A PERSPECTIVE ON SYMBOLIC MATHEMATICAL COMPUTING AND ARTIFICIAL INTELLIGENCE

J. Calmet[1] and J.A. Campbell[2]

[1] Institut für Algorithmen und kognitive Systeme, Universität Karlsruhe, Am Fasanengarten 5, 76131 Karlsruhe, Germany, calmet@ira.uka.de
[2] Department of Computer Science, University College London, Gower Street, London WC1E 6BT, England, jac@cs.ucl.ac.uk

Abstract. The nature and history of the research area common to artificial intelligence and symbolic mathematical computation are examined, with particular reference to the topics having the greatest current amount of activity or potential for further development: mathematical knowledge-based computing environments, autonomous agents and multi-agent systems, transformation of problem descriptions in logics into algebraic forms, exploitation of machine learning, qualitative reasoning, and constraint-based programming. Knowledge representation, for mathematical knowledge, is identified as a central focus for much of this work. Several promising topics for further research are stated.

As an introduction to the proceedings of the first international conference that was devoted specifically to symbolic mathematical computing (SMC) and artificial intelligence, we wrote a combination of a short survey and a summary of our predictions and suggestions for the future development of the territory common to those two subjects[1]. The present paper revisits the same areas, as well as remarking on results that have been reported during some of the time since the first conference in 1992.

In the earlier paper, we reviewed the history of SMC with reference to its associations with - and, for some years, detachment from - AI. Following that review, we indicated several topics where SMC had contributed to modern AI or was in a position to do so, and then repeated the exercise for the actual and likely influences of AI on SMC. In addition, we singled out knowledge representation as the area in which the links between the two subjects were strongest and also appeared to have the greatest potential for future exchanges of mutual benefit. The present paper reflects this kind of organisation: progress since the first conference has followed the lines that we predicted in [1]. A selection of papers on significant work that had started before the conference, and that was reported there, is published in the current issue of this journal.

Historical Perspective

In the earliest period of AI research, the nearest thing to "applied AI" was SMC. That is to say, papers that were clearly about SMC made their appearance

commonly in media where AI was most often represented, and - unlike other AI papers at the time - concerned software or computed results that had been sought by, and were of immediate value to, people outside both AI and SMC. Celestial mechanics and high-energy particle physics were the first two serious areas of application of SMC, as papers like [2] and [3] and earlier references cited there indicate. Even though it occurred before the Dartmouth College summer workshop that is often regarded as marking the transition from prehistory to history in AI [4], the writing of two programs for symbolic differentiation has been quoted subsequently as an example of early AI. More visibly than any of these events, AI used SMC because of the choice of integration in finite terms as a problem that could stimulate advances in its techniques. In terms of software, the two highlights of this line of activity were Slagle's SAINT [7] and the programs of J. Moses that were later absorbed into the MACSYMA project on SMC [8].

Soon after the period covered above, which lasted until about 1970, SMC material became harder to find in AI conferences and publications. The transition to near-invisibility was very rapid. One reason for this was the increased production of theoretical or conceptual papers in AI for its own sake. But the major reason, as we can interpret it after the event, is also something that has happened to other parts of AI since then: the transition of the information used for representation and reasoning from heuristics to algorithms. This also lies behind the popular saying among AI practitioners that "AI exports its successes". Once a topic where the knowledge was originally expressed heuristically arrives at a state where algorithms express its most efficient methods of computation, it tends to disappear from papers at AI conferences. Historically, this happened first for SMC, through the transition from the basically heuristic methods used by Slagle and Moses to the implementation of what is now known as the Risch-Norman algorithm [9] for integration in finite terms. This evolution has been repeated for other topics, such as 3-dimensional interpretation of edges in 2-dimensional views (with respect to the Waltz filtering procedure [10]) and the Hough transform [11] in feature detection in images.

For nearly 20 years, the resulting separation between AI and SMC continued. But we are now seeing an increasing amount of convergence between the two subjects. The overlap has become steadily more evident with each successive biennial conference (of which there have now been three) on AI and SMC. The explanation for the convergence is that each subject, in its present state, stands to gain some advantage from the other.

The actual range of capabilities of SMC systems has not changed much for some years. Their character has changed, but primarily because of improved algorithms in their existing applications, or purpose-built modules to draw together what were previously users' programs for specific variants of these applications. In many instances, probably a large majority, users are at home with this situation. In other instances, one sees users carrying out pen-and-paper manipulations on problems in order to state them in forms that are acceptable inputs for current SMC. According to at least one of the earliest implementers of SMC software, the work was partly to remove the need for pen-and-paper manipulations among

people who wished to carry their formulations of problems to a stage that could serve as input to Fortran programs. If a good SMC capability exists, it should ideally eliminate any "off-line" manipulations that users would prefer to pass to a computer. Answering this challenge in general calls for mathematical knowledge to be embedded in SMC environments: hence the relevance of AI, which deals centrally with the representation of knowledge and its use in reasoning. AI has a particular track record in dealing with multiple heterogeneous representations of knowledge, which is the situation one faces necessarily in using mathematical knowledge (e.g. expressions in logics, algorithms, hierarchical collections of information, heuristics, examples of problems and their solutions) as mathematicians use it.

Just as many desirable advances in SMC will have to rely on methods of AI, AI in some of its most active areas can benefit from access to SMC. Two areas in particular fit this statement: qualitative reasoning, and multi-agent systems. In the former, the mathematical expressions used resemble those of differential calculus but require also the flexibility of defining simplifications and other manipulations that typical SMC systems do not offer. In the latter, some sub-problems have a strong element of mathematical modelling, and demand the kinds of knowledge that a good human modeller deploys. At present, the mathematical parts of such computations are usually reduced to pen-and-paper exercises, or processed by programs inside the larger AI-based software that merely reproduce the simple SMC capabilities of 30 or more years ago. Some environments for qualitative reasoning and for multi-agent computations can do better than that, but the integration of AI and SMC in these areas is still quite a new issue, and much useful work remains to be done.

There is one further reason for the convergence of AI and SMC. The description that we gave in [1] is still valid: "....specialists in knowledge representation are becoming interested in capturing and using the considerable amount of heuristic knowledge that mathematicians possess about suitable symbolic structures. Apart from the technical interest of doing this, there is the attraction that success will make it possible to build practical systems that combine operational mathematical knowledge with the calculational capabilities of present systems like MACSYMA and REDUCE. Any substantial progress in this area will give such systems their first significant boost in functionality since the early 1970s".

In the past history of SMC, capturing heuristic knowledge about some topic T has been the first step towards finding algorithms for T that are at least equally effective in actual use. Where there has been a final step for a T where AI and SMC have both been involved, as we have said, the final step has often been that T has disappeared into the most technical depths of SMC publications and has become invisible to AI. In the future history of AI and SMC, this is rather less likely. Both heuristics and algorithms for any interesting T have a place, e.g. in different parts of a mathematical knowledge base, and can illuminate each other. For example, features of one can be used to support proofs or explanations of the other, or treated as cues for retrieval of the other if problem-solving computations include any case-based or analogical reasoning. The underlying point is that

the design and construction of mathematical knowledge bases is desirable (and practicable) for progress in the theoretical and applied sides of both AI and SMC.

A Central Issue: Knowledge Representation

In typical SMC systems the representation(s) that the builders have selected are for their own convenience or for computational efficiency. Sometimes these two are the same thing. Most users have paid no special attention to this, as it has coincided with their own interests. The data structures are therefore fixed, as are the "knowledge structures", which are primarily programs expressing the most efficient algorithms in the most efficient ways. Inspectability of this latter material is not in question. On occasions, multiple ways of structuring the data and showing this information to the user are possible: as one-dimensional (Fortran-like) or two-dimensional (imitating typeset mathematical expressions) transcriptions, or as graphs after numerical substitutions for symbolic variables, for example.

To apply methods of AI to their fullest extent, this approach to representation must be replaced by schemes in which the mathematical knowledge is available explicitly and in which arbitrary computations on that knowledge can be carried out without any redesign or rebuilding of the framework in which it is held. Here, SMC can borrow directly from the collected experience of AI research (e.g. [12] [13]) on knowledge representation, which has led to suggestions of between 10 and 20 different representations for knowledge. Initially, at least, SMC can exploit the most standard representations that are quoted in general AI textbooks: rules, inheritance networks, and frames. Many examples of knowledge of mathematical properties fit each of them; to do full justice to mathematical reasoning, some of the more exotic representations (in addition to "logic", which is both a representation and a reasoning device) may recommend themselves. We return to that point later. It may even happen that future research on the nature of mathematical reasoning will lead to the development of representations that will be new in the AI literature.

The simplest (and historically the earliest) way of combining SMC facilities with knowledge (about how and when to use them) has been to write the knowledge into sets of rules, and to make these sets control the use of parts of the SMC software. In effect, the control mechanism is a small expert system, although in typical implementations the SMC environment retains the basic control and consults the rules for indications of when its usual behaviour should be modified. Vivet [14] gives an early example. Later examples (e.g. [15]) are not uncommon, and at least one general SMC environment (Mathematica) allows its users to state rule-like information about how a computation should run in some circumstances.

While these examples are literally mathematical knowledge-based systems, their use is limited quite tightly to the problems for which they have been defined. To be more generally valuable, systems should permit ready use of their

mathematical knowledge, when it is relevant, for computations other than the ones for which they were built. In other words, expanding the range of the mathematical knowledge and reasoning capabilities should be a cumulative process without constant rewriting and redesigning. An example of an approach driven by these considerations (though with much larger and more general ambitions about knowledge), and containing useful ideas about points and methods to consider when one is aiming to specify a framework for common use of knowledge on different topics, is the CYC project [16].

If we grant that mathematical knowledge in a general-purpose system will be heterogeneous, then an early practical requirement that we face is to ensure that navigation inside the knowledge base goes as quickly as possible to items that are relevant for a given computation, and ignores material or possible steps of reasoning that are not appropriate. Here, computer science offers a strong hint: if one has such a requirement and the information that one is processing contains types, demand that the type be given along with each piece of information, and use the types in the computations that follow. The kind of declaration and exploitation of types that one sees in (for example) debuggers and compilers for typed languages is one model of design that can be applied to mathematical knowledge-based systems, especially where domain objects (e.g. polynomial rings, finite fields) are naturally modular. These objects can then be arranged in hierarchies, and each one can carry a type schema with names, names of objects from which properties are inherited, associated operators, axioms governing operators and elements, etc. This approach appeared first in SMC in the series of systems that have led to AXIOM [17]; it has also been used successfully in environments such as MANTRA (Modular Assertional, Semantic Network and Terminological Representation Approach) where both AI and SMC considerations have applied [18] [19]. We mention this as one example of how to respond to the considerations that we have given above.

The goal for MANTRA was to introduce algebraic algorithms, and indeed computer algebra systems, as a novel knowledge representation paradigm. At the beginning of that work, no existing system fitted that description. Implementation was guided by the following opinions about design:

1. Several cooperating formalisms are better than a unique representation formalism;
2. A clear semantics explaining the meaning of the knowledge representation language is fundamental;
3. All algorithms involved must be decidable, and reasonably fast.

In a knowledge engineering perspective, MANTRA may also be regarded as a general-purpose shell for building large mathematical knowledge-based systems.

The system provides four different representation formalisms that can interact through hybrid inference algorithms. The motivation is that several cooperating formalisms ought to enhance the expressiveness and inferential power of a system. The knowledge-representation approach consists of a representational theory, explaining which knowledge is to be represented by which formalisms,

and a common semantics to define the relationship between expressions of different formalisms in a semantically sound manner. The decidability of all algorithms involved is ensured by adopting a four-valued semantics based on works of Belknap, P.F. Patel-Schneider, A.M. Frisch, R.H. Thomason and others. The four knowledge representation formalisms mentioned above are all supported, along with the usual AI inference mechanisms for each.

It is convenient to regard the MANTRA architecture in terms of levels. The lowest consists of components involving logic, frames and inheritance networks (semantic nets), and is an epistemological level. The second level, the logical level, consists of the inference mechanisms, including those that permit hybrid inferences among the different lowest-level representations, and means for management of the knowledge bases. The highest level is a heuristic level, with representation of procedural knowledge of domains, and rules. The heuristic-level knowledge is expressed in production rules that are processed by an OPS5-like rule interpreter. The overall system was written in Common Lisp using the object-oriented extension CLOS, and is supplemented by a graphical user interface produced in C with XToolkit.

As a test of the effectiveness of the architecture, the computer algebra system REDUCE was added as a component at the epistemological level, and was found to communicate in the expected way with the other knowledge-representation formalisms. One aim of that exercise was to show that the extended system could work as a tutoring system for REDUCE users.

The MANTRA framework above does not take account of types. More recent work in the same project has included the design of a specification language, FORMAL [17], which allows enough richness of type structure to accommodate mathematical knowledge (for which specification languages in more traditional areas of computer science are not yet adequate). The ultimate goal is "to execute" specifications, e.g. to query properties of the specified structures, by transforming the specifications into another language - MANTRA - for which inference mechanisms exist.

Two branches of theoretical computer science play the main roles here: algebraic specification and type theory. The former studies ADTs or abstract data types (signatures, i.e. sets of type- and function-names, and sets of axioms). One then reasons about models of the ADTs, which can be regarded as "implementations" of the types and functions of the signatures that fulfil the requirements (axioms) of the specification. Thus, reasoning about ADTs and reasoning about "implementations" should ultimately be the same thing, and the theory for an ADT should be the set of theorems that are valid for it.

Existing specification languages in computer science are often based on universal algebras and category theory - as is FORMAL - but their many-sorted and order-sorted algebras are not rich enough to describe all the useful properties of material that is likely to occur in a good mathematical knowledge-based system. Further research is needed on types, e.g. type hierarchies, parametrisation of types (both of which can be treated in FORMAL), types of higher-order functions, and higher-order types in general. For example, one tries to find models

for such type systems and to classify them according to considerations like

- Does each possible expression have a unique type?
- Is the type of an expression computable?
- Is the correctness of the type of an expression decidable?
- What dependencies exist? (i.e. What can be parametrised by what?)

A research agenda in this direction, to develop appropriate knowledge-representation schemes for SMC, should have other components also. One of those components should involve finding (semi)automatic means to enable simultaneous typing and specification. This capability is not needed if, say, one can write specifications along with the algorithms to which they refer. However, this is not sufficient if one wants to couple computing (use of the algorithms) with deduction or other reasoning about what the algorithms are doing to the current data. Another component that deserves further research is the checking of completeness of specifications (e.g. of the list of properties for a given operator). Present completion algorithms, where they exist, are generally very inefficient. Therefore it may pay to investigate whether methods of machine learning can generate heuristics to achieve completion. Usually, when a property (equation) is encountered, one checks immediately whether it is a new property - but it is possible to delay such a check and to consider the new equation as a training instance that will be processed later during a learning phase. It is possible to prove that algorithms of this kind exist, but there is scope for much additional work.

The discussion above is at the formal end of the implications of knowledge representation. There is also an informal end, where the knowledge and the reasoning have not drawn significant support from logics and from any intrinsic mathematical properties. This involves the use of patterns, examples, narrations etc. of successful instances (and sometimes unsuccessful instances, as guides to what to avoid in tackling similar exercises in the future) of solution of a problem. It is not hard to find literature on mathematical problem-solving behaviour of this kind; some (e.g. [20] [21]) has classic status. Even when the sources for the behaviour are learners and not professional mathematicians, it is fair to describe the actions and the results in a "case". A case is a recognised knowledge representation in AI, though not primitive: it can be written in a variety of ways in terms of simpler representations. Case-based reasoning [22] is a well-known and expanding area of AI, but apparently not reported in papers on AI and mathematics so far. The existence of this gap in coverage is a challenge for AI and SMC: how far can one use case-based methods from elsewhere in AI to represent and exploit mathematical knowledge, whether for solution of significant problems or for "debugging" misconceived behaviour of students or even inexperienced users of computer algebra systems?

Areas of Interaction between AI and Symbolic Mathematical Computing

We organise this section of the paper according to the areas in which the most significant activity has been taking place recently.

Autonomous Agents and Multi-Agent Systems

The traditional AI approach to multi-agent systems (MAS) has been in terms of logic, or at least the notation of logic (e.g. in the design of primitives and schemes for inter-agent communication, and formation of individual and joint plans). But it is increasingly evident that the character of MAS requires alternative or extra forms of description that are not available via this traditional path. It is sometimes advantageous to view the systems in a social or socio-economic perspective, sometimes as competitors for limited resources, and sometimes as dynamical systems in the sense of classical or statistical mechanics. In all of these perspectives, the common tool for a better understanding of the systems' properties and behaviour is mathematical modelling. Moreover, a variety of mathematical techniques and knowledge occurs in such modelling in the subjects that have exploited those perspectives in the past. Some of the recent history of MAS research has been the identification of correspondences between aspects of MAS and aspects of problems in other fields where modelling has been effective, followed by application of the same kinds of modelling to clarify properties of agents and their interactions. Use of SMC software and representations of mathematical knowledge has the potential to enhance such activities in the future, in two ways. First, SMC can increase the speed and reliability of modellers' reasoning about MAS and derivation of theorems and other results. A good example is a study of the possible behaviours of agents in changeable situations where the breaking of contracts, at some cost, may be advisable [23]. The mathematical manipulations involve a mixture of integral expressions, examples, inequalities and the drawing of graphs by way of clarification. Second, individual agents reason about other agents, themselves, and their environment. Typically, this reasoning is expressed in the logic-based form that we have mentioned at the beginning of this subsection. But there are aspects of what can be reasoned about (e.g. behaviours that can be reduced to observation, generation or prediction of trajectories in coordinate or other spaces) that do not fit comfortably into logical notation. An agent dealing with such phenomena can benefit from viewing reasoning as a mathematical modelling process, and having access to SMC to support it.

Modelling not only involves re-using existing terms and methods after analogies are made between MAS and the areas where they have been used in the past. If we are trying to understand the nature of MAS and their behaviour, it may sometimes be necessary to call upon concepts that are new in modelling. For example, Pfalzgraf, Sigmund and Stokkermans [24] have considered deductive planning activities, e.g. path planning, in a simple robotic MAS, and have shown how to use logical fiberings (on agents' state spaces) to add semantics to the planning process. This approach opens further questions for research, e.g. on how to handle expressions dependent on space and time coordinates, and on hierarchical planning.

An alternative to path planning with logic is the use of classical mechanics. Here, locations that should lie on a path are given attractive potentials, and locations such as obstacles, which should be avoided, are associated with repulsive potentials. The path is then determined as the solution to a variational problem.

The same general method can deal with highly complicated practical instances, e.g. collision-free motion of articulated robots for welding in a shipyard. Overgaard, Petersen and Perram [25] use this underlying discipline in a scheme where each link of each jointed robot is treated as a single agent, and where equations of motion with kinematic constraints govern the required motion and specify that the robots do not collide with other objects or with themselves. Although sets of equations for real robots will usually be solved numerically, SMC allows development of such equations and explorations of their gross properties. We quote this paper as just one example of quite a large literature on constrained dynamics for robots and robotic agents.

An example of the social or socio-economic approach to MAS is a paper on dilemmas by Glance and Hogg [26]. Here, in particular, the paradox that adding resources to what is available to a group of agents can degrade the performance of the group is examined. Another such example is a study of the dynamics of computational ecosystems [27]. What is common to such studies is the manipulation of sets of equations and inequalities, where SMC has the same relevance for its ability to support the development and analysis of the sets.

Other topics in MAS where SMC can act as a support tool for researchers are the design of algorithms for basic activities such as communication (e.g. by parametrising families of algorithms and examining the effects of varying the parameters) and in equational deduction. Papers that describe the topics themselves in suggestive ways are by Decker and Lesser [28] and Denzinger [29] respectively.

As agents become more complex, it is likely that they will need their own access to mathematical modelling facilities in order to interpret their surroundings better. In MAS research it is already recognised that agents can and should maintain models of (beliefs about) other agents, for example, These models are expressed in rather simple logical notation, but do not capture in transparent or efficient ways the kinds of information about their environment (e.g. trajectories) that are easy to handle in SMC systems. It is an interesting research question, not yet tackled, to consider how to package or give access to the rather large range of capabilities of SMC systems for the relatively small blocks of software that constitute agents. Existing work on the kinds of modelling that agents will need to carry out suggests, further, that some mathematical structures and knowledge that are not normally present in SMC systems are required. In particular, there are approaches that rely on graphs [30] and graph theory. Further, models about agent behaviour, which could usefully be made available to the agents themselves, often start from ideas involving queues [31], loads on resources, and queueing theory.

Transforming Logic into Algebra

In any substantial system that holds and uses mathematical knowledge, both "computer algebra" (the area of the most traditional SMC systems) and logic have a place. There are topics, e.g. making and annotating proofs of theorems, where a logic component is irreplaceable. However, computations based on logics

are usually quite slow, and expensive in storage, at least by comparison with the performance of computer algebra systems. More than once, we have heard comments of the form "If only my program for deduction/induction/abduction/etc. could run as efficiently as REDUCE/Mathematica/Maple/etc.". Therefore, if a problem initially posed in a logical formalism could be transformed into a problem suitable for such a system, it would bring computational benefits, as well as extending our stock of mathematical knowledge. We have made that point already in [1].

In that paper, we referred to examples of problems transformed in this way, for theorem-proving in geometry [32], reasoning about the solution of certain transcendental equations [33], and cylindrical algebraic decomposition [34] as a means of either exploring the satisfiability of systems of equations, inequalities and inequations [35] or (in its original guise) solving the quantifier-elimination problem of Tarski. The use of integer programming to solve certain transformed problems of deduction [36] is another example. It is still desirable to search for new topics where a transformation from logic to some mathematical form that traditional SMC systems can process efficiently, and to expand existing results. The area where the greatest amount of activity and progress has been occurring is geometrical theorem-proving, as evidenced by papers by Rege and Canny [37] and Wang [38].

Machine Learning

Machine learning is a wide field whose techniques are accessible in many places (e.g. [39]). Here we concentrate on a rather simple part of the subject, which has been known for some time but which has only attracted occasional SMC research-level attention. It deserves much more.

Typically, the generalisations that are the outputs of machine-learning methods are generated and expressed in a logic-based notation. An early example of such activity in a field highly relevant to SMC, the AM project [40], has been famous in its time, but also controversial [41]. In general, AM-style automated discovery of interesting terms, conjectures etc. in mathematics is a difficult job. As we have said above, certain difficulties (inefficiencies) in logic-based computations can be reduced or removed if one manages to transform them into something - informally, "algebraic" - that fits better what traditional SMC systems can handle.

The particular aspect of learning that we emphasise starts from algebraic information; no logical expressions or related theorem-proving steps are needed. The simplest example that we have met involves counting the number of terms in each of the first several members of a one-parameter family of functions (e.g. functions occurring in exact or approximate solutions of equations in mathematical physics), after generating these members by SMC, and then searching for symbolic expressions that reproduce the numbers of terms when values for the parameter are substituted into them. If there are regularities, it is possible that these follow from some properties of the functions that can be exploited in design of specialised compact data structures and methods for computing higher

members of the family of functions. There have been situations where users have asked for those members to be computed but where the more general and therefore loose data structures maintained by SMC systems have made it impossible for the systems to finish their computations before running out of space: hence the need to find special ways of reducing the demand on memory.

In the same general area of inferring symbolic mathematical regularities, there have been several programs such as BACON [42] for (re)discovery of equations of physics, e.g. Kepler's Law, from sets of data, and an example [43] of finding candidate solutions of some integral equations in mathematical physics by a similar process plus the use of dimensional analysis to control the search for candidates. Correct solutions were then identified by substitution into the equations, and simplification. More recent work following from the former reference has now led to additional results, and a more general outlook, on "scientific discovery" [44]. Particular instances have involved searches for new invariants: quantities that remain the same before and after reactions that are observed to occur, but that would be different between the "before" and the "after" of reactions that are not observed in experimental data. In high-energy particle physics, these quantities are scalars, and the two forms of data are expressed by linear equations and inequations respectively. Both Kocabas [45] and Valdes-Perez [46] have written programs to suggest conserved quantities in this way. Their results are not always the same: the program PAULI [46] searches for results that involve the smallest possible number of new invariants, while BR-3 [45] suggests invariants that are more often rediscovered copies of quantities (e.g. baryon number, lepton number) that physicists have proposed and used already. Valdes-Perez makes several suggestions, related to the nature of investigation and discovery in science, about reconciliation of the different results. None is conclusive, and some give openings for further research on methods and heuristics (e.g. for partitioning the sets of data before the main part of the search for invariants occurs) that can be added to the knowledge base for any SMC system that may be used for similar explorations of machine learning in the future.

Because the precise focus of this kind of computation is better adapted to computer algebra systems than many logic-based learning computations, it is surprising that there has been almost no apparent contact between its specialists and SMC. This is clearly an area that deserves more attention.

Qualitative Reasoning

When this subject emerged in AI, it had the name "naive physics" [47]. The original goal was to identify the terms and reasoning methods that people use to make their own practical models of physical phenomena. It was expected that these models would not resemble the logical formulations made by AI specialists for "planning" programs or the equations that are taught to students of applied mathematics and physics. Nevertheless, the practical models must have some form of consistency (perhaps weak) and relevance, because the people who use them to cope with the real world seem to survive and avoid injury quite well.

After a short time, the emphasis in AI shifted towards something non-naive but more relevant to SMC: the use of physicists' and mathematicians' equations, where the variables were "qualitative" (e.g. drawn from the set +,0,-) rather than of integer or real type. Quite substantial schemes of interpretation and of qualitative calculi (e.g. [48]) were developed on this foundation. The manipulations involved were well suited to SMC, although researchers in qualitative reasoning tended to write their own modules for symbolic manipulation rather than calling on SMC systems to make use of the calculi.

The "pure" version of this trend has now moved into the background, mainly because the qualitative computations were more expensive than working with the original equations in terms of integer and real variables. The foreground of qualitative reasoning is occupied by the study of choices and of the activity of modelling in design, particularly engineering design. Thus, for example, diagnosis of faulty operation does not wait for a device to be built and to become defective, but a design is inspected diagnostically to identify possible faults in advance of construction [49]. Struss [50] gives further details of the present state of qualitative reasoning.

Given the change of emphasis in the subject, as we observed in [1], an automated qualitative reasoner should be able to handle not only traditional computer algebra but also knowledge about design, in any of the standard knowledge representations of AI. Knowledge that a designer normally draws in diagrams or other forms that express kinematic and spatio-temporal information should be permitted to fit easily into the resulting SMC systems, e.g. through access to Prolog-like modules. We stated in [1] that a challenge for SMC research is to integrate modules of this kind with modules for traditional computer algebra. That challenge remains at present.

The topic of spatio-temporal reasoning was discussed in [1] under a separate heading, with the suggestion that it was potentially fruitful for research. A more recent development in qualitative reasoning has been the active use of its methods and concepts to treat just that topic. A recent survey by Cohn [51] gives information about achievements and about research questions that are still open.

Constraint-Based Programming

Two types of constraints are common in mathematical knowledge-based computations and SMC. In SMC they are written as inequalities, as in [23]. Whenever mathematical knowledge is put into in a representation such as Horn-clause logic and can then be treated as a program in a suitable language, e.g. Prolog, it is usually reasonable to describe the operation of the program as constraint-based computing. Instantiations for uninstantiated variables are tried until one or more make all the logical assertions of the relevant part of program true. The constraint is that an instantiation must make them true in order to be acceptable.

Although these two kinds of constraint go naturally together, e.g. in allowing automated reasoning about exercises in linear programming and optimisation,

they have not yet led to many SMC environments in which the two are integrated. The best-known example is the programming language Prolog III [52]. Rueher [53] gives examples of its use. Bouhineau [54] has recently discussed further examples that are more directly concerned with SMC: geometric constraint problems, in terms of the algebraic forms described by Chou [32], and "constructible numbers".

A system that accepts both kinds of constraint, and which can also deal with the consequences of overdetermined collections of equations and inequalities, is UniCalc [55]. This system apparently does not have an explicit Prolog-like component, but can use some kinds of information that would be appropriate for such a component, if the user presents them in a way that fits UniCalc's internal notation.

Apart from the approach represented by Prolog III, CLP(R) [56] and UniCalc, there are explicitly "constraint-based" programming languages [57] [58]. We have not seen any use of those languages specifically for applications of SMC interest, but it would be of some interest to investigate their capabilities and any lessons that they might offer for builders and users of SMC systems.

The examples quoted above have close connections to optimisation. A more general meaning of "constraint" is: any condition that restricts the set of possible solutions of a problem because some candidate solutions may fail to respect it. A good example of work that starts from this observation is by Ladkin and Reinefeld [59]. Although primarily theoretical, it makes the implicit point that availability of SMC systems embodying good mathematical knowledge-representation facilities can help future studies in similar areas by providing a laboratory in which exercises on the design and testing of algorithms and heuristics can be carried out.

Environments for Manipulation of Mathematical Knowledge

Systems for the expression and use of mathematical knowledge (most often in the notation of predicate calculus) and SMC systems (largely algorithmic, though sometimes with the capability of accepting rules that modify their control regimes) have grown up in parallel, without much contact. The earliest gestures from each side towards the other have added features of the other type that were rather rudimentary by comparison with the state of the art in the other field. CAMELIA [14] and PRESS [33] are examples taken from the two different sides.

Because of the strengthening of contacts between AI and SMC, software systems that can accommodate the two types of material are developing in the direction of increased integration, with more of a balance between the types. Three examples, in what appears to an increasing order of integration, are Oyster, which has been used for constraint-based program optimisation with the help of proof plans [60], APS [61], for solving school-level algebraic problems, e.g. to help in the teaching of mathematics, and MANTRA [19] [62], which aims to integrate explicitly components for reasoning, mathematical knowledge representation, and computer algebra. To date, published reports of progress on such

projects indicate that they are at experimental or prototype stages; hence there is scope for further work on design of such systems and, equally important, the collection and representation of larger bodies of mathematical knowledge, including the heuristic problem-solving knowledge of practising mathematicians. We expect these topics to be much in evidence in the AI and SMC media in the near future.

Outlook

The intersection of AI and SMC has been in a constant state of expansion since 1992, after initial close contacts up to about 1970 followed by a period of separation and perhaps near-divorce. The separation started when SMC, which had had a large heuristic component in its early days, turned into a mainly algorithmic subject. While it is true that other topics that were once regarded as part of AI have disappeared from the AI perspective as their heuristics have been replaced by algorithms, SMC is now mature enough that we can see a more complex picture. For example, it is appreciated that heuristic knowledge about how to apply the algorithmic techniques of SMC increases the flexibility and the scope of SMC systems, and that the combination of the two approaches has value both for SMC researchers and for users. There are many intellectually challenging questions for the former, on the way to the production of more powerful systems that embody mathematical knowledge. For the users there is the prospect that the phenomenon of pen-and-paper mathematical manipulations to put problems into forms that SMC systems can accept as input will shrink or even disappear. After all, some of the earliest SMC systems were built to take this chore (which then preceded the writing or running of Fortran programs) away from the computer user. This original motivation is still useful as a spur to further research in AI and SMC.

We conclude by listing some issues that we expect to drive future research:

- Determination of the usefulness of each of the knowledge representations that are available inside AI, with respect to the storage of mathematical knowledge and its use in reasoning (which, for some representations, implies the development of improved methods of reasoning)
- Construction of mathematical knowledge-based computing environments that combine AI and SMC features more closely and in greater volumes than at present;
- Re-use of algorithmic knowledge in such environments, e.g. by methods of annotation of the algorithms, where the annotations describe the knowledge held in the algorithms, and how and where it is appropriate for the algorithms to be used;
- Evaluation of the respective roles of formal (e.g. based on logics and/or typing) and informal (e.g. case-based and analogical) mathematical knowledge and reasoning, and incorporation of both approaches inside knowledge-based computing environments;

- Examination of the behaviour of environments containing significantly larger bodies of mathematical knowledge than are represented in such environments at present - in particular, larger bodies of case-like or plan-like heuristic knowledge about mathematical problem-solving;
- Specific focusing on the activity and the associated knowledge of mathematical modelling;
- Representation and use of mathematical material that is often required in modelling but that has not yet received much attention in SMC: particularly graphs and graph theory, and the mathematics appropriate to queueing theory and to the distribution and loading of resources;
- Investigation of the mathematical knowledge that will be most useful for incorporation in autonomous agents in multi-agent systems;
- Adaptation of the mathematical knowledge-based environments that have been discussed above (and possibly the development of purpose-built new ones) for the teaching of mathematics.

References

1. J. Calmet and J.A. Campbell, *Artificial Intelligence and Symbolic Mathematical Computations*, in Artificial Intelligence and Symbolic Mathematical Computing, eds. J. Calmet and J.A. Campbell, LNCS **737**, p. 1. Springer-Verlag, Berlin, (1993)
2. A. Deprit, J. Henrard and A. Rom, Science, **168**, 1569 (1970)
3. J.A. Campbell and A.C. Hearn, J. Comput. Phys.,**5**, 280 (1970)
4. P. McCorduck, *Machines who Think*, Freeman, San Francisco (1979)
5. H.G. Kahrimanian, *Analytical Differentiation by a Digital Computer*. M.S. thesis, Temple University, Philadelphia (1953)
6. J. Nolan, *Analytical Differentiation on a Digital Computer*. S.M. thesis, Massachusetts Institute of Technology, Cambridge, MA (1953)
7. R. Slagle, *A Heuristic Program that Solves Symbolic Integration Problems in Freshman Calculus*, in Computers and Thought, eds. E.A. Feigenbaum and J. Feldman. McGraw-Hill, New York (1963)
8. J. Moses, Comm. ACM, **14**, 548 (1971)
9. R.H. Risch, Trans. Amer. Math. Soc., **139**, 167 (1969)
10. D.L. Waltz, *Understanding Line Drawings of Scenes with Shadows*, in The Psychology of Computer Vision, ed. P.H. Winston, p. 19. McGraw-Hill, New York (1975)
11. J. Illingworth and J. Kittler, Comp. Vision, Graphics and Image Proc., **44**, 87 (1988)
12. R. Brachman and H. Levesque (eds.), *Readings in Knowledge Representation*. Morgan Kaufmann, Los Altos, CA (1985)
13. G. Ringland and D.A. Duce (eds.), *Approaches to Knowledge Representation*. Science Research Associates/Wiley, Chichester, England (1986)
14. M. Vivet, *Expertise Mathématique: CAMELIA, un Logiciel pour Raisonner et Calculer*. Thèse d'Etat, Université de Paris IV, Paris (1984)
15. B. Silver, *Meta-Level Inference*. Elsevier, Amsterdam (1986)
16. D.B Lenat and S.V. Guha, *Building Large Knowledge-Based Systems*. Addison Wesley, Reading, MA (1990)

17. R.D. Jenks and R.S. Sutor, *AXIOM*. Springer-Verlag, Berlin (1992)
18. J. Calmet and I.A. Tjandra, *A Unified Algebra-Based Specification Language for Symbolic Computing*, in Design and Implementation of Symbolic Computation Systems, ed. A. Miola, LNCS **722**, p. 122. Springer- Verlag, Berlin (1993)
19. G. Bittencourt, J. Calmet, K. Homann and A. Lulay, MANTRA: *A Multi-Level Knowledge Representation System*, in Proc. 11th Brazilian Symposium on Artificial Intelligence (SBIA '94), eds. T. Pequeno and F. Carvalho, p. 493 (1994)
20. J. Hadamard, *The Psychology of Invention in the Mathematical Field*. Princeton University Press, Princeton, NJ (1949)
21. G. Polya, *Mathematics and Plausible Reasoning*, 2 vols, Princeton University Press, Princeton, NJ (1954)
22. J. Kolodner, *Case-Based Reasoning*. Morgan Kaufmann, San Mateo, CA (1994)
23. T. Sandholm and V. Lesser, *Advantages of a Leveled Commitment Contracting Protocol*, Artificial Intelligence, to be published.
24. J. Pfalzgraf, U.C. Sigmund and K. Stokkermans, *Modeling Cooperating Agents Scenarios by Deductive Planning Methods and Logical Fiberings*, in Integrating Symbolic Mathematical Computation and Artificial Intelligence, eds. J. Calmet and J.A. Campbell, LNCS **958**, p. 167. Springer-Verlag, Berlin (1995)
25. L. Overgaard, H.G. Petersen and J.W. Perram, *Motion Planning for an Articulated Robot: A Multi-Agent Approach*, in Distributed Software Agents and Applications, eds. J.W. Perram and J.-P. Müller, p. 206. Springer-Verlag, Berlin (1996)
26. N.S. Glance and T. Hogg, *Dilemmas in Computational Societies*, in Proc. First Int. Conf. on Multi-Agent Systems, ed. V. Lesser, p. 117. AAAI Press/MIT Press, Menlo Park, CA, and Cambridge, MA (1995)
27. J.O. Kephart, T. Hogg and B.A. Huberman, Phys. Rev. A, **40**, p. 404 (1989)
28. K.S. Decker and V.R. Lesser, *Designing a Family of Coordination Algorithms*, in Proc. First Int. Conf. on Multi-Agent Systems, ed. V. Lesser, p. 73. AAAI Press/MIT Press, Menlo Park, CA, and Cambridge, MA (1995)
29. J. Denzinger, *Knowledge-Baaed Distributed Search Using Teamwork*, in Proc. First Int. Conf. on Multi-Agent Systems, ed. V. Lesser, p. 81. AAAI Press/MIT Press, Menlo Park, CA, and Cambridge, MA (1995)
30. D. Maio and S. Rizzi, *Unsupervised Multi-Agent Exploration of Structured Environments*, in Proc. First Int. Conf. on Multi-Agent Systems, ed. V. Lesser, p. 269. AAAI Press/MIT Press, Menlo Park, CA, and Cambridge, MA (1995)
31. A. Knoll and J. Meinkoehn, *Hierarchical and Lateral Coordination in Multi-Agent Systems: an Analysis of Message Traffic Flow*, in Proc. First Int. Conf. on Multi-Agent Systems, ed. V. Lesser, p. 225. AAAI Press/MIT Press, Menlo Park, CA, and Cambridge, MA (1995)
32. S-C. Chou, *Mechanical Geometry Theorem Proving*. D. Reidel, Dordrecht, Netherlands (1988)
33. L. Sterling, A. Bundy and L. Byrd, *Solving Symbolic Equations with PRESS*, in Proc. EUROCAM '82, ed. J. Calmet, LNCS **144**, p. 108. Springer-Verlag, Berlin (1982)
34. G.E. Collins and H. Hong, J. Symb. Comp., **12**, p. 299 (1991)
35. H. Hong, *Heuristic Search Strategies for Cylindrical Algebraic Decomposition*, in Artificial Intelligence and Symbolic Mathematical Computing, eds. J. Calmet and J.A. Campbell, LNCS **737**, p. 151. Springer-Verlag, Berlin, (1993)
36. R. Hähnle, *A New Translation from Deduction into Integer Programming*, in Artificial Intelligence and Symbolic Mathematical Computing, eds. J. Calmet and J.A. Campbell, LNCS **737**, p. 262. Springer-Verlag, Berlin, (1993)

37. A. Rege and J. Canny, *A Practical Algorithm for Geometric Theorem Proving*, in Integrating Symbolic Mathematical Computation and Artificial Intelligence, eds. J. Calmet and J.A. Campbell, LNCS **958**, p. 10. Springer-Verlag, Berlin (1995)
38. D. Wang, *Geometry Machines: from Artificial Intelligence to Symbolic Mathematical Computation*, in Artificial Intelligence and Symbolic Mathematical Computation, eds. J. Calmet, J.A. Campbell and J. Pfalzgraf, p. 213. Springer-Verlag, Berlin (1996)
39. R. Michalski, J. Carbonell and T.M. Mitchell (eds.), *Machine Learning*, 3 vols. Tioga Press, Palo Alto, CA, and Springer-Verlag, Berlin (1983 and subsequent years)
40. D.B. Lenat, *AM: an Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search*, in Knowledge-Based Systems in Artificial Intelligence, eds. R. Davis and D. Lenat, p. 3. McGraw-Hill, New York (1982)
41. G.D. Ritchie and F.K. Hanna, Artificial Intelligence, **23**, p. 249 (1984)
42. P. Langley, *Strategy Acquisition Governed by Experimentation*, in Progress in Artificial Intelligence, eds. L. Steels and J.A. Campbell, p. 52. Ellis Horwood, Chichester, England (1985)
43. J.A. Campbell, J. Phys. A, **12**, p. 1149 (1979)
44. P. Langley, H.A. Simon, G. Bradshaw and J.M. Zytkow, *Scientific Discovery: Computational Explorations of the Creative Process*. MIT Press, Cambridge, MA (1987)
45. S. Kocabas, Machine Learning, **6**, p. 277 (1991)
46. R.E. Valdes-Perez, Artificial Intelligence, **82**, p. 331 (1996)
47. P.J. Hayes, *The Naive Physics Manifesto, in Expert Systems in the Microelectronic Age*, ed. D. Michie, p. 139. Edinburgh University Press, Edinburgh (1979)
48. K.D. Forbus, Artificial Intelligence, **24** , p. 85 (1984)
49. J. de Kleer, A.K. Mackworth and R. Reiter, Artificial Intelligence, **56**, p. 197 (1992)
50. P. Struss, paper in the present issue of this journal.
51. A.G. Cohn, *Calculi for Qualitative Spatial Reasoning*, in Artificial Intelligence and Symbolic Mathematical Computation, eds. J. Calmet, J.A. Campbell and J. Pfalzgraf, LNCS **1128**, p. 124. Springer-Verlag, Berlin (1996)
52. A. Colmerauer, Comm. ACM, **28**, p. 418 (1990)
53. R. Rueher, Software - Practice & Experience, **23**, (1993)
54. D. Bouhineau, *Solving Geometrical Constraint Systems using CLP Based on Linear Constraint Solver*, in Artificial Intelligence and Symbolic Mathematical Computation, eds. J. Calmet, J.A. Campbell and J. Pfalzgraf, LNCS **1128**, p. 274. Springer-Verlag, Berlin (1996)
55. A. Semenov, A. Babichev and A. Leshchenko, *Subdefinite Computations and Symbolic Transformations in the UniCalc Solver*, in Integrating Symbolic Mathematical Computation and Artificial Intelligence, eds. J. Calmet and J.A. Campbell, LNCS **958**, p. 264. Springer-Verlag, Berlin (1995)
56. J. Jaffar, S. Michaylov, P-J. Stuckey and R. Yap, ACM Trans. Prog. Lang. Syst., **14**, p. 339 (1992)
57. W. Leler, *Constraint Programming*. Addison Wesley, Reading, MA (1989)
58. H-W. Güsgen, *CONSAT: a System for Constraint Satisfaction*. Pitman, London (1989)
59. P. Ladkin and A. Reinefeld, paper in the present issue of this journal.
60. P. Madden and I. Green, *A General Technique for Automatically Optimizing Programs through the Use of Proof Plans*, in Integrating Symbolic Mathematical Computation and Artificial Intelligence, eds. J. Calmet and J.A. Campbell, LNCS **958**, p. 64. Springer-Verlag, Berlin (1995)

61. Y.V. Kapitonova, A.A. Letichevsky, M.S. L'vov and V.A. Volkov, *Tools for Solving Problems in the Scope of Algebraic Programming*, in Integrating Symbolic Mathematical Computation and Artificial Intelligence, eds. J. Calmet and J.A. Campbell, LNCS **958**, p. 30. Springer-Verlag, Berlin (1995)
62. K. Homann and J. Calmet, *Combining Theorem Proving and Symbolic Mathematical Computing*, in Integrating Symbolic Mathematical Computation and Artificial Intelligence, eds. J. Calmet and J.A. Campbell, LNCS **958**, p. 18. Springer-Verlag, Berlin (1995)

This article was processed using the LaTeX macro package with LLNCS style