Preface

In Memory of William W. McCune (1953–2011)

This volume is a tribute to the memory of William (Bill) McCune, whose sudden death on May 4, 2011, left the field of automated reasoning deprived of one of the founders of practical theorem proving and model building. While he was an accomplished computer scientist all around, Bill McCune was especially a fantastic system builder and software engineer. He developed a series of systems of astounding power, robustness, useability, and portability, including the theorem provers *Otter*, *EQP*, and *Prover9*, the parallel prover *ROO*, the proof checker *Ivy*, the prototype SAT-solver *ANL-DP*, and the model builders *Mace2* and *Mace4*.

Bill McCune's Scientific and Professional Contributions

Originary of New Hampshire, Bill did his undergraduate studies in mathematics at the University of Vermont, and completed his education with an MS and a PhD in computer science from Northwestern University, with adviser Larry Henschen. After the PhD, he started his career as Assistant Computer Scientist at the Mathematics and Computer Science (MCS) Division of the Argonne National Laboratory (ANL), at Argonne, near Chicago. He was soon promoted to Computer Scientist, and later became Senior Computer Scientist. Bill spent most of his professional life in the red-brick building hosting the MCS Division on the vast and quiet ANL campus. He stayed there until the unexpected demise of the automated reasoning program in 2006, when he joined the Department of Computer Science of the University of New Mexico as Research Professor.

ANL is a research laboratory mostly funded by the federal government of the United States through its Department of Energy. A primary mission of its MCS Division is to advance research in computing and mathematics to enable the solving of the mathematical and computational challenges posed by research in physics and other natural sciences. Perhaps surprisingly, and due in part to the lack of specialization in the early days of computer science, Argonne, was, and still remains from a historic point of view, the cradle of automated reasoning. J. Alan Robinson worked on *resolution* and *unification* during summer jobs at Argonne during 1962–1964, writing the milestone article on "A Machine-Oriented Logic" in the summer of 1963. Approximately in the same years, Larry Wos started at ANL a research program in automated reasoning, where *paramodulation*, or resolution with equality built-in, *demodulation*, or the *ad hoc* use of equations for rewriting, and the *set of support strategy* were invented in the years 1965–1969. Work on automated reasoning at Argonne continued during the 1970s and early 1980s. When Bill McCune started at ANL in 1984, he joined Larry Wos, Ewing L. (Rusty) Lusk, and Ross Overbeek. The Argonne theorem provers in that period were ITP, LMA/ITP, and AURA, which already featured an early version of the *given-clause algorithm* later popularized by Otter.

A positive consequence of Argonne's great inventions was the persuasion that time was ripe to focus on implementation and system building. A less positive one was the notion that research in theory was almost over. In reality, in the mid-1980s the quest for building equality into resolution was not over. The *Wos-Robinson conjecture*, namely, whether paramodulation is complete without paramodulating into variables and adding instances of reflexivity – the *functionally reflexive axioms* – was not settled. Also, it was not known how to control the termination of demodulation, that was called k-demodulation, because it depended on an *ad hoc* bound k on the number of allowed rewriting steps.

In the same years when Larry Wos and G. Robinson invented paramodulation, Don Knuth and his student Peter B. Bendix devised a *completion procedure*, featuring *superposition*, or paramodulation between the left sides of two rewrite rules, and *simplification* of a rewrite rule by another. A key idea was that equations were oriented into rewrite rules by a *well-founded ordering* on terms. In 1981, Gérard Huet proved that Knuth-Bendix completion is correct: if it does not fail by generating an equation that cannot be oriented, it generates in the limit a *confluent rewriting system*, and it semi-decides the validity of equational theorems, as suggested independently also by Dallas Lankford. At the IEEE Symposium on Logic in Computer Science (LICS) of 1986, Leo Bachmair, Nachum Dershowitz, and Jieh Hsiang reobtained these results in a more general framework based on *well-founded proof orderings*. At the 8th International Conference on Automated Deduction (CADE-8), held at Oxford in July 1986, Jieh Hsiang and Michäel Rusinowitch showed that *ordered resolution* and *ordered paramodulation*, restricted by a well-founded ordering, are refutationally complete *without functionally reflexive axioms*. During 1987–1989, Jieh Hsiang and Michäel Rusinowitch, on one hand, and Leo Bachmair, Nachum Dershowitz, and David A. Plaisted, on the other, came up independently with *unfailing*, or *ordered*, completion, which works for equations with no need of orienting them once and for all into rewrite rules. Throughout the 1980s, Nachum Dershowitz, David A. Plaisted, and others worked systematically on well-founded orderings, their properties, and termination of rewriting for theorem proving.

Bill McCune's greatness at that time was that he deeply understood the rewriting and completion research developed elsewhere, and united it with the best results of the Argonne tradition in a new theorem prover named *Otter*. Otter stands for *Organized techniques for theorem-proving and effective research*, and it is also the name of a rare semiaquatic mammal, that inhabits rivers and unites a playful, shy nature with the determination of a skilled hunter. The release of Otter at CADE-9 in 1988 was a turning point in the history of automated reasoning. Never before had the computer science community seen a theorem prover of such awesome power. Otter proved theorems in full first-order logic with equality with amazing speed, relative to the technology of the day. It was almost surely sound,[1] and endowed with both complete and incomplete strategies, with

[1] Though Bill used to joke that he would not jump off a bridge, if a soundness bug were exposed in Otter or any other of his systems.

the latter often most useful in practice. Otter quickly became the touchstone for an entire field.

Already in the early 1990s, Otter matured into a robust, reliable, and portable prover, with *options* and *parameters* that the experimenter could set to define the adopted strategy. Over time, Bill added features such as the *pick-given ratio* to mix clause evaluation functions, *hints* and a *hot list* to guide the search, and an *auto(matic) mode* enabling the prover to choose by itself the strategy based on the input's syntax. Several of these enhancements came from Bill's readiness to learn from experiments, including those carried out by others, and to integrate users' suggestions or requests with his own apparently infallible sense of what was practical to implement. Notwithstanding its wealth of features, Otter was remarkably easy to use, and therefore a significant user community grew world-wide, including scientists not working in theorem proving, and especially mathematicians and logicians. Indeed, Larry Wos and Bill McCune shared a keen interest in applying theorem proving to mathematics, especially algebra, geometry, and logic, also building on historic ties that the MCS Division of ANL had with the Department of Mathematics of the University of Chicago.

However, perhaps Otter's greatest impact was due to Bill's generous and far-looking decision to make its source code publicly available. It is impossible to describe completely a reasoning program in research papers. There is always some amount of knowledge, often a surprising amount, that is written only in the code, and therefore remains hidden, if the code is not public or is too hard to read. Bill's code was admirably readable and well organized. Other researchers, including those whose systems eventually overtook Otter in speed or in variety of inference rules, also learnt from Bill's code data structures, algorithms, and indexing schemes, which are fundamental for implementing theorem provers.

Although Bill developed Otter for several years, he had a clear sense that it may not be wise to try to put too many features in one system. For instance, he refused to implement in Otter *reasoning modulo associativity and commutativity* (AC). Rather, he built another theorem prover, called *EQP*, for *equational prover*, that had *AC-matching* and *AC-unification*, but worked only for equational theories. Bill always considered EQP as a prototype to be used by himself and a few others, rather than a system for all like Otter. EQP was written with a specific goal in mind: proving the *Robbins conjecture*, an open problem in mathematics whose existence Larry Wos had discovered in his continuous quest for challenges for theorem provers.

The Robbins conjecture dated back to 1933, when a mathematician, named E. V. Huntington, demonstrated by hand that a certain equation, later called *Huntington axiom*, was sufficient, together with associativity and commutativity of addition, to axiomatize Boolean algebra. In the same year, another mathematician, Herbert Robbins, conjectured that the same was true of another equation, later called *Robbins axiom*. A proof was not found, and algebras presented by Robbins axiom, together with associativity and commutativity of addition, became known as Robbins algebras. In 1990-1992, S. Winker proposed two equations, later called *first Winker condition* and *second Winker condition*,

and proved by hand that if each of them is added to the Robbins axiom, the Huntington axiom follows. This led to decomposing the problem into proving that Robbins axiom implies the second Winker condition, the second Winker condition implies the first, and the first implies Huntington axiom. While the first step was relatively easy, the other two remained beyond the possibilities of theorem provers.

In 1996, EQP proved them both, thereby solving a problem that had challenged mathematicians for over 60 years. The first successful run for the hardest lemma, the one showing that the second Winker condition implies the first, in February 1996, took an impressive 12.5 days of computation on a 486DX2/66. The experiment was repeated on an RS/6000 taking 7.2 days. In the same year, Bill also succeeded in having EQP prove that Robbins axiom implies Huntington axiom in one run, in order to have a single mechanical proof. A human version of the proof was extracted from the mechanical one for readability and persuasion. The field reacted with awe, and this momentous achievement brought unprecedented visibility to automated deduction, artificial intelligence, and the whole of computer science. Bill McCune and EQP made it onto the pages of the *New York Times*. One of the very early dreams of artificial intelligence, namely, machines capable of proving mathematical theorems that human experts could not prove, was no longer only a dream: it was a reality. The Robbins algebra proof is an ideal example of a successful blending of multiple strands of research in automated reasoning: the extensive experimentation characteristic of Argonne, new theory about equality reasoning, and associative-commutative unification.

While obtaining these outstanding results in theorem proving, Bill was also among the first to understand the importance of the dual problem of *model building*, or *theorem disproving*. A precipitating event was the solution of open problems of *quasigroup existence* in 1992. The problems were posed by mathematician Frank Bennett, and solved initially by Masayuki Fujita, with ICOT's Model Generation Theorem Prover (MGTP), and by John Slaney, with his model builder FINDER. The opportunity and excitement of solving open problems stimulated a lot of activity by researchers, including the second editor of this volume, Hantao Zhang, and Bill McCune, on new, advanced implementations of the Davis-Putnam-Loveland-Logemann (DPLL) procedure, known since 1960-1962, to decide propositional satisfiability (SAT), and find models. A Japanese-American Workshop on Automated Theorem Proving that focused on finite domain theorem proving was held at Duke University in March 1994. Frank Bennett, Masayuki Fujita, Bill McCune, Hantao Zhang, and the second editor of this volume, were among the attendees. A few months later, in June 1994, Bill McCune announced his DPLL-based SAT-solver ANL-DP, which was already being applied to solving quasigroup existence problems.

Bill was too involved with first-order reasoning to delve into SAT, and ANL-DP remained a prototype. Since DPLL works by trying to build a model, and reports unsatisfiability when it has found that none exists, ANL-DP was preparatory work for Bill's next great system, the SAT-based finite model finder *Mace*, whose most successful version was *Mace2*. The mathematical community that

Bill supported benefited enormously from his providing a model builder as well as a theorem prover. As we shall see, several chapters of this volume report results that depend on both. For all these achievements, in 2000 Bill McCune received *the Herbrand Award*, the highest honor in automated reasoning and one of the highest in computer science.

Bill McCune was not the kind who would rest on his laurels. Although he maintained Otter through August 2003, which is the date of that glorious theorem prover's last manual, Bill knew that Otter had become too old to continue developing it. Thus, he embarked in designing a brand new theorem prover for first-order logic with equality, called *Prover9*, and with the ever optimistic, forward-looking subtitle "*the future of theorem proving.*" The years 2005-2010 were devoted to Prover9 and *Mace4*, a new Mace, no longer SAT-based, but using a more general constraint solving approach. Prover9 and Mace4 inherited all the great qualities of their predecessors Otter and Mace2, as witnessed by the fact that they are still very much in use today.

Not only was Bill McCune a marvelous system builder, he also wanted to make it easy for others to build reasoning programs. In addition to making his own code available, already in 1992–1993, he had the idea of building a software library to assemble theorem provers. The version dating from those years was named OPS, for *Otter parts store*. Later it evolved into a new library, called LADR, or *Library for Automated Deduction Research*. Bill was probably also the first to think of a web-based interface to let anyone play with an automated reasoner: *Son of BirdBrain* gave access to Otter and Mace2, and *Son of BirdBrain II* gave access to Prover9 and Mace4.

Since he worked for most of his career in a research laboratory, Bill did not have PhD students. However, he mentored through cooperation several junior colleagues. Also, he served the scientific community as the first Secretary of the Association for Automated Reasoning (AAR) in 1986–1993, as co-organizer of CADE-9 at Argonne in 1988, as Program Chair of CADE-14 in Townsville, Australia, in 1997, and as Trustee of CADE Inc. in 1997–2000. Bill McCune was remembered with heart-felt speeches by several colleagues at CADE-23 in Wrocław, Poland, in August 2011, and at the 8th International Workshop on First-Order Theorem Proving (FTP) in Bern, Switzerland, in July 2011.

Some Recollections of Bill McCune by Maria Paola Bonacina

It is likely that my first interaction with Bill was by e-mail, when he was AAR Secretary, I was a graduate student, and I wanted to become a member of the AAR. As part of my PhD work at SUNY Stony Brook, I implemented a distributed version of Otter, called *Aquarius*, later presented at the Third International Symposium on Design and Implementation of Symbolic Computation Systems (DISCO), in Gmunden, Austria, in September 1993. The availability of Bill's code helped the implementation part of my thesis enormously. It meant that I could focus on implementing the methodology for distributed deduction that I was proposing in the thesis, called *clause diffusion*, reusing Bill's code

for everything else. It was my first opportunity to appreciate the clarity and robustness of his code. Also, Bill's turn-around time by e-mail was fantastic.

My adviser, Jieh Hsiang, suggested inviting Bill to my PhD defense committee. Bill kindly accepted, and travelled from Argonne to Stony Brook for the defense in December 1992. Later Larry Wos told me that Bill went back to Argonne so excited about distributed deduction, that he implemented right away a prototype with a master-slave architecture for equational reasoning modulo AC. Indeed, that was when Bill was starting to expand the OPS with code that would lead later to the development of EQP. After my defense, Bill invited me to visit Argonne in January–February 1993, before starting a postdoc at INRIA Lorraine in Nancy. Thus, I had a wonderful opportunity to work side by side with Bill McCune, and discuss research with Larry Wos, Rusty Lusk, and Ross Overbeek. At that time, Rusty and Ross were interested in parallel programming, and, together with other colleagues at Argonne, had developed the *the p4 Parallel Programming System*. It was a perfect match, and Bill and I worked together on implementing a new clause diffusion prover, using p4 and the OPS code for equational reasoning modulo AC. Upon Bill's suggestion, the prover was called *Peers*, because in clause diffusion there was no master-slave concept, and all deductive processes cooperated as peers. The Peers prover was presented at CADE-12 in Nancy in June 1994.

When I started as Assistant Professor in the Department of Computer Science of the University of Iowa in the fall of 1993, I was asked to serve as Colloquium Chair, and encouraged to invite leading figures in my field, including mentors. Because Jieh Hsiang had moved from Stony Brook to the National Taiwan University, my first invitation was for Bill McCune. During my years at the University of Iowa, our scientific paths diverged somewhat. Bill delved into the implementation of EQP and the final attack to the Robbins problem. I pursued other topics, including a new version of clause diffusion, called *modified clause diffusion*. However, I continued to benefit from Bill's code: my next distributed theorem prover, *Peers-mcd*, implemented modified clause diffusion using MPI (message passing interface), and EQP as sequential base. In July 1997, Peers-mcd was presented at CADE-14, and at the Second International Symposium on Parallel Symbolic Computation (PASCO), in Maui, Hawaii, where the experimental report included instances of super-linear speed-up, made possible by distributed search, in the proofs for the Robbins problem.

I have fond memories of working and discussing with Bill, especially during my visits at Argonne in January-February 1993 and June 1998. In addition to being such a great computer scientist, Bill McCune was a very fine gentleman: he had an admirable self-control, used very few words, and was very kind. I never saw him upset or losing his temper about anything. It seemed impossible that such a thing could ever happen. In January-February 1993, Bill had just gotten his beloved dog, and was very concerned that the puppy would cry if left alone all day. Thus, he would quit the office early and continue work at home. His usage of time was extremely effective. He used to say, "There is a whole lot we can do in 15 minutes!", and it was true. Bill was very open towards people.

When I arrived at Argonne in January 1993 he asked me how long I had been in the States, and when I answered, "Three years and half," he said, "You're one of us!" When I visited again in June 1998, and we found that ANL had tightened security checks on visitors without a US passport, Bill commented that it was nonsense. He could not see how the trustworthiness of people had anything to do with their passport. During my visits at Argonne we used to have lunch at the cafeteria on campus. A couple of times we took extended walks to go see a historic linear accelerator of the Physics Division of ANL.

I never knew how Bill really felt about the termination of the automated reasoning program at ANL. I did not dare to inquire too much. In earlier conversations, Bill had told me that he did not necessarily see himself spending the rest of his career at Argonne. He said at some point he might have moved into teaching, preferably at some university or college in New England. When his position at ANL was terminated, for some time it looked like he would stay in Illinois as a sort of free-lance researcher. Then, he joined the Department of Computer Science of the University of New Mexico. I invited him to visit my new Department in Verona, but at that time he was not too keen on long flights. We mentioned me visiting his new department in Albuquerque, but regrettably it did not happen.

I would like to dedicate the work I personally put in the editing of this volume to the memory of my mother, Annita Fusari, who unexpectedly passed away on May 21, 2011, a couple of weeks after Bill McCune.

Some Recollections of Bill McCune by Mark E. Stickel

Although I was not one of Bill McCune's collaborators or long-term visitors, our joint attendance at workshops and conferences was always a welcome opportunity for discussion.

A memorable encounter was at the Workshop on Empirically Successful First Order Reasoning (ESFOR) at the International Joint Conference on Automated Reasoning (IJCAR) 2004 in Cork, Ireland. Bill and I enthused about Stephan Schulz's *feature vector indexing* (described and refined in his chapter in this volume) over beers the evening after his presentation. Bill was quick to test the idea, reported six months later to Stephan and me that he had tried it, and praised it highly. Indexing is crucial to the performance of theorem provers and, as a top system builder, Bill paid close attention to it. Back in 1989 he was also quick to implement my then new *path indexing* method, alongside with *discrimination tree indexing*, in Otter. Through comprehensive testing, he created test sets of terms that were long used by other researchers to evaluate indexing methods.

Also in 1989, Argonne asserted the near indispensability of subsumption in automated reasoning in the article "Subsumption, a Sometimes Undervalued Procedure" by Ross Overbeek, Larry Wos, and Ewing Lusk, that appeared in 1991 in the volume edited by Jean-Louis Lassez and Gordon Plotkin in honor of Alan Robinson. Approaches like my *Prolog technology theorem prover* (PTTP), which is based on logic programming, generally lack subsumption for controlling redundancy. As another instance of his openness to appreciating different

theorem-proving paradigms, Bill once told me that he *liked* PTTP, a great encouragement and then a cherished memory.

PTTP can also be criticized for lacking another feature Argonne found to be nearly indispensable: making inferences from clauses in order of ascending weight in the given-clause algorithm, instead of using first-in-first-out order (i.e., level saturation) or depth-first search from the goal as in PTTP. There is considerable justification for these criticisms; however, PTTP and other systems that lack subsumption and weight-ordered search occasionally surprised the field by finding proofs contrary to expectation. This revealed that an overreliance on weight-ordered search may be a weakness. I do not know if PTTP influenced his thinking, but Bill also saw the weakness and a solution. His pick-given ratio allows the system to choose clauses for inference either by weight or first-in-first-out ordering in alternation according to a user supplied ratio. This can be used to avoid the problem of large clauses, especially those derived early but necessary for a proof, being postponed too long.

Bill and I were both system builders who learned from each other's systems. I often consult Otter or Prover9 code to see how Bill did things, and Bill looked at my implementation of the DPLL procedure when developing ANL-DP and my implementation of AC-unification when developing EQP. We both strived to build a library of code that others could use for building systems. I would have liked to use his LADR, but my preference for programming in Lisp is too strong. We shared the attitude inherited from the Argonne tradition that new problems or application areas often require considerable user input. Fully automated proof is not always feasible and opportunities for user control of the proving process should be provided in abundance. This is illustrated, for example, in the vital role of Robert (Bob) Veroff's hint mechanism in chapters in this volume. As a system builder myself, I wish to emphasize that Bill was peerless in making his systems valuable to a large community of users, especially mathematicians, by excellent design, implementation, documentation, outreach, and support.

Outline of This Volume

We had the idea of this volume back in May 2011, when the field of automated reasoning was still under shock at the news of Bill McCune's sudden passing. We were encouraged by Larry Wos, Deepak Kapur, and Bob Veroff: we thank them for their support. A first call for papers appeared in September 2011 and was repeated a few months later: we thank Carsten Schürmann for helping with the publicity. We received 15 submissions and accepted 13 of them. Each paper had at least two and up to four reviewers, who wrote accurate and detailed reviews: we thank them all for their precious cooperation. All accepted papers were thoroughly revised, and in some cases extended, by their authors.

The volume is organized as follows. The first article is a recollection of Bill McCune by Larry Wos, his main colleague and friend at ANL. Larry describes how Bill approached proving theorems in mathematics, especially with Otter, and how Bill and Larry cooperated in the search for shorter or otherwise more

elegant proofs. We thank Gail Pieper for helping Larry with the editing. Then there are four articles on core topics where Bill gave fundamental contributions: *strategies*, *indexing*, *superposition-based theorem proving*, and *model building*.

Leonardo De Moura and Grant Olney Passmore discuss the strategy challenge in SMT-solving, where SMT stands for satisfiability modulo theories. This article debunks the notion that SMT-solvers are push-button tools, whereas first-order theorem provers depend heavily on heuristics, and user-defined parameters and options. Both kinds of system need heuristic search, feature search strategies, and may involve user-defined settings, based on the problem's difficulty. Indeed, SMT-solvers *are* theorem provers and model builders. The importance of strategies is such that the authors propose enabling users to exert strategic control, so that they can program, in a sense, their strategies. Thus, this work also advances Bill's vision of enabling others to build reasoners.

Stephan Schulz presents a simple and efficient indexing technique for *clause susbumption*, called *feature vector indexing*. This article focuses on *clause indexing*, when most of the literature emphasizes *term indexing* for unification and matching. As Bill McCune was among the initiators of the research in indexing, this article continues a central topic in his research program. Thomas Hillenbrand and Christoph Weidenbach contribute an article on *superposition-based decision procedures* for theories of *bounded domains*, including, for example, the theory of *bitvectors*. This article connects two fundamental themes in Bill McCune's research: superposition-based deduction and reasoning about finite domains.

The latter topic leads to the article on *finite model generation* by Jian Zhang and Hantao Zhang. After touching briefly on SAT-based model generators, such as Bill McCune's Mace2 and Hantao Zhang's ModGen, the authors analyze model builders based on constraint solving, comparing Mace4 with their SEM. The article presents the paradigm of *backtracking search* for constraint solving applied to model finding, and then treats in greater detail two main issues of this paradigm: *heuristics to choose assignments* and inference rules for *constraint propagation*. Apparently, there has not been much exchange between this kind of work and SMT-solving: this volume might help establish connections.

The core of the volume is devoted to the *application of automated reasoning to mathematics*, which Bill pursued throughout his career. Ranganathan Padmanabhan was Bill's companion of investigations in the realms of ternary Boolean algebras, cubic curves, lattices, ortholattices, and more. Together they wrote a book on *Automated Deduction in Equational Logic and Cubic Curves* that appeared in the LNAI series of Springer as volume 1249. Ranganathan Padmanabhan contributed two articles in *geometry*. The one with Bob Veroff reports on proving new theorems about *cubic curves* with Prover9, and discusses the pros and cons of building theory properties in the inference system versus using generic inference rules, an ever-returning issue in automated deduction. The one with Eric Ens describes using Prover9 to prove theorems connecting *projective planes* and *abelian groups*.

This leads us from geometry to *algebra*: Michael Kinyon, Petr Vojtěchovský, and Bob Veroff contributed an article on applying Prover9 to reason about alge-

braic structures, including *loops* and *quasigroups*, by using proof hints and proof sketches. The next two articles take us from algebra to *logic*: Rob Arthan and Paulo Oliva investigate *continuous logic*, finding counter-examples with Mace4 and proofs with Prover9; Branden Fitelson studies an axiomatic approach to proving a theorem in *sentential logic* called *Gibbard's collapse theorem*. Prover9 is used to prove it, while Mace4 is used to show that the axioms in the proposed axiomatization are pair-wise independent, by exhibiting counter-models to the conjecture that one depends on the others.

The third part of the volume collects articles on applications of automated reasoning that Bill's work contributed to make possible: *program verification*, *data mining*, and *computer formalized mathematics*. Deepak Kapur, Zhihai Zhang, Hengjun Zhao, Matthias Horbach, Qi Lu, and Thanvu Nguyen investigate deductive methods for *invariant generation* by quantifier elimination. Zachary Ernst and Seth Kurtenbach explore how to apply data mining and statistical learning techniques, which paid off in fields such as *computational biology* or *computational linguistics*, to theorem proving, which also deals with dazzling amounts of data. The volume is closed by an article on a grand project that Bill McCune cared for, as witnessed by his engagement with the *QED manifesto*: *computer formalization of mathematics*. Josef Urban and Jiří Vyskočil survey recent work on interfacing the Mizar library and proof assistant for formalized mathematics with automated theorem provers and model builders.

As computer scientists designing algorithms, we are trained to refrain from brute-force solutions, and seek to instill in our programs as much intelligence as possible. However, as computer scientists we are fascinated with computers, and with the unavoidably brute-force character, in a way, of mechanical solutions: there is simultaneously intelligence and brute force in a machine playing chess *à la* Deep Blue, answering questions *à la* Jeopardy, and proving theorems *à la* Otter. The balance between the two is a constantly renewed challenge of artificial intelligence. Thus, we would like to close this preface with a quote from the article by Zachary Ernst and Seth Kurtenbach: "We do not expect a mathematician to work from scratch re-proving everything each time, so why would we want that from a theorem prover?" Indeed, when it comes to machines, humans sometimes set the threshold unreasonably high, perhaps because there is still a certain reluctance to renounce the assumption of a human monopoly on intelligence, and admit that there is intelligence in machines, and there is intelligence in non-human animals. Bill McCune did engineer wonderful intelligent machines, and the best way to honor his legacy is to continue pursuing this research.

January 2013

Maria Paola Bonacina
Mark E. Stickel
Editors

Referees

Vincent Aravantinos
Peter Baumgartner
Nikolaj Bjørner
Thierry Boy de la Tour
Mnacho Echenim
Raúl Gutiérrez
Ulrich Furbach
Jaap Kamps
Deepak Kapur
Alexander Leitsch
Antoine Miné
Erik Parmann
Lawrence C. Paulson
Stefan Ratschan
Alexandre Riazanov
Christoph Ringeissen
Renate A. Schmidt
Viorica Sofronie-Stokkermans
Geoff Sutcliffe
Cesare Tinelli
Laurent Vigneron
Uwe Waldmann
Hantao Zhang

Please insert here on a page of its own the photograph wwm-canterbury.jpg

Fig. 1. William W. McCune (1953–2011)

Table of Contents

Automated Reasoning and Mathematics: Essays in Memory of William W. McCune

The Legacy of a Great Researcher*

Larry Wos

Argonne National Laboratory
Argonne, Illinois, USA

* This work was supported by the U.S. Department of Energy, under Contract DE-AC02-06CH11357.

The Strategy Challenge in SMT Solving

Leonardo de Moura[1] and Grant Olney Passmore[2,3]

[1] Microsoft Research, Redmond
[2] Clare Hall, University of Cambridge
[3] LFCS, University of Edinburgh

Simple and Efficient Clause Subsumption with Feature Vector Indexing

Stephan Schulz

Institut für Informatik, Technische Universität München,
D-80290 München, Germany, schulz@eprover.org

Superposition for Bounded Domains

Thomas Hillenbrand and Christoph Weidenbach

Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
D-66123 Saarbrücken, Germany
{hillen,weidenbach}@mpi-inf.mpg.de

Mace4 and SEM: A Comparison of Finite Model Generators

Hantao Zhang[1] and Jian Zhang[2]

[1] Department of Computer Science
The University of Iowa
Iowa City, IA 52242, USA
hantao-zhang@uiowa.edu
[2] State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing 100190, China
zj@ios.ac.cn

Group Embedding of the Projective Plane PG(2, 3)

Eric Ens and Ranganathan Padmanabhan

University of Manitoba, Winnipeg, Manitoba, Canada

A Geometric Procedure with Prover9

Ranganathan Padmanabhan[⋆1] and Robert Veroff[2]

[1] University of Manitoba, Winnipeg, Manitoba, Canada
padman@cc.umanitoba.ca
[2] University of New Mexico, Albuquerque, New Mexico, USA
veroff@cs.unm.edu

⋆ Partially supported by a University of Manitoba research leave grant.

Loops with abelian inner mapping groups: an application of automated deduction*

Michael Kinyon[1], Robert Veroff[2], and Petr Vojtěchovský**[1]

[1] Department of Mathematics, University of Denver,
Denver, CO 80208 USA
mkinyon@math.du.edu www.math.du.edu/~mkinyon
petr@math.du.edu www.math.du.edu/~petr
[2] Department of Computer Science, University of New Mexico,
Albuquerque, NM 87131 USA
veroff@cs.unm.edu www.cs.unm.edu/~veroff

* Dedicated to the memory of William McCune (1953–2011).
** Partially supported by Simons Foundation Collaboration Grant 210176.

(Dual) Hoops Have Unique Halving

Rob Arthan and Paulo Oliva

Queen Mary University of London
School of Electronic Engineering and Computer Science
Mile End Road, London E1 4NS, UK

Gibbard's Collapse Theorem for the Indicative Conditional: An Axiomatic Approach

Branden Fitelson

Department of Philosophy
Rutgers University
New Brunswick, New Jersey, USA

Geometric Quantifier Elimination Heuristics for Automatically Generating Octagonal and Max-plus Invariants*

Deepak Kapur[1], Zhihai Zhang[2], Matthias Horbach[1], Hengjun Zhao[3], Qi Lu[1], and ThanhVu Nguyen[1]

[1] Department of Computer Science, University of New Mexico, Albuquerque, NM, USA
[2] School of Mathematical Sciences, Peking University, Beijing, China
[3] Institute of Software, Chinese Academy of Sciences, Beijing, China

* Partially supported by NSF grants CCF-0729097 and CNS-0905222, by a fellowship from the Postdoc Program of the German Academic Exchange Service (DAAD), and by EXACTA and the China Scholarship Council.

Toward a Procedure for Data Mining Proofs

Zachary Ernst[1] and Seth Kurtenbach[2]

[1] Department of Philosophy, University of Missouri, USA
[2] Department of Computer Science, University of Missouri, USA

Theorem Proving in Large Formal Mathematics as an Emerging AI Field

Josef Urban[1]* and Jiří Vyskočil[2] **

[1] Radboud University Nijmegen, The Netherlands
[2] Czech Technical University

* Supported by The Netherlands Organization for Scientific Research (NWO) grants *Knowledge-based Automated Reasoning* and *MathWiki*.
** Supported by the Czech institutional grant MSM 6840770038.

Author Index