Cognitive Programming

Loizos Michael[1], Antonis Kakas[2], Rob Miller[3], and Gyorgy Turán[4]

[1] Open University of Cyprus, loizos@ouc.ac.cy
[2] University of Cyprus, antonis@ucy.ac.cy
[3] University College London, r.s.miller@ucl.ac.uk
[4] University of Illinois at Chicago and MTA-SZTE
Research Group on Artificial Intelligence, gyt@uic.edu

Abstract. The widespread access to computing-enabled devices and the World Wide Web has, in a sense, liberated the ordinary user from reliance on technically-savvy experts. To complete this emancipation, a new way of interacting with, and controlling the behavior of, computing-enabled devices is needed. This position paper argues for the adoption of *cognitive programming* as the paradigm for this user-machine interaction, whereby the machine is no longer viewed as a tool at the disposal of the user, but as an assistant capable of being supervised and guided by the user in a natural and continual manner, and able to acquire and employ common sense to help the user in the completion of everyday tasks. We argue that despite the many challenges that the proposed paradigm presents, recent advances in several key areas of Artificial Intelligence, along with lessons learned from work in Psychology, give reasons for optimism.

1 An Emerging Need and The Overall Challenge

Today's huge market pressure for the use of smart systems by everyone and in every aspect of their daily life is forcing Artificial Intelligence (AI) to stand up and deliver. What was perhaps thought out of reach in the past needs to become a reality to satisfy the ever increasing desire of humans to use their new machines — computer devices linked with the Internet — in their everyday activities.

Unlike anything we have seen to date, this new vision of user-machine interaction will allow ordinary users without technical background to instruct or program their devices in a natural and personalized manner, and will allow the devices to assist (and enhance the abilities of) their users in dealing with everyday tasks. This *symbiotic* relation splits the burden of communication among the user and the device, offering a "programming paradigm for the masses", avoiding the extremes of using natural languages that are too complex for ordinary devices, or programming languages that are too complex for ordinary users.

Early examples of such interactions already exist, ranging from the personal assistant softwares provided by major smart-device manufacturers, to the (expected) applications for expert analysis of problems in specialized domains built on top of the Watson engine. But perhaps the clearest example of this emerging form of interaction, which we shall call ***cognitive programming***, is that

of searching for information on the World Wide Web. The use of web search engines constitutes a form of programming exercised by billions, independently of technical ability, through a programming language of keywords in natural language, in a manner compatible with the cognitive abilities of humans. Through their searches, users gradually develop a sense of how to improve the way they program or instruct the search engine with queries that achieve the users' intended aim. On the other side, search engines capture the preferences or typical behaviors of users, to help propose search queries or choose how to rank results.

We will refer to systems interacting with users through cognitive programming as ***cognitive systems***, as these systems are, in spirit at least, of the same kind as the cognitive systems proposed relatively recently in several works in AI; see, for example, the new journal of Advances in Cognitive Systems, the journal of Cognitive Systems Research, and works such as [26–28, 50].

Unlike work in existing autonomous agents / systems, we think of a cognitive system as having an operational behavior similar or parallel with that of a human personal assistant. Its domain of application is limited to certain common everyday tasks, and its operation revolves around its interaction with its user in a manner that is compatible with the cognitive reasoning capabilities of the latter. To understand (and correct when needed) the reasoning process of the system, the user expects the system to use *common sense* to fill-in important relevant information that the user leaves unspecified, and to be able to keep learning about the domain and the user's personal preferences through their interaction.

The goal for building systems that are *cognitively compatible with humans* ultimately imposes a set of considerations on cognitive programming, as this determines the communication channel between the user and the system. The overall challenge of developing the proposed paradigm of cognitive programming ultimately rests on fleshing out and addressing these considerations:

- Cognitive programming should be a process akin to human-human communication. The need for detailed operational instructions should be minimized.
- There should be a level of interaction between the user and the system where the two understand and can anticipate the behavior of each other.
- Cognitive compatibility with the user should be accommodated by acknowledging the central role that natural language has in human communication, and in the way humans store, retrieve, and use commonsense knowledge.
- Cognitive programs should develop incrementally to meet the aims of the user through an open-ended process. Cognitive systems should be able to learn, and be able to improve from their past interaction with the user.
- Cognitive programs should be robust, never failing, but continuously improving / completing their ability to offer personalized solutions to the user, while adapting to a possibly new or changing user position, stance, or profile.

The emphasis of this position paper is on describing the desirable characteristics and the technical challenges resulting from the aforementioned considerations. It examines the salient and foundational issues that need to be considered, and offers possible suggestions for a first version of a cognitive programming language. This proposal is grounded in our recent experience of trying to automate

the cognitive task of story comprehension,[5] and on the comparison of the resulting psychologically-informed approach with earlier work in AI for addressing other types of scientifically-oriented problems, such as problems of diagnosis and planning that span beyond the ordinary capabilities of human intelligence.

1.1 Scientific Position for Cognitive Programming

The scientific position underlying our approach and proposal for cognitive programming is that *symbolic AI* can offer the tools needed for the aforementioned considerations, as long as one *abandons the traditional view* of the role of logic for reasoning, and one is strongly guided by work in Cognitive Psychology. To a certain extent, then, this position takes us back to the early days of AI.

We embrace McDermott's view in his paper "A critique of pure reason" [31], that developing a logical theory alone — even a non-monotonic one — without consideration of the reasoning process can not lead to human commonsense intelligence. A vast amount of empirical work from Psychology (see, e.g., [12]) shows that commonsense inferencing has a looser form than that of scientific reasoning, and that the conventional structure and form of logical reasoning, as epitomized by mathematical or classical logic, is not appropriate. Given strong evidence from recent work in Psychology (see, e.g., [33]) in support of an argumentation-based theory for human reasoning, we adopt a form of argumentation as the basis for a cognitive system's reasoning process. Drawing from work in Cognitive Psychology (see, e.g., [13, 21, 23, 44]) on how human knowledge is (or might be) structured and used, we base our approach on the cognitive process of *comprehension*, within which logical inference is only one component.

Although work in logic-based AI may accept, to a certain extent, the need to deviate from strict logical reasoning (e.g., non-monotonicity, belief revision, logic programming), efforts to automate reasoning still typically proceed on the basis of developing proof procedures that are sound and complete against some underlying semantics of "ideal inferences". Unlike such work, on which cognitive programming may be based and from which it may be guided, cognitive programming shifts the emphasis from deep and elaborated reasoning to richly structured knowledge, assuming that commonsense intelligence resides in the "complexity of knowledge representation" rather than the "complexity of thought". As in many cases of Computer Science, data structures and data organizations matter and can make all the difference in having an effective and viable solution.

2 Computational Model and System Architecture

The central notion underlying the computation of a cognitive system is that of ***comprehension***, a notion adopted from story or narrative text comprehension in Cognitive Psychology (see, e.g., [22]). In our setting, comprehension proceeds

[5] The system *STAR: Story Comprehension through Argumentation*, along with benchmark stories and other material, is available at: http://cognition.ouc.ac.cy/narrative/

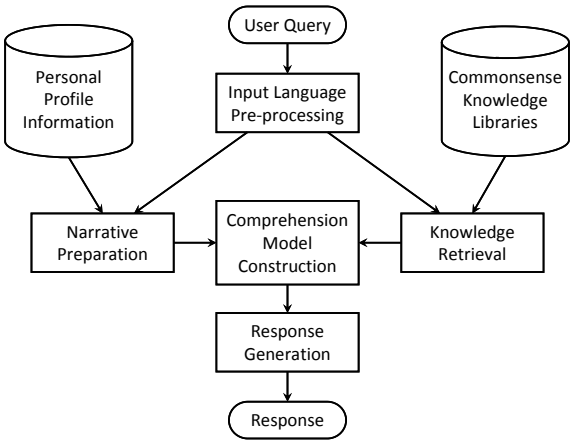

Fig. 1. General Architecture of a Cognitive System

by first combining the explicit input given by the user with information that is available in the user's profile (i.e., personal facts), forming an ***input narrative*** of the task at hand. This narrative is then synthesized with information that the system has about the domain (i.e., commonsense knowledge) and its user (i.e., personal preferences), leading to the construction of a ***comprehension model***.

A comprehension model is an elaboration of the input narrative with new information, or ***inferences***, capturing the (or a possible) implicit meaning or intention of the narrative. Critically, the comprehension model is ***coherent***, and includes only inferences that are important for successful understanding, while omitting cluttering details and speculations. If, for example, a user enquires for "private celebration of wedding anniversary", it is essential for the comprehension model to include the inference "place for two people", but not the side inference "married for at least one year" or the mere possibility "dinner at fancy restaurant".

The *central hypothesis* of our proposed cognitive programming framework is, then, that the availability of a comprehension model allows the system to better act and assist its user in the requested task. The general high-level architecture of cognitive systems that follows from this hypothesis is depicted in Figure 1.

We shall analyze the various components of this architecture in subsequent sections. For now, we shall discuss the interaction of the user with the system.

2.1 Cognitive Programming Interaction Modes

The most basic form of user-machine interaction is ***querying***, whereby the user, or the cognitive assistant of some other user, or even some sensor device, inputs a specific request or query to the cognitive system. The system then identifies or compiles relevant commonsense knowledge, perhaps even invoking a process of online learning, and responds with some action (e.g., a suggestion of whether to

accept or not an offer) that would help in addressing the task that has prompted the query. When an output is thus produced by the cognitive system, another form of interaction, that of ***supervising***, allows the user to give feedback to the system on the appropriateness of its output. For example, a user may override the suggestion or decision of a cognitive assistant with or without an explanation. The overridden output is then treated as training data for the system to learn (better) the user's personal opinion or preference on the particular case at hand.

Independently of any given query, the user may interact by ***personalizing*** the cognitive system through general statements about the user's preferences, such as "I like to spend the evenings with my family" or "Family is more important than work for me". The system responds by transforming such statements in an appropriate internal language, and recording them in the user's profile, which, in turn, personalizes other aspects of the user's interaction with the system.

In the context of a particular domain of application or discourse, interaction through ***guiding*** allows the user to offer general information that would aid the cognitive system to understand the salient aspects of the domain. Such information is also provided indirectly when, for instance, the user interacts with the system in any of the preceding ways. No matter how information is provided, guiding initiates a process to recognize concepts that are relevant and important for the user. In turn, this information can be used to prepare relevant knowledge on these concepts, by directing a background process of offline or batch learning of general commonsense knowledge that is related to the particular domain.

In what is arguably the lowest (i.e., closest to the machine, and analogous to the use of traditional programming languages) level of interaction, ***instructing*** allows the user to input particular pieces of knowledge to the cognitive system on how to operate or react under very specific circumstances. Such inputs are expressed in the system's internal language, and can be imputed directly in the user's personal profile or personalized knowledge libraries. We do not envisage that this would be the prevalent way of user interaction with cognitive systems.

2.2 Illustrative Example of a Cognitive System

Suppose that Bob wishes to manage his evening work appointments with the assistance of a cognitive system. He cognitively programs the system by guiding it with domain-specific information like "dinner plans, family time, work appointments, dietary constraints", prompting the system to gather relevant commonsense knowledge. Bob further personalizes the system with facts, such as "Bob is vegetarian", and preferences, such as "I like to spend evenings at home", "Customers from abroad are very important", and "I should never miss my children's birthday parties". Some of this latter type of information might have also been learned by the system by finding regularities in Bob's past queries to the system (e.g., if Bob often specified the keyword "vegetarian" in past queries), or through supervision of past proposed suggestions by the system (e.g., if Bob often declined suggestions by the system for late dinner outside his house).

When Bob's cognitive system receives a request from Bob's immediate boss, John, for "Working dinner today with John", the system combines this input with

facts in Bob's profile or other current information the system has from sensors, calendars, etc., to construct an expanded *input narrative*. This narrative is then comprehended through the use of the system's commonsense libraries, and the comprehension model is used to decide on whether the request is to be accepted.

If no additional information is given to the cognitive system, the system will reject the request, since having dinner with John would mean going to a restaurant that evening, which would conflict with Bob's preference to be at home in the evenings. Such inferences would be supported by commonsense knowledge of the form "Normally, working dinners are at restaurants", and "Normally, dinner is in the evening". In a more advanced case the system could generate alternative suggestions, such as to have dinner with John at home that evening. The request would also be rejected if the system were to receive from the calendar the information that "Today is the wedding anniversary of Bob", giving an additional reason for Bob's inability to have dinner with John, since "Normally, a wedding anniversary is celebrated privately"; this piece of common sense supporting the decision could be offered as an explanation of the system's response.

If (possibly after the initial rejection of the request) additional information is given that "John will be accompanied by important customers from abroad", this new piece of the *story* will be incorporated in the input narrative, leading to a *revision* of the comprehension model, and to the retraction of the system's earlier decision, as now the request is supported by Bob's preferences. The system would then suggest to accept the request, and perhaps reschedule the celebration of the wedding anniversary for another evening. Had further additional information been available that "Bob's son is having a birthday party tonight", a further revision would have been caused that would again reject the request, but possibly suggesting an alternative plan through the use of commonsense knowledge such as "Normally, a pre-dinner drink (and an apology) is an alternative to dinner".

3 Foundations of Cognitive Programming

What is an appropriate theoretical model of computation and semantics of programming that would underlie the development of the cognitive programming paradigm? What is the form of the internal language of the cognitive system, which would support the *computational cognitive metaphor* of story or narrative text comprehension as the central form of program execution? This internal language ultimately determines the *form of representation* of knowledge used by the cognitive system. Adopting a symbolic representation raises several questions: What is an appropriate logic and form of reasoning? Is logic alone sufficient to capture the cognitive requirements, such as that of a natural language user-interface and a computational model of comprehension? If not, what are the cognitive elements that would need to accompany a logical approach?

We turn again to Cognitive Psychology (see, e.g., [16, 21, 53]) for guidance:

- Knowledge is composed of *loose associations* between concepts, that, unlike logic rules, are stronger or weaker depending on the context.

– Reasoning gives rise to a *single comprehension model*, avoiding the cognitively expensive task of considering possible non-deterministic choices.
– Reasoning proceeds *lazily* by drawing only inferences that are *grounded* directly on the explicit concepts given in the narrative, in an *incremental* manner as parts of the narrative become available. When conflicting information is encountered, the comprehension model is suitably *revised* [43].
– *Cognitive economy* — necessitated by human cognitive limitations, which are bound to appear also in cognitive systems with massive knowledge libraries — is achieved by requiring the comprehension model to be *coherent*, including inferences that are tightly interconnected, and excluding inferences (even undisputed ones) that are peripheral to the understanding of the given narrative [1, 15, 32, 49], or to the completion of another cognitive task [45].

The above guidelines leave, nonetheless, several key issues on the treatment of knowledge unanswered. Below we elaborate on two of those: a more detailed view of knowledge representation, and the process of knowledge acquisition.

3.1 Representation of Cognitive Programs

In constructing the comprehension model, the cognitive system needs to *retrieve relevant commonsense knowledge* and possibly to *adapt* this to the narrative (and hence to the particular query and task) at hand for subsequent reasoning. This imposes two desired properties for knowledge representation that seem at odds with each other: knowledge should be represented in a fashion sufficiently flexible to be easily accessible and adaptable (e.g., in terms of the vocabulary and syntax being used), but at the same time knowledge should be represented in a fashion sufficiently concrete to be amenable to symbolic reasoning. We refer to this problem of representation as ***the challenge of knowledge plasticity***.

A way to address this challenge might be the adoption of *multiple representations* for the internal language of the cognitive system, and hence, of the commonsense knowledge that the system handles. Representations can exist, for instance, to capture a general categorization of the knowledge, typical or exemplar entities and situations, detailed knowledge for specific cases, etc. Perhaps the system's commonsense knowledge is represented at a more general and abstract level when it is initially acquired through offline or batch learning. When queries are provided by the user, a form of ***knowledge compilation*** might turn the relevant general knowledge into a task-specific form that can be directly used to link the knowledge with the input query (and resulting narrative) for reasoning.

How the knowledge is structured in such levels and how a user input is compiled down these levels to the specific one on which the execution / reasoning occurs presents one of the central challenges for cognitive programming. We posit that an *argumentation perspective* might be useful in capturing the important aspects of the most specific of these levels, where knowledge is already compiled into a form appropriate for formal reasoning. This representation framework falls under the general scheme of abstract argumentation frameworks [11] that have been used to formalize and study several problems in AI (see, e.g., [3, 4]),

including story comprehension [5, 9], and natural language interpretation [6]. Abstract argumentation will need to be suitably relaxed and adapted to reflect the cognitive requirements that we have set for cognitive systems (see, e.g., [39]).

Based on our work on story comprehension [9] and our attempts to develop a cognitive programming language for that task [10], we offer below some pointers on what a cognitively-guided argumentation framework might look like.

Arguments are built via simple association rules, each comprising a small set of concepts as its premise and a single concept as the conclusion that is supported or promoted (but not necessarily logically entailed) when the premise holds. In relation to the example discussed in Section 2.2, a relevant association rule would be "{dinner_at(Person,Place), with_boss(Person)} $\rightsquigarrow$ restaurant(Place)", capturing the argument that having dinner with one's boss normally happens at a restaurant. We view such association rules not as components of scientific theories (e.g., of causality, of norms and obligations, of the mind), relying on elaborative and careful reasoning, but rather as *phenomenological* manifestations of the inferences that would follow from such theories, via a "flat" representation.

Even so, not all association rules can be applied in parallel. Different association rules may promote conflicting conclusions, not all of which can be included in a comprehension model. Resolving conflicts is the essence of the argumentative stance we employ. We adopt the view that association rules are annotated to denote their (possibly relative) level of strength, so that when in conflict, these strengths ensure that the stronger rules will draw inferences, effectively qualifying (by offering a strong counter-argument to) the use of the weaker rules.

With the addition of a *time* dimension, such association rules are sufficiently expressive to represent causality. Thus, if we mark the conclusion of an association rule as holding *temporally after* the premise, the conclusion could correspond to the effect that is brought about when the premise holds. Such causal links are known from Psychology to be important in ascertaining the coherence of a comprehension model. Analogously, if we mark the conclusion of an association rule as holding *temporally before* the premise, the conclusion could correspond to an explanation of why the premise came to be. Drawing such explanatory inferences (when justified to do so) is again critical in the process of comprehension.

Such aspects of causality in world knowledge have featured prominently in the foundations of Artificial Intelligence (cf. the Situation Calculus [30], the Event Calculus [25], and several action languages [14, 19, 29, 46]). The central problems of *frame*, *ramification*, and *qualification* will need to be addressed within the cognitive programming framework, but only in a simplified and qualitative form, as it suffices for our treatment of cognitive programs as phenomenological theories.

3.2 Acquisition of Cognitive Programs

Key in a cognitive system's working is the availability of relevant knowledge, or cognitive programs. Even though the user could contribute to this knowledge by directly instructing the system, we envision that the main mechanism through which cognitive programs would be acquired will be offline or batch learning.

The most promising source of training material for learning commonsense knowledge is currently natural language text, both because of the existence of parsing and processing tools that are more advanced than those that exist for other media (e.g., images), but also because of the high prevalence of textual corpora. The World Wide Web has, typically, played the role of such a textual corpus for machine learning work seeking to extract facts (see, e.g., [41]). When seeking to extract, instead, knowledge appropriate for reasoning, an additional consideration comes into play: knowledge encoded in text from the World Wide Web is biased and incomplete in several ways with respect to our commonsense real-world knowledge, and would be more aptly called *websense* [36]. We posit, however, that certain deficiencies that a cognitive system could have by employing websense would be overcome through the user's feedback and supervision.

Acquisition of knowledge could proceed in several ways. For one, the cognitive system may memorize fragments of text that describe exemplars of certain concepts or scenarios (e.g., a typical restaurant scenario). In a somewhat more structured form, the cognitive system may compute and store statistics about word co-occurrences, e.g., in the form of n-grams, or in the form of frequencies of words appearing in a piece of text conditioned on certain other words also appearing. This last form of statistical information can be interpreted as a weighted association rule, with the weight indicating the "strength" or "probability" of the association holding. In an even more structured form, statistics as above can be stored not on words, but on relations extracted by parsing the text.

Beyond statistical information, one can attempt to learn reasoning rules over words or relations, using typical machine learning techniques. Some such techniques represent learned rules in a form understandable by humans (e.g., DNF formulas). Recent work has shown, in fact, that one can learn not only deductive rules, but also abductive ones, which provide possible explanations given a certain input to be explained [18]. Learning *causal* rules can also proceed naturally by treating consecutive sentences in a textual corpus as the before and after states needed for causal learnability [35]. Treating fragments of texts as partial observations of some underlying, even if unknown, truth or reality can be shown to *guarantee* [34] that rules learned in this manner will draw inferences that are not explicitly stated in, but follow from, a given piece of text. This task, known as *textual entailment* [8], contributes to one of the necessary processes (namely, the drawing of relevant inferences) for constructing a comprehension model.

The amount of knowledge that can be extracted from text is massive, and measures need to be taken to account for this. Section 2.1 has already pointed out that the user guides, explicitly or implicitly, the cognitive system on what concepts the system needs to focus on, and in turn these concepts determine what training material the system will seek for learning knowledge. Even with such guidance, the system may need to refrain from learning knowledge in the most specific form possible, since that would commit the knowledge to a very rigid representation that could not be used later in the context of different queries. Instead, the system should probably choose to retain the learned knowledge in a general representation, some examples of which we have discussed above.

This type of batch and query-independent learning could operate continuously, with the learned knowledge guiding its further development by identifying those concepts for which more training is needed. This process ensures, then, the gradual improvement of a system's cognitive programs, and hence their performance. When a query is posed, the process of knowledge compilation may invoke a further (online) form of learning, treating the offline-learned general knowledge as training data. This query-driven learning is much more focused (and could, in fact, be done implicitly [17]), and should, therefore, be sufficiently efficient to be carried out in real time between the user posing a query and receiving a response. The results of this online learning may be stored, and be reused for future query answering. Supervision by the user may provide additional training material for online learning, which would produce, therefore, user-specific knowledge.

In all cases, learning should proceed in a manner that anticipates reasoning. Valiant's *Probably Approximately Correct* (PAC) semantics for learning and reasoning [47, 48] points to how one could establish formal guarantees on the quality of learned cognitive programs and the comprehension models and inferences they produce. Recent work has proposed PAC semantics for two situations that are of particular interest to cognitive systems: when reasoning involves the chaining of multiple pieces of knowledge [37]; and, when a user's interaction with a cognitive system is personalized by learning to predict the user's intentions [38, 40].

4 Major Challenges for Cognitive Programming

Developing cognitive systems through the cognitive programming paradigm poses major technical challenges. We group and summarize below certain such challenges that would need to be overcome to make progress in this direction.

User-Machine Interaction. Cognitive systems need to interact with human users in a natural way through some fragment of natural language. Hence, the natural language processing capabilities of the supporting modules of cognitive programming are important. In particular, central questions include:

- How do we structure and restrict the complexity of natural language for the user-interface fragment of natural language, without, on the one hand, losing the expressiveness required by the applications, and while keeping, on the other hand, a form of natural communication with human users?
- How can we use existing natural language processing (NLP) systems for the syntactic and grammatical analysis of the user input to ascertain the concepts involved and to extract the narrative information? The use of better NLP tools should help us develop incrementally improved cognitive systems.
- How does the user become aware of the language and knowledge capabilities of the underlying cognitive programming framework? How can we develop useful schemes of dialogues between the user and cognitive systems for user feedback and for natural forms of supervision of the system by the user?

Reasoning with Common Sense. The basic form of argumentative cognitive reasoning and comprehension depends critically on many factors, when this is to be scaled up to be applied in many (if not all the) domains of discourse of common sense. The major questions that need concrete technical answers are:

- Does commonsense knowledge have a generic and task-independent vocabulary and form? What is an appropriate such form and how is this adapted (in real time, through knowledge compilation) into a useful task-specific form? In particular, how do we address the need for *syntactic plasticity* of commonsense knowledge, so that it can be adapted in a manner syntactically compatible with the vocabulary that the current input narrative is using?
- How are relevant parts of commonsense knowledge identified efficiently and reliably given an input narrative? In particular, how do we address the need for *conceptual plasticity* of commonsense knowledge, so that the concepts referred to in the input narrative are matched to concepts in the knowledge base? Is a meta-level form of "context indexing" of the knowledge needed?
- How do we integrate effectively the "pure reasoning" with the process of comprehension, while being guided by the central principle of coherence?

Acquiring Common Sense. Given that we have an appropriate representation for commonsense knowledge, we are then faced with the challenge of how to automatically learn and populate a commonsense library. Questions include:

- Is an offline or batch learning process for commonsense knowledge acquisition the only form of learning required, or do we also need a form of online learning at the time of query processing and knowledge compilation?
- How do we distinguish learned user-specific knowledge from learned generic commonsense knowledge given that the user supervises both processes, and how could learned knowledge be *reused* across users and cognitive systems?
- What are the main technical problems of "mining" commonsense association rules from the World Wide Web? What NLP techniques, search and download tools, storage and indexing schemes would be required? How do we overcome the possibly biased and incomplete nature of learned knowledge?
- How do we learn the annotations and priority tags of commonsense association rules? Can this process be automated, or is it ultimately user-specific?

To address many of these challenges, further *empirical study* with the help of Cognitive Psychology will be needed to help reveal possible answers and guide the development of the computational framework. The availability of a computational framework would then facilitate the experimental examination of the computational viability and effectiveness of various guidelines in improving the cognitive programming framework and the programming experience of the users. In particular, the central and major issues of knowledge plasticity and knowledge compilation are amenable to empirical psychological investigation.

In general, the development of cognitive programming needs to be informed and guided by the psychological understanding at different levels of human cognitive processes. Understanding how the mind operates at some higher conceptual level when dealing with everyday cognitive tasks can help us in developing

possible models of computation in cognitive programming. On the other hand, understanding how humans introspectively perceive or understand the operation of their cognitive processes can help us develop *human-compatible* models of computation: models of computation that humans can naturally relate to.

5 Concluding Remarks

Ideas and proposals related to one form or another of cognitive systems go back to the very beginning of the history of AI, and it would be an interesting topic in itself to explore the development and confluence of these ideas. Among work carried out in more recent years on cognitive computing and systems, Watson is, perhaps, closest to a complete system, and has attracted the most attention from the media. Unlike its emphasis towards "help[ing] human experts make better decisions by penetrating the complexity of Big Data",[6] our proposal focuses on assisting ordinary people by supporting their everyday decision making.

Although both Watson and our envisioned systems seek to solve a cognitive task, the difference in emphasis outlined above suggests that for the latter systems it is crucial that the problem-solving process itself be cognitive, inspired by human heuristics and transparent to the ordinary people's way of thinking. It could be argued that the label "cognitive" should be reserved for such types of systems, and not be conferred to every system that solves a cognitive task.

Adopting this more stringent view of cognitive systems points to a second — in addition to developing intelligent machines — end for building them. Through their operation, cognitive systems could be used to empirically validate or falsify the theoretical models they implement, supporting the scientific process of hypothesizing, predicting, and revising. This iterative process would allow AI to contribute to the refinement of psychological theories of human cognition.

Following a vision where humans and machines share a similar level of common sense, we have proposed cognitive programming as a means to build cognitive systems. Cognitive programming adopts the view of a machine as a personal assistant: a human asks for the completion of a task, perhaps without fully and unambiguously specifying what is needed, but relying on the assistant's experience, and, ultimately, common sense, to perform the task. Cognitive programming aims to bring the flexibility of traditional programming to the masses of existing technology users, enabling them to view their personal devices as novice assistants, amenable to training and personalization through natural interaction.

Our proposal offers a blueprint of what needs to be done and the challenges that one will have to face. We are optimistic that it can be realized to a large extent by building on existing techniques and knowhow from Artificial Intelligence, especially when one takes a pragmatic view by synthesizing the theory and methods of AI with empirical results and ideas from Cognitive Psychology.

Unsurprisingly, the representation and reasoning requirements for cognitive programming are reminiscent of those of production rules as one finds in Computational Cognitive Psychology (see, e.g., [2, 20, 52]). For cognitive programming,

[6] See, for instance, this website: http://www.research.ibm.com/cognitive-computing/

production rules need to include the element of causality in their representation, be enhanced with a declarative form of representing and handling conflicts, and use some notion of (relative) strength of knowledge — or, of the arguments built from the underlying commonsense knowledge — when drawing inferences.

Logic Programming, and recent developments from this [24], have moved in this direction of production or reactive systems with such enhancements, but remain largely bound to the strict formal logical semantics. Similarly, frameworks for autonomous agents, such as BDI agents [42] and robotic agent programming [7], which aim amongst other things to give cognitive abilities to agents, also rely on strict logical or operational semantics. These approaches serve, therefore, a different class of problems from those aimed to by cognitive systems based on commonsense knowledge, and for which the role of comprehension is important.

One may argue that progress on natural language understanding would suffice to realize our vision of cognitive programming. Despite the important role of such progress, a fully automated natural language system would seem to require a machine architecture similar to that of the human brain. Given the gap between the formal logic-driven machine architectures of today (with long, rigid, and error-intolerant chains of computation — a limitation already identified by von Neumann [51]), and the cognitive capabilities and constraints of the human mind, our proposal of cognitive programming hopes to provide the middle-ware needed today to move closer to the ideal of an automated natural language system.

Acknowledgements

We are grateful to our colleague Irene-Anna Diakidoy (Department of Psychology, University of Cyprus) for her collaboration in our joint work on story comprehension, and the invaluable help that she has given us in our effort to understand the computational implications of relevant work in Cognitive Psychology. We also wish to thank Hugo Mercier and Francesca Toni for many valuable discussions on the psychology of reasoning and the nature of cognitive systems.

References

1. J. E. Albrecht and E. J. O'Brien. Updating a Mental Model: Maintaining Both Local and Global Coherence. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 19(5):1061–1070, 1993.
2. J. R. Anderson, B. E. John, M. A. Just, P. A. Carpenter, D. E. Kieras, and D. E. Meyer. Production System Models of Complex Cognition. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, pages 9–12, 1995.
3. P. Baroni, M. Caminada, and M. Giacomin. An Introduction to Argumentation Semantics. *Knowledge Engineering Review*, 26(4):365–410, 2011.
4. T. J. M. Bench-Capon and P. E. Dunne. Argumentation in Artificial Intelligence. *Artificial Intelligence*, 171(10–15):619–641, 2007.
5. F. J. Bex and B. Verheij. Story Schemes for Argumentation about the Facts of a Crime. In *Proceedings of the 2nd Workshop on Computational Models of Narrative*, 2010.

6. E. Cabrio and S. Villata. Natural Language Arguments: A Combined Approach. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pages 205–210, 2012.
7. K. L. Clark and P. J. Robinson. Robotic Agent Programming in TeleoR. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5040–5047, 2015.
8. I. Dagan, D. Roth, M. Sammons, and F. M. Zanzotto. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2013.
9. I.-A. Diakidoy, A. Kakas, L. Michael, and R. Miller. Story Comprehension through Argumentation. In *Proceedings of the 5th International Conference on Computational Models of Argument*, volume 266 of *Frontiers in Artificial Intelligence and Applications*, pages 31–42, 2014.
10. I.-A. Diakidoy, A. Kakas, L. Michael, and R. Miller. STAR: A System of Argumentation for Story Comprehension and Beyond. In *Proceedings of the 12th International Symposium on Logical Formalizations of Commonsense Reasoning*, pages 64–70, 2015.
11. P. M. Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artificial Intelligence*, 77(2):321–358, 1995.
12. J. S. Evans. Logic and Human Reasoning: An Assessment of the Deduction Paradigm. *Psychological Bulletin*, 128(6):978–996, 2002.
13. S. L. Frank, M. Koppen, L. G. M. Noordman, and W. Vonk. Computational Models of Discourse Comprehension. *Discourse Processes*, 45(6):429–463, 2008.
14. M. Gelfond and V. Lifschitz. Representing Action and Change by Logic Programs. *Journal of Logic Programming*, 17(2/3–4):301–321, 1993.
15. R. J. Gerrig. The Scope of Memory-Based Processing. *Discourse Processes*, 39(2–3):225–242, 2005.
16. A. C. Graesser, K. K. Millis, and R. A. Zwaan. Discourse Comprehension. *Annual Review of Psychology*, 48:163–189, 1997.
17. B. Juba. Implicit Learning of Common Sense for Reasoning. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 939–946, 2013.
18. B. Juba. Learning Abductive Reasoning Using Random Examples. In *Proceedings of the 1st Workshop on Cognitive Knowledge Acquisition and Applications*, 2015.
19. A. C. Kakas, L. Michael, and R. Miller. Modular-$\mathcal{E}$ and the Role of Elaboration Tolerance in Solving the Qualification Problem. *Artificial Intelligence*, 175(1):49–78, 2011.
20. D. E. Kieras and D. E. Meyer. An Overview of the EPIC Architecture for Cognition and Performance with Application to Human-Computer Interaction. *Human-Computer Interaction*, 12(4):391–438, 1997.
21. W. Kintsch. The Role of Knowledge in Discourse Comprehension: A Construction-Integration Model. *Psychological Review*, 95:163–182, 1988.
22. W. Kintsch. *Comprehension: A Paradigm of Cognition*. Cambridge University Press, 1998.
23. W. Kintsch and P. Mangalath. The Construction of Meaning. *Topics in Cognitive Science*, 3:346–370, 2011.
24. R. Kowalski and F. Sadri. Programming with Logic without Logic Programming. *manuscript*, 2015.
25. R. Kowalski and M. Sergot. A Logic Based Calculus of Events. *New Generation Computing*, 4(1):67–95, 1986.

26. P. Langley. The Cognitive Systems Paradigm. In *Proceedings of the 1st Annual Conference on Advances in Cognitive Systems*, pages 3–13, 2012.
27. P. Langley, J. E. Laird, and S. Rogers. Cognitive Architectures: Research Issues and Challenges. *Cognitive Systems Research*, 10(2):141–160, 2009.
28. A. Lieto, A. Minieri, A. Piana, and D. P. Radicioni. A Knowledge-Based System for Prototypical Reasoning. *Connection Science*, 27(2):137–152, 2015.
29. N. McCain and H. Turner. Causal Theories of Action and Change. In *Proceedings of the 14th AAAI Conference on Artificial Intelligence*, pages 460–465, 1997.
30. J. McCarthy and P. J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. *Machine Intelligence*, 4:463–502, 1969.
31. D. McDermott. A Critique of Pure Reason. *Computational Intelligence*, 3:151–160, 1987.
32. D. S. McNamara and J. Magliano. Toward a Comprehensive Model of Comprehension. *The Psychology of Learning and Motivation*, 51:297–384, 2009.
33. H. Mercier and D. Sperber. Why Do Humans Reason? Arguments for an Argumentative Theory. *Behavioral and Brain Sciences*, 34(2):57–74, 2011.
34. L. Michael. Reading Between the Lines. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 1525–1530, 2009.
35. L. Michael. Causal Learnability. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1014–1020, 2011.
36. L. Michael. Machines with WebSense. In *Proceedings of the 11th International Symposium on Logical Formalizations of Commonsense Reasoning*, 2013.
37. L. Michael. Simultaneous Learning and Prediction. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning*, pages 348–357, 2014.
38. L. Michael. Introspective Forecasting. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 3714–3720, 2015.
39. L. Michael. Jumping to Conclusions. In *Proceedings of the 2nd International Workshop on Defeasible and Ampliative Reasoning*, 2015.
40. L. Michael. The Disembodied Predictor Stance. *Pattern Recognition Letters*, 64:21–29, 2015. Special Issue on Philosophical Aspects of Pattern Recognition.
41. T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. Never-Ending Learning. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2302–2310, 2015.
42. A. S. Rao and M. P. Georgeff. BDI Agents: From Theory to Practice. In *Proceedings of the 1st International Conference on Multi-Agent Systems*, pages 312–319, 1995.
43. D. N. Rapp and P. Van den Broek. Dynamic Text Comprehension: An Integrative View of Reading. *Current Directions in Psychological Science*, 14:297–384, 2005.
44. A. J. Sanford and S. C. Garrod. The Role of Scenario Mapping in Text Comprehension. *Discourse Processes*, 26(2–3):159–190, 1998.
45. P. Thagard. *Coherence in Thought and Action*. MIT Press, 2002.
46. M. Thielscher. From Situation Calculus to Fluent Calculus: State Update Axioms as a Solution to the Inferential Frame Problem. *Artificial Intelligence*, 111:277–299, 1999.
47. L. G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
48. L. G. Valiant. Robust Logics. *Artificial Intelligence*, 117(2):231–253, 2000.

49. P. Van den Broek. Comprehension and Memory of Narrative Texts: Inferences and Coherence. In M. A. Gernsbacher, editor, *Handbook of Psycholinguistics*, pages 539–588. Academic Press, 1994.
50. D. Vernon, G. Metta, and G. Sandini. A Survey of Artificial Cognitive Systems: Implications for the Autonomous Development of Mental Capabilities in Computational Agents. *Transactions on Evolutionary Computation*, 11(2):151–180, 2007.
51. J. von Neumann. The General and Logical Theory of Automata. In A. H. Taub, editor, *John von Neumann: Collected Works. Volume V: Design of Computers, Theory of Automata and Numerical Analysis*, chapter 9, pages 288–328. Pergamon Press, 1961. Delivered at: Hixon Symposium, September 1948.
52. R. M. Young. Production Systems in Cognitive Psychology. In N. J. Smelser and P. B. Baltes, editors, *International Encyclopedia of the Social & Behavioral Sciences*. Elsevier, 2001.
53. R. A. Zwaan and G. A. Radvansky. Situation Models in Language Comprehension and Memory. *Psychological Bulletin*, 123:162–185, 1998.