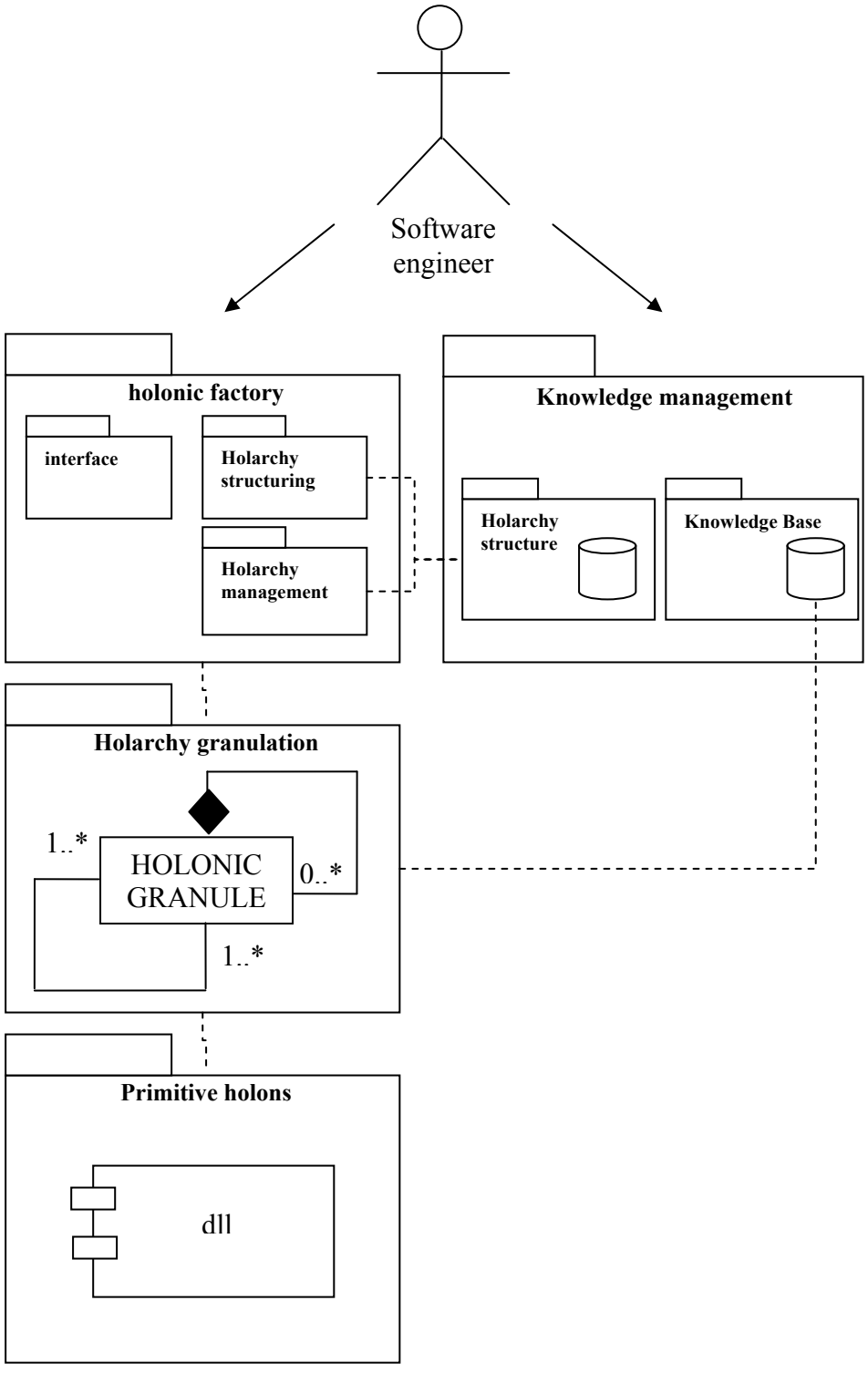

Figure 4.15: Package diagram describing HGHM in software-oriented terms.

4.3.3.2 Communication aspects

Communication aspects are less crucial in HGHM than in agent-oriented application. This because holons are, by design, similar with respect to their information processing role. Intelligence is in fact at the holarchy level and hence does not need to be communicated but is intrinsic to the information process as a whole.

Nevertheless, some considerations can be drawn. Information flow traverses the HGHM-based architecture following two opposite directions: upward and downward. The upward flow accounts for data raising towards higher-level components; the downward flow triggers commands directed towards actuators or lower layer components. Both commands and data communications are implemented, by default, through asynchronous message exchange for at least reasons of two orders:

1. since HGHM is ultimately conceived for human-centric applications, user requests can occur at any time. In this sense, the architecture must be as flexible as to balance appropriately heavy load requests with real-time constraints.
2. data require time to be processed. After processing, it can happen that the processed output is considered irrelevant with respect to the local ontological model and does not produce a data communication act to the upper layer. From this perspective, HGHM is more suited to event-driven programming, a technique that is experiencing an increasing interest in modern software engineering due to the ability of handling multi-level abstraction coding more straightforwardly than traditional programming techniques (Meyer, 2009).

Both the two points are compatible with well-known communication standards such as Agent Communication Language (ACL) and related Java-based implementations like the Java Agent DEvelopment Framework (JADE) (Bellifemine et al, 2007). Similarly, it is useful to mention that agent-sensors or agent-actuators communications based on the eXtensible Markup Language (XML) have also been proposed in recent years (Acampora & Loia, 2005) and can be applied straightforwardly in HGHM settings as well.

5. CONCLUDING REMARKS

In this thesis, an original model for information processing at various granularity levels called Hierarchical-Granularity Holonic Model (HGHM) has been introduced. HGHM empowers traditional complex systems engineering approach, such as object-oriented or agent-oriented modelling, by natively embedding into a unique model the concept of hierarchic granular knowledge representation and processing. Both theoretical and operational aspects along with some potential applications of HGHM-based approach have been devised in the thesis. This concluding chapter is aimed at summarizing the relevant aspects raised by our proposal and at drawing a line towards future developments on the subject.

5.1 Relevant aspects in HGH modelling, border domains and prospective works

GrC and the holonic field have remained quite distant from one other during these years. However, the expressive power behind the notion of holon, a self-similar entity for managing complex systems, is well suited for exploring in depth granular structures with a simple and efficient conceptual framework. To achieve this end, the concept of holonic granule (HG) has been introduced and a number of example applications that emerge from the theory of HGs has been presented. HG enlightens the complementarity of the two fields, tightening them into a unique model. Furthermore, thanks to introduction of a particular grammar both for writing HG-oriented software and designing HG-based systems, our proposal is close to the field of CWW. By means of HGs we are able to provide hierarchical-granularity level descriptions for complex system analysis.

The author is confident that the study about Holonic Systems can leverage connections among holonic approaches and GrC and CWW applications. By definition, holons provide an abstract framework for dealing with (granular) entities that can be aggregated into higher-level entities or decomposed into lower-level entities maintaining always the same computational structure. Consequently, they are suitable for describing processes at different granularity levels on the base of available data. In this sense, holons can be an ‘alphabet’ in which to inflect GrC and CWW ideas. Hopefully, the proposed technique represents a first step in this direction.

In the authors’ view, HGH modelling is worth considering for at least reasons of two orders:

- it combines theoretical and practical aspects in a unique framework;
- it allows for handling system granularity under a holonic perspective.

The first point represents a remarkable point with respect to the current GrC community where the debate on the trade-off between the theory and implementation is still wide open. The second point is in the line of the Occam’s razor principle: since the holonic modelling has already a well-grounded authorship, its translation to the GrC field is more straightforward than building a new theory from scratch.

Since our HGHM is based mainly on the architectural aspects of granular systems, we believe that our proposal is additive to several existing methodologies. For example, a more in-depth investigation should be pursued between HGH modelling and FL. One major point is that in both approaches knowledge of observed phenomena is expressed in terms of linguistic propositions rather than numerical equations.

In this thesis, the focus was on crisp granular modelling alone: HGs accounting for linguistic hypotheses on data have been considered principally in case they were true or false with no degree of uncertainty attached. However, reasoning in FL-like terms, holonic rules can be assigned a membership value indicating the uncertainty about the truthfulness of the underpinning logical implication. This makes holonic rules prospectively suitable for automated inference under uncertainty and hence they can be a valuable means for those real-world settings where agents have to take decisions with only noisy and incomplete information available. In this sense, well-established solutions like FL-based agents can be a source of inspiration and comparison for future HGHM-based applications.

It is necessary to point out that, from a CI perspective, our HG-based approach compared to FL-based ones is at a very early stage of evolution. At the moment, it is hardly predictable which similarities and which differences will emerge in the next future. Time will tell: if we look at FL development for example, we notice that some decades have been required to mould new concepts such as GrC and type-II FL.

For what program coding and software structure parsing are concerned we found out that HGHM, thanks to its inherent recursive nature, allows for separating neatly between computational element (which remains always the same, at any hierarchical-granularity level) and KR, leaving this one on the behalf of the software engineer.

HGH modelling, stemming from the original Koestler's idea of holon as an entity being a while and a part at the same time, allows to deal with holistic approaches in a computational way. In the literature, with particular reference to the scientific one, the holistic view as been considered as too philosophical and scarcely applicable to real-world situations. In fact, we, as engineers, are generally thought to decompose complex problems according to a reductionist view, which is certainly efficient in practical terms, but has the disadvantage of loosing the concept of the whole as more than the sum of its parts, which is something difficult to grasp only with reductionist-oriented approaches.

Since HGH modelling is not only a theoretical frame, but, as shown in the thesis, also a practical methodology for managing different problems in the realm of complex systems modelling, we are hopeful that new debates and new proposals will arouse in the field of CI following our direction.

5.2 Open Questions and Future Developments

In this thesis, a novel holonic-based methodology for supporting complex system modelling at multiple granularity levels has been presented.

From an engineering perspective, the relevant aspect of the proposal lays in the possibility to handle multiple granularity-level descriptions within a single

framework. This was made possible thanks to the noteworthy property of holon considered as a computational entity able to play the part of a whole and a part of a whole at the same time. Such a kind of ambiguous definition has been made operational by introducing the concept of HG, a re-definition of the Zadeh's concept of information granule in holonic terms.

HGs are data structures arranged in a recursive fashion: i.e., they can be described as nested hierarchies of other instances of themselves until some primitive form is reached out. HGs can be encoded within a hashtable to provide the basic structure for holonic-inspired programs: we provided some example for this. These programs have the interesting features of being high modular, since they all inherit the same archetype class which is specialized at run-time depending on the granularity-level the holonic program is being processed.

A heuristics for extracting HGs from data has also been reported. It allows for inducing HG-based structure directly from observation but also for automated signal analysis and prediction.

Holarchy extraction and management algorithms have been then fused within a single computational model that we referred to as HGHM.

HGHM is both a conceptual and operational framework to deal with complex systems according to a (holonic) agent-oriented view. For example, HGHM can be used as a software modelling approach basing on the idea that primitive processing functions can be considered as instances of an archetype HG-based class. More generally, HGHM is useful for modelling any complex system where a (physical or conceptual) hierarchy exists. For example, we devised a HGHM-based re-interpretation of the electric power distribution network, showing how the problem of energy balancing can be handled in holonic terms.

This thesis should be considered as a first insight into the problems of holonic modelling from a CI standpoint. Much work is to be done and hypothetical connections between the proposed approach and other CI solutions should be further investigated.

For example, we only gave some hints about the possible relationships between HGH Modelling and FL, while the evolutionary aspects of HG-based holarchies have been completely skipped.

There remain several implementing questions that should be addressed with more care such as the combinatorial aspects in holarchy unrolling mechanism (when holonic rule are highly polysemous) and the technical problems related to the depth of recursion during computation.

A first attempt to provide some symbolic-logic characterization to the process of holarchy rolling/unrolling has been shown. However, this would require a deeper study also from the perspective of Automata Theory and Mathematical Logic. The holonic grammar here proposed has in fact to be considered more as an easy-to-understand computational tool to describe HG-based process like abstraction and enrichment rather than a formal framework with solid theoretical foundations.

Finally, at the epistemic level, HGH Modelling poses several questions in terms of re-definition of the concept of environment as nothing but a super-holon of the holonic system under scope. This position is in contrast with our current view of agent and environment as entities that must be kept separated to perform an efficient design.

Last but not least, our modelling approach assumes implicitly the super-holon playing the role of the environment as being the true reference knowledge during the knowledge extraction phase. From an epistemic point of view, this implies that ground observations would have some sort of inherent truth that allows for producing correct inferences of the observed phenomena.

REFERENCES

1. R. Abilemona, E. M. Petriu, T. E. Whalen (2010), Distributed intelligent sensor agent system for environment mapping, *Journal of Ambient Intelligence and Humanized Computing*, Springer, 1(2): 95-110
2. G. Acampora, V. Loia (2005), Fuzzy control interoperability and scalability for adaptive domotic framework, *IEEE Trans. Industrial Informatics* 1(2): 97-111.
3. G. Acampora, V. Loia (2008), A proposal of ubiquitous fuzzy computing for Ambient Intelligence, *Information Sciences: an International Journal*, Elsevier Science Inc., 178(3): 631-646
4. E. Adam, R. Mandiau, C. Kolski (2000), HOMASCOW: a holonic multi-agent system for cooperative work, Proc. of the 11th International Workshop on Database and Expert Systems Applications, pp. 247-253
5. R. Agrawal, T. Imieliński, A. Swami (1993), Mining association rules between sets of items in large databases, Proc. of the ACM SIGMOD international conference on Management of data, pp. 207 – 216
6. M. Albert, T. Laengle, H. Woern, M. Capobianco, A. Brighenti (2003), Multi-agent systems for industrial diagnostics. Proc. of the 5th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes, pages 483-488
7. C. Alippi, P. Braione (2006), Classification Methods and Inductive Learning Rules: What We May Learn From Theory, *IEEE Transactions on Systems, Man, and Cybernetics—part C: Applications and Reviews*, 36(5): 649-655.
8. C. Alippi, S. Ferrari, V. Piuri , M.Sami, F. Scotti (1999), New trends in intelligent system design for embedded and measurement applications, *IEEE Instrumentation & Measurement Magazine*, 2(2): 36-44
9. R. N. Anthony (1965), *Planning and Control Systems: A Framework for Analysis*, Harvard Business School Division of Research
10. G. Antoniou, F. van Harmelen (2004), Web Ontology Language: OWL. *Handbook on Ontologies*, pp.: 67-92
11. B. Auprasert, Y. Limpiyakorn (2009), Representing Source Code with Granular Hierarchical Structures, Proc. of the IEEE 17th International Conference on Program Comprehension, pp. 319-320
12. G. Ausiello , F. D'Amore , G. Gambosi (2008), *Linguaggi, Modelli, Complessità* (italian only), 2nd ed. Franco Angeli, Milan
13. S. Bandini, R. Serra, Complex Systems, AI*IA *Intelligenza Artificiale* 3(1): 102-108
14. R. Basili, et al. (2002), Knowledge-Based Multilingual Document Analysis, Proc. of the International Conference on Computational Linguistics (COLING 2002) on SEMANET: building and using semantic networks - Volume 11: 1-7
15. F. L. Bellifemine, G. Caire, D. Greenwood (2007), *Developing Multi-Agent Systems with JADE*, Wiley ed.
16. F. L. Bellifemine, A. Poggi, G. Rimassa (2001), Developing multi-agent systems with a FIPA-compliant agent framework, *Software, Practice and Experience*, 31(2):103-128
17. T. Berners-Lee, J. Hendler, O. Lassila (May 2001), The Semantic Web, *Scientific American*
18. D. Bertolini, P. Busetta, A. Molani, M. Nori, A. Perini (2003), Designing peer-to-peer applications: An agent-oriented approach, In R. Kowalczyk, J. P. Muller, H. Tianfield, and R. Unland, editors, Agent Technologies, Infrastructures, Tools, and Applications for E-Services, volume 2592 of LNCS, pages 92-106, Springer
19. A. Bieszczad, T. White, B. Pagurek (1998), Mobile agents for network management, *IEEE Communications Surveys & Tutorials*, 1(1): 2-9
20. V. Bocchio, M. Masoero (1992), CH4, Energia, Metano (italian only), 2: 15-20.
21. C. Böhm, G. Jacopini (May 1966), Flow diagrams, Turing machines and languages with only two formation rules, *Communications of the ACM*, 9(5): 366 – 371
22. M. P. Bonacina, A. Martelli, Automated reasoning, AI*IA *Intelligenza Artificiale* 3(1):14-20
23. L. Breiman, J. Friedman, C. J. Stone, R.A. Olshen (1984), Classification and Regression Trees, 1st Edition, Taylor & Francis

24. R. W. Brennan, J. H. Christensen, W. A. Gruver, Dilip B. Kotak, D. H. Norrie, E. van Leeuwen (2005), Holonic Manufacturing Systems – A Technical Overview, *Industrial Information Technology Handbook*, Richard Zurawski (ed), CRC Press
25. R. W. Brennan, W. A. Gruver, Ken H. Hall (eds) (2011), Special Issue on Industrial Applications of Holonic Systems, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 41(1): 1-3
26. H. van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, P. Peeters (1998), Reference architecture for holonic manufacturing systems: PROSA, *Computers in Industry*, 37(3): 255-274
27. M. Calabrese (2010), Self-Descriptive IF THEN Rules from Signal Measurements. A holonic-based computational technique, Proc. of the 2010 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA 2010), pp.: 102-106
28. M. Calabrese, A. Amato, V. Di Lecce, V. Piuri (2010), Hierarchical-granularity holonic modelling, *Journal of Ambient Intelligence and Humanized Computing* , Springer, 1(3): 199-209
29. M. Calabrese, V. Di Lecce, V. Piuri (2007), ANN Residential Load Classifier for Intelligent DSM System, Proc. of CIMSA 2007 – IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, Ostuni - Italy, pp. 33 – 38, June 27-29.
30. L. Camarinha-Matos, H. Afsarmanesh (2001), Virtual enterprise modelling and support infrastructures: applying multi-agent system approaches, In J. Carbonell and J. Siekmann, editors, Multi-agents systems and applications, pp.: 335—364. Springer-Verlag, New York, NY, USA
31. A. Camurri, A. Coglio (1998), An architecture for emotional agents, *IEEE Multimedia*, 5(4): 24-33
32. S. Carberry, L. Lambert (1999), A process model for recognizing communicative acts and modelling negotiation subdialogues, *Computational Linguistics*, 25(1): 1 – 53, MIT Press.
33. S. Chakrabarti, B.E. Dom, D. Gibson, J.M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins (1999), Mining the Link Structure of the World Wide Web, *IEEE Computer*
34. P. P.S. Chen (1976), The Entity-Relationship Model: Toward a Unified View of Data, *ACM Transactiona on Database Systems*, 1(1): 9-36
35. B. Chen, M. Sun, M. Zhou (2009), Granular Rough Theory: A representation semantics oriented theory of roughness, Applied Soft Computing, 9(2): 786-805
36. F.F. Chen, R.F Babiceanu, R.H. Sturges (2005), Real-time holonic scheduling of material handling operations in a dynamic manufacturing environment, Robotics and Computer-Integrated Manufacturing, 21: 328–337
37. C. Chi-Bin, C.-C.H. Chan, Lin Cheng-Chuan (2005), Buyer-supplier negotiation by fuzzy logic based agents, Proc. of the Third International Conference on Information Technology and Applications, Vol. 1, 137 - 142
38. N. Chomsky (1956), Three models for the description of language. *IRE Transactions on Information Theory*, (2): 113–124
39. N. Chomsky (1959), On certain formal properties of grammars, *Information and Control*, 2 (2): 137–167
40. J.H. Christensen (1994), Holonic Manufacturing Systems: Initial Architecture and Standards Directions, Proc. of the 1st Euro Workshop on Holonic Manufacturing Systems, HMS Consortium, pp.: 1-20
41. J.H. Christensen (2007), IEC 61499: A Standardized Architecture for Adding Value in Industrial Automation, Kitara seminar, HTC High Tech Center, Available at www.holobloc.com.
42. B.T. Clegg (2007), Building a Holarchy Using Business Process-Oriented Holonic (PrOH) Modelling, IEEE *Trans. Syst., Man, Cybern.*—Part A: Systems And Humans, 37(1): 23-40
43. B.T. Clegg, D. Shaw (2008), Using process-oriented holonic (PrOH) modelling to increase understanding of information systems, *Information Systems Journal*, 18: 447-477
44. V. Crespi, A. Galstyan, K. Lerman (2008), Top-down vs bottom-up methodologies in multi-agent system design, *Autonomous Robots*, Springer, 24 (3): 303-313

45. B. Clegg, S. Duncan (Sept. 2008), Using process-oriented holonic (PrOH) modelling to increase understanding of information systems, *Information Systems Journal*, 18(5): 447-477(31)
46. A.W. Colombo, R. Schoop, R. Neubert (2006), An Agent-Based IntelligentControl Platform for Industrial Holonic Manufacturing Systems, IEEE *Transactions on Industrial Electronics*, 53(1): 322-337.
47. D. D. Corkill (Sept. 1991), Blackboard Systems, *AI Expert*, 6(9):40-47
48. P. Davidsson, L. Henesey, L. Ramstedt, J. Tornquist, F. Wernstedt (2005), Agent-Based Approaches to Transport Logistics, Whitestein Series in Software Agent Technologies and Autonomic Computing, pp.: 1-15.
49. R. Davis, H. Shrobe, P. Szolovits (1993), What is a Knowledge Representation? AI Magazine, 14(1):17-33
50. K. Decker, K.Sycara (1997), Intelligent adaptive information agents. *Journal of Intelligent Information Systems*, 9(3):239—260.
51. K. Dellschaft, S. Staab (2006), On How to Perform a Gold Standard Based Evaluation of Ontology Learning, Proc. Of the 5th International Semantic Web Conference, Athens, GA, USA, pp. 173-190
52. D. DeVault, M. del Rey, M. Stone (2009), Learning to interpret utterances using dialogue. Proc. of the 12th Conference of the European Chapter of the Association for Computational Linguistics, pp. 184-192
53. L. Devroye, L. Györfi, G. Lugosi (1996), *A Probabilistic Theory of Pattern Recognition*. New York: Springer.
54. E. Dijkstra, Go To Statement Considered Harmful, *Communications of the ACM*, 11(3) (March 1968): 147–148
55. V. Di Lecce, M. Calabrese, R. Dario (2010), Computational-based Volatile Organic Compounds discrimination: an experimental low-cost setup, Proc. of the 2010 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, pp. 54-59
56. V. Di Lecce, A. Amato, M. Calabrese, A. Quarto, Multi Agent System to promote electronic data interchange in port systems (2008), Proc. of IEEE 21st Canadian Conference on Electrical and Computer Engineering, pp. 729 – 734
57. V. Di Lecce, M. Calabrese (2008), Taxonomies and Ontologies in Web Semantic Applications: the New Emerging Semantic Lexicon-Based Model, Proc. Of the IEEE International Conference on Intelligent Agents, Web Technologies and Internet Commerce (IAWTIC'08), pp. 277-283
58. V. Di Lecce, M. Calabrese, D. Soldo (2009), Semantic Lexicon-based Multi-Agent System for Web Resources Markup, Proc. of the 4th International Conference on Internet and Web Applications and Services (ICIW 2009), pp. 143-148
59. V. Di Lecce, M. Calabrese, D. Soldo (2009), Semantic Lexicon-Based Multi-Agent System for Web Resources Markup, Proc. of the Fourth International Conference on Internet and Web Applications and Services (ICIW 2009), pp. 143-148.
60. V. Di Lecce, M. Calabrese, D. Soldo (2009), Fingerprinting lexical contexts over the Web, Journal of Universal Computer Science, 15(4): 805-825
61. V. Di Lecce, C. Pasquale, V. Piuri (2004), A Basic Ontology for Multi Agent System Communication in an Environmental Monitoring System, Proc. of the International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA 2004), pp. 45-50
62. A. Di Stefano, C. Santoro, G. Pappalardo, E. Tramontana (2004), Enforcing agent communication laws by means of a reßective framework. In H. Haddad, A. Omicini, R. L. Wainwright, and L. M. Liebrock, editors, 2004 ACM Symposium on Applied Computing (SAC), pages 462—468
63. F. Doctor, H. Hagras, V. Callaghan, *A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments*, 35(1): 55 - 65
64. N.J. van Eck, L. Waltman, J. van den Berg, U. Kaymak (2006), Visualizing the computational intelligence field, *IEEE Computational Intelligence Magazine*, 1(4):6-10.
65. D. Dubois, H. Prade (2009), Formal representations of uncertainty, in D. Bouyssou, D. Dubois, M. Pirlot, H. Prade (2009), Concepts and Methods of the Decision-Making Process, ISTE, London, UK & Wiley, Hoboken, N.J. USA.

66. F. Esposito, A. Giordana, L. Saitta (2006), Machine Learning and Data Mining, AI*IA *Intelligenza Artificiale* 3(1): 63-71
67. C. Fellbaum (1998), *WordNet: An electronic lexical database*, MIT Press, Cambridge.
68. J. Ferber (1999), *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Addison Wesley Longman
69. FIPA - Foundation for Intelligent Physical Agent - (2002), FIPA specifications, http://www.fipa.org.
70. K. Fischer, M. Schillo, J. Siekmann (2004), Holonic Multiagent Systems: A Foundation for the Organisation of Multiagent Systems, *Lecture Notes in Computer Science*, Springer, Vol. 2744: 1083-1084
71. M. Fleetwood, D.B. Kotak, W. Shaohong, H. Tamoto (2003), Holonic System architecture for scalable infrastructures, Proc. of the IEEE International Conference on Systems, Man and Cybernetics, Vol. 2, pp. 1469- 1474.
72. M. Fletcher, E. Garcia-Herreros, J.H. Christensen, S.M. Deen, R. Mittmann (2000), An open architecture for holonic cooperation and autonomy, Proc. of the 11th International Workshop on Database and Expert Systems Applications, pp. 224-230
73. M. Fletcher, S.M. Deen (2001), Fault-tolerant holonic manufacturing systems, Concurrency and Computation: Practice and Experience, Special Issue on High Performance Agent Systems, 13(1): 43 – 70
74. R. A. Flores-Mendez (1999), Towards a standardization of multi-agent system framework, ACM *Crossroads*, 5(4): 18-24
75. N. Fornara, F. Viganò, M. Verdicchio, Marco Colombetti (2008), Artificial institutions: a model of institutional reality for open multiagent systems, *Artificial Intelligence and Law*, Springer, 16 (1): 89-105
76. Fujita N. (2001), Holonic controller and assembly task planning, Proc. of the IEEE International Symposium on Assembly and Task Planning, pp. 67-72.
77. A. Gangemi, N. Guarino, A. Oltramari, R. Oltramari (2001), Conceptual Analysis of Lexical Taxonomies: The Case of WordNet Top-Level, Proc. of the International Conference on Formal Ontology in Information Systems (FOIS-2001), pp. 285-296
78. G.S. Gardiner, M.J. Gregory (1996), An audit based approach to the analysis, redesign and continuing assessment of a new product introduction system, *Integrated Manufacturing Systems*, 7: 52–59
79. A. Giret, V. Botti (2004), Holons and agents; *Journal of Intelligent Manufacturing*, 15: 645-659.
80. F. Giunchiglia, T. Walsh (1992), A theory of abstraction, *Artificial Intelligence*, 57(2-3): 323 – 389, ACM
81. A. Greasley (2004), *Simulation Modelling for Business*, Ashgate Press, London, UK
82. T. R. Gruber (1993), A Translation Approach to Portable Ontology Specifications, *Journal of Knowledge Acquisition*, Academic Press, 5(2): 199 - 220.
83. T. R. Gruber (1995), Toward principles for the design of ontologies used for knowledge sharing, *International Journal of Human and Computer Studies*, 43: 907–928
84. W. A. Gruver, D. Kotak, E. van Leeuwen, D. H. Norrie (2003), Holonic manufacturing systems: Phase II, in *Holonic and Multiagent Systems for Manufacturing*, V. Marik, D. McFarlane, P. Valckenaers, (eds.), Berlin, Germany: Springer-Verlag, HoloMAS 2003, LNAI 2744: 1-14
85. Y. Gurevich (2000), Sequential Abstract State Machines Capture Sequential Algorithms, *ACM Trans. Computational logic*, 1(1): 77 – 111
86. H. Hagras (2007), Type-2 FLCs: A New Generation of Fuzzy Controllers, IEEE *Computational Intelligence Magazine*, 2(1): 30-43
87. H. Hagras, V. Callaghan, M. Collry (2001), Outdoor mobile robot learning and adaptation, IEEE *Robotics & Automation Magazine*, 8(3): 53 – 69
88. H. Hagras, F. Doctor, V. Callaghan, A. Lopez (2007), An Incremental Adaptive Life Long Learning Approach for Type-2 Fuzzy Embedded Agents in Ambient Intelligent Environments, IEEE *Transactions on Fuzzy Systems*, 15: 41 - 55
89. J. Han, J. Dong (2007), Perspectives of Granular Computing in Software Engineering, Proc. of the IEEE International Conference on Granular Computing, pp. 66-71

90. Y. Hayashi, T. Ishida (2006), A Dictionary Model for Unifying Machine Readable Dictionaries and Computational Concept Lexicons, Proc. of the 5th international conference on Language Resources and Evaluation (LREC 2006), pp.1-6
91. J.R. Hobbs (1985), Granularity, Proc. of the 9th International Joint Conference on Artificial Intelligence, pp. 432-435
92. Hoang Thi Thanh Ha, M. Occello, Nguyen Thanh Binh (2009), Applying Type Theory to Formal Specification of Recursive Multiagent Systems, Proc. of the International Conference on Computing and Communication Technologies, pp. 1-8
93. Fu-S. Hsieh (2008a), Holarchy formation and optimization in holonic manufacturing systems with contract net, *Automatica*, 44: 959-970
94. Fu-S. Hsieh (2008b), Robustness analysis of holonic assembly/disassembly processes with Petri nets, *Automatica*, 44: 2538–2548
95. Fu-S. Hsieh (2009), Collaborative reconfiguration mechanism for holonic manufacturing systems, *Automatica*, 45: 2563-2569
96. E. Hutchins (1995), *Cognition in the Wild* (Chapter 9), MIT Press
97. D. Inkpen (2001), Building A Lexical Knowledge-Base of Near-Synonym Differences, Proc. of the Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations, pp. 47-52
98. M.C. Jackson, P. Keys (1984), Towards a system of system methodologies, *Journal of Operations Research*, 35, 473–486
99. A. K. Jain, M. N. Murty, P. J. Flynn (1999), Data clustering: a review, ACM Computing Surveys, 31(3): 264-323
100. N.R. Jennings (1994), The ARCHON system and its applications, Proc. of the 2nd International Working Conference on Cooperating Knowledge Based Systems (CKBS-94), pp. 13-29
101. N.R. Jennings, P. Faratin, M. J. Johnson, T. J. Norman, P. O Brien, M. E. Wiegand (1996), Agent-based business process management, *International Journal of Cooperative Information Systems* , 5(2-3): 105-130
102. N. R. Jennings, M. Wooldridge (1998), Applications of intelligent agents. In Agent technology: foundations, applications, and markets, pp. 3-28. Springer-Verlag New York, Inc., Secaucus, NJ, USA
103. J.J. Jiang, D.W. Conrath (1997), Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy, Proc. of the International Conference on Research in Computational Linguistics, pp. 19-33.
104. J. Kegl (1995), Machine-readable dictionaries and education, Walker, Donald E., Antonio Zampolli and Nicoletta Calzolari, eds., Automating the Lexicon: Research and Practice in a Multilingual Environment, Oxford University Press, New York, pp. 249 – 284
105. M. Klusch (2001), Information agent technology for the Internet: a survey, *Data Knowledge Engineering*, 36(3): 337—372
106. A. Koestler (1967), *The Ghost in the Machine*, (1st Edition) Hutchinson, London
107. A. Koestler (1969), Some General Properties of Self-Regulating Open Hierarchic Order (SOHO), *Panarchy*, @http://www.panarchy.org/koestler/holon.1969.htm
108. N. L. Komarovaa, M. A. Nowak (May 2001), The Evolutionary Dynamics of the Lexical Matrix, *Bulletin of Mathematical Biology*, 63(3): 451-485, Springer
109. P. Kopacek (1999), Intelligent Manufacturing: Present State and Future Trends, *Journal of Intelligent & Robotic Systems*, 26(3-4): 217-229, Springer
110. R. Kremer, D. Norrie (2000), Architecture and design of a holonic visual interface, Proc. of the IEEE Conference on Systems, Man, and Cybernetics, Vol. 3, pp. 1715-1720
111. Yu-K. Kwok, I. Ahmad (1998), Benchmarking the Task Graph Scheduling Algorithms, Proc. of the 12th. International Parallel Processing Symposium on International Parallel Processing Symposium, pp. 531-537
112. J. Lagorse, M.G. Simoes, A. Miraoui (2009), A Multiagent Fuzzy-Logic-Based Energy Management of Hybrid Systems, IEEE *Transactions on Industry Applications*, 45(6): 2123-2129
113. P. Langley (2000), The science of machine learning. Preface, In Proc. of the Seventeenth International Conference on Machine Learning (ICML-2000), Morgan Kaufmann

114. P. Leitão, F. Restivo (2008), Implementation of a Holonic Control System in a Flexible Manufacturing System, IEEE *Trans. Syst., Man, Cybern.*, Part C 38(5): 699-709
115. T. Y. Lin (1997), From Rough Sets and Neighborhood Systems to Information Granulation and Computing in Words, Proc. of European Congress on Intelligent Techniques and Soft Computing, pp. 1602-1607.
116. M. Ljungberg, A. Lucas (1992), The OASIS air-traffic management system, Proc. of the 2nd Pacific Rim International Conference on Artificial Intelligence (PRICAI' 92)
117. S. A. Long, A. C. Esterline (2000), Fuzzy BDI Architecture for Social Agents, Proc. of the IEEE Southeastcon 2000, pp.: 68 – 74
118. G. Ma, C. Shi (2000), Modelling Social Agents in BDO Logic, Proc. of the Fourth International Conference on MultiAgent Systems, pp. :411 - 41
119. A. Maedche, S. Staab (2001), Ontology Learning for the Semantic Web, *IEEE Intelligent Systems*, 16(2): 72 – 79
120. B. Magnini, C. Strapparava, F. Ciravegna, and E. Pianta (1994), A Project for the Construction of an Italian Lexical Knowledge Base in the Framework of WordNet, IRST Technical Report #9406-15
121. V. Marik, Michal Pechoucek (2002), Holons and agents: Recent developments and mutual impacts, Proc. 12th International Workshop on Database Expert Systems Applications, in Multi-Agent Systems and Applications II, LNCS, Springer, Vol. 2322: 89-106
122. T.J. McCabe (1976), A Complexity Measure, *IEEE Transactions on Software Engineering*, 2(4): 308-320
123. J.M. Mendel (2007), Computing with Words: Zadeh, Turing, Popper and Occam, IEEE *Computational Intelligence Magazine*, 2(4): 10-17
124. J. M. Mendel, R. I. B. John (2002), Type-2 Fuzzy Sets Made Simple, IEEE *Transactions on Fuzzy Systems*, 10(2): 117-127
125. J. Mendel, L. Zadeh, E. Trillas, R. Yager, J. Lawry, H. Hagras, S. Guadarrama (2010), What Computing with Words Means to Me, IEEE *Computational Intelligence Magazine*, 5(1): 20-26
126. E. Mendelson (1964), *Introduction to Mathematical Logic*, D. Van Nostrand Company -Princeton, New Jersey
127. J. Mennis, J.W. Liu (2005), Mining association rules in spatio-temporal data: An Analysis of Urban Socioeconomic and Land Cover Change, *Transactions in GIS*, 9(1): 5–17
128. M.D. Mesarović, D. Macko, Y. Takahara (1970), *Theory of Hierarchical Multilevel systems*, New York, Academic Press
129. B. Meyer (2009), Touch of Class: Learning to Program Well with Object and Contracts, Springer-Verlag.
130. G. Miller (1995), WordNet: a lexical database for English, *Communications of the ACM*, 38(11): 39-41
131. Minghua He, Ho-fung Leung, N.R Jennings, A fuzzy-logic based bidding strategy for autonomous agents in continuous double auctions, IEEE *Transactions on Knowledge and Data Engineering*, 15(6): 1345 - 1363
132. M. Minsky (1961), Steps toward Artificial Intelligence, Proc. of the Institute of Radio Engineer (IRE), 49(1): 8-30
133. M. Minsky (1967), *Computation: Finite and Infinite Machines*. New Jersey: Prentice-Hall
134. M. Minsky (1988), *The Society of Mind*, Simon and Schuster, New York
135. E. Motta, J. Domingue, L. Cabral, and M. Gaspari (2003), Irs-ii: A framework and infrastructure for semantic web services. In D. Fensel, K. P. Sycara, and J. Mylopoulos, editors, 2nd International Semantic Web Conference, volume 2870 of LNCS, pages 306—318, Springer.
136. J. Mylopoulos (1980), An Overview of Knowledge Representation, Workshop on Data Abstraction, Databases and Conceptual Modelling, pp. 5-12
137. R. Navigli, P. Velardi (July 2005), Structural Semantic Interconnections: a knowledge-based approach to word sense disambiguation, Special Issue - Syntactic and Structural Pattern Recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7)

138. R. Navigli. Word Sense Disambiguation: a Survey (2009). *ACM Computing Surveys*, 41(2): 1-69
139. E. Nichols, F. Bond, D. Flickinger (2005), Robust ontology acquisition from machine-readable dictionaries, In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI-2005), pp. 1111–1116.
140. H. Ning., D. Shihan, Structure-Based Ontology Evaluation (2006), Proc. Of the International Conference on e-Business Engineering, 2006. IEEE, pp. 132 – 137
141. Object Management Group (2007), Documents associated with Architecture-Driven Modernization (ADM)/ Knowledge Discovery Meta-Model (KDM), v1.0, @http://www.omg.org/spec/KDM/1.0/
142. Object Management Group (2007), OMG Unified Modelling Language, Infrastructure, V2.1.2, @http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF
143. J. Odell, H. Van Dyke Parunak, M. Fleischer, S. Brueckner (2003), Modelling Agents and their Environment: The Physical Environment, *Journal of Object Technology*, 2(2): 43-51
144. S. Okamoto, P. Scerri, K. Sycara (2008), The Impact of Vertical Specialization on Hierarchical Multi-Agent Systems, Proc. of the 23rd AAAI Conference on Artificial Intelligence, pp. 138-143.
145. A. Omicini, A. Poggi (2006), Multiagent Systems, AI*IA *Intelligenza Artificiale* 3(1): 79-86
146. N. Ordan, S. Wintner (2005), Representing Natural Gender in Multilingual Databases, *International Journal of Lexicography*, 18(3): 357-370
147. H. V. D. Parunak (1987), Manufacturing experience with the Contract Net. In M. N. Huhns, editor, Distributed Artificial Intelligence, pages 285—310, Pitman
148. H. V. D. Parunak, J. Odell (2002), Representing Social Structures in UML, International Workshop on agent-oriented software engineering, Vol. 2222, pp. 1-16.
149. M. J. Pazzani (2000), Knowledge discovery from data? *IEEE Intelligent Systems and their Applications*, 15(2): 10-12
150. Z. Pawlak (1982), Rough sets, *International Journal of Computer and Information Sciences*, 11: 341-356
151. M. Pechoucek, Vladimir Marik (2008), Industrial Deployment of Multi-Agent Technologies: Review and Selected Case Studies, *International Journal on Autonomous Agents and Multi-Agent Systems*, Springer, AAMAS 17:397-431
152. W. Pedrycz (2001), Granular computing: an introduction, Proc. of the Joint 9th IFSA World Congress and 20th NAFIPS International Conference, pp. 1349 - 1354
153. W. Pedrycz (2002), Computational intelligence as an emerging paradigm of software engineering, Proc. of the 14th international conference on Software engineering and knowledge engineering, pp. 7-14
154. W. Pedrycz, P. Rai (2008), A Multifaceted Perspective at Data Analysis: A Study in Collaborative Intelligent Agents, IEEE *Transactions on Systems, Man, and Cybernetics*—part b: Cybernetics, 38(4): 1062-1072
155. F. Pichler (2000), Modelling Complex Systems by Multi-Agent Holarchies, Lecture Notes in Computer Science, Computer Aided Systems Theory - EUROCAST'99, Springer Berlin / Heidelberg, pp. 154-168
156. A. Poggi, M. Tomaiuolo, G. Vitaglione (2005), A security infrastructure for trust management in multiagent systems. In R. Falcone, K. S. Barber, J. Sabater-Mir, and M. P. Singh, editors, Trusting Agents for Trusting Electronic Societies, volume 3577 of LNCS, pages 162-179. Springer
157. D. Poole, A. Mackworth, R. Goebel (1998), *Computational Intelligence: A Logical Approach*, Oxford University Press, New York
158. T. Qian, B. Van Durme, L. Schubert (2009), Building a Semantic Lexicon of English Nouns via Bootstrapping, In Proc. of the NAACL HLT Student Research Workshop and Doctoral Consortium, pp. 37–42.
159. S. Robinson (2003), *Simulation: The Practice of Model Development and Use*, John Wiley and Sons, Chichester, UK.
160. S. Russell, P. Norvig (2003), *Artificial Intelligence: A Modern Approach*, 2nd ed., Prentice Hall
161. J. R. Searle (1969), *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge.

162. J. Searle, (1980), Minds, Brains and Programs, *Behavioral and Brain Sciences* 3 (3): 417–457
163. W. Shen, D. H. Norrie (1999), Agent-based systems for intelligent manufacturing, A state-of-the-art survey, *Knowledge Information Systems*, 1(2): 129-156
164. M. Schillo, K. Fischer (2003), Holonic Multiagent Systems, *Zeitschrift für Künstliche Intelligenz*, No. 3
165. S. Shafaei, N.G. Aghaee (2008), Biological Network Simulation Using Holonic Multiagent Systems, 10th International Conference on Computer Modelling and Simulation, pp.: 617 – 622.
166. G. Shafer (1976), *A Mathematical Theory of Evidence*, Princeton University Press, 1976
167. J.M. Simao, C.A. Tacla, P. C. Stadzisz (2009), Holonic Control Metamodel, IEEE *Trans. Syst., Man, Cybern.*, Part A 39(5): 1126-1139.
168. V. Snasel, P. Moravec, J. Pokorny (2005), WordNet Ontology Based Model for Web Retrieval, Proc. of International Workshop on Challenges in Web Information Retrieval and Integration (WIRI '05), pp. 220-225.
169. K. Sycara (1998), MultiAgent Systems, *AI Magazine* 19(2): 79-92
170. A.S. Tannenbaum (2006), *Structured Computer Organization*, 5th Edn, Pearson Education
171. D. Thompson, D.R. Hughes (1998), Holonic Modelling, *Manufacturing Engineer*, 77(3) pp. 116-119
172. P. Tichy, P. Slechta, R.J. Staron, F. Maturana, K.H. Hall (2005), Multiagent technology for fault tolerance and flexible control, IEEE *Trans. Syst., Man, Cybern. C*, Appl. Rev., 36(5): 700–705
173. L. van der Torre (2003), Contextual deontic logic: normative agents, violations and independence, Annals of Mathematics and Artificial Intelligence 37: 33–63, Kluwer Academic Publishers, the Netherlands.
174. A. M. Turing (1950), Computing machinery and intelligence, *Mind*, 59: 433-460
175. M. Ulieru, R. W. Brennan, S.S. Walker (2002), The holonic enterprise: a model for Internet-enabled global manufacturing supply chain and workflow management, *Integrated Manufacturing Systems*, 13(8): 538-550
176. M. Ulieru, M. Cobzaru (2005), Building Holonic Supply Chain Management Systems: An e-Logistics Application for the Telephone Manufacturing Industry, IEEE *Transactions on Industrial Informatics*, 1(1): 18-30
177. M. Ulieru, R. Doursat (2010), Emergent Engineering: A Radical Paradigm Shift, *ACM Transactions on Autonomous and Adaptive Systems* (submitted April 2010)
178. M. Ulieru, R. Este (2004), The Holonic Enterprise and Theory of Emergence, International Journal Cybernetics and Human Knowing (Imprint Academic), 11(1): 79-99
179. M. Uschold, M. Gruninger (1996), Ontologies: Principles, methods and applications, *Knowledge Engineering Review*, 11(2): 93-155
180. M. Uschold, M. King, S. Moralee, Y. Zorgios (1998), The enterprise ontology, *The Knowledge Engineering Review*, Vol. 13: 31–89
181. J.S. Valacich, J.F. George, J.A. Hoffer (2006), *Essentials of Systems Analysis and Design,* 3rd edn. Prentice Hall, Upper Saddle River, NJ, USA
182. V. Vapnik (1979), *Estimation of Dependencies Based on Empirical Data*. Nauka, Moscow, (In Russian). English translation. New York. Springer Verlag, 1982
183. P. Velardi, A. Cucchiarelli, Michael Pétit (2007), A Taxonomy Learning Method and its Application to Characterize a Scientific Web Community, *IEEE Transaction on Data and Knowledge Engineering* (TDKE), 19(2): 180-191
184. C. Wagner, Hani Hagras (2010), *Toward General Type-2 Fuzzy Logic Systems Based on zSlices*, IEEE *Transactions on Fuzzy Systems*, 18(4): 637-660
185. S. S. Walker, R.W. Brennan, D.H. Norrie (2005), Holonic Job Shop Scheduling Using a Multiagent System, IEEE *Intelligent Systems*, 2: 50-57
186. Y. Wang (2008), A Hierarchical Abstraction Model for Software Engineering, Proc. of the International Conference on Software Engineering, pp. 43-48
187. W. Wang, V. Portnoy, I. Pollak (2007), A Stochastic Context-Free Grammar Model for Time Series Analysis, Proc. Of the IEEE International Conference on Acoustics, Speech and Signal Processing, III: 1245-1248

188. M. Winikoff (2005), JACK intelligent agents: An i ndustrial strength platform, In R. Bordini, M.Dastani, J.Dix, and A. E. Fallah-Seghrouchni, editors, Multi-Agent Programming, pages 175-193. Springer, Berlin, Germany
189. M. Wooldridge, N. R. Jennings (1995), Intelligent Agents: Theory and Practice, *Knowledge Engineering Review*, 10(2): 115-152
190. M. Wooldridge (2009), *An Introduction to Multi Agent Systems*, 2nd ed. , John Wiley & Sons
191. World Wide Web Consortium (W3C), RDF Vocabulary Description Language 1.0: RDF Schema, @http://www.w3.org/TR/rdf-schema/
192. World Wide Web Consortium (W3C), OWL 2 Web Ontology Language Primer, @http://www.w3.org/TR/2009/REC-owl2-primer-20091027/
193. T. Wu (2005), Granular Computing in Programming Language Design, Proc. of the IEEE International Conference on Granular Computing, pp. 296 – 302.
194. Z. Xiaokun, D.H. Norrie (1999), Dynamic reconfiguration of holonic lower level control, Proc. of the Second International Conference on Intelligent Processing and Manufacturing of Materials, 2: 887-893
195. Y. Yao (2005), Perspectives of Granular Computing, Proc. of 2005 IEEE International Conference on Granular Computing, 1: 85-90
196. L. A. Zadeh (1965), Fuzzy sets, *Information Control*, 8: 338-353
197. L.A. Zadeh (1979), Fuzzy sets and information granularity, in: Advances in Fuzzy Set Theory and Applications, Gupta, N., Ragade, R. and Yager, R. (Eds.), Amsterdam: North-Holland, pp.: 3-18
198. L.A. Zadeh (1996), Fuzzy logic = computing with words, IEEE *Transactions on Fuzzy Systems*, 4(2): 103-111
199. Zadeh LA (1997) Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic, *Fuzzy Sets and Systems,* 90: 111-127
200. L. A. Zadeh (1998), Some reflections on soft computing, granular computing and their roles in the conception, design and utilization of information/intelligent systems, Springer-Verlag *Soft Computing* (2): 23—25
201. L.A. Zadeh (2004), Soft computing and fuzzy logic, *IEEE Software*, 11(6): 48-56