Mathematical Foundation of Cognitive Computing Based Artificial Intelligence

Tamás Gergely(✉) and László Ury

Applied Logic Laboratory, Budapest, Hungary
gergely@all.hu, uss@t-online.hu

Abstract. Today Cognitive computing and Artificial Intelligence (AI) face the same challenges namely, simulate human thought processes and mimic the way human brain works. The main difference between Cognitive computing and AI is: (i) AI models various functions of human intelligence, where computer is one of the modelling means though often the most important one, i.e. intelligence is in the focus while (ii) Cognitive computing models human thought processes and simulates the hypothetical way human brain works as computation.

Our aim is to develop a theoretically and methodologically well-founded theory of AI together with a unified computational theory, which will provide specific tools and methods for Cognitive computing.

To achieve our goal we follow a methodology triangle, consisting of a conceptual-philosophical, a system theoretical and a logical-mathematical component. Computing will play a fundamental role in both system-theoretical and logical-mathematical methodological components.

Hereby we concentrate on the development of the logical-mathematical foundation in detail by the use of category theory, which provides an excellent frame for defining all notions necessary for developing a universal theory for computing, specification, cognitive reasoning, information, knowledge and their various combinations. Foundation theory is by the use of the so-called constitutions, the mathematical basis for the cognitive computation. Logical foundation will be developed as a special constitution and cognitive computing processes are defined by using situations, infons and information. The main properties are discussed with some examples.

Keywords: Categorical theoretical foundation · Cognitive computing · Specification theory · Cognitive reasoning · Computing theory · Logic programming

1 Introduction

1.1 Artificial Intelligence Today

What is meant by artificial intelligence? So far not a single conceptual apparatus has been formed, nor there is a single conceptual justification and a single scientifically based methodology. Besides, there has not yet been developed a

G. S. Osipov et al. (Eds.): Artificial Intelligence, LNAI 11866, pp. 29–64, 2019.
https://doi.org/10.1007/978-3-030-33274-7_3

generally accepted philosophical foundation in the form of such epistemology and ontology, which would consider and respond to the challenges that arise during the development of the field of artificial intelligence (AI).

The High-Level Expert Group on AI of the European Commission dealing with the definition of AI provides the following definition for AI [1] as a scientific discipline: *AI includes several approaches and techniques, such as machine learning (of which deep learning and reinforcement learning are specific examples), machine reasoning (which includes planning, scheduling, knowledge representation and reasoning, search, and optimization), and robotics (which includes control, perception, sensors and actuators, as well as the integration of all other techniques into cyber-physical systems).*

Today, modern dictionary definitions focus on AI since it is a field that applies computer science and how machines can imitate human intelligence (human-like rather than becoming human). The English Oxford Living Dictionary [31] gives this definition: The theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages.

At the same time The Encyclopedia Britannica defines artificial intelligence as it the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience [32].

The term *artificial intelligence* is frequently applied to systems endowed with the intellectual processes characteristic of humans, such as the ability to make decisions, to solve problems, to understand texts, to recognize pictures, to learn from an actual activity and a past experience, etc. There are different technologies that get ranked as artificial intelligence and there are different types of AI.

The current wave of AI innovation focuses on several real-life applications of artificial intelligence that often start with words such as *smart*, *intelligent*, *predictive* and, indeed, *cognitive*, depending on the exact application and vendor. One major issue is that artificial intelligence is indeed a broad concept and reality, covering many technologies and realities that lead to misunderstandings about what it exactly means. Some people are actually speaking about machine learning when they talk about AI. The most advertised AI tools by Google, Facebook etc. are mainly or only machine learning and, mostly, deep learning related. This is why the wide public thinks that all new AI applications are carried out only with this type of machine learning. However, neither machine learning, nor deep learning are synonyms for AI. They are only one of the many areas of AI research. Moreover, deep learning is a technology of the 1980's while trained with more data, 1970's *neural networks with hidden layers* gave better results; and then it was renamed as deep learning and was hyped as such.

Today, AI is a conglomerate of techniques, technologies and of various research and development directions. Machine learning and especially deep learning are the most common methods. However, deep learning technology, from

the application point of view has been close to its limit. Artificial intelligence urgently needs to be promoted to a new stage, and to achieve breakthroughs in the development of an appropriate underlying theory.

1.2 Cognitive Computing

As AI applications became more and more widespread, a new name, Cognitive computers appeared. This actually is a renaming that has not brought much new to the content of AI. Cognitive computing is a term really, that has been popularized by IBM mainly to describe the current wave of artificial intelligence with a twist of purpose, adaptiveness, self-learning, contextuality and human interaction. The latter, the human one is the key here and without a doubt, also easier to digest than all those AI-related science fiction scenarios.

Note that thanks to science fiction, many people think of artificial intelligence as a computer or robot thinking like a person, including self-awareness and independent will. Instead, what we call cognitive computing uses the ideas behind neuroscience and psychology to **augment** human reasoning with better pattern matching while determining the optimal information a person needs to make decisions.

However, if we peel off the marketing catches from the notion of Cognitive computing used by IBM then we get back AI with one important difference. Namely, Cognitive computing emphasizes the augmentation of human intelligence instead of mimicking it. This is why it is claimed that IBM's Watson is armed with perception and understanding that is refined and expanded with every interaction. Moreover, it should be appropriate for supporting the solution of problems that encompass enormous amounts of information and discernment.

Cognitive computing is primarily a marketing term indicating a computing service that is able to understand, reason and learn from the data it is supplied with. In essence, it is the application of machine learning and artificial intelligence to data processing. IBM is the flag bearer for cognitive computing. Presumably, it wanted a term that differentiated its Watson cloud based service from the ocean of other such services. IBM has its own definition of cognitive computing, cited below [8]:

Cognitive computing refers to next-generation information systems that understand, reason, learn, and interact. These systems do this by continually building knowledge and learning, understanding natural language, and reasoning and interacting more naturally with human beings than traditional programmable systems.

IBM's volatile, often science-fiction-like allegations have provoked serious criticism (see e.g. Schank [23]) . Independently from IBM's way Cognitive computing has its history, see for a brief state of the art Gutierrez-Garcia and López-Neri [15]. The notion of Cognitive computing appeared in Schank [22], where natural language understanding and knowledge structures were in the focus. e structures were in the focus.

Valiant [25] defined cognitive computing as *a discipline that links together neurobiology, cognitive psychology and artificial intelligence.* Brasil et al. [6] state

that cognitive computing is *a collection of emerging technologies inspired by the biological processing of information in the nervous system, human reasoning, decision making, and natural selection.*

Today, Cognitive computing refers to the ability of automated systems to handle conscious, critical, logical, attentive, reasoning modes of thought. Semantic computing facilitates and automates the cognitive processes involved in defining, modelling, translating, transforming, and querying the deep meanings of words, phrases, and concepts. This claim is very similar to that of AI.

Therefore, Cognitive computing faces the same challenges that AI does. Cognitive computing aims to simulate human thought processes and mimic the way human brain works, addressing complex situations are characterized by ambiguity and uncertainty. AI aims to perform operations analogous to learning and decision making in humans. Intelligent personal assistants can recognize voice commands and queries, respond with information, or take desired actions quickly, efficiently, and effectively.

Today the main difference between Cognitive computing and AI is the following:

1. AI aims to model various functions of human intelligence with different levels of detail and abstraction, where computer is one of the modelling means but very often the most important one, i.e. intelligence is in the focus while,
2. Cognitive computing aims to model human thought processes and simulate the hypothetical way the human brain works, i.e. computing is in the focus.

The fundamental shortcoming of the two areas is that neither has a uniform system of concepts, theoretical and methodological foundations. Instead, both consist of a conglomerate of technologies and methods. For example, Cognitive computing would also need a unified computational theory that should be developed with specific tools and methods that support the modelling of the main features of cognizing and thinking processes. However, such theory does not exist as yet, though there had been some attempts, see e.g. Amir [3].

At the same time there is a complex approach to provide a theoretical framework for cognitive computing together with some advances in the study of cognitive computing theories and methodologies in cognitive informatics, soft computing, and computational intelligence, see Wang [27] and [28]. This approach provides conceptual and behavioural models of cognitive computing. It also introduces mathematical tools such as inference algebra and denotational mathematics to deal with the design and implementation of cognitive computing systems.

Note Wang [27] defines Cognitive computing as the conglomerate of *more intelligent technologies beyond imperative and autonomic computing, which embodies major natural intelligence behaviours of the brain such as thinking, inference, learning, and perceptions.*

However, it is important to emphasize that the formal modelling of the cognitive processes aims to mimic the fundamental mechanisms of the brain. This approach develops a model for the brain architecture called Layered Reference Model of the Brain (LRMB), see Wang [26]. This model formally and

rigorously explains the functional mechanisms and cognitive processes of the natural intelligence. A comprehensive and coherent set of mental processes and their relationships is identified in LRMB, that encompasses 37 cognitive processes at six layers known as the sensation, memory, perception, action, meta cognitive, and higher cognitive layers from the bottom-up. The modelling tools are computers. Therefore this approach leads to the area where the processes of human intelligence are to be modelled by the use of computers. This is the area of computational mind. Computationalism is the main way of seeing the cognitive processes. Computationalism is a family of theories about the mechanisms of cognition. The main relevant evidence for testing computational theories comes from neuroscience, though psychology and AI are relevant, too. Computationalism comes in many versions, which continue to guide competing research programs in philosophy of mind as well as psychology and neuroscience. Computation theoretic approach is grounded in the idea that the mind, in many ways works like a digital computer; the mind is parsing internal representations (symbols) in algorithmic ways.

In order to appropriately use the notion of computing it is important to clarify its nature. Computation is an ambiguous concept and computer scientists, philosophers and cognitive scientists who use the concept can contest some claim using it and do not realise they are not actually in disagreement with each other, even though it looks as if they were. For a deep and detailed analysis of the notion of computing we refer to Fresco [10].

Moreover, instead of going into detail review of the main approaches to the computation theory of mind we refer to some good works that represent the current state of the art in this area, such as Ivancevic [17], Milkowski [20] and Piccinini [21].

1.3 What We Offer

Our aim is to develop a theoretically and methodologically well founded theory of AI, where this abbreviation means Amplifier for Intelligence. This AI will be able to act as genuine problem-solving companion understanding and responding to complex problem situations. This AI system will be able to act either as a partner system for cooperative functioning with a human agent or as an autonomous cognitive system for a well-defined problem area. As to become a cooperative partner for human agents, the system has to function very similarly to humans. e.g., it should be able to communicate and understand natural language, reason in a compatible way, learn from its experience, etc. AI will be able to cognize the environment and itself including the co-operative partner. Namely, this AI will be able to self-reflection.

The cognizing activity may run on a wide scale from learning objects, events till discovering various tendencies and regularities. This expected activity would realise the data → information → knowledge transformation and processing at different computation levels. Special attention will be devoted to the knowledge change and management that results during the data → information → knowledge transformation and processing.

The long-term vision is to develop a theoretically well-founded, coherent, integrated theory, technology and design methodology for a new computation paradigm – the so-called **CO**gnitive **I**ntelligence co-**O**perating **S**ystem (*COIOS*). *COIOS* supports Collaborative Intelligence, where humans and AI systems are joining their abilities. Therefore, in our case AI will be a symbiosis rather, instead of a replacement.

Unlike traditional computers within the von Neumann paradigm, COIOS-systems will be able to interpret and gain novel insights from data, solve problems and make decisions without explicit algorithmic instructions from humans. Instead of being programmed to perform pre-defined tasks, they will act as genuine problem-solving companions able to understand and respond to complex problem situations. Unlike data-centric processing of the traditional computers COIOS analyses data and processes information in a cognitive way and deals with knowledge in a goal-oriented way. Essentially this processing targets the reduction of the uncertainty of a problem situation of ignorance.

According to the proposed vision COIOS-systems take problem situations with various uncertainties as their input, and they resolve or decrease these uncertainties via cognitive reasoning, without relying on predefined problem-solving algorithms known in advance. Bearing this in mind COIOS will realise a reasoning-based, uncertainty-driven, upper-level computation.

To achieve our goal we follow a methodology triangle which consists of a conceptual-philosophical, a system theoretical and a logical-mathematical component. In the proposed approach computing will also play a fundamental role in both system-theoretic and logical-mathematical methodological components. The conceptual-philosophical component provides a formal epistemology with cognizing agents and ontology characterising the world to be cognized. Formal epistemology deals with data analysis, information extraction and knowledge acquisition with respect to an actual problem situation and the active cognizing agent. System-theoretic component provides the main principles for the organisation of cognizing processes, which are controlled by directed thinking. Directed thinking is goal-oriented and connected with a cognizing agent's problem solving activity.

It will be developed by the use of category theory, which provides an excellent frame for defining all notions necessary for elaborating a universal theory for computing, specification, cognitive reasoning, information, knowledge and there various combinations. The foundation theory is provided by the use of so-called constitutions, which form the mathematical basis for cognitive computation. The logical foundation will be developed as a special constitution.

Two constructive versions of set theory will be provided:

1. *clFSA* theory of finite sets with atoms and classes,
2. *cclFSA* theory of finite sets with atoms, classes and co-classes.

It is shortly described how *clFSA* permits to describe the entire traditional computation theory including e.g. program semantics, computation power of various programming languages, various programming paradigms like instructional, declarative programming and program specification. It is shown how

cclFSA can support the description of the programs that use metadata called information and knowledge in a strong mathematical frame.

It is shown how various specific approaches such as granular programming or probabilistic programming can be represented in the proposed approach.

In a general setting, the mathematical theory provides an important method to extend a given theory according to specific needs. This method is the inductive and co-inductive extension in constitutions.

By the use of the proposed mathematical tools it is shown how cognitive reasoning can be handled in the proposed logical-mathematical framework. The main specificities of cognitive reasoning for cognitive computing are connected with the possibility to handle

1. The dynamic nature of the reasoning-based cognitive computation processes. The dynamic characterisation of the cognitive computation processes will be based on the representation of a cognitive reasoning process as a motion from ignorance to knowledge.
2. Spatial strategies, which permit to combine data driven statistical and cognitive data processing with the logic based modification calculi.
3. The *semantic* or *contentual* aspects of reasoning, whereas traditional instructional and declarative programming articulate statements inferentially, according only to their shape, without regard to reference. A referential reasoning is proposed, which is based on a special dual (semantic-syntactic) approach, which uses a set of axioms, which entirely and uniquely describes the semantic structure. The latter is considered as a model of our initial knowledge about the subject domain.
4. Indeterminacy and temporal contradictions of the cognitive computation processes in contrast to correctness of traditional instructional and declarative programs. A special formal approach can be provided to deal with the logical contradictions.
5. It provides a scientifically well-founded general approach that possesses methods and tools for modelling, designing and generating information processing responsible for the formation of cognitive processes of artificial cognitive systems. The proposed approach, at the same time, provides an innovative logical foundation for the entire area of cognitive reasoning and it provides support for cognitive system development at the following three levels of abstraction: conceptual, formal, and realisational levels.

Cognitive computing processes will be defined using situations, infons and information. However, the well-known constructions of situations, infons and information (see e.g. Barwise [5] or Devlin [9]) are used in a modified way, which will be formalised by the use of the proposed methods of extension. Here Cognition Kernel will be one of the main constructs, which generalises (i) the information theory and the corresponding data analysis together with a referential reasoning system, and (ii) the multilevel organisation of situation related information and knowledge management processes. Thus Cognitive Kernel will provide an adequate framework to handle the data → information → knowledge transformation and processing. Cognitive computing is defined by the use of Cognition Kernel.

2 The Mathematical Foundation

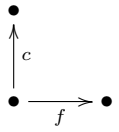
Considering the formalism used in the present paper we first of all assume that the reader is familiar with the basics of set theory and with a basic course on mathematical logic. Moreover, reading Sect. 2 also requires familiarity with a basic course in category theory. However, the reader not familiar with the latter can avoid Sect. 2 and can read the paper as a constructive specification theory based on the theory of finite sets which uses a special first order language as a specification language.

2.1 Constitution Theory

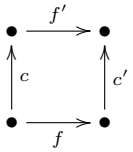
The basic foundational construct is the so-called constitution theory, which provides a logical and category theoretical frame for the development of the cognitive computation theory within the framework of the proposed approach. The first order logic provides the operative tool-set for the constitution theory. Compact constitutions form an important class, which provides the foundation for (i) descriptive theory to describe and investigate various aspects of costructivity necessary for any type of computations, (ii) specification theory to provide a framework for specifying computational objects.

Definition 1. *A* **pre-constitution** $\mathbb{P}$ *is a pair (Uni,Cons) where*

1. *Uni is a category.*
2. *Cons is a subcategory of Uni such that* $Ob\,Cons = Ob\,Uni$.
3. *For any diagram*



where $c \in Mor\ Cons$ *there exists its colimit*



such that c' *also belongs to* $Mor\ Cons$. *This property is called* **hierarchy persistence** *property of the pre-constitution* $\mathbb{P}$.

Example 1. Let *First* denote the category of all first order theories and theory morphisms between them. A morphism $Th_1 \to Th_2$ is called conservative iff it is a composition of a renaming map and of a conservative extension in the original sense. Let *FCons* denote the subcategory of *First* generated by the class of all conservative theory morphisms. It is an easy exercise using Craig interpolation theorem to prove that the pair $(First, FCons)$ is a pre-constitution, further on be denoted as $\mathbb{FOL}$.

Example 2. The following logical systems satisfy the Craig interpolation property and hence their theories with the conservative extensions form a pre-constitution:

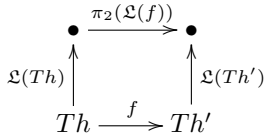
- classical first order logic;
- classical higher-order logic;
- ω-logic;
- intuitionist first order logic;
- intuitionist type theory;
- classical temporal logic;
- intuitionist temporal logic.

Example 3. In Burstall, Goguen [7] introduced the notion of institution to give a categorical-theoretical approach to model theory. For an institution can be defined the interpolation property as well. It can easily prove that institutions with interpolation property form a pre-constitution.

Example 4. (For details see Gergely, Ury [12]) It is well-know that a Cartesian-closed category having a subobject classifier is called topos. A topos form a pre-constitution. Moreover if *Uni* is an arbitrary finitely cocomplete category within which partial maps are representable then $(Uni, Mon(Uni))$ is a pre-constitution. Accordingly a famous theorem partial maps in topoi are representable. For details see Gergely, Ury [12]

Definition 2. *A* **constitution** $\mathfrak{L}$ *on a pre-constitution* $\mathbb{P} = (Uni,\ Cons)$ *is a function* $\mathfrak{L}$ *from Uni to the category* $Mor(Uni)$ *such that:*

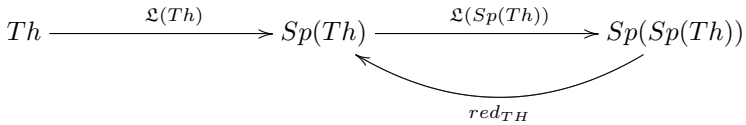
1. *if* Th *is an object of Uni then* $\mathfrak{L}(Th)$ *is a conservative morphism over* Th *called the* **superstructure** *of* Th*;*
2. *if* f *is a morphism then* $\pi_1(\mathfrak{L}(f)) = f$*;*
3. *for all* $Th \in Obj(Uni)$ *the following diagram is a colimit with respect to* $(\mathfrak{L}(Th), f)$ *where:*



Definition 3. *Let* $(Uni, Cons)$ *be a pre-constitution. A constitution* $\mathfrak{L}$ *on it is said to be* **perfect for an object** $Th \in Ob\ Uni$ *iff there is a morphism* red_{Th} *such that*

$$(\mathfrak{L}(cod(\mathfrak{L}(Th)), red_{Th})$$

is a projection system, i.e. the diagram below commute:



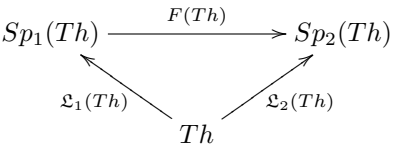
where $Sp(Th)$ and $Sp(Sp(Th)$ is the codomain of $\mathfrak{L}(Th)$ and $\mathfrak{L}(\mathfrak{L}(Th))$, respectively. $Sp(Th)$ is called the **superstructure** *of Th.*

Definition 4. *A constitution $\mathfrak{L}$ on a pre-constitution $\mathbb{P} = (Uni, Cons)$ is compact iff for all object of $Th \in Ob\ Uni$ $\mathfrak{L}$ is perfect for Th.*

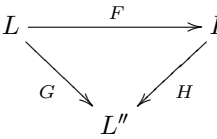
Definition 5. *Fix a pre-constitution $\mathbb{P} = (Uni, Cons)$. Let $C(\mathbb{P})$ be the following category:*

$ObC(\mathbb{P}) \rightleftharpoons \{\mathfrak{L} \mid \mathfrak{L}$ *is a constitution on* $\mathbb{P}\}$
$Mor(C(\mathbb{P})) \rightleftharpoons \{F : \mathfrak{L}_1 \to \mathfrak{L}_2 \mid F : Obj(Uni) \to Mor(Uni)$ *such that* $\mathfrak{L}_1(Th) \circ F(Th) = \mathfrak{L}_2(Th)\}$.

The last condition in the definition of $Mor(C(\mathbb{P}))$ means that the following diagram commutes:



Theorem 1. *Let $\mathbb{P} = (Uni, Cons)$ be a pre-constitution and let $\mathfrak{L}$ be a constitution on it. Let us suppose that Uni has countable coproducts. There is a unique (up to natural isomorphism) constitution $\mathfrak{L}'$ and a morphism $F : \mathfrak{L} \to \mathfrak{L}'$ such that*

1. *$\mathfrak{L}'$ is compact;*
2. *for any $G : \mathfrak{L} \to \mathfrak{L}''$ with closed $\mathfrak{L}''$ there is a factorization through F; i.e. there is a (unique) H such that*  *commutes in $C(\mathbb{P})$.*

3. *This $\mathfrak{L}'$ is denoted as $\mathfrak{L}^*$*

2.2 *FOL*-Based Constitutions

In this subsection we define a compact constitution $\mathbb{Y}$ called **fixed-point constitution**. The superstructures of this constitution add least and greatest fixed-point to each monotone operators. From now we will work in the first-order pre-constitution $\mathbb{FOL}$. A constitution called *FOL*-based if it is a constitution on a full subcategory of $\mathbb{FOL}$. In the sequel we give some examples for such constitution usable in computing, AI and cognitive computing.

The well-know notion of transitive closure can be turned into a constitution.

Definition 6. *Let $Th = (\sigma, Ax)$ be a fixed first-order theory.*

1. *A first-order operator in Th is a triple (Φ, R, X) written as $\Phi(R, X)$ where R is an X-type new relation symbol, Φ is a $\sigma \cup \{R\}$-type formula free variables of which belongs to X. The set of variables X is called the type of $\Phi(R, X)$*
2. *$\Phi(R, X)$ is called monotone iff for any new X-type relation symbols S*

$$Ax \vdash (\forall X\ R(X) \to S(X)) \to \forall X\ (\Phi(R, X) \to \Phi(S, X))$$

3. *For any $\psi(X)$ let*

$$\tilde{\Phi}(\psi)(X) \text{ be the formula } \Phi(R, X)[\psi/R].$$

It is clear that $\tilde{\Phi}$ is a function of type $Form_\sigma(X) \to Form_\sigma(X)$
4. *Let us denote Δ_{Th} be denote the set of all* **monotone first-order operator** *in Th.*

Let $\top_X$ and $\bot_X$ be denote the X-type truth and falsity, respectively. If a first-order operator $\Phi(R, X)$ is monotone in Th then we got an infinite chain

$$\bot_X \to \tilde{\Phi}(\bot_X) \to \tilde{\Phi}(\tilde{\Phi}(\bot_X)).....\tilde{\Phi}(\tilde{\Phi}(\top_X)) \to \tilde{\Phi}(\top_X) \to \top_X$$

where each individual implication is provable in Th.

Definition 7. *$Th = (\sigma, Ax)$ be a fixed first-order theory and fix a set $\mathcal{F}$ of monotone first-order operators.*

1. *Let $\tilde{\Phi} \in \mathcal{F}$ be an X-type monotone first-order operation. A formula $\psi(X)$ is called a left fixed-point of $\tilde{\Phi}$ iff $Ax \vdash \tilde{\Phi}(\psi) \to \psi$ and called a right fixed-point of $\tilde{\Phi}$ iff $Ax \vdash \psi \to \tilde{\Phi}(\psi)$.*
2. *For any $\tilde{\Phi} \in \mathcal{F}$ let add two new X-type relation symbols Φ^μ and Φ^ν together the following axioms:*
   (a) $\tilde{\Phi}(\Phi^\mu) \to \Phi$ and $\Phi \to \tilde{\Phi}(\Phi^\mu)$
   (b) for an arbitrary σ-type formula $\psi(X)$: $(\psi \to \tilde{\Phi}(\psi)) \to (\Phi^\nu \to \psi)$
   (c) for an arbitrary σ-type formula $\psi(X)$: $(\tilde{\Phi}(\psi) \to \psi) \to (\psi \to \Phi^\mu)$
   (d) Let $Ind(\mathcal{F})$ denote this new set of axioms.

Theorem 2. *Let $Th = (\sigma, Ax)$ be a fixed first-order theory. For an arbitrary $\mathcal{F} \subset \Delta_{Th}$ the axiom system $Ax \cup Ind(\mathcal{F})$ is conservative over Ax. It means that there is a constitution Ind on the pre-constitution $\mathbb{FOL}$ which renders each Th the conservative extension $Ax \cup Ind(\Delta_{Th})$*

Let us suppose that the restriction of first-order operations are categorical. It means that for all Th there is a $Th_{\mathcal{F}} \subset \Delta_{Th}$ given in such a way that a theory morphism $Th^1 \to Th^2$ transfer $Th^1_{\mathcal{F}}$ into a subset of $Th^2_{\mathcal{F}}$. If so then there is a constitution $Ind_{\mathcal{F}}$ on the pre-constitution $\mathbb{FOL}$ which renders each Th the conservative extension $Th_{Ax} \cup Ind(Th_{\mathcal{F}})$. One of the most well-known example of such a restriction is as follows. For any σ let us consider the positive existential formulas of the form $\psi(X, X')$ where ψ free variables as stated and X is a copy of X'. Any such formulas generate a monotone first-order operators as $\exists U R(U) \wedge \psi(U, X)$. For any Th let $\mathcal{TR}$ let denote the set of all monotone first-order operator defined in such a way. Let $Tran$ be denote this constitution on the pre-constitution $\mathbb{FOL}$.

Theorem 3. *$Tran$ is a compact constitution on the pre-constitution $\mathbb{FOL}$.*

2.3 Inductive and Coinductive Extensions in Constitutions

It is obvious that it is not enough to just add fixed-points to underlying theory. Let $Th = (\sigma, Ax)$ be a first order theory. There are many situations when we need to add new types, functions and relations to the original similarity type σ. All these additional symbols we can collect into a new similarity type θ. Of course the similarity type has not, but the axiom system Ax has to improve new axioms and axiom schemas. If these axioms are short to inductive and coinductive ones then we can formulate a theorem similar to Theorem 1.

In details. Let us fix a similarity type η. Let $\mathbb{FOL}(\eta)$ denote the pre-constitution containing only those theories (σ, Ax) where $\eta \subset \sigma$. Let ζ be a new similarity type and let Eq and Eq^c be a set of quasi-equations and quasi-coequations, respectively based on the similarity type $\zeta + \eta$.

Theorem 4. *Let η, ζ and Eq, Eq^c as in above. There is a least (up to natural isomorphism) constitution denoted by $\mathbb{Y}(\eta, \zeta, Eq, Eq^c)$*

1. *$\mathbb{Y}(\eta, \zeta, Eq, Eq^c)$ is compact,*
2. *All superstructures satisfy $Eq + Eq^c$,*
3. *All monotone operators in each superstructures have least and greatest fixed points,*
4. *If $\mathfrak{L}$ is such a constitution on $\mathbb{FOL}(\eta)$ that satisfies the previous assumptions then there is a unique (up to natural isomorphism) morphism (up to natural isomorphism) $\mathbb{Y}(\eta, \zeta, Eq, Eq^c) \to \mathfrak{L}$.*

For any $Th \in Obj\mathbb{FOL}(\eta)$ the superstructure of Th, i.e. the codomain of the morphism $\mathbb{Y}(\eta, \zeta, Eq, Eq^c)(Th)$ is called **canonical d-inductive extension of** Th (d for double because extension was constructed by using both equations and coequations. If Eq^c is empty we simply say inductive extension or if we want to emphasize that Eq^c is empty we say **simple inductive extension**. If $\mathfrak{L}$ is such a constitution that satisfies the first three assumption of the theorem then $\mathfrak{L}$ is also called d-inductive extension of the extension system (η, ζ, Eq, Eq^c).

In Subsect. 3.2 we demonstrate that for a large part of computing theory the simple inductive extensions are sufficient. Coequations need to speak about cognitive aspects of computation.

Remark 1. There is a clear definition of quasi-equations. Let σ be an arbitrary similarity type. A σ-type formulas in the form $\tau_1 = \tau_2$ where $\tau_i (i = 1, 2)$ are σ terms are called **equation**. A formula in the form

$$\bigwedge_{j=1..n} e_j \to e$$

where $e_j (j = 1..n), e$ are σ-type equations is called a **quasi-equation**. If n = 0 then we get back the notion of equation. See Grätzer [14]. Unfortunately there is no such an elegant and easily usable definition for *quasi-coequations.*

One of the main advantages of the d-inductive extensions is, that there are universal ones. Let η, ζ be fixed and let $\mathbb{EQ}_(\eta, \zeta)$ denote the set of all $(\eta + \zeta)$-type equations. Let $d\nabla(\eta, \zeta)$ be the category of all d-inductive extensions on (η, ζ, Eq, Eq^c)

where $Eq, Eq^c \subset \mathbb{EQ}_(\eta, \zeta)$. The morphisms are the natural transformations between the functors. Also let $\nabla(\eta, \zeta)$ be the category of all simple inductive extensions on $(\eta, \zeta, Eq, \emptyset)$

Theorem 5. *Let η, ζ be arbitrary but fixed similarity types.*

- *Both $d\nabla(\eta, \zeta)$ and $\nabla(\eta, \zeta)$ are really categories;*
- *$d\nabla(\eta, \zeta)$ has terminal objects. Any terminal object of $d\nabla(\eta, \zeta)$ is called* **d-universal extension** *on (η, ζ).*
- *$\nabla(\eta, \zeta)$ has terminal objects. Any terminal object of $\nabla(\eta, \zeta)$ is called* **simple universal extension** *or just* **universal extension** *on (η, ζ).*

2.4 Constitutional Set Theories

Two versions of set theory are provided by the modification of the von Neumann-Bernays-Gödel set theory. The modification augments the set theoretic operations with fixed points of monotone operators. Namely, the so obtained set theories are as follows:

- *cHF* is a compact constitution where superstructures are the finite sets with atoms and classes augmented with least fixed points of positive existential operators (*cl* stands for classes),
- *clFSA* is a compact constitution where superstructures are the finite sets with atoms and classes augmented with least fixed points of monotone operators (*cl* stands for classes),
- *cclFSA* (*ccl* stands for classes and coclasses) a compact constitution where superstructures are the finite sets with atoms, classes and co-classes augmented with least and greatest fixed points of monotone operators.

It is shortly explained how *clFSA* permits to describe the entire traditional computation theory including e.g. program semantics, computation power of various programming languages, various programming paradigms like instructional, declarative programming and program specification, see e.g. Ury, Gergely [24]. For a short description of how *clFSA* looks like see 7.1. It is clear that for a fixed similarity type σ and the axiom system Ax the new axioms of $clFSA(\sigma, Ax)$ depends only on σ. Let $cFSA_\sigma$ denote this set of axioms.

The unifying theories developed in [16] can build using *cHF* instead of ZFC.

Note that the use of the above two set theories is useful because they are universal as can be seen in the following theorem.

Theorem 6. *Let η, ζ be arbitrary but fixed similarity types.*

- *$clFSA(\eta)$ is a simple universal extension on (η, ζ)*
- *$cclFSA(\eta)$ is a d-universal extension on (η, ζ).*

Remark 2. We emphasize that there was not any assumption "how large" is ζ or the set of equations. However we suppose that both of them are recursively enumerable then $clFSA(\eta)$ and $cclFSA(\eta)$ are also recursively enumerable.

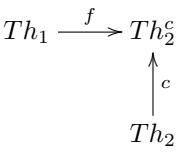
3 Examples

3.1 Specifications as Constitutions

Given a pre-constitution $\mathbb{P} = (Uni, Cons)$ we can think that a specification itself defines the object perfectly. Later on we shall see that the elements of $Ob(Uni)$ are generally a set of algebraic equations or formulas. However, in our definition the notion of a specification is in an abstract form without any fixed meaning. Whatever we can say about the specifications it is identical with the properties of the category *Uni*.

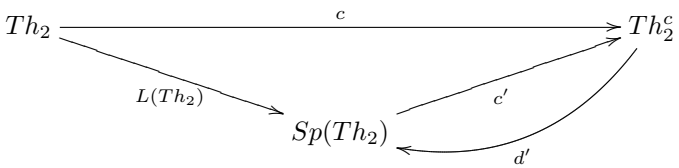
One of the most important properties of the specifications is that they can be interpreted by each other. In the case of algebraic and logical specification theories these interpretations turn out to be homomorphisms or theory presentations. However, in our definition the notion of interpretation is as abstract as those of the specifications. See in Maibaum [19].

Definition 8. *Let $\mathbb{P} = (Uni, Cons)$ be a pre-constitution and let $\mathfrak{L}$ be a constitution on it. An* **L-refinement** *is a pair of morphisms (f, c) as shown below iff there are two morphisms c', d' such that if*



then c splits such that

1. the diagram below commutes:



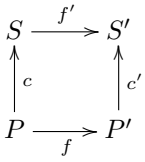
2. and (c', d') is a projection pair.

Theorem 7. *Let $\mathbb{P} = (Uni, Cons)$ be a pre-constitution and let $\mathfrak{L}$ be a constitution on it. $\mathfrak{L}$-refinements are closed under composition iff $\mathfrak{L}$ is a compact constitution.*

An abstract specification theory is developed by the use of category theory, which allows the characterization of specification languages and the provision of the necessary conditions to operate with specifications, e.g. to put them together or to refine or to modularize them. Here only shown the condition necessary for stepwise refinement of specifications is shown. A specification classical first order language which can be used as a specification one. As it can be seen this theory

is appropriate to support formal specifications with a "constructive" definition theory. It is shown how the specification language Z can be developed in this set theoretical framework so that it becomes more transparent from semantical point of view and more useable due to the "constructive" fixed point theory. Moreover, we develop the constructive version of the specification theory by using logic programming ideas. First, logic programming is defined for the abstract specification theory and then logic programming is developed in the proposed set theoretic framework.

One of the desired properties of a specification theory is the correct handling of the hierarchical specification. In our frame any interpretation $c : P \to S$ can be considered as a hierarchical specification. Specification P contains the so called primitive specification part and by using c and S we add some extra to this primitive specification part. It is a natural assumption with respect to the hierarchical specifications that any interpretation of the primitive part can be extended to an interpretation of the entire hierarchical specification. It means that any $f : P \to P'$ can be extended to a commutative diagram below:



However, in most cases initial and terminal specifications do not exist at all. This is the reason why we restrict the hierarchical specifications to conservative ones. Definition 8 (1) axiomatizes the existence of the initial specification for conservative hierarchical specifications.

3.2 Computing Theory

3.2.1 Instructional Programs

Intuitively it is evident that the main problem in defining the IO-relation of a program is connected with the definition of program iteration. Since e.g. denotational semantics renders a relation to a program, the reflexive and transitive closure of this relation corresponds to iteration. We are interested in internalizing, thus we deal with the formulas defining the relations. Therefore the main question is whether the reflexive and transitive closure of an arbitrary formula is definable. It is an easy exercise to prove that using least fixed-points the denotation of the **while** programs can be defined. However, Hoare, Jifeng [16] shows that the correct definition of the denotation of recursive procedure calls requires the greatest fixed-points.

Programs operate on their data environments. If we are interested in the change caused by the execution of a program in its environment then the input-output semantics defined as a binary relation on data sequences, is suitable. A great variety of program properties are connected with the relational semantics, e.g. partial correctness, quasi-total correctness, pseudo-total correctness, etc. See e.g. Gergely, Ury [13].

3.2.2 Logic Programs

The traditional way of programming according to Wirth can be represented as programs = algorithms + data structures (see Wirth [29]). An important combination of traditional programming with the declarative one can take place in the case of data declaration. The latter means that data structure component of the Wirth's characterization of programs should be given by declarative tools, i.e. by the use of logic. In this case programs = algorithm + logic + realisation, where logic may consists of functional and relational parts. The logic component allows to define abstract data types which by the use of realization define the constructive model over which the execution of algorithms, i.e. the computation takes place.

Logic programming takes place in models constructed according to the logic programs. We suppose that a similarity type δ given. Any logic program contains definitions of new relation symbols. The goal of a logic program, i.e. the question which one should be answered reflects these new relations, see e.g. Gergely, Szőts [11]. Let us fix a rich enough similarity type η and a constructive interpretation $\mu : \delta \to cTerm_\eta$. We consider how we can define relations new with respect to μ by the use of logical program's approach.

Since we aim to develop logic programming in *cclFSA*, so the constructivity should be defined with respect to this system axioms. Intuitive meaning of a *constructive* model is that any component of that is computable in *cclFSA*. To define the required notion of constructive model first we have to define an appropriate notion of model, and the notion of computability over this model.

If η a given similarity type and Ax is an η-type set of formulas let $cFSA_\eta(Ax)$ denote the axioms of the superstructure $cclFSA(\eta, Ax)$. If Ax is empty we simply write $cFSA_\eta$.

Definition 9. *Let σ be a fixed similarity type. A function $\mu : \sigma \to cTerm_\eta$ is called an* **interpretation of σ in $cFSA_\eta$.** *An interpretation η* **generates a σ-type model** *in a model $\mathbf{V} \in Mod(cFSA_\eta)$ iff the followings hold in $\mathbf{V}$:*

(a) for all $s \in sort(\sigma)$, $\eta(s)$ is a non-empty class in $\mathbf{V}$;
(b) for all $\rho \in rel(\sigma)$, $\mu(\rho)$ is a subclass of $\top\{\mu(s_i) | i = 1, \ldots, n\}$, where $\rho : s_1, \ldots, s_n$ is the arity of ρ;
(c) for all $f \in fun(\sigma)$, $\mu(f)$ is a functional class of the form $\top\{\mu(s_i) | i = 1, \ldots, n\} \to \mu(s)$, where $f : s_1, \ldots, s_n \to s$ is the arity of f.

We say that μ generates a σ-type model in $cFSA_\eta$ iff for all $\mathbf{V} \in Mod(cFSA_\eta)$ μ generates a σ-type model in $\mathbf{V}$. The σ-type model $\mathbf{A}$ generated by μ in $\mathbf{V}$ is denoted by $\mathbf{V}(\mu)$.

Proposition 1. *Let μ be an interpretation. There is a class formula $IM?(\mu)$ which is valid in a model iff μ generates a model in that set-theory, i.e.*

$$\mathbf{V} \models IM?(\mu) \text{ iff } \mathbf{V}(\mu) \text{ exists.}$$

An interpretation $\mu : \sigma \to cTerm_\eta$ is said to be **correct in $\mathbf{V}$** *iff the model $\mathbf{V}(\mu)$ exists.*

Let us see how to restrict the investigation of formulas to a given interpretation only.

Definition 10. *Let $\mu : \sigma \to cTerm_\eta$ be fixed and let $\varphi \in cForm_\sigma$. Let us define the* **relativization** φ^μ **of** φ **along** μ *by induction on the complexity of φ in the following way.*

(A) First for any term $\tau \in Term_\sigma$ let us define a relation $R_\tau(x, y^\tau)$ (where $x = var(\tau)$ and y^τ is a new variable) which expresses the fact that $\tau(x) = y$:

- *if $\tau = x$ is an s-sorted variable then $R_\tau(x, y^\tau) \leftrightharpoons x = y^\tau \wedge x \in \mu(s)$;*
- *if $\tau = f(x_1, \ldots, x_n)$ and $f : s_1, \ldots, s_n \to s$ then $R_\tau(x, y^\tau) \leftrightharpoons \bigwedge\{R_{\tau i}(x, y^{\tau i}) | i \in n\} \wedge (y^{\tau 1}, \ldots, y^{\tau n}, u^\tau) \in \mu(f) \wedge y^\tau \in \mu(s)$.*

We remark that $R_\tau(x, y)$ implies that $y \in \mu()s$ where $s = sort(\tau)$.

(B)

- *if $\varphi = \rho(\tau_1, \ldots, \tau_n)$ then $\rho^\mu \leftrightharpoons \exists y^{\tau 1} \ldots \exists y^{\tau n} \bigwedge\{R_{\tau i}(x, y^{\tau i}) | i \in n\} \wedge (y^{\tau 1}, \ldots, y^{\tau n}) \in \mu(\rho)$*
- *if $\varphi = \neg\psi$ then $\varphi^\mu \leftrightharpoons \neg(\psi^\mu)$*
- *if $\varphi = \psi_1 \vee \psi_2$ then $\varphi^\mu \leftrightharpoons ({\psi_1}^\mu) \vee ({\psi_2}^\mu)$*
- *is $\varphi = \exists x\psi$ then $\varphi^\mu \leftrightharpoons \exists x(x \in \mu(s) \wedge (\psi^\mu))$, where $s = sort(x)$*

The following statement shows that relativization restricts the investigation of validity of formulas to the given interpretation.

Theorem 8. *Let $\mu : \sigma \to cTerm_\eta$ be an interpretation and let $\mathbf{V}(\mu)$ be the model generated by μ in $\mathbf{V}$. Let $\varphi \in cForm_\sigma$ and let $k \in Val(\mathbf{V}(\mu))$*

$$\mathbf{V}(\mu) \models \varphi[k] \text{ iff } \mathbf{V} \models \varphi^\mu[k]$$

Definition 11. *Let $\mu_\sigma : \sigma \to cV$ be a fixed injection. μ_σ is called the* **canonical interpretation of** σ **in cFSA**$_\eta$. *Let φ^σ denote the relativization of φ along μ_σ.*

Definition 12. *Let $\mathbf{V} \models^\sigma \varphi$ denote the fact that a closed formula is true in every interpretation of σ in $\mathbf{V}$. If so we say that φ* **is valid in** $\mathbf{V}$. *Take $cFSA_\eta \models^\sigma \varphi$ iff for all $\mathbf{V} \in Mod(cFSA_\eta)$, $\mathbf{V} \models^\sigma \varphi$. If so we say that φ* **is valid in cFSA**$_\sigma$.

Let us consider the main properties of this notion of validity. Let $\mu : \sigma \to cTerm_\eta$ be an arbitrary interpretation. Let $\mu = \mu_\sigma$ be a shorthand for the formula $\bigwedge\{\mu(l) = \mu_\sigma(l) | l \in \sigma\}$.

Theorem 9. *Let us suppose that φ is a closed σ-type formula. Let $\mathbf{V}$ be an arbitrary model of $cFSA_\eta$. Let $\mu : \sigma \to cTerm_\eta$ be an interpetation.*

(A) Let us suppose that u generates a model in $\mathbf{V}$. The followings are equivalent:

- $\mathbf{V}(\mu) \models \varphi$;
- $\mathbf{V} \models \varphi^\mu$;

– $Th(\mathbf{V}) + IM?(\mu_\sigma) + (\mu = \mu_\sigma) \models \varphi^\sigma$.

(B) $\mathbf{V} \models^\sigma \varphi$ *iff* $Th(\mathbf{V}) + IM?(\mu_\sigma) \models \varphi^\sigma$

(C) $cFSA_\eta \models^\sigma \varphi$ *iff* $cFSA_\eta + IM?(\mu_\sigma) \models \varphi^\sigma$.

Now we define the computability in $cFSA_\eta$ by the use of the theory of programs developed for the programming language P_η. The necessary notions used bellow can be found in Appendix 1.

Definition 13. *(A) Let C be a class in a model $\mathbf{V}$ of $cFSA_\eta$. C is called* ***computable in V*** *iff there is a program $p \in P_\eta$ such that*

$$C = dom\ Den_{\mathbf{V}}[\![p]\!]$$

(B) A class C is called ***enumerable in V*** *iff there is a computable surjection $\omega \to C$.*

(C) The class C is called ***p∃-definable in V*** *iff there is a term*

$$t \in cTerm\Sigma^+(\eta; \{D\})$$

such that $C = Yt$, i.e. C is the least fixed point of the equation $D = t(D)$.

Theorem 10. *Let $\mathbf{V}$ be a model of $cFSA_\eta$, and suppose that the class $Atom \leftrightharpoons \{x | \underline{atom}(x)\}$ is enumerable in $\mathbf{V}$. The p∃-definable, the enumerable and the computable classes are the same in $\mathbf{V}$.*

We remark that the assumption of Theorem 5.9 in most cases is true, e.g. if Atom is finite or equivalent with ω.

Theorem 11. *If Atom is enumerable in $\mathbf{V}$ then AC holds in $\mathbf{V}$.*

Definition 14. *An interpretation $\mu :\to cTerm_\eta$ is called* ***constructive in V*** *iff for all $l \in \sigma$, $\mu(l)$ is computable in $\mathbf{V}$.*

Example 5. Let σ contain the following symbols:

$0, 1 :\to d$
$+, \times : d, d \to d$

Let interpret these symbols in the 'usual' way:

$\mu(d) \leftrightharpoons \omega$
$\mu(0) \leftrightharpoons 0$
$\mu(1) \leftrightharpoons \{0\}$
$\mu(+) \leftrightharpoons \lambda x, y.\ x + y$
$\mu(\times) \leftrightharpoons \lambda x, y.\ x * y$

where + and * is the sum and product of natural numbers. It is clear that μ generates a model in $cFSA_\sigma$ and in any $\mathbf{V}$ it is constructive.

Example 6. From the point of view of functional programming it is a very interesting question whether a factor of a constructive model is constructive again. Consider the following simple example. Again let σ be as in 5.12. Take the following interpretation:

$\xi(d) \leftrightharpoons \omega \times \omega$
$\xi(0) \leftrightharpoons (0,0)$
$\xi(1) \leftrightharpoons (\{0\},0)$
$\xi(+) \leftrightharpoons \lambda(a,b),(c,d)\,.\,(a+c,b+d)$
$\xi(*) \leftrightharpoons \lambda(a,b),(c,d)\,.\,(ac+bd,ad+bc)$

Clearly ξ is a correct interpretation in any model of $cFSA_\sigma$. Moreover, ξ is clearly constructive.

Let us define the following equivalence on $\xi(d)$:

$$(a,b) \equiv (c,d) \text{ iff } a+d=b+c$$

One can check that *equiv* is a congruence relation on $\mathbf{V}(\xi)$. Clearly $\mathbf{V}(\xi)/\equiv$ is a model for integer numbers $(\underline{Z},0,1,+,*)$. It is a question whether $\mathbf{V}(\xi)/\equiv$ is constructive.

The following theorem gives an answer to this question.

Theorem 12. *Let $\mu : \sigma \to cTerm_\eta$ be a constructive and correct interpretation of σ in a model $\mathbf{V}$. Let us suppose that $\equiv$ is a decidable congruence relation on $\mathbf{V}(\eta)$, i.e. both $\equiv$ and its complement are computable in $\mathbf{V}$. Moreover, let us suppose that all the classes $\mu(s) \in$ sort σ are enumerable in $\mathbf{V}$. If so then $\mathbf{V}(\mu)/\equiv$ is constructive in $\mathbf{V}$.*

Now we can turn to prove the existence of $\mu/\equiv$. Take

$$\mu/\equiv(s) \leftrightharpoons \{x | x \in \mu(s) \wedge \forall y(\ulcorner y \urcorner < \ulcorner x \urcorner \to \neg y \equiv x)\}$$

By using the computability of $\ulcorner\urcorner$ and the decidability of $\equiv$ the right hand side of this equation is a $p\exists$ term in $cFSA_\eta$. Hence $\mu/\equiv$ is well defined on sort(σ). Take

$$\mu/\equiv(\rho) \leftrightharpoons \{(x_1,\ldots,x_n) | \exists y_1,\ldots,\exists y_n \bigwedge x_i \equiv y_i \wedge \rho(y_1,\ldots,y_n)\}$$

$$\mu/\equiv(f) \leftrightharpoons \{(x_1,\ldots,x_n,x) | \exists y_1,\ldots,\exists y_n,\exists y \bigwedge x_i \equiv y_i \wedge x \equiv y \wedge f(y_1,\ldots,y_n) = y\} \quad (1)$$

It is clear that $\mu/\equiv : \sigma \to cTerm_\eta$ is an interpretation. Since $\equiv$ is a congruence relation $\mu/\equiv$ is correct in $\mathbf{V}$. By using the enumerability of classes one can check that $\mu/\equiv$ is a computable relation.

Returning to the example it is easily provable that $\omega \times \omega$ is enumerable in $cFSA_\eta$ and of course $\equiv$ is decidable. It means that $\xi/\equiv$ exists and it is computable in any model of $cFSA_\eta$.

Definition 15. *Let σ be a similarity type. Let*

$$\Gamma_\sigma \leftrightharpoons \{P_s | s \in sort\ \sigma\} \cup \{P_f | f \in func\ \sigma\} \cup \{P_\rho | \rho \in rel\ \sigma\}$$

be a fixed set of class variables. Any function $\Phi : \Gamma_\sigma \to cTerm\Sigma^+(\eta; \Gamma_\sigma)$ is called a ***presentation of σ in $\mathbf{cFSA}_\sigma$. A presentation is correct*** *in a model $\mathbf{V}$ of $cFSA_\sigma$ if the interpretation $Y\Phi : \sigma \to cTerm_\eta$ is correct, i.e. if $\mathbf{V}(Y\Phi)$ exists.*

Theorem 13. *Let us suppose that Atom is enumerable in $\mathbf{V}$. Let Φ be a correct presentation of σ in $\mathbf{V}$. $V(Y\Phi)$ is a constructive model of σ in $\mathbf{V}$.*

It is clear that the models of $cFSA_\sigma$, within which Atom is enumerable, have significant properties. So we give the following definition:

Definition 16. *A model $\mathbf{V}$ of $cFSA_\sigma$ is called textbiconstructive iff Atom is enumerable in $\mathbf{V}$.*

Proposition 2. *'Atom is enumerable' is definable in $cFSA_\eta$.*

Proof. *Let $U(x, y)$ be a universal computable relation with respect to the computable classes. 'Atom is enumerable' is expressible with the following class formula:*

$$\exists x(\{y | U(x, y)\} = Atom)$$

□

Theorem 14. *'Atom is enumerable' is independent from $cFSA_\eta$.*

Proof. *Clearly $cFSA_\eta$ + 'Atom is enumerable' is a conservative extension of $cFSA_\eta$. Let A be an uncountable set and fix a standard model $\mathbf{V}$ in such a way that Atom in $\mathbf{V}$ is just A. One can check that*

$$\mathbf{V} \models \neg \text{'}Atom\ is\ enumerable\text{'}.$$

We note, that by the downward Lövenheim-Skolem theorem there is also a computable $\mathbf{V}$ within which Atom is not enumerable. □

Theorem 15 *(on the existence on constructive models). Let $\mathbf{V}$ be a constructive model of $cFSA_\eta$. In $\mathbf{V}$ the $p\exists$-definable and computable classes are the same and therefore any correct presentation $\Phi : \sigma \to cTerm_\eta$ generates a constructive model $\mathbf{V}(Y\Phi)$ of σ in $\mathbf{V}$.*

This theorem plays an important role in the forthcoming chapters.

Definition 17. *Let $\eta_1 \supset \eta$ and let $\mu : \eta_1 \to cTerm_\eta$ be such a correct interpretation of η_1 in $cFSA_\eta$ that for all $\mathbf{V} \in Mod(cFSA_\sigma)$ $\mathbf{V}(\mu) \restriction_\eta = \mathbf{V}$. Moreover let μ be constructive. In this case μ is called a* **constructive extension of** *$cFSA_\eta$.*

Theorem 16. *Let $\eta_2 \supset \eta_1 \supset \eta_0$ and μ_{i+1} be a constructive extension of $cFSA_{\eta i}$ $(i = 0, l)$. μ_2 is a constructive extension of $cFSA_{\eta 0}$.*

Definition 18. *Let R be a set of new relation symbols. A rule is of the form*

$$\rho(\tau_1, \ldots, \tau_2) \Leftarrow \varphi$$

where $\tau_i \in Term_{\eta 1}$, $\varphi \in p\exists(0, \delta \cup R)$.

A logic program is a finite set of rules. A logic program u is called **well-formed** *iff whenever u contains two rules of the form*

$$\rho({\tau_1}^i, \ldots, {\tau_{ni}}^i) \Leftarrow \varphi_i \ (i = 1, 2)$$

then $\rho_1 = \rho_2 (\in R)$ implies $n_1 = n_2$ and $sort({\tau_j}^1) = sort({\tau_j}^2)$ $(j = 1, \ldots, n_1)$. Let $LP_\delta(R)$ denote the set of all well-formed logic programs.

Let $S = sort\ \delta$. Let S' be a copy of S with bijection ι. Define an S'-sorted similarity type σ_u for any $u \in LP_\eta(R)$. Take sort $\sigma_u \leftrightharpoons S'$. Define

$$rel\ \sigma_u \leftrightharpoons \{\rho \in R | \text{ there is a rule } \rho(\ldots) \Leftarrow \varphi \ \text{ in} u\} \cup rel\ \delta.$$

The arity of ρ is $\rho : \iota(s_1), \ldots, \iota(s_n)$ and u contains a rule of the form

$$\rho(\tau_1, \ldots, \tau_n) \Leftarrow \varphi$$

and for all $i = 1, \ldots, n \ \ \iota_i = sort(\tau_i)$. Since u is well-formed, this definition of arity is correct. Let

$$func\ \sigma_u \leftrightharpoons func\ \delta.$$

Let $X \leftrightharpoons (x_1, \ldots, x_n)$. Let $X \in \Gamma$ be a shorthand for $\bigwedge\{x_i \in \Gamma_{si} | i \in n\}$, where s_i is the sort of x_i. Let $\varphi[\Gamma]$ denote the relativization of σ along $\mu_{\sigma u}$.

Definition 19. *Let $u \in LP(R)$ be a well–formed logic program. By using u we define a presentation $u\hat{}$ in the following way:*

(a) Let $\rho \in rel\ \sigma_u$. Take
$u\hat{}(\rho) \leftrightharpoons$
$\cup \{\{x | \exists y_1, \ldots, \exists y_n \bigwedge\{y_i = \tau_i | i \in n\} \wedge \varphi[\Gamma]\} | \rho(\tau_1, \ldots, \tau_n) \Leftarrow \varphi' \in u\}$

(b) $u\hat{}(s) \leftrightharpoons$
$\cup \{\{x | \exists x_1, \ldots, \exists x_n \ x = x_i \bigwedge(x_1, \ldots, x_n) \in \Gamma_\rho\} | \rho \in rel_{\sigma n} sort\ x_i = s\}$
$\cup \{\{f(x_1, \ldots, x_n) | X \in \Gamma\} | y \in func\ \sigma\ f : {s'}_1, \ldots, {s'}_n \to s\}$

(c) $u\hat{}(s) \leftrightharpoons$
$\{(x_1, \ldots, x_n, y) | y = f(x_1, \ldots, x_n) \wedge X \in \Gamma\}$

where $f : {s'}_1, \ldots, {s'}_n \to s'$.

Proposition 3. *Let u be a well–formed logic program. Then $u\hat{}$ is a correct presentation with respect to μ.*

Definition 20. *Any logic program u generates a set of axioms $Ax(u)$ which is called the* ***logic generated by the logic program u*** *in the following way:*

$$Ax(u) \leftrightharpoons \{\varphi \to \rho(\tau_1, \ldots, \tau_n) | \rho(\tau_1, \ldots, \tau_n) \Leftarrow \varphi' \in u\}.$$

Similarly, you can give another set of axioms which describes how $u\hat{}$ was generated from the original sorts and functions. Let $Gen(u\hat{})$ denote this fact. Namely if $\sigma \leftrightharpoons \sigma_u$ then take:

$$\begin{aligned} Gen(u\hat{}) \leftrightharpoons\ & \{\mu_\sigma(s) \subset \mu(s) | s \in sort\ \delta\} \\ & \cup \{\mu_\sigma(f) \subset \mu(f) | f \in func\ \delta\} \\ & \cup \{\mu_\sigma(\rho) = \mu(\rho) \restriction \mu_\sigma(s_1) \times \ldots \times \mu_\sigma(s_n) | \rho : s_1, \ldots, s_n \in rel\ \delta\} \end{aligned}$$

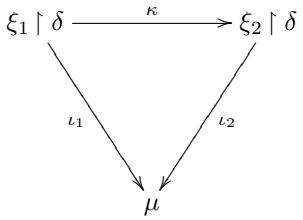
Theorem 17. *Let u be a well-formed logic program and $\mathbf{V} \in Mod(cFSA_\eta)$ be a model. $\mathbf{V}(Yu\hat{})$ is called* ***the denotation of u*** *and it is denoted by $Den_{\mathbf{V}}[\![u]\!]$. Clearly $Den_{\mathbf{V}}[\![u]\!] \in Mod_\sigma$. If $\mathbf{V}$ and $\mu : \delta \to cTerm_\eta$ are constructive then so is $Den_{\mathbf{V}}[\![u]\!]$.*

Theorem 18. *Let u be a well-formed logic program and fix a $\mathbf{V} \in Mod(cFSA_\eta)$. Take $\sigma \leftrightharpoons \sigma_u$. Let $\rho(\tau_1, \ldots, \tau_k)$ be a positive ground atomic formula from $Form_\sigma$. Then*

$$\begin{aligned} & Den_{\mathbf{V}}[\![u]\!] \models \rho(\tau_1, \ldots, \tau_k)\ \textit{iff} \\ & Th(\mathbf{V}) + IM?(\mu_\sigma) + Gen(u) \vdash Ax(u)^\sigma \to \rho(\tau_1, \ldots, \tau_n)^\sigma \end{aligned}$$

To formalize the initial property of $Den_{\mathbf{V}}[\![u]\!]$ let us consider the following category $\mathbf{C}$. Let us fix a model $\mathbf{V} \in Mod(cFSA_\sigma)$. The objects of $\mathbf{C}$ are pairs (ξ, ι) where $\xi : \sigma_u \to cTerm_\sigma$ and $\iota : \xi \restriction func\ \sigma \to \mu$ is a morphism between two interpretations in $\mathbf{V}$. Moreover, we suppose that ι is an embedding.

The morphisms of $\mathbf{C}$ are the morphisms $\kappa : \xi_1 \to \xi_2$ which commute with ι's:



This category is called **the model category generated by u** and denoted by $\mathbf{C}(u)$.

Theorem 19. *$Den_{\mathbf{V}}[\![u]\!]$ with the natural injection is the initial object in $\mathbf{C}(u)$.*

We show that logic programming based on Horn-formulas is only a particular case of our definition. Indeed let δ be a similarity type such that rel δ is empty. Let $\underline{Herb}(0, \delta)$ be the minimal Herbrand model generated from δ in $cFSA_\eta$. Clearly, there is an interpretation $\Theta : \delta \to cTerm_\eta$ which defines this model. Clearly Θ is constructive in $cFSA_\eta$.

Definition 21. *A well-formed logic program $u \in LP_\delta(R)$ is called a* **Horn-type logic program** *iff each rule $\rho(\tau_1, \ldots, \tau_n) \Leftarrow \varphi$ belonging to u is Horn-type, i.e. $\varphi \to \rho(\tau_1, \ldots, \tau_n)$ is a Horn-formula.*

It is clear that the mathematical tools defined in Chaps. 5.1–5.3 fits well for Horn-type logic programs. By using Theorem 5.29 and the construction of Θ we can give another version of initiality. Let $u \in LP_\Theta(R)$ be a fixed Horn-type logic program. Take $\sigma \leftrightharpoons \sigma_u$. Let $\mathbf{V}$ be a fixed constructive model of $cFSA_\eta$.

Theorem 20. *Let $\mathbf{C}$ be the category the objects of which are σ-type correct interpretations in a constructive model $\mathbf{V}$, and the morphisms of which are the interpretations between them. Then $\mathbf{V}(Yu\hat{})$ is the initial object of $\mathbf{C}$.*

Corollary 1. *Let u be a well-formed Horn-type logic program. Let $\rho(\tau_1, \ldots, \tau_n)$ be a positive ground atomic formula from $Form_\sigma$.*

$$cFSA_\eta \models \rho(\tau_1, \ldots, \tau_n) \text{ iff}$$
$$cFSA_\eta + IM?(\mu_\sigma) \vdash Ax(u)^\mu \to \rho(\tau_1, \ldots, \tau_n)^\mu$$

Moreover, let $\mathbf{V}$ be a fixed model of $cFSA_\sigma$. Then

$$\mathbf{V}(Yu\hat{}) \models \rho(\tau_1, \ldots, \tau_n) \text{ iff}$$
$$Th(\mathbf{V}) + IM?(\mu_\sigma) \vdash Ax(u)^\mu \to \rho(\tau_1, \ldots, \tau_n)^\mu$$

By the use of a fixed point theory, which allows us to have solutions definable in $cFSA_\sigma$ we will be able to work with definable least fixed points. To achieve this we use the positive existential (or constructive) functionals over which the fixed point theory with the usual properties may be developed. Note that this functional class consists of only the functions considered as computable. By the use of this fixed point theory we get the traditional logic programming case. Let us suppose that η' contains finitely many new constant symbols: $A_1, \ldots, A_n$. Let us denote the formula

$$\bigwedge\{\underline{atom}(A_i) | i = 1, \ldots, n\} \wedge \forall x \underline{atom}(x) \to \bigvee\{x = A_i | i = 1, \ldots, n\}$$

by $\underline{Const}$.

Theorem 21. *$cFSA_\eta{}' + \underline{Const}$ has an initial term model $\mathbf{V}_i$.*

Let μ be the same as in the previous example.

Theorem 22. *Let u be a well-formed logic program. Let $\rho(\tau_1, \ldots, \tau_n)$ be a positive ground atomic formula from $Form_\sigma$. Then*

$$\mathbf{V}_i(u) \models \rho(\tau_1, \ldots, \tau_n) \text{ iff}$$
$$cFSA_\eta + IM?(\mu_\sigma) + \underline{Const} \vdash Ax(u)^\mu \to \rho(\tau_1, \ldots, \tau_n)^\mu$$

3.2.3 Other Programming Paradigm

The proposed theoretic frame is appropriate to describe and analyse various other approaches in programming. We mention here only the rough set theory, which was proposed as an approach to support intelligent data analysis and data mining. Approximation is the basic concept of rough set theory. Let us suppose that a set X should be described with the terms of attribute values from a given set A. Then according to the rough set theory two operations are defined assigning to every X two sets A*(X) and A*(X) called the A-lower and the A-upper approximation of X, respectively. Note that the A-lower approximation of a set is the union of all A-granules that are included in the set, whereas the A-upper approximation of a set is the union of all A-granules that have a nonempty intersection with the set. Rough set theory gives us one of the important backgrounds for that type of computing when the aim is to deal with inexact solutions of computational problems. Rough set theory plays an important role in granular computing. The basic ingredients of granular computing are granules such as subsets, classes, objects, clusters, and elements of a universe. These granules are composed of finer granules that are drawn together by distinguishability, similarity and functionality. Based on complexity, abstraction level and size, granules can be measured in different levels. A problem domain may exist at the highest and coarsest granule. Granules at the lowest level are composed of elements of the particular model that is used. Granulation is one of the key issues in granular computing for problem solving. See e.g. Akama et al. [2], Kumar et al. [18] and Yao [30]. Note that the two-sided approximation of the sets used by rough set theory can be provided by the fixed point equations. The approximation itself is realized by the smallest and largest fixed points. At the same time the granules to be used in the approximation will be given in the universe that will correspond to the actual problem domain.

4 Cognitive Computing

4.1 Motivations

Computing is the basic method for representing, model and investigate processes of human intelligence in the fields of Artificial Intelligence, Cognitive computing and Computational theory of mind. Thus, our goal is to develop a general computation theory that considers all the important aspects of this modelling. It is essential to handle the various levels of computation, from computation that uses purely syntactic digits to computation at the content level that among others interprets data, information and knowledge. Our further aim is to provide a theoretical framework, which will able to consider all the processes of the data $\rightarrow$ information $\rightarrow$ knowledge transformation and processing.

The theoretical framework - developed above - will permit to represent, model and investigate the main processes of cognizing under the control of direct thinking by the use of appropriately defined and constructed computing, which we call cognitive computing.

4.2 Basic Definitions

Let $Th = (\sigma, Ax)$ be a fixed theory. Cognitive processes are defined by the use of situations, infons and information. The proposed extension will formalise the well-known constructions of situations and information. See Devlin [9] or Barwise [5]. Let ι be a new similarity type containing the followings.

1. A sort s for **situations**,
2. A sort b, i for **basic infons** and **infons** respectively,
3. A sort tv for **truth values**,
4. A functional symbol $\kappa : s, i \to tv$. κ gives the truth value to an infon in a situation
5. A sort k for **knowledge**.
6. A function $\tau_i : k, s \to i$. τ is the **query function** which in a given situation produces an infon by the use of the actual knowledge.

Let us add an axiom stating that **infons** form the greatest set containing all the basic infons and if A is an infon and S is a situation then the triple κ, S, A is also an infon. Let $\mathcal{I}$ be the fixed-point equation describing this. According to Barwise [5] well-founded infons are the elements of the least fixed-points of the equation $\mathcal{I}$. Let BC_ι denote this set of axioms.

It is clear that $Th \to Th + BC_\iota$ form a constitution on $\mathbb{FOL}$. According to Theorem 1 there is a least compact constitution containing this constitution. Let $\mathbb{CB}$ denote this constitution. $\mathbb{CB}$ is the so-called **cognitive base constitution** (compact by definition).

4.3 Cognitive Processes

We recall the **modification calculi** from Anshakov, Gergely [4] see Chapters 10 and 18. To treat non-monotonity of reasoning process we need to

1. differentiate external and internal truth values
2. add a new type r for **reasoning** to clarify why we think that about the truth value of an infon in a given situation
3. have inference rules for handling records which can contain infons, information and/or knowledge (see Anshakov, Gergely [4] pp. 145–149).

Now we are ready to give a short description of what we mean by **cognitive process**. Again, fix a theory $Th = (\sigma, Ax)$. Using the superstructure $SP_{\mathbb{CB}}(Th)$ a cognitive process

1. is a logic program in the sense of 3.2.2, its inner logic is the above stated modification calculus. This is able to extract infons, to generate new information and knowledge.
2. is also a process that can generate new processes, start, terminate and eliminate processes and can communicate with other processes.
3. can have a lot of query functions to interact with its environment.
4. can have a predefined goal.

Let PC_ι denote the axioms describing the aboves. Again we can define a closed constitution $\mathbb{CK}$ over $\mathbb{FOL}$ being the least compact-one containing the axiom systems BC_ι and PC_ι.

Note that only those processes are called cognitive processes, which can generate, store and use new elements of the sort k, i.e. new information and knowledge elements.

Now we define the important notion Cognition Kernel as follows.

Definition 22. Cognitive Kernel *is a compact constitution $\mathcal{C}$ on $\mathbb{FOL}$ together with an embedding (in functor category) of $\mathbb{CK}$ into itself.*

5 Cognitive Computing

A goal-oriented organisation of cognitive processes form cognitive computing. The goal is usually related to the solution of a given problem situation, i.e. to the reduction of the uncertainty level of a problem situation.

The Cognitive computing theory will provide a mathematically well-defined classification of the possible types of computing and it will consider a special theory of realisation. The Cognitive computing theory allows the investigation and the determination of the theoretical limitations of the computing based modelling of intelligent processes. Thus in addition to the computational capabilities and limitations of different programming paradigms, the multi-layer cognitive computing can be explored and a framework can be developed that support the realization of the data → information → knowledge transition processes that use thought-driven cognitive processes. At the beginning generic and/or specific data analytical methods will provide infons from the data and then from the generated set of infons the corresponding information will be built. This information is used to decrease the uncertainty of the actual problem situation. The methods and information successfully used in the solution of the problem situation will form knowledge candidates. The latters will become knowledge only after a successful checking done by the use of the existing knowledge repository.

In this context, generalizations of the concept of Church- and Turing-computa-bility can be given and the computational possibilities and limitations of cognitive computing can be investigated. A multilayer theory of complexity can be defined to characterise the problem situations. It can be shown that cognitive processes necessary for the solution of problems of certain complexity will not be cognitive computable, but they will become so by a cognitive computing system with oraculum.

6 AI Based on Cognitive Computing

The proposed cognitive computing theory permits to design systems, which are able to lean on and interact naturally with users to extend the capability of humans and/or machines. So that humans would able to do more of what they could do on their own normally without cognitive computing support.

A cognitive computing system will be able to respond to the environment in an autonomous regime too, without pre-programming. It can sense, learn, infer and interact. Cognitive computing systems can sense or perceive the environment and collect the data on the basis of needs and situations. They understand, interpret and analyse the context based on collected data and information, and they make decision based on reasoning and act accordingly. Various semantics and knowledge-driven cognitive data analysis methods can be represented and realised in a constructive way within the proposed Cognitive computing approach.

Cognitive computing theory provides a possibility for the use of computational approach to realise the understanding processes of natural language. Namely, this theory provides tools to interpret natural language texts in the various levels of cognitive computing.

The development of cognitive computing covers the basics of computing from language design to evaluator implementation with the aim of explaining existing systems at a deep enough level. This will help to adopt and use any of both, the languages and systems that are currently used in artificial intelligence and cognitive system area thus enabling the next generation of cognitive computing designers and implementers to use this as a foundation to build upon which. This is associated/augmented with a methodology that supports the selection of an appropriate specification method together with a constructive language that permits to describe the problem situation so that it prescribes the realization of the cognitive computing processes necessary for the solution of the actual problem situation.

Therefore, the proposed cognitive computing theory provides a constructive foundation of AI, where this abbreviation means Amplifier for Intelligence. This AI will be able to act as genuine problem-solving companion understanding and responding to complex problem situations. This AI system will be able to act either as a partner system for cooperative functioning with a human agent or as an autonomous cognitive system for a well-defined problem area. As to become a cooperative partner for human agents, the system has to function very similarly to humans. E.g., it should be able to communicate and understand natural language, reason in a compatible way, learn from its experience, etc. AI will be able to cognize the environment and itself including the co-operative partner.

7 Appendixes

7.1 Axiomatization of *clFSA*

In order to interpret program execution and different data and control structures, our theory of programming needs appropriate models to be obtained from a relation structure (models) of a given similarity type by building up the corresponding superstructure as we have seen so far. However, to use these superstructures in our theory of programming we have to introduce an appropriate formalism which allows to provide a theory (an axiomatization) the models of

which are the structures in question and by the use of which, statements can be formulated and proved about these structures. According to our aim to develop a first order theory of programming the axiomatization of superstructures will be done in an appropriate first order language.

Superstructure construction is followed by a set–theoretic approach so the signature of the language has to contain at least:

– a unary relation symbol $\underline{atom} : d$

to distinguish the elements of the original relation structure from which the superstructure is built up. These elements may be considered as elementary data;

– the 'element' relation symbol $\in: d, d$
– and the constant symbol $0 :\rightarrow d$

which reflects the empty set.

A similarity type σ is called **rich enough** iff it contains the above symbols. We use $\notin$ for the negation of element relation.

The variable symbols of the language correspond to sets and atoms. Therefore, sets and their elements are of the same nature, if the latter ones are not atoms. In other words, we consider hereditary sets the elements of which are either atoms or hereditary sets etc. Though atoms have no elements they are not equivalent to the empty set. Therefore we have to be careful while providing the Axiom of Extensionality and defining some of the set-theoretic operations.

A relation symbol of σ is called non set-theoretical iff it is not equal with either $\in$ or 0 or to $\underline{atom}$.

The system of axioms FSA_σ axiomatizes the hereditarily finite sets with atoms. Why do we need atoms? As we know the Zermalo-Fraenkel axiomatization is powerful enough to make atoms unnecessary. Set theory, as formalized in ZFC, provides an elegant and powerful way to organize mathematics but it is too strong for the programming theory. The aim to build up an adequate axiom system for this theory dictates to develop a set theory weaker than ZFC, weak in the principles of set existence which they attempt to formalize e.g. by allowing atoms. The latters just have a programming interpretation as elementary data or, if you think about the relation structures as the object modelling computers where the programs run then atoms represent the registers where the data are stored. Atoms also break the finiteness which we intend to axiomatize since they may be infinitely many. We axiomatize the hereditarily finite sets with atoms by modifying the axiom system ZF as follows:

FSA_0: Existential axiom of atoms:
$\exists x\ \underline{atom}(x)$

FSA_1: Extensionality axiom:
$(\neg\underline{atom}(x) \wedge \neg\underline{atom}(y)) \rightarrow ((x = y) \leftrightarrow \forall z(z \in x \leftrightarrow z \in y))$

FSA_2: Empty set axiom:
$\forall x(\neg x \in 0)$

FSA_3: Significance axiom of atoms:
$\forall z(atom(z) \leftrightarrow (z \neq 0 \wedge \forall x(x \notin z)))$

This axiom together with FSA_1 declares that though atoms have no elements they differ from the empty set 0.

FSA_4: Foundation axiom:
$\forall x(\exists y(y \in x) \rightarrow \exists y(y \in x \wedge \neg \exists z(z \in x \wedge z \in y)))$

This axiom says that every non-empty set has a minimal element with respect to $\in$.

$FSA_{5\sigma}$: Comprehension Scheme. For each σ-type formula φ:
$\forall z \forall w_1 \ldots \forall w_n \exists y \exists x(x \in y \leftrightarrow [x \in z \wedge \varphi(x, z, w_1, \ldots, w_n)])$

The y asserted to exist is unique by Extensionality Axiom and it is denoted by

$$\{x | x \in z \wedge \varphi(x, y, w_1, \ldots, w_n)\} \text{ or } \{x \in z | \varphi\}.$$

Intuitively for a given formula $\varphi(x)$ there need not necessarily exist a set $\{x : \varphi(x)\}$ this collection may be too large to form a set. However, Comprehension Scheme says that if the collection is a subcollection of a given set then it does exist. The following axioms say that certain sets, which should exist, really do exist.

FSA_6: Pairing axiom:
$\forall x \forall y \exists z(x \in z \wedge y \in z)$
FSA_7: Union axiom:
$\forall F \exists G \forall y((y \in F \wedge x \in y) \rightarrow x \in G)$
$FSA_{8\sigma}$: Replacement scheme. For each σ-type formula φ:

$$\forall F \forall w_1 \ldots \forall w_n(\forall x \in F \exists! y\ \varphi(x, y, F, w_1, \ldots, w_n) \rightarrow$$
$$\exists G(\forall x \in F \exists y \in G\ \varphi(x, y, F, w_1, \ldots, w_n))$$

Intuitively (using also $FSA_{5\sigma}$) this axiom says that if $H(x)$ is the unique y satisfying $\varphi(x, y, \ldots)$ then $\{H(x) | x \in F\}$ is a set.

FSA_9: Finiteness axiom:

$$\forall x(set(x) \rightarrow \exists y \exists z(z \text{ is a bijection between } x \text{ and } y) \wedge$$
$$(y \text{ is finite ordinal}))$$

Definition 23. *The set of axioms of the hereditarily finite sets with atoms is:*

$$FSA_\sigma \leftrightharpoons \{FSA_i | i = 0, 1, 2, 3, 4, 6, 7, 9\} \cup \{FSA_{i\varphi} | i = 5, 8,\ \varphi \in Form_\sigma\}.$$

If $R_1, \ldots, R_k; f_1, \ldots, f_n$ are new relation and function symbols respectively then $FSA_\sigma(R_1, \ldots, R_k; f_1, \ldots, f_n)$ stands for FSA_{σ^*} where $\sigma^* = \sigma \cup (R_1, \ldots, R_k; f_1, \ldots, f_n)$.

The Finiteness Axiom implies the Axiom of Choice, i.e.:

Proposition 4. *In FSA_σ the following statements hold:*

(i) Each set can be well-ordered.
(ii) There exists a choice function on sets.

Moreover, basically from the Finiteness Axiom, it follows that for any set there exists the power set, i.e. the Power Set Axiom is a consequence of FSA_σ.

Proposition 5. *The axioms of FSA_σ ensure that for any set there exists the power set:*

$$FSA_\sigma \models \forall x(\neg \underline{atom}(x) \rightarrow \exists y \forall z(z \in y \leftrightarrow z \subset x)).$$

Therefore, the axiom system FSA_σ is equivalent with the Zermalo-Fraenkel axiom system with the Axiom of Infinity and the Power Set Axiom deleted and Finiteness Axiom added.

Note that all the notions introduced in ZFC can be introduced in FSA_σ as well. E.g. relation, domain, range, function, bijection, surjection, injection are such notions. The expressions that define these notions can also be used as a definition of new relation or function symbols. Adding these new symbols to the similarity type o and their definitions to FSA_σ, we obtain a conservative extention of FSA_σ.

Since in FSA_σ all sets are finite, therefore the following preposition holds for ordinals:

Proposition 6. *Each ordinal is finite in FSA_σ and the usual addition and multiplication on ordinals are commutative.*

Note that in set theory natural numbers are identified with finite ordinals. Namely, an ordinal α is a natural number if for all $\beta \leq \alpha$ if $\beta \neq 0$ then β is a successor of some γ. Again the Finiteness Axiom implies the following:

Proposition 7. *The natural numbers, the ordinals (and the cardinals) are the same in FSA_σ.*

A programming theory needs, among others, tools to handle infinite objects e.g. to represent infinite computation processes. Therefore, beyond finite sets as finite objects, we also have to be able to speak about infinite objects.

Different approaches provide different techniques for this aim, e.g. denotation approach to semantics makes the topological space complete. We introduce the notion of class to handle infinity. This notion is also important in ZFC axiom system, where e.g. the class of all ordinals On is often used. In the axiom system FSA_σ the notion of class has a more important role, since the majority of the usual sets (namely, all infinite sets) cannot be identified with any set in FSA_σ.

Intuitively, a class is but a conglomerate of elements x which satisfy a given formula $\varphi(x)$. Since each σ-type model of FSA_σ has constructive objects (atoms or finite sets) as elements a class is but a defined or specified conglomerate of

these objects. Due to the significant role of classes in our further investigations they will frequently appear and we therefore have to precisely define what type of statements can be stated about classes. The definition is based on the followings. Having a given model a class does not consist of arbitrarily collected elements of the universe, but they may be collected only by the use of a given formula. Therefore, we extend the language such that it may contain statements about classes. Let us first fix an arbitrary set cV of the so called class variable. The intended meaning of a class variable in a model of FSA_σ is a conglomerate of objects of the universe.

Let σ be a rich similarity type. In order to handle classes we extend the σ-type classical first order language by adding class terms which provide definable conglomerate of objects of the universe of the models and the class formulas which allow to formulate statements about classes.

Definition 24. *(a) The set of* σ**-type class terms** *(***cTerm**$_\sigma$*) consists of the terms in the form* $\{x|\varphi\}$ *where* $x \in V$ *and* $\varphi \in cForm_\sigma$.

(b) The set of σ**-type class formulas** *(***cForm**$_\sigma$*) is the minimal set satisfying the followings:*

- $Atom_\sigma \subset cForm_\sigma$;
- *if* $x \in Term_\sigma$ *and* $C \in cV$ *then* $x \in C$ *belongs to* $cForm_\sigma$;
- *if* φ, ψ *are of* $cForm_\sigma$ *then* $\neg\varphi$ *and* $\varphi \wedge \psi$ *also belong to* $cForm_\sigma$;
- *if* φ *is of* $cForm_\sigma$, $x \in V$ *then* $\exists v\varphi$ *also belong to* $cForm_\sigma$.

Now let us see how the semantics of $cForm_\sigma$ can be defined. First of all, we extend the notion of valuation for class variables. A valuation of a class variable is a subset of the universe of the model under consideration.

Definition 25. *Let* $\mathbf{A}$ *be an arbitrary but fixed* σ*-type model. A* **class valuation** *is a function* k *such that*

- $dom(k) = C \cup cV$;
- $k(x) \in A$ *for any* $x \in V$;
- $k(x) \in Sb(A)$ *for any class variable* $x \in cV$.

As usual let $cVal_\mathbf{A}$ *denote the family of all class valuations.*

Having the valuation we can define the meaning of class terms and class formulas with respect to a given valuation in an arbitrary but fixed model.

Definition 26. *Let a model* $\mathbf{A} \in Mod_\sigma$ *be given.*

(a) The meaning of a class term $t = \{x|\varphi\}$ *in the model* $\mathbf{A}$ *with respect to* k *is the following family:*

$$t_\mathbf{A} \leftrightharpoons \{a|\mathbf{A} \models \varphi[k + (x, a)]\}.$$

(b) For any class formula $\varphi \in cForm_\sigma$ *and valuation* $k \in cVal_\mathbf{A}$ *we define the validity* φ *in* $\mathbf{A}$ *with respect to* k *(written as* $\mathbf{A} \models \varphi[k]$*) by induction on the complexity of* φ*:*

if $\varphi \in Atom_\sigma$ then the validity is defined as in (i) and (ii) of 1.2.10; if $\varphi \leftrightharpoons \tau \in C$ then $\mathbf{A} \models (\tau \in C)[k]$ iff $\tau_{\mathbf{A}}[k]$ is an element of $k(C)$; and (iv) are defined as in (iii) of 1.2.10 (i.e as usual).

In order to make the class valuation more transparent the σ-type models may be extended such that they will contain an entity which refers to classes.

Definition 27. *A pair $\mathbf{V} = (\mathbf{A}, \underline{Class})$ is called a class extention of $\mathbf{A}$ iff*

- *$\mathbf{A}$ is a σ-type model;*
- *$\underline{Class} \subset Sb(A)$;*
- *for any class term $t \in cTerm_\sigma$ and valuation $k \in cVal_{\mathbf{A}}$ we have $k(cV) \subset \underline{Class}$ implies $t_{\mathbf{A}}[k] \in \underline{Class}$.*

Let $cMod_\sigma$ denote the family of all class extentions of σ-type models. If $\mathbf{V} \in cMod_\sigma$ then let $Val_{\mathbf{V}}$ denote those class valuations k for which $k(cV) \subset \underline{Class}$ holds.

Definition 28. *Let $\mathbf{V} = (\mathbf{A}, \underline{Class})$ be an arbitrary class extention belonging to $cMod_\sigma$. Let $\varphi \in cForm_\sigma$.*

(i) We say that φ ***is valid in V with respect to a valuation k*** *iff $k \in Val_{\mathbf{V}}$ and $\mathbf{V} \models \varphi[k]$.*

(ii) The class formula φ is valid in $\mathbf{V}$ iff it is valid with respect to all $k \in Val_{\mathbf{V}}$ i.e.

$$\mathbf{V} \models \varphi \text{ iff for all } k \in Val_{\mathbf{V}}, \mathbf{V} \models \varphi[k]$$

(iii) A class formula φ is said to be valid in a σ-type model $\mathbf{A}$ iff for all class extention $\mathbf{V} = (\mathbf{A}, \underline{Class})$ we have $\mathbf{V} \models \varphi$.

So we have defined the σ-type class language as a triple

$$CL_\sigma \leftrightharpoons (cForm_\sigma, cMod_\sigma, \models)$$

This language is really a two-sorted one. The first sort corresponds to sets and atoms and the second one to classes. However, quantification is allowed only for set variables.

A variable $x \in V \cup cV$ can also be considered as a shorthand for the class term $\{y | y \in x\}$ To have a clearer view of a variable let us see how the "element relation" $\in$ and the equality are defined for class terms.

- $\{x|\varphi\} \in \{y|\psi\} \leftrightharpoons \exists y(\forall x(x \in y \leftrightarrow \varphi) \wedge \psi)$;
- $\{x|\varphi\} = \{y|\psi\} \leftrightharpoons \forall x(\varphi \leftrightarrow \psi[x/y])$.

Depending on how we look at x as a variable or a=j a shorthand for $\{y|y \in x\}$ the class formulas $x \in y$ and $x \in C$ have different meanings. However, they are equivalent if we take the Extensionality Axiom (FSA_1).

Proposition 8. *(i) $FSA_1 \models (x \in y) \leftrightarrow (\{z|z \in x\} \in w|w \in y)$*

(ii) $FSA_1 \models (x \in C) \leftrightarrow (\{z|z \in x\} \in \{w|w \in C\})$

Proof. We prove only the statement (i). Working in axiom system $\{FSA_1\}$ *we have the following chain of semantic equivalences:*

$$\{z|z \in x\} \in \{w|w \in y\} \equiv \exists u(\forall z(z \in x \leftrightarrow z \in u) \wedge u \in y) \equiv u(x = u \wedge u \in y) \equiv x \in y.$$

□

By using the above proposition we can define the following abbreviations:

$C = D \leftrightharpoons \forall x(x \in C \leftrightarrow x \in D)$
$C = y \leftrightarrow \forall x(x \in C \leftrightarrow x \in y)$
$C \in x \leftrightharpoons \exists y(C = y \wedge y \in x)$
$C \in D \leftrightharpoons \exists y(C = y \wedge y \in D)$

For the classical first order languages we have defined the simultaneous substitution of terms. This notion can be extended even to the class language. However, we have to make a careful distinction between substitution for variables and for class–terms. The only question is how substitute into a formula $x \in C$?

First let τ be a term belonging $Term_\sigma$. If so then take $(x \in C)[\tau/C] \leftrightharpoons x \in \tau$. Clearly $x \in \tau$ belongs to $cForm_\sigma$. In the case of class terms take $(x \in C)[\tau/C] \leftrightharpoons \varphi[x/y]$ where $\tau = \{y|\varphi\}$. This definition is comform with the fact that $x \in \{y|\varphi\}$ is just a shorthand for $\varphi[x/y]$. In the end if D is a class variable then take $(x \in C)[D/C] \leftrightharpoons x \in D$.

Without spelling out the whole definition we use the notation $\varphi[\tau_i/x_i]_i^k$ for the simultaneous substitution of terms τ_i for variables x_i, respectively.

Lemma 1. *Let* $\mathbf{V} \in cMod_\sigma$ *and let* $\varphi \in cForm_\sigma$ *be arbitrary. Let us suppose that* $var(\varphi) \cap cV = \{C_1, \ldots, C_k\}$. *Then for any class terms* $\tau_1, \ldots, \tau_n$ *if* $\mathbf{V} \models \varphi$ *then* $\mathbf{V} \models \varphi[\tau_i/C_i]_i^k$.

In order to handle classes axiomatically the axiom system FSA_σ is to be appropriately extended. The extended axiom system denoted by $cFSA_\sigma$ consists of

– axioms which remain the same as they were in FSA_σ dealing with sets only;
– axioms the scope of which is extended to classes;
– the extentions of the axiom schemas by allowing class formulas.

Namely, by extending the axiom system we get the following axiom system $cFSA_\sigma$ where variables $f, g, x, y, z, w, w_1, \ldots, w_n$ are from V and C, D are from cV.

FSA_0 Existential axiom of atoms:
$\exists x\ \underline{atom}(x)$
FSA_1 Extensionality axiom:
$(\neg\underline{atom}(x) \wedge \neg\underline{atom}(y)) \rightarrow ((x = y) \leftrightarrow \forall z(z \in x \leftrightarrow z \in y))$
FSA_2 Empty set axiom:
$\forall x(\neg x \in 0)$

FSA_3 Significance axiom of atoms:
$\forall z(atom(z) \leftrightarrow (z \neq 0 \wedge \forall x(x \notin z)))$

This axiom together with FSA_1 declares that though atoms have no elements they differ from the empty set 0.

FSA_4 Foundation axiom:
$\forall z(\exists x(x \in z) \rightarrow \exists x(x \in z \wedge \neg\exists y(y \in x \wedge y \in z)))$
$FSA_{5\sigma}$ Comprehension schema:

$$\forall z \forall w_1 \ldots \forall w_n \exists y \exists x(x \in y \leftrightarrow [x \in z \wedge \varphi(x, z, w_1, \ldots, w_n)])$$

(where $\varphi \in cForm_\sigma$!)
FSA_6 Pairing axiom:
$\forall x \forall y \exists z(x \in z \wedge y \in z)$

We remark that since classes have only sets as elements the Pairing Axiom is not extended to classes.

Before reformulating FSA_7 we remark that for any term t one can define its union by taking $\cup t = \{x | \exists y(x \in y \wedge y \in t)\}$. However, our original axiom states that if t is not proper then $\cup t$ is also not proper! Therefore, we have to use FSA_7 without any changes.

FSA_7 Union axiom:
$\forall x \exists z[\forall y \forall w(y \in x \wedge w \in y) \rightarrow w \in z]$
$FSA_{8\sigma}$ Replacement scheme. For each σ-type formula φ:

$$\forall f \forall w_1 \ldots \forall w_n(\forall x \in f \exists! y\ \varphi(x, y, f, w_1, \ldots, w_n) \rightarrow$$
$$\exists g(\forall x \in f \exists y \in g\ \varphi(x, y, f, w_1, \ldots, w_n))$$

(where $\varphi \in cForm_\sigma$)

Similarly to the modification of the Comprehension schema we allow the use of class formulas in the scope of the Replecement schema as well.

FSA_9 Finiteness axiom:
Each set is equivalent with a finite ordinal

Now having the axiom system $cFSA_\sigma$ we clarify some notions. Let $\mathbf{V} = (\mathbf{A}, \underline{Class})$ be an arbitrary class extention of $\mathbf{A}$.

- The elements of A are called **objects**.
- Let $a \in A$ be an object. If $\mathbf{V} \models \underline{atom}(x)[(a, x)]$ then a is said to be an $\underline{atom}$, otherwise it is a $\underline{set}$.
- The elements of $\underline{Class}$ are said to be classes.
- An object a and a class U are called equal iff $\mathbf{V} \models \forall y(y \in x \leftrightarrow y \in C)[(x, a) + (C, U)]$.
- A class $U \in \underline{Class}$ is called a **proper** class iff it is not equal to any object of $\mathbf{V}$. Namely U is a proper class if it satisfies the formula $\mathbf{V} \models \neg\exists x(x = C)[(C, U)]$.

We redefine predicate 'set' by taking:

$$\underline{cset}(C) \leftrightharpoons \exists x(x = C \wedge \neg \underline{atom}(x)).$$

The $\neg \underline{atom}(x)$ part of the conjunction is needed only when one substitutes a variable x for class-variable C:

$$\underline{cset}(C)[x/C] \equiv \exists y(y = x \wedge \neg \underline{atom}(y) \equiv \neg \underline{atom}(x)).$$

Next we omit 'c' from the name of this redefined predicate because on sets the new and old meanings are the same.

The predicate 'proper' can be defined by taking:

$$\underline{proper}(C) \leftrightharpoons \neg \exists x(x = C).$$

According to the above defined predicates a class term τ is called relational or functional iff $Rel(\tau)$ or $Func(\tau)$ holds respectively.

The followings are two useful proper classes:

- $\underline{Universe} = \{x | x = x\}$;
- $\omega = \{x | Nat(x)\}$.

References

1. A definition of AI: main capabilities and scientific disciplines, Brussels (2018). https://ec.europa.eu/digital-single-market/en/news/definition-artificial-intelligence-main-capabilities-and-scientific-disciplines
2. Akama, S., Murai, T., Kudo, Y.: Reasoning with Rough Sets - Logical Approaches to Granularity-Based Framework. Springer, Switzerland (2018). https://doi.org/10.1007/978-3-319-72691-5
3. Amir, A. et al.: Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores, In: Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN) (2013)
4. Anshakov, O., Gergely, T.: Cognitive Reasoning - A Formal Approach. Springer, Berlin (2010). https://doi.org/10.1007/978-3-540-68875-4
5. Barwise, J.: The situation in logic. CSLI Lecture Notes Number, vol. 17 (1989)
6. Brasil, L.M., et al.: Hybrid expert system for decision supporting in the medical area: complexity and cognitive computing. Int. J. Med. Inform. **63**(1), 19–30 (2001)
7. Goguen, J.A., Burstall, R.M.: Introducing institutions. In: Clarke, E., Kozen, D. (eds.) Logic of Programs 1983. LNCS, vol. 164, pp. 221–256. Springer, Heidelberg (1984). https://doi.org/10.1007/3-540-12896-4_366
8. Cognitive Catalyst. https://www.ibm.com/downloads/cas/OMZMGNP5
9. Devlin, K.: Logic and Information. Cambridge University Press, Cambridge (1991)
10. Fresco, N.: Physical Computation and Cognitive Science. Springer, Berlin (2014). https://doi.org/10.1007/978-3-642-41375-9
11. Gergely, T., Szőts, M.: Cuttable formulas for logic programming, In: Proceedings of the Symposium on Logic Programming, IEEE Press (1984)
12. Gergely, T., Ury, L.: Programming in topoi. a generalized approach to program semantics, In: Categorical and Algebraic Methods in Computer Science and System Theory, Herdecke, Germany (1980)

13. Gergely, T., Ury, L.: First-Order Programming Theories. EATCS Monographs on Theoretical Computer Science, vol. 24. Springer-Verlag, Berlin (1991). https://doi.org/10.1007/978-3-642-58205-9
14. Grätzer, G.: Universal Algebra, 2nd edn. Springer-Verlag, New York (1979). https://doi.org/10.1007/978-0-387-77487-9
15. Gutierrez-Garcia, J.O., Lopez-Neri, E.: Cognitive computing: a brief survey and open research challenges, In: 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence, pp. 328–333 (2015)
16. Hoare, C.A.R., Jifeng, H.: Unifying Theories of Programming. Prentice Hall, New Jersey (1998)
17. Ivancevic, V.G., Ivancevic, T.T.: Computational Mind - A Complex Dynamics Perspective, Studies in Computational Intelligence 60. Springer-Verlag, Berlin (2007). https://doi.org/10.1007/978-3-540-71561-0
18. Kumar, V.S., Dhillipan, J., Shanmugam, D.B.: Survey of recent research in granular computing. Int. J. Emerg. Technol. Comput. Sci. Electron. **24**(3), 976–1353 (2017)
19. Maibaum, T.S.E.: Role of abstraction in program development. In: Kugler, H.J. (ed.) Information Processing 1986. Elsevier Science Publisher, Amsterdam (1986)
20. Milkowski, M.: Explaining the Computational Mind. The MIT Press, Cambridge (2013)
21. Piccinini, G.: The computational theory of cognition. In: Müller, V.C. (ed.) Fundamental Issues of Artificial Intelligence, Synthese Library 376, pp. 203–221. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-26485-1_13
22. Schank, R.: The Cognitive Computer: On Language Learning and Artificial Intelligence. Addison Wesley, Reading (1984)
23. Schank, R.: The fraudulent claims made by IBM about Watson and AI (2019). https://www.rogerschank.com/fraudulent-claims-made-by-IBM-about-Watson-and-AI
24. Ury, L., Gergely, T.: A constructive specification theory. In: Declarative Systems Elsevier Science Publishers, pp. 33–83 (1990)
25. Valiant, L.G.: Cognitive computation. In: Proceedings of IEEE 36th Annual Foundations of Computer Science, Milwaukee, WI, USA, pp. 2–3 (1995)
26. Wang, Y., et al.: A layered reference model of the brain (LRMB). IEEE Trans. Syst. Man Cybern. (Part C) **36**(2), 124–133 (2006)
27. Wang, Y.: On cognitive computing. Int. J. Softw. Sci. Comput. Intell. **1**(3), 1–15 (2009)
28. Wang, Y.: On denotational mathematics foundations for the next generation of computers: cognitive computers for knowledge processing. J. Adv. Math. Appl. **1**(1), 121–133 (2012)
29. Wirth, N.: Algorithms + Data Structures = Programs. Prentice Hall, New Jersey (1976)
30. Yao, Y.: Artificial intelligence perspectives on granular computing. In: Pedrycz, W., Chen, S.-M. (eds.) Granular Computing and Intelligent Systems ISRL 13, pp. 17–34. Springer-Verlag, Berlin (2011). https://doi.org/10.1007/978-3-642-19820-5_2
31. https://www.lexico.com/en/definition/artificial-intelligence
32. https://www.britannica.com/technology/artificial-intelligence