Volume 2, Number 1
January 2015

Published by the Association for Computing Machinery
Special Interest Group on Logic and Computation

SIGLOG *news*

TABLE OF CONTENTS

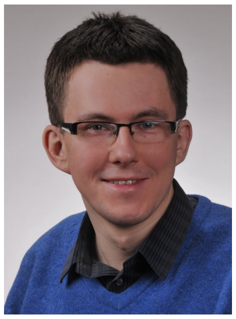
Association for Computing Machinery
Advancing Computing as a Science & Profession

SIGLOG NEWS

Published by the ACM Special Interest Group on Logic and Computation

SIGLOG News (ISSN 2372-3491) is an electronic quarterly publication by the Association for Computing Machinery.

From the Editor



Welcome to the January issue of SIGLOG News!

In this issue

– SIGLOG's Chair unveils the new logo.

– We pay tribute to the late *Robert McNaughton (1924-2014)* with an obituary by John Corcoran, Paliath Narendran and Wolfgang Thomas. The significance of McNaughton's Theorem is further discussed in the Automata Column by Mikołaj Bojańczyk.

– Anuj Dawar surveys fixed-point logic with counting in Neil Immerman's Complexity column.

– In Matteo Maffei's Security and Privacy column, Cătălin Hriţcu's reports on the recently organized "Joint EasyCrypt-F*-CryptoVerif School 2014" and Véronique Cortier presents solutions and challenges in verifying e-voting protocols.

– Homotopy Type Theory takes centre stage in Mike Mislove's Semantics column: Steve Awodey and Robert Harper provide an account of the latest developments.

– Neha Rungta sets out her vision for the Verification column.

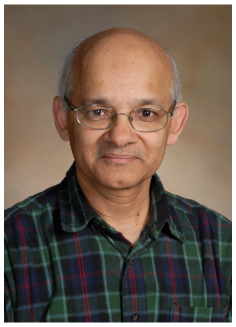
– We end with a survey on Theory in Practice for System Design and Verification by Rajeev Alur, Thomas Henzinger and Moshe Vardi.

SIGLOG News is looking for volunteers for coordinating sections on conference reports and book reviews. Please email `editor@siglog.org` if you are interested. Prakash will be going to POPL this January and will kick off with a report on that.

Happy New Year!

Andrzej Murawski
University of Warwick
SIGLOG News Editor

Chair's Letter



The season of holidays and rushing to meet the major conference deadlines is upon us. Best wishes to everyone on both counts.

SIGLOG has a new logo. Congratulations to Peter Selinger for his simple but elegant design, so like Peter's own outstanding research!

The most important recent news is that ACM and EATCS and EACSL have approved reciprocity agreements for discounted memberships. This is monetarily a small step but a major step in fostering the spirit of cooperation between SIGLOG and our allied organizations. This summer, the ACM-IEEE Symposium on Logic in Computer Science (LICS) and the International Colloquium on Automata, Languages and Programming (ICALP) will take place in Kyoto, Japan. The last such co-located event was 2007 in Wroclaw. Coming on the heels of the highly successful FLoC last summer in Vienna, this is another opportunity to strengthen ties with related communities.

Our membership is close to 200 which is good, but we aim to do better. I will start a serious recruiting effort in the coming conference season.

Prakash Panangaden
McGill University
ACM SIGLOG Chair

TRIBUTE

Robert McNaughton (1924-2014)

The scientific community receives with sadness the news of the death of Robert McNaughton, a pioneer of theoretical computer science who has shaped our field by his ingenious contributions reported in a large number of lucid and highly influential papers. Bob McNaughton grew up in Brooklyn in New York City. His undergraduate degree is from Columbia University and his doctorate from Harvard, where his advisor was Willard Van Orman Quine. Several of his fellow Quine students became distinguished logicians: William Craig, Henry Hiz, Hughes Leblanc, John Myhill, and Hao Wang. Starting from his dissertation *On Establishing the Consistency of Systems* (1951), McNaughton's early work - often in collaboration with Wang - was devoted to set theory and problems of relative consistency. At the same time he made fundamental contributions to philosophy of mathematics (in *Philosophical Review*) and to the metamathematics of number theory (in *Transactions of the American Mathematical Society*).

In the late 1950's and early 1960's, at the Moore School of Electrical Engineering of the University of Pennsylvania, he turned to the theory of finite automata and regular languages. An influential paper with his Penn PhD student Hisao Yamada supplied a lucid treatment of finite automata in relation to regular expressions. In the 1960's, he moved to Rensselaer Polytechnic Institute, where he stayed until his retirement.

In the sequel McNaughton was one of the key researchers founding a classification theory of regular languages. This is documented by the monograph (with S. Papert)

Counter-Free Automata, in which he tied the class of star-free languages to first-order logic, supplying also several other characterizations, e.g., in terms of permutation-free automata and loop-free nerve nets, and taking up also Schützenberger's Theorem on the equivalence between star-free expressions and group-free monoids. Further topics of this research were the notions of "loop complexity" of finite automata and their relation to star-height, and a characterization (with Kim and McCloskey) of the class of locally testable languages.

By the mid-sixties McNaughton had become an established name in four fields: philosophy, mathematical logic, formal linguistics, and computer science.

For many researchers, McNaughton is best known for his work on automata over infinite strings. In his landmark paper of 1966, *Testing and Generating Infinite Sequences by a Finite Automaton*, he demonstrated the central theorem of omega-automata theory, namely the determinization of nondeterministic Büchi automata by a transformation into deterministic Muller automata. Often this result is referred to as "McNaughton's Theorem". His ingenious construction motivated research that is pursued until today, aiming at reducing the number of states of deterministic automata and at unifying determinization with other constructions (such as complementation).

At the same time he addressed "Church's synthesis problem", which asks for a construction of a non-terminating finite-state transducer, given a specification of the desired relation between input stream and output stream. It was McNaughton who proposed to study this problem in the framework of infinite games, thus starting a development which has led to a fruitful and active branch of current theoretical computer science. McNaughton's proposal was first presented in a 1965 MIT technical report *Finite-State Infinite Games*. Also his later paper (of 1993) *Infinite Games Played on Finite Graphs* and his last contribution in this area, titled *Playing Infinite Games in Finite Time* (2000), were influential in developing this theory and motivating new approaches to solve infinite games.

Another important area of McNaughton's research, into which he entered in the early 1980's, was string rewriting systems (or Thue systems). One of the outcomes of this research, done together with his doctoral student P. Narendran and with F. Otto, was the concept of "Church-Rosser Languages" or "McNaughton Languages", leading to the JACM (1988) paper *Church-Rosser Thue Systems and Formal Languages*. He also made contributions to the study of special string rewriting systems (where one side of the identities is the empty word) and to the termination problem for single-rule string rewriting systems.

Bob McNaughton was a great teacher. His PhD students include John Corcoran, Anthony Dos Reis, David Hannay, Robert McCloskey, Paliath Narendran, Gil Porter, John Spagnuolo, Robert Winder, and Hisao Yamada. For many other former students, among them Aravind Joshi and Samuel Litwin, he was an inspiring mentor and role model. His textbook *Elementary Computability, Formal Languages, and Automata* (1982) is a masterpiece in clarity, supplied with many brilliant exercises.

All those who knew Bob McNaughton will keep precious memories of him as a warm-hearted and generous person. The community of theoretical computer science will miss his pioneering leadership and masterful creativity.

John Corcoran, Paliath Narendran, Wolfgang Thomas

AUTOMATA COLUMN

MIKOŁAJ BOJAŃCZYK, University of Warsaw
bojan@mimuw.edu.pl

McNaughton's Theorem

In the logic and automata community, Robert McNaughton is perhaps best known for his determinisation result for automata on infinite words, as mentioned in the obituary. Below, I try to explain the significance of this result.

Automata can also be run on infinite words, i.e. words where positions are indexed by natural numbers. When automata are used as a tool for logics, such as temporal logics used in verification, infinite words are at least as important as finite words.

For infinite words, it is not immediately clear what the acceptance condition should be, since the automaton runs forever. In a seminal paper, J. R. Büchi proposed what is now called the Büchi condition: a run of an automaton is considered accepting if accepting states are visited infinitely often. He proved that nondeterministic automata with this acceptance condition have the same expressive power as monadic second-order logic (MSO). The key lemma in Büchi's proof says that the class of languages recognised by nondeterministic Büchi automata is closed under complementation. To prove this lemma, Büchi did not use determinisation – instead, he showed how to directly convert a nondeterministic automaton into another nondeterministic automaton recognising the complement language. In fact, determinisation fails for Büchi automata, e.g. the language "infinite words over the alphabet $\{a,b\}$ with finitely many occurrences of the letter b" is recognised by a nondeterministic Büchi automaton, but not by any deterministic Büchi automaton.

After Büchi's paper, it was known that, over infinite words, nondeterministic automata (with the Büchi acceptance condition) are equivalent to MSO, and both are equivalent to ω-regular expressions (a natural notion of regular expressions for infinite words). This left open the problem: is there also a deterministic model of automata for infinite words?

This problem was solved by McNaughton in 1966, in a paper called "Testing and Generating Infinite Sequences by a Finite Automaton". This paper shows that every ω-regular expression can be converted into a deterministic automaton, which easily implies that nondeterministic Büchi automata can be converted into deterministic automata. As mentioned before, the Büchi condition is not sufficient for deterministic automata, so McNaughton used deterministic automata with a more general accepting condition, which is currently called the Muller condition. To specify the Muller condition, one gives a family of subsets of states (as opposed to the single set of states used in the Büchi condition). A run is considered accepting if the set of states that it visits infinitely often belongs to the family. A typical Muller condition would be: "either states p and q appear infinitely often but not state r, or states p and r appear infinitely often but not state q". The condition was proposed by D. E. Muller in 1963, in a paper on determinisation which unfortunately contained a flaw.

McNaughton's result was a true breakthrough. Apart from answering a very natural and deep problem, the determinisation result became a crucial element of later fundamental results on automata and logic, like Rabin's theorem MSO on infinite trees, or Büchi and Landweber's solution to Church's synthesis problem. There is now a rich bibliography on determinisation, with many alternative approaches, of which probably the best known is the Safra construction. After almost fifty years, the topic is still a lively research area, and new papers on determinisation appear every year (e.g. at least four papers in the last two years). Nevertheless, even with the simplest modern proofs, determinisation remains difficult, and there are no proofs which are substantially shorter than McNaughton's original 4-page proof.

Complexity Column

NEIL IMMERMAN, University of Massachusetts Amherst
immerman@cs.umass.edu

In the 1980's Eric Lander, Jin-Yi Cai, Martin Fürer and I began to explore the power of fixed point logic with counting (FPC). This natural class is deeply entwined with the challenge of capturing the class of polynomial-time graph properties and determining the complexity of graph isomorphism — problems that remain open. Despite that, so much more is now known about the remarkable power of FPC. Anuj Dawar explains in the following lovely survey.

The Nature and Power of Fixed-Point Logic with Counting

Anuj Dawar, University of Cambridge

In 1982, Neil Immerman proposed an extension of fixed-point logic by means of counting quantifiers (which we denote FPC) as a logic that might express all polynomial-time properties of unordered graphs. It was eventually proved (by Cai, Fürer and Immerman) that there are polynomial-time graph properties that are not expressible in FPC. Nonetheless, FPC is a powerful and natural fragment of the complexity class PTime. In this article, I justify this claim by reviewing three recent positive results that demonstrate the expressive power and robustness of this logic.

1. INTRODUCTION

Neil Immerman showed in [Immerman 1982] that fixed-point logic—FP, a formalism extending first-order logic with a mechanism for defining predicates recursively—could define exactly the polynomial-time decidable properties of finite structures provided that the formulas had access to a linear order on the elements of the structures (a result established independently by Vardi [1982]). In his paper, Immerman asks the question whether the requirement of having a linear order could be replaced by allowing instead the logic to express the cardinalities of definable sets. This leads to the definition of *Fixed-Point Logic with Counting* (FPC), the extension of first-order logic with a mechanism for iteration and a mechanism for counting. Immerman's question was motivated by two considerations. On the one hand, the requirement for a linear order imposes an extraneous condition on the structures allowing us to express properties that are not really properties of the underlying structure at all, and one would like to avoid this. On the other hand, the only known examples of structural properties that could not be expressed in FP were cardinality properties, which could be expressed once a mechanism for counting is included.

It was not long after the question was raised that it was shown, by Cai et al. [1992] that there are, indeed, polynomial-time properties of graphs that cannot be expressed in FPC. The question of whether there is a natural logic in which one can express all and only the polynomial-time (PTime) decidable properties of finite structures remains open. There have been several proposals of logics that properly extend the expressive power of FPC while remaining within PTime. These include logics with choice operators [Gire and Hoang 1998], choiceless machines [Blass et al. 1999] and logics with matrix rank operators [Dawar et al. 2009]. However, none of these has been as extensively studied as FPC (see, for instance, the monograph by Otto [1997] for an extensive discussion of the logic). FPC, though no longer a candidate for capturing PTime, seems to capture an intriguing class of problems within PTime worthy of study in its own right. That makes the logic an important focus of continuing investigation.

In this article, I take a look at three recent results about the expressive power of FPC that support the case that the FPC-definable properties form a natural and powerful fragment of PTime. First of these is the monumental result of Grohe that shows that on any class $\mathcal{C}$ of structures that excludes some graph minor (precise definitions of these notions are given in Section 3) FPC captures all of PTime. This is the culmination of a decade of work exploring the expressive power of FPC on restricted classes of structures using the methods of structural graph theory. The second result (explored in Section 4) is one that establishes the surprising power of FPC to express linear-programming problems and uses this to settle a fifteen-year-old conjecture of Blass, Gurevich and Shelah. The third result, reviewed in Section 5, characterizes the expressive power of FPC in terms of families of circuits restricted by a natural symmetry condition. This frames the class of problems definable in FPC in the context of circuit complexity and reinforces the view that this is a natural and robust class. In this article, the aim, of course, is not to prove these results but to explain the context in which they are proved, the nature of their contribution and explore some of the ideas involved in the proofs. But, first of all, we begin by reviewing the definitions and some background of the logic FPC.

2. FIXED-POINT LOGIC WITH COUNTING

Fix a relational vocabulary σ, and let $\mathrm{FO}[\sigma]$ denote first-order logic over this vocabulary. By this we may mean the collection of first-order σ-formulas or we may also mean the collection of σ-classes of finite structures that are definable in first-order logic. In general, we may drop the mention of the vocabulary σ if it is implicit and just write FO. FP denotes the extension of FO with a fixed-point operator. Fixed-point operators come in different varieties, such as the least-fixed-point operator and the the inflationary-fixed-point operator. For our purposes here, it is most convenient to just consider inflationary fixed-points. For details of the different logics one can obtain with such fixed-point operators and their expressive power, the interested reader may consult [Ebbinghaus and Flum 1999]. By Immerman [1986] and Vardi [1982], we know that on ordered structures, FP expresses exactly those properties that are decidable in polynomial-time. However, in the absence of order, simple counting properties, such as saying that the number of elements in a structure is even, are not definable.

The Logic. Fixed-point logic with counting (FPC) extends FP with the ability to express the cardinality of definable sets. The logic has two sorts of variables: $x_1, x_2, \ldots$ ranging over the domain elements of the structure, and $\nu_1, \nu_2, \ldots$ ranging over the non-negative integers. If we allow unrestricted quantification over non-negative integers, the logic would be powerful enough to express undecidable properties. In Immerman's original proposal, number variables were restricted to taking values in the set $\{0, \ldots, n\}$ where n is the number of elements in the domain of the structure of interpretation. In the formal definition below, we adopt another convention (suggested by Grohe) of requiring quantification of number variables to be bounded by a term. In addition, we also have second order variables $X_1, X_2, \ldots$, each of which has a type which is a finite string in $\{\mathrm{element}, \mathrm{number}\}^*$. Thus, if X is a variable of type $(\mathrm{element}, \mathrm{number})$, it is to be interpreted by a binary relation relating elements to numbers. The logic allows us to build up *counting terms* according to the following rule:

If ϕ is a formula and x is a variable, then $\#x\phi$ is a term.

The intended semantics is that $\#x\phi$ denotes the number (i.e. the non-negative integer) of elements that satisfy the formula ϕ. The formulas of FPC are now described by the following set of rules:

— all atomic formulas of first-order logic are formulas of FPC;
— if τ_1 and τ_2 are counting terms (that is each one is either a number variable or a term of the form $\#x\phi$) then each of $\tau_1 < \tau_2$ and $\tau_1 = \tau_2$ is a formula;
— if ϕ and ψ are formulas then so are $\phi \wedge \psi$, $\phi \vee \psi$ and $\neg\psi$;

— if ϕ is a formula, x is an element variable, ν is a number variable, and η is a counting term, then $\exists x\,\phi$ and $\exists \nu \leq \eta\,\phi$ are formulas; and

— if X is a relation symbol of type α, $\mathbf{z}$ is a tuple of variables whose sorts match the type α and $\mathbf{t}$ is a tuple of terms of type α, then $[\mathbf{fp}_{X,\mathbf{z}}\phi](\mathbf{t})$ is a formula.
The intended semantics here is that the tuple of elements denoted by $\mathbf{t}$ is in the inflationary fixed-point of the operator defined by ϕ, binding the variables in $X, \mathbf{z}$.

For details of the semantics and a lot more about the logic I refer the reader to Otto's excellent monograph [Otto 1997]. Here, I illustrate the use of this logic with some examples. In what follows, we use $\mathbf{A}$ and $\mathbf{B}$ to denote finite structures over a vocabulary σ and A and B to denote their respective universes.

A standard example of the power of inductive definitions is that we can use them to say a graph is connected, a property that is not expressible in FO.

Example 2.1. The following formula:

$$\forall u \forall v[\mathbf{fp}_{T,xy}(x = y \vee \exists z(E(x,z) \wedge T(z,y)))](u,v)$$

is satisfied in a graph (V, E) if, and only if, the graph is connected. Indeed, the least relation T that satisfies the equivalence $T(x,y) \equiv x = y \vee \exists z(E(x,z) \wedge T(z,y)))$ is the reflexive and transitive closure of E. This is, therefore the fixed point defined by the operator **fp**. The formula can now be read as saying that every pair u, v is in this reflexive-transitive closure.

The standard example of a property not definable in FP is to say that the number of elements in a structure is even.

Example 2.2. The following sentence is satisfied in a structure $\mathbf{A}$ if, and only if, the number of elements of $\mathbf{A}$ that satisfy the formula $\phi(x)$ is even.

$$\exists \nu_1 \leq [\#x\phi] \exists \nu_2 \leq \nu_1(\nu_1 = [\#x\phi] \wedge (\nu_2 + \nu_2 = \nu_1))$$

In particular, taking ϕ to be a universally true formula such as $x = x$, we get a sentence that defines evenness. Here we have used the addition symbol in the subformula $\nu_2 + \nu_2 = \nu_1$. It should be noted that this denotes a relation that is easily definable by induction on the domain of numbers.

Besides using formulas of FPC to define classes of structures, we also use it to define a new structure from a given one. An FPC-*interpretation* of a vocabulary τ in a vocabulary σ is a sequence of σ-formulas, including a formula θ_R for each symbol R of τ which, when interpreted in a σ-structure $\mathbf{A}$, yield a definition of a τ-structure $\Phi\mathbf{A}$. More precisely, the universe U of $\Phi\mathbf{A}$ is the set of tuples of elements (which may be tuples involving numbers as well as elements of $\mathbf{A}$) satisfying some particular formula θ_U, and the interpretation of each τ symbol R is given by the set of those tuples in U satisfying θ_R. We omit some technical details here and the interested reader may consult [Immerman 1999] where an interpretation is also called a *query* or [Grohe 2014, Chapter 2], where it is called a *transduction*.

Finite Variable Logics. A key tool in analysing the expressive power of FPC is to look at equivalence in a weaker logic—first-order logic with counting quantifiers and a bounded number of variables. For each natural number i, we have a quantifier $\exists^i$ where $\mathbf{A} \models \exists^i x\,\phi$ if, and only if, there are at least i distinct elements $a \in A$ such that $\mathbf{A} \models \phi[a/x]$. While the extension of first-order logic with counting quantifiers is no more expressive than FO itself (in contrast to the situation with counting terms), the presence of these quantifiers does affect the number of variables that are necessary to express a query. Let C^k denote the k-variable fragment of first-order logic with counting quantifiers. That is, C^k consists of those formulas in which no more than k variables appear, free or bound. For two structures $\mathbf{A}$ and $\mathbf{B}$, we write $\mathbf{A} \equiv^{C^k} \mathbf{B}$ to denote that the two structures are not distinguished by any sentence of C^k. The link between this and FPC is the following fact, established by Immerman and Lander [1990]:

THEOREM 2.3. *For every sentence ϕ of* FPC*, there is a k such that if $\mathbf{A} \equiv^k \mathbf{B}$, then $\mathbf{A} \models \phi$ if, and only if, $\mathbf{B} \models \phi$.*

Indeed, this theorem follows from the fact that for any formula ϕ of FPC, there is a k so that on structures with at most n elements, ϕ is equivalent to a formula θ_n of C^k Additionally, it can be shown that the quantifier depth of θ_n is bounded by a polynomial function of n, but the important bound for us is that the number of variables k is bounded by a constant that only depends on ϕ.

The equivalence relations $\equiv^{C^k}$ have many different characterisations, some of which arose in contexts removed from the connection with logic, such as the Weisfeiler-Lehman family of equivalences arising in the study of graph isomorphism (see the discussion in [Cai et al. 1992] for the connection). Particularly interesting are the characterisations of $\equiv^{C^k}$ in terms of two-player games, including the counting game of Immerman and Lander [1990] and the bijection game of Hella [1992]. These provide a means of arguing that two structures $\mathbf{A}$ and $\mathbf{B}$ are $\equiv^{C^k}$-equivalent. By Theorem 2.3, this can then be used to show that some property is not definable in FPC, by showing that it is not closed under $\equiv^{C^k}$ for any fixed k.

Inexpressibility Results. The suggestion that FPC might be sufficient to express all properties in PTime arose from the intuition that it addresses the two obvious shortcomings of FO by providing a means to express inductive definitions and a means of counting. In a sense, all algorithms that are "obviously" polynomial-time can be translated into the logic. Nonetheless, an ingenious construction by Cai et al. [1992] uses games to give a polynomial-time decidable class of graphs that is not definable in FPC. More precisely, they show how to construct a sequence of pairs of graphs (G_k, H_k), one for each $k \in \omega$ such that:

— for each k, $G_k \equiv^{C^k} H_k$;
— for each k, $G_k \not\cong H_k$, i.e. the graphs are not isomorphic; and
— G_k and H_k have maximum degree 3.

It is an immediate consequence of these facts that the problem of graph isomorphism for graphs of degree 3 is not expressible in FPC. However, this problem is known to be in PTime by a result of Luks [1982].

A number of other conclusions on the limitations of the expressive power of FPC can be drawn from the result of Cai et al. Such results typically fall into one of two classes. In the first category are non-expressiblity results that follow from the result of Cai et al. by means of reductions. For instance, there is no FPC sentence that defines the graphs with a Hamiltonian cycle. This is because, by a result of Dahlhaus [1984], this problem is NP-complete under first-order reductions, and FPC is closed under such reductions. Hence, as long as we have some problem in NP that is not definable in FPC, it follows that Hamiltonicity is not definable. By an older result of Lovász and Gács [1977] we also know that satisfiability of Boolean formulas in CNF (suitably encoded as a relational structure) is NP-complete under first-order reductions and therefore this is also not definable in FPC. Interestingly, it remains an open question whether 3-SAT is NP-complete under first-order reductions.

In the second category of non-expressibility results that follow on from Cai et al. are those proved by adapting their methods. For instance, it is known that graph 3-colourability is *not* NP-complete under first-order reductions (indeed, the class of problems that reduce to it obeys a 0-1-law, see [Dawar and Grädel 2010]). Still, it has been shown [Dawar 1998] that 3-colourability is not definable in FPC by a construction of graphs adapting that of Cai et al. More recently, Atserias et al. [2009] show that the problem of determining whether a system of linear equations (modulo 2) is solvable cannot be expressed in FPC, though this is a problem in PTime. By means of first-order reductions, they then show that this implies that a host of constraint satisfaction problems (characterised by algebraic properties) are not definable in FPC, including 3-colourability and 3-SAT.

It should be noted that in all cases mentioned, the problem is shown to be not definable in FPC by establishing the stronger statement that the problem is not invariant under $\equiv^{C^k}$ for any fixed k. Of course, there are problems that are invariant under $\equiv^{C^k}$ for some k but still not definable in FPC, for instance, problems that are not in PTime at all. Could it be that every problem in PTime that is invariant under $\equiv^{C^k}$ for some k is also in FPC? This remains an open question and I refer the interested reader to Otto [1997] for a thorough discussion.

Capturing Results. Alongside results establishing the limits of the expressive power of FPC, there has been a series of results showing that FPC is powerful enough to express all polynomial-time decidable properties, provided that we restrict the class of structures in some way. Of course, we have already noted above that FPC (even without the counting extension) suffices to capture PTime on ordered structures. One important line of investigation has sought to examine classes of structures based on *sparse graphs*. An early result in this vein was due to Immerman and Lander, who showed that FPC captures PTime on trees. This was generalized in two distinct directions by Grohe [1998], who showed that FPC capture PTime on planar graphs and Grohe and Mariño [1999] who show that FPC captures PTime on graphs of bounded treewidth. These results were the beginning of a long progression which leads to Grohe's theorem establishing that FPC captures PTime on any class of structures whose adjacency graphs form a proper minor-closed class. This generalizes the prevous mentioned results and that is the subject of the next section.

3. MINOR-CLOSED CLASSES

With any σ-structure $\mathbf{A}$, we associate a (loop-free, undirected) graph $\Gamma\mathbf{A}$, which we call the *adjaceny graph* of $\mathbf{A}$ (also known as the Gaifman graph of $\mathbf{A}$). This is the graph with vertex set A and in which two vertices u and v are adjacent if there is some relation R in σ and some tuple $\mathbf{t} \in R^{\mathbf{A}}$ such that both u and v occur in $\mathbf{t}$. A very useful methodology for studying well-behaved classes of finite structures is to restrict attention to structures with adjacency graphs from some restricted graph class. This has enabled the deployment of techniques from structural graph theory in the study of finite model theory (see [Dawar 2007] for a short survey). In particular, well-behaved classes of finite structures can be defined using the graph minor relation. In the rest of this section, I talk of graphs and classes of graphs. All the results about definability and capturing PTime that are stated for a class of graphs $\mathcal{C}$ apply equally well to a class of relational structures all of whose adjacency graphs are in $\mathcal{C}$.

Graph Minors. We say that a graph H is a *minor* of a graph G and write $H \preceq G$, if H can be obtained from G by means of repeated applications of the operations of *(i)* deleting an edge; *(ii)* deleting a vertex, and all edges incident on it; and *(iii)* contracting an edge. Here, the last operation is one where we remove an edge (u, v) and replace the two vertices u and v by a new vertex whose neighbours are all neighbours of u and v (other than u and v themselves). A more formal definition is that $H = (U, F)$ is a minor of $G = (V, E)$ if there is a set $U' \subseteq V$ and a surjective map $M : U' \to U$ such that

— for each $u \in U$, $M^{-1}(u)$ induces a connected subgraph of G; and

— for each edge $(u, v) \in F$, there is an edge in E between some $x \in M^{-1}(u)$ and some $y \in M^{-1}(v)$.

Intuitively, H is a minor of G if the "structure" of H can be found inside G. As a simple example, if H is a triangle (denoted K_3), then H is a minor of G if, and only if, G contains a cycle.

Say that a class $\mathcal{C}$ of graphs excludes H as a minor if H is not a minor of any graph in $\mathcal{C}$. Thus, the class of graphs that exclude K_3 as a minor is exactly the class of acyclic graphs. Say $\mathcal{C}$ is *minor-closed* if any minor of any graph in $\mathcal{C}$ is also in $\mathcal{C}$. It is a proper minor-closed class if it is *minor-closed* and is not the class of all graphs. Clearly, any proper minor-closed class of graphs excludes some graph as a minor. Indeed it is characterised by the set of minor-minimal graphs that it excludes. Also, any graph class that excludes some graph H as a minor is included in a proper minor-closed class. In particular, it is included in the class of all graphs that do not have H as a minor, which is clearly minor-closed, by transitivity of the minor relation.

A famous theorem of Wagner [1937] (building on earlier work by Kuratowski [1930]) states that a graph G is planar if, and only if, neither K_5 (the clique on 5 vertices) nor $K_{3,3}$ (the complete bipartite graph on two sets of three vertices) is a minor of G. Thus, planar graphs are a proper minor-closed class that is characterised by two *forbidden minors*. The theory of graph minors was developed through a series of papers by Robertson and Seymour culminating in the proof of the graph minor theorem:

THEOREM 3.1 ([ROBERTSON AND SEYMOUR 2004]). *In any infinite collection $\{G_i \mid i \in \omega\}$ of graphs, there are i, j with $G_i \preceq G_j$.*

In other words, there are no infinite anti-chains in the graph minor relation. Since the set of minor-minimal elements that are excluded from some proper minor-closed class $\mathcal{C}$ form an anti-chain, an immediate corollary to the theorem is that any such class is characterised by a finite set of forbidden minors.

COROLLARY 3.2. *For any minor-closed class $\mathcal{C}$, there is a finite collection $\mathcal{F}$ of graphs such that $G \in \mathcal{C}$* if, and only if, *$F \not\preceq G$ for all $F \in \mathcal{F}$.*

This corollary has important algorithmic consequences. In particular, since Robertson and Seymour also show [Robertson and Seymour 1995] that for every H, there is a cubic time algorithm that decides if a given G has H as a minor, it follows that every proper minor-closed class is decidable by a cubic time algorithm. Indeed, if $\mathcal{C}$ is such a class and $\mathcal{F}$ the finite collection of forbidden minors that characterize it, an algorithm for testing membership in $\mathcal{C}$ is obtained by taking a graph G and checking for each $H \in \mathcal{F}$ that H is not a minor of G.

Capturing PTime. We can now state the theorem of Grohe.

THEOREM 3.3 (GROHE [GROHE 2014]). FPC *captures* PTime *on any proper minor-closed class $\mathcal{C}$.*

Note that, since any class $\mathcal{C}$ that excludes a minor is included in some proper-minor closed class, it follows that FPC captures PTime on any class of structures whose adjacency graphs exclude some minor.

The full proof of Theorem 3.3 has not yet been published but appears in a monograph of more than 400 pages that is available from the author's website [Grohe 2014]. The statement of the result, and a proof of the important special case of classes of graphs $\mathcal{C}$ embeddable in a fixed surface, have been published in [Grohe 2012]. It is clearly not possible, in the confines of this column, to even sketch the proof of Theorem 3.3, but I will try to highlight some of the important ideas.

Say that a class $\mathcal{C}$ of σ-structures admits FPC *canonization* if there is an FPC interpretation that maps each structure $\mathbf{A}$ in $\mathcal{C}$ to a $\sigma \uplus \{\leq\}$-structure $\mathbf{A}^{\leq}$ whose σ-reduct is isomorphic to $\mathbf{A}$ and on which $\leq$ is interpreted as a linear order of the universe. In other words, the canonization allows us to define an ordered version of each structure in $\mathcal{C}$ from the structure itself. In particular, by the invariance properties of FPC, a canonization has the property that two structures are isomorphic if, and only if, the canonization takes them to isomorphic *ordered* structures. Checking isomorphism of ordered structures is trivial, so an FPC canonization on a class $\mathcal{C}$ also gives us an FPC-definable isomorphism test on $\mathcal{C}$. Since FPC captures PTime on ordered structures, it is not difficult to see that for any class $\mathcal{C}$ that admits FPC canonization, FPC captures PTime on $\mathcal{C}$. Grohe shows that for any graph H, there is an FPC interpretation that is a canonization on the class of graphs that omit H as a minor.

In order to prove the result, Grohe develops a version of the structure theory that Robertson and Seymour developed in the proof of the graph minor theorem, but in a version that respects definability. An essential ingredient of Robertson and Seymour's proof of Theorem 3.1 is known as the structure theorem [Robertson and Seymour 2003]. Introducing all the elements required for a formal statement of the theorem would take us too far afield, so we will be content with an informal statement. The structure theorem essentially says that for each H, there is a k such that a graph G that excludes H as a minor admits a tree decomposition in which each bag is *almost embeddable* in a surface of genus k. The difficulty in using this decomposition theorem to derive an FPC canonization for graphs excluding H as a minor is that the particular tree decomposition constructed by Robertson and Seymour is not *generic*. That is to say, the construction relies on a particular presentation of the graph, is not invariant under automorphisms of the graph and so cannot be defined by formulas that necessarily respect such automorphisms. Indeed, it is not clear that any generic form of such a tree decomposition can be defined.

To get around this difficulty, Grohe defines the notion of a *treelike decomposition* of a graph G. This is not a tree decomposition at all. Rather, it is a directed acyclic graph decorated with *bags* that are sets of vertices of G. The definition includes technical conditions on connectivity and consistency of these bags that ensure that it includes within it a tree decomposition of G. Indeed, a treelike decomposition of G can be obtained from a tree decomposition by taking the images of all bags under automorphisms of G. An essential step in Grohe's construction is a definable version of Robertson and Seymour's structure theorem, showing that there is an FPC interpretation which, on graphs that exclude H as a minor, defines a treelike decomposition into bags that are almost embeddable in a fixed surface.

The definable structure theorem is arrived at by a series of steps. First, Grohe shows that there is an FPC-definable decomposition of planar graphs into their 3-connected components. This is lifted to graphs embeddable in an arbitrary surface, by an inductive argument on the surface. More heavy lifting is required, based on elements of Robertson and Seymour's structure theory, to obtain from this a definable treelike decomposition. The final element is that from such a definable treelike decomposition, one can show that the class of graphs excluding H as a minor admits FPC canonization, leading to Theorem 3.3.

There are two significant consequences of this construction to note here. One is that every proper minor-closed class of graphs is itself definable by some FPC sentence. The other is that for any such class $\mathcal{C}$, there is a k such that $\equiv^{C^k}$ is the same as isomorphism on graphs in $\mathcal{C}$. In other words, the k-dimensional Weisfeiler-Lehman algorithm suffices to decide graph isomorphism on $\mathcal{C}$. Of course, the exact value of k depends on the finite set of excluded minors that characterise the class $\mathcal{C}$ and for many proper minor-closed classes, we do not know this set. But, in other cases, such as planar graphs, it would be interesting to pinpoint the precise value of k for which this happens.

4. LINEAR PROGRAMMING AND MATCHING

We noted above (see page 11) that the original intuition that led to the question of whether FPC captures PTime was that all problems that admit "obvious" polynomial-time algorithms are expressible in FPC. The last section showed a whole class of problems—the proper minor-closed classes of graphs—whose membership in PTime is far from obvious, but which nonetheless turn out to be definable. In this section, we turn to another fundamental algorithmic technique which is used in a rather non-obvious way to establish that some problems are in PTime.

Linear programming is a widely used approach to solving combinatorial optimization problems. It provides a powerful framework within which optimization problems can be represented, as well as efficient methods for solving the resulting programs. In particular, it is known since the work of [Khachiyan 1980] that there are polynomial-time algorithms that solve linear programs.

Linear Programs as Structures. In general, a linear programming instance consists of a set C of constraints over a set V of variables. Each constraint c consists of a vector $a_c \in \mathbb{Q}^V$ and a rational $b_c \in \mathbb{Q}$. The *feasibility problem* is to determine for such an instance if there exists a vector $x \in \mathbb{Q}^V$ such that $a_c^T x \leq b_c$ for all $c \in C$. An instance of the *optimization problem* is obtained if we have, in addition, an objective function represented by a vector $f \in \mathbb{Q}^V$. The aim is then to find an $x \in \mathbb{Q}^V$ which satisfies all constraints in C and maximizes the value of $f^T x$.

We want to consider instances of linear programming as relational structures that may act as interpretations for formulas of FPC. For this, we consider structures whose universe consists of three disjoint sets: V, C and B. The last of these is equipped with a linear order and may be thought of as $\{0, \dots, t\}$, i.e. an initial segment of the natural numbers where t is large enough that all the rational numbers required to represent our instance can be written down using at most t bits for both numerator and denominator. We can then encode the linear programming instance through suitable relations for the numerators, denominators and signs of the values involved. For instance, we have a ternary relation N so that for $c \in C$, $v \in V$ and $b \in B$, $N(c, v, b)$ holds if the bth bit in the numerator of $a_c(v)$ is 1. It should be stressed that while the set B is ordered, there is no order on the sets V and C. If there were, then feasibility of such linear programming instances would be definable in FPC simply by the fact that all polynomial-time decidable properties of ordered structures are definable.

Ellipsoid Method. In [Anderson et al. 2013], it is shown that the feasibility problem and, indeed, the optimization problem, for linear programming, is definable in FPC. The proof proceeds by means of expressing in the logic a version of Khachiyan's ellipsoid method [Khachiyan 1980]. In short, the ellipsoid method for determining the feasibility of a linear program proceeds by choosing a vector $x \in \mathbb{Q}^V$ (one may as well begin from the vector of all zeroes) and calculating, based on the bit complexity of the instance, an ellipsoid around x which is guaranteed to include the polytope defined by the constraints C. Now, if x itself does not satisfy all the constraints in C, we can choose one $c \in C$ for which $a_c^T x > b_c$ and use it to calculate a new vector x' and an ellipsoid centred on x' which still includes the polytope defined by C. The construction guarantees that the volume of the new ellipsoid is at most half that of the original one. This means that in a number of steps that is bounded by a polynomial in the size of the instance, the process converges to a centre x that satisfies all the constraints in C *or* the volume of the ellipsoid is small enough that we know for certain that the polytope is empty.

It is not difficult to show that all the calculations involved in computing the ellipsoid and centres can be expressed in FPC (see [Holm 2010] for details on how linear algebra using matrices without an order on the rows and columns can be expressed). The one step in the algorithm outlined above that causes difficulty is the *choice* of a violated constraint, which is used to define a new centre. There is no way, in any logical formalism that respects automorphisms of the structure on which it is interpreted (as any reasonable logic must) to choose an arbitrary element. However, the ellipsoid method is quite robust and it still works as long as, at each stage, we can construct some hyperplane that separates our current centre x from all points in the polytope defined by C. And, as shown in [Anderson et al. 2013], such a construction can be done canonically by taking the set of all constraints in C that are violated by x and taking their sum as our separating hyperplane. We thus get that feasibility of linear programs (and also the linear programming optimization problem) are definable in FPC.

Separation Oracles. So far, we have considered linear programs represented *explicitly*. That is, all the constraints are written out as part of the input structure. Relying on the robustness of the ellipsoid method, we can take this method further. In many applications of linear programming, we are not given an explicit constraint matrix, but some succinct description from which explicit constraints can be derived. In particular, the number of constraints may be exponential in the size of the input. The ellipsoid method can be used in such cases as long as we have a means of determining, for any vector x, whether it is in the polytope P described by the constraint matrix and, if it is not, a hyperplane that separates x from the polytope. This is known as a *separation oracle* for P. It is shown in [Anderson et al. 2013] that, as long as a separation oracle for a polytope P is itself definable in FPC, then the corresponding linear programming optimization problem can also be defined in FPC.

This reduction of optimization to separation reveals an interesting relationship of linear programming with the equivalence relations $\equiv^{C^k}$. We are given a polytope $P \subseteq \mathbb{Q}^V$ in the form of some relational structure $\mathbf{A}$ and an FPC interpretation Φ which acts as a separation oracle. That is, given an $x \in \mathbb{Q}^V$ (and we assume that V is part of the universe of $\mathbf{A}$, though the constraint set C is not and may, in general, be exponentially larger than $\mathbf{A}$), Φ interpreted in a suitable expansion of $\mathbf{A}$ with x either determines that x is in the polytope P or defines a hyperplane separating x from P. Then, there is some k (the exact value will depend on Φ) so that we can take the quotient $V' = V/_{\equiv^{C^k}}$ (where $\equiv^{C^k}$ is defined with respect to the structure $\mathbf{A}$) and project P to a polytope $P' \subseteq \mathbb{Q}^{V'}$ in such a way that feasibility and optimization with respect to P can be reduced to similar questions about P'. This should be compared with [Grohe et al. 2014] where it is shown, essentially, that in the special case of explictly represented constraint matrices, taking $k = 2$ gives a similar result. This is then used as a preprocessing tool for optimizing the performance of linear programming solvers.

Maximum Matching. An important application given in [Anderson et al. 2013] of the FPC definability of linear programming is to the *maximum matching problem*. Recall that a matching in a graph G is a set of edges M so that each vertex in G is incident on at most one edge in M. The maximum matching problem is then to find a matching in G of maximum size. Note that it would not be possible, in FPC or any other logic, to give a formula that would actually define the set M that is a maximum matching. This is because M would not, in general, be invariant under automorphisms of G. For instance, if G is K_n—the n-vertex complete graph—then it contains an exponential (in n) number of distinct maximum matchings all of which map to each other under automorphisms of G. However, it is shown in [Anderson et al. 2013] that the *size* of a maximum matching in G can be defined in FPC. It is known, thanks to Edmonds [1965] that we can associate with a graph G a *matching polytope* with an exponential number of constraints (and Rothvoß [2014] has shown that this is essentially optimal) so that optimizing over this polytope yields a maximum matching. Anderson et al. [2013] show that a separation oracle for the matching polytope can be defined in FPC interpreted on the graph G and hence deduce that the size of the maximum matching is definable. This settles a fifteen-year-old question posed by Blass et al. [1999] who conjectured that the existence of a perfect matching in G was not definable even in the stronger formalism of Choiceless Polynomial Time with Counting.

5. SYMMETRIC CIRCUITS

Let us now turn to the relationship between definability in FPC and circuit complexity. We start with a brief introduction to the formalism of circuit complexity.

A language $L \subseteq \{0,1\}^*$ can be described by a family of *Boolean functions*:

$$(f_n)_{n\in\omega} : \{0,1\}^n \to \{0,1\}.$$

Each f_n can be represented by a *circuit* C_n which is a directed acyclic graph where we think of the vertices as gates suitably labeled by Boolean operators $\wedge, \vee, \neg$ for the internal gates and by inputs $x_1, \ldots, x_n$ for the gates without incoming edges. One gate is distinguished as determining the output. If there is a polynomial $p(n)$ bounding the size of C_n (i.e. the number of gates in C_n), then the language L is said to be in the complexity class $\mathrm{P/poly}$. If, in addition, the family of circuits is *uniform*, meaning that the function that takes n to C_n is itself computable in polynomial time, then L is in PTime. For the definition of either of these classes, it does not make a difference if we expand the class of gates that we can use in the circuit beyond the Boolean basis to include, for instance, *threshold* or *majority* gates. The presence of such gates can make a difference for more restricted circuit complexity classes, for instance when we limit the depth of the circuit to be bounded by a constant, but not when we allow arbitrary polynomial-size circuits. Also, in the circuit characterization of PTime, it does not make a difference if we replace the uniformity condition with a stronger requirement. Say, we might require that the function taking n to C_n is computable in $\mathrm{DLogTime}$.

Circuits on Graphs. We are interested in languages that represent properties of relational structures such as graphs. For simplicity, in what follows, let us restrict attention to directed graphs, i.e. structures in a vocabulary with one binary relation. A property of such graphs that is in PTime can be recognised by a family $(C_n)_{n\in\omega}$ of Boolean circuits of polynomial size and uniformity, as before, where now the inputs to C_n are labelled by the n^2 *potential edges* of an n-vertex graph, each taking a value of 0 or 1. Of course, given an n-vertex graph G, there are many ways that it can be mapped onto the inputs of the circuit C_n. So, to ensure that the family of circuits is really defining a property of graphs, we require it to be *invariant* under the choice of this mapping. That is, each input of C_n carries a label of the form (i,j) for $i,j \in [n]$ and we require the output to be unchanged under any permutation $\pi \in S_n$ acting on the inputs by the action $(i,j) \mapsto (\pi(i), \pi(j))$. It is clear that any property of graphs that is invariant under isomorphisms of graphs and is in PTime is decided by such a family of circuits.

From Logic to Circuits. It turns out that the properties of graphs that are definable in logics such as FPC are decided by circuits with a stronger invariance condition. Say that a circuit C_n is *symmetric* if any permutation $\pi \in S_n$ can be extended to an *automorphism* of C_n which takes each input (i,j) to $(\pi(i), \pi(j))$. It is clear that symmetric circuits are necessarily invariant and it is not difficult to come up with examples that show that the converse is not true.

It is also not difficult to show we can translate a formula of first-order logic, say, in the language of graphs into a family of symmetric circuits. Given a first-order sentence ϕ and a positive integer n, we define the circuit C_n^ϕ by taking as gates the set of pairs $(\psi, \mathbf{a})$ where ψ is a subformula of ϕ and $\mathbf{a}$ is a tuple of values from $[n]$, one for each free variable occurring in ψ. The input gates correspond to the atomic formulas and the output gate is the one labelled by ϕ. If the outermost connective in ψ is a Boolean operation, the gate $(\psi, \mathbf{a})$ is labelled by that operator and it is connected to the gates corresponding to subformulas in the obvious way. If ψ is $\exists x\theta$, then $(\psi, \mathbf{a})$ would be an OR gate with inputs from $(\theta, \mathbf{a}a)$ for all values $a \in [n]$ and similarly a universal formula attaches to a large AND gate. It is not difficult to see that the circuits constructed are symmetric as for any $\pi \in S_n$, the map that takes $(\psi, \mathbf{a})$ to $(\psi, \pi(\mathbf{a}))$ is an automorphism of the circuit. Moreover, the *depth* of the circuit is a function of ϕ, so a fixed formula yields a family of circuits of constant depth. On the other hand, we can take a formula ϕ of fixed-point logic, FP, and for each n, obtain a first-order formula θ_n with k (independent of n) variables and quantifier depth bounded by a polynomial in n so that ϕ and θ_n are equivalent on structures with at most n elements. Converting this into a circuit by the translation above yields a polynomial-size family of symmetric Boolean circuits (the size is bounded by $c \cdot n^k$ where c is the number of sub-formulas of ϕ) that is equivelent to ϕ. If we start with a sentence ϕ of FPC instead, we can use the translation to C^k (see Theorem 2.3) to obtain a family of symmetric circuits with threshold (or majority) gates. We need these additional gates to translate the counting quantifiers that appear in the C^k formula and Boolean gates will no longer suffice. Symmetric circuits have been studied for the repesentation of properties defined in logic under different names in the literature (see [Denenberg et al. 1986; Otto 1996]).

From Circuits to Logic. The recent paper [Anderson and Dawar 2014] establishes that the correspondence between decidability by a family of symmetric circuits and definability in logic is tight by giving translations in the other direction. In particular, the following is shown there.

THEOREM 5.1. *A class of graphs is accepted by a polynomially uniform symmetric family of Boolean circuits* if, and only if, *it is definable by an* FP *formula interpreted in* $G \uplus ([n], <)$.

Here, $G \uplus ([n], <)$ denotes a structure consisting of a graph G together with a disjoint set of elements $[n]$ of the same cardinality as the set of vertices of G. In this structure, we have the edge relation on the vertices of G and a linear order on the elements $[n]$ and no other relations. In particular, the vertices of G are not ordered and there is no relation that connects the vertices of G with the elements of $[n]$. One can write an FP formula for such structures that states that the number of vertices is even, since this amounts to saying that the length of the linear order $([n], <)$ is even. But, it is not difficult to show by a pebble game argument that no FP formula can say that the number of edges in the graph G is even, even in the presence of the linear order on the side. It follows from Theorem 5.1 that this simple *invariant* property of graphs is not decidable by any polynimially-uniform family of *symmetric* circuits. It also follows that allowing *threshold* or *counting* gates in symmetric circuits leads to a model that is more powerful than just Boolean circuits, in contrast to the situation of general polynomial-size circuits. The exact power of polynomially uniform families of symmetric circuits with threshold gates is determined by the following theorem from [Anderson and Dawar 2014]:

THEOREM 5.2. *A class of graphs is accepted by a polynomially uniform symmetric family of threshold circuits* if, and only if, *it is definable in* FPC.

A consequence of Theorems 5.2 and 5.1 is that any translation of *invariant* Boolean circuits into equivalent *symmetric* circuits, even allowing additional threshold gates in the latter, will necessarily involve a super-polynomial blow-up in the size of the circuits.

Theorem 5.2 gives a natural and purely circuit-based characterization of FPC definability, justifying the claim that this is a natural and robust class. It also shows that inexpressibility results for FPC can be understood as lower bound results against a natural class of circuits. The holy grail of circuit complexity is to prove lower bounds against the class of polynomial-size circuit families. So far, lower bounds have been proved by imposing various restrictions on the families, such as monotonicity (where a famous result of Razborov [1985] showed that no polynomial size family of *monotone* circuits can decide the clique problem) or bounded depth. We can now add symmetric circuits to the catalogue of circuit classes against which we are able to prove lower bounds. It is instructive to put this in the context of the expressibility and inexpressibility results for FPC mentioned previously. In particular, it follows that there is a polynomially uniform family of symmetric threshold circuits for deciding whether a graph has a perfect matching. But, there is no such family that decides whether a graph contains a Hamiltonian cycle. A natural circuit model that separates these two problems is certainly an intriguiing development.

Proof Idea. To conclude this section, I present some of the ideas underlying the proofs of Theorems 5.1 and 5.2. One direction is easy, by the argument given above about translating formulas into circuits. In the other direction, we want to use the symmetry of the circuit to derive an equivalent formula in the fixed-point logic. If C_n is a symmetric circuit taking n-vertex graphs as input, we can assume without loss of generality, that the automorphism group of C_n is exactly the symmetric group S_n acting in the natural way on its inputs. For a gate g in C_n, we say that a set $X \subseteq [n]$ *supports* g if for every $\pi \in S_n$ such that $\pi(x) = x$ for all $x \in X$, we also have $\pi(g) = g$. Now, when we consider the circuit families $(C_n)_{n\in\omega}$ that arise as translations of FPC formulas, we can note that there is a constant k such that all gates have a support of size at most k. This is because the gates are labelled by pairs (ψ, a) where a is a k-tuple of elements from $[n]$ and it is easily checked that the set of elements that appear in a is a support of the gate. The main technical lemma in the proof of Theomem 5.2 establishes that this is, in fact, necessary for all symmetric circuits: if $(C_n)_{n\in\omega}$ is a family of symmetric circuits of polynomial size then there is a k such that all gates in C_n have a support of at most k elements. Moreover, given a description of the circuit C_n, there is a polynomial-time algorithm that will determine a minimal size support of each gate. Now, given a graph G on n vertices and a bijection $\gamma : [n] \to V(G)$ that determines how this graph is mapped to the inputs of C_n, whether a gate g in C_n evaluates to TRUE is completely determined by γ restricted to the support of g. We can thus represent the set of all maps γ that make g true as a k-ary relation on G. These k-ary relations admit an inductive definition (by induction on the construction of the circuit C_n) which allows us to turn the circuit family into a formula of FP (for Boolean circuits) or FPC (if the circuit also has threshold gates). This relies on the fact that the map that takes n to C_n is polynomial-time computable and therefore definable in FP on ordered structures. We use the linear order available "on the side" in Theorem 5.1 for this. It is not necessary in the proof of Theorem 5.2 as it can be replaced by the use of numerical variables.

6. CONCLUSION

I noted in the introduction that the conjecture that FPC captures PTime was based on the intuition that algorithmic techniques that are obviously polynomial-time are all expressible in this logic. The result of Cai et al. and subsequent results on inexpressiblity in FPC essentially show that one important algorithmic technique, that of Gaussian elimination, is not expressible in the logic. Nonetheless, the recent results surveyed in this article show that many powerful and certainly non-obvious polynomial-time algorithmic techniques are indeed expressible in FPC. In particular, we have seen that FPC can express maximum matching in graphs, feasibility of linear programs and arbitrary minor-closed classes of graphs. For each of these, the fact that the problem is in polynomial-time was a major result of its day. Finally, the results of Section 5 show further that FPC is a natural and robustly defined class by giving a characterization of it by a natural and independently-motivated circuit model. Taken together, these justify the claim that FPC is a *natural* and *powerful* class of problems.

ACKNOWLEDGMENTS

This article is a write-up of a talk I first gave as part of the workshop on *Logic and Computational Complexity* at Vienna in July 2014, devoted to Neil Immerman's 60th birthday. I would like to thank the organizers of that workshop, and I would like to thank Neil Immerman for inviting me to write it up in this column.

REFERENCES

M. Anderson and A. Dawar. 2014. On Symmetric Circuits and Fixed-Point Logics. In *31st Intl. Symp. Theoretical Aspects of Computer Science (STACS 2014)*. 41–52.

M. Anderson, A. Dawar, and B. Holm. 2013. Maximum Matching and Linear Programming in Fixed-Point Logic with Counting. In *28th Annual ACM/IEEE Symp. Logic in Computer Science*. 173–182.

A. Atserias, A. Bulatov, and A. Dawar. 2009. Affine Systems of Equations and Counting Infinitary Logic. *Theoretical Computer Science* 410, 18 (2009), 1666–1683.

A. Blass, Y. Gurevich, and S. Shelah. 1999. Choiceless Polynomial Time. *Annals of Pure and Applied Logic* 100 (1999), 141–187.

J-Y. Cai, M. Fürer, and N. Immerman. 1992. An Optimal Lower Bound on the Number of Variables for Graph Identification. *Combinatorica* 12, 4 (1992), 389–410.

E. Dahlhaus. 1984. Reduction to NP–Complete Problems by Interpretation. In *LNCS 171*. Springer-Verlag, 357–365.

A. Dawar. 1998. A Restricted Second Order Logic for Finite Structures. *Information and Computation* 143 (1998), 154–174.

A. Dawar. 2007. Finite Model Theory on Tame Classes of Structures. In *MFCS (Lecture Notes in Computer Science)*, Vol. 4708. Springer, 2–12.

A. Dawar and E. Grädel. 2010. Properties of Almost All Graphs and Generalized Quantifiers. *Fundam. Inform.* 98, 4 (2010), 351–372.

A. Dawar, M. Grohe, B. Holm, and B. Laubner. 2009. Logics with Rank Operators. In *Proc. 24th IEEE Symp. on Logic in Computer Science*. 113–122.

L. Denenberg, Y. Gurevich, and S. Shelah. 1986. Definability by Constant-depth Polynomial-size Circuits. *Information and Control* 70 (1986), 216–240.

H-D. Ebbinghaus and J. Flum. 1999. *Finite Model Theory* (2nd ed.). Springer.

J. Edmonds. 1965. Maximum Matching and a Polyhedron with $0, 1$ Vertices. *J. Research National Bureau of Standards* 69 B (1965), 125–130.

F. Gire and H. Hoang. 1998. An Extension of Fixpoint Logic with a Symmetry-Based Choice Construct. *Information and Computation* 144 (1998), 40–65.

M. Grohe. 1998. Fixed-Point Logics on Planar Graphs. In *Proc. 13th IEEE Annual Symp. Logic in Computer Science*. 6–15.

M. Grohe. 2012. Fixed-point definability and polynomial time on graphs with excluded minors. *J. ACM* 59, 5 (2012), 27:1–27:64.

M. Grohe. 2014. Descriptive Complexity, Canonisation, and Definable Graph Structure Theory. (2014). http://www.automata.rwth-aachen.de/~grohe/cap/index.en Draft of Monograph.

M. Grohe, K. Kersting, M. Mladenov, and E. Selman. 2014. Dimension Reduction via Colour Refinement. In *Algorithms - ESA 2014 - 22nd Annual European Symposium*. 505–516.

M. Grohe and J. Mariño. 1999. Definability and Descriptive Complexity on Databases of Bounded Tree-Width. In *Proc. 7th International Conference on Database Theory (LNCS)*, Vol. 1540. Springer, 70–82.

L. Hella. 1992. Logical hierarchies in PTIME. In *Proc. 7th IEEE Symp. on Logic in Computer Science*. 360–368.

B. Holm. 2010. *Descriptive Complexity of Linear Algebra*. Ph.D. Dissertation. University of Cambridge.

N. Immerman. 1982. Relational Queries Computable in Polynomial Time (Extended Abstract). In *Proc. 14th Annual ACM Symp. Theory of Computing*. 147–152.

N. Immerman. 1986. Relational Queries computable in Polynomial Time. *Information and Control* 68 (1986), 86–104.

N. Immerman. 1999. *Descriptive Complexity*. Springer.

N. Immerman and E. S. Lander. 1990. Describing Graphs: A First-order Approach to Graph Canonization. In *Complexity Theory Retrospective*, A. Selman (Ed.). Springer-Verlag.

L. G. Khachiyan. 1980. Polynomial Algorithms in Linear Programming. *U. S. S. R. Comput. Math. and Math. Phys.* 20, 1 (1980), 53–72.

K. Kuratowski. 1930. Sur le Probléme des Courbes Gauches en Topologie. *Fundamenta Mathematicae* 15 (1930), 271283.

L. Lovász and P. Gács. 1977. Some Remarks on Generalized Spectra. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* 23 (1977), 27–144.

E. M. Luks. 1982. Isomorphism of Graphs of Bounded Valence Can be Tested in Polynomial Time. *J. Comput. System Sci.* 25 (1982), 42–65.

M. Otto. 1996. The Logic of Explicitly Presentation-Invariant Circuits. In *Computer Science Logic, 10th International Workshop, CSL '96, Annual Conference of the EACSL*. 369–384.

M. Otto. 1997. *Bounded Variable Logics and Counting — A Study in Finite Models*. Lecture Notes in Logic, Vol. 9. Springer-Verlag.

A. A. Razborov. 1985. Lower Bounds on the Monotone Complexity of some Boolean Functions. *Dokl. Akad. Nauk. SSSR* 281 (1985), 798–801.

N. Robertson and P. D. Seymour. 1995. Graph Minors. XIII. The Disjoint Paths Problem. *J. Comb. Theory, Ser. B* 63 (1995), 65–110.

N. Robertson and P. D. Seymour. 2003. Graph Minors. XVI. Excluding a non-planar graph. *J. Comb. Theory, Ser. B* 89 (2003), 43–76.

N. Robertson and P. D. Seymour. 2004. Graph Minors. XX. Wagner's conjecture. *J. Comb. Theory, Ser. B* 92 (2004), 325–357.

T. Rothvoß. 2014. The Matching Polytope has Exponential Extension Complexity. In *Symp. Theory of Computing, STOC 2014*. 263–272.

M. Y. Vardi. 1982. The Complexity of Relational Query Languages. In *Proc. of the 14th ACM Symp. on the Theory of Computing*. 137–146.

K. Wagner. 1937. Uber eine Eigenschaft der Ebenen Komplexe. *Math. Ann.* 114 (1937), 570–590.

SECURITY & PRIVACY COLUMN

MATTEO MAFFEI, CISPA, Saarland University
maffei@cs.uni-saarland.de

The technical column on Security and Privacy of the SIGLOG newsletter kicks off with two invited contributions.

— Cătălin Hriţcu (INRIA, Paris-Rocquencourt) reports on the recently organized “Joint EasyCrypt-F*-CryptoVerif School 2014”. These three state-of-the-art security verification tools constitute an example of how logic and program verification may bring crucial contributions in the security domain. The presented material is available online and is certainly of great interest for the readers of the SIGLOG newsletter.

— Véronique Cortier (Loria) overviews a topic that is of public interest and constitutes an extraordinary research opportunity for logic and program verification researchers, namely the security of electronic voting. Véronique’s work demonstrates the effectiveness of formal methods in rigorously reasoning about the security of cryptographic protocols in general, and electronic voting systems in particular. In this column, Véronique reviews the state of the art in this field, pointing out research challenges of particular interest for the SIGLOG community.

A special thanks to both authors for accepting the invitation and for their great contributions!

The Joint EasyCrypt-F*-CryptoVerif School 2014

Cătălin Hriţcu, Inria Paris-Rocquencourt

1. OVERVIEW

Formal security verification tools are slowly reaching maturity. The Joint EasyCrypt-F⋆-CryptoVerif School took place between 24 and 28 November 2014 in Paris and taught participants how to use three state-of-the-art security verification tools, as well as gave participants a glimpse of the formal foundations behind these tools. The school brought together over 80 participants from 12 countries, with backgrounds spanning security, cryptography, programming languages, and formal methods. The time of the school was roughly evenly split between lectures and hands-on exercise sessions.[1] For the exercise sessions participants installed and tried EasyCrypt, F⋆, and CryptoVerif on their own laptops, under the guidance of members from the developer teams of the tools. The school also contained three short invited talks: Hubert Comon on *Unconditional Soundness*, Véronique Cortier on *Type-Based Verification of Electronic Voting Protocols*, and Matteo Maffei on *Logical Foundations of Secure Resource Management in Protocol Implementations*.

2. EASYCRYPT

EasyCrypt is a toolset for reasoning about relational properties of probabilistic computations with adversarial code. Its main application is the construction and verification of game-based cryptographic proofs. EasyCrypt was used to prove the security of complex cryptographic constructions, including the Cramer-Shoup encryption scheme, the Merkle-Damgaard iterative hash function design, and the ZAEP encryption scheme. More recently, it has been used for proving the security of protocols based on garbled circuits, and for the proof of security for authenticated key-exchange protocols and derived proofs under weaker assumptions.

The EasyCrypt lectures were given by Gilles Barthe, François Dupressoir, Benjamin Grégoire, Benedikt Schmidt, Pierre-Yves Strub, who also jointly run the exercise-sessions. The lectures presented (a) the code-based game-playing approach to computer-aided cryptographic proofs and the connection to program verification; (b) the foundations of EasyCrypt: probabilistic relational Hoare logic and program transformations; (c) the ambient (classical higher-order) logic of EasyCrypt and the most widely-used tactics; (d) grouping related concepts and lemmas into theories and structuring proofs with sections; (e) using modules to achieve abstraction and support high-level reasoning steps; (f) high-level cryptographic proof principles; (g) EasyCrypt case studies; (h) verifying "real-world" security at the source-code level; (i) automated proofs and synthesis; and (j) perspectives for the future.

The EasyCrypt exercise sessions involved proving (a) security against chosen plaintext attacks (IND-CPA) for the Bellare and Rogaway 1993 and the Hashed ElGammal encryption schemes; (b) correctness of a stateful random generator that uses a pseudo-random function (PRF); and (c) the PRP (pseudo-random permutation)/PRF switching lemma.

[1] The materials, including slides and exercises, are available on the website of the school: https://wiki.inria.fr/prosecco/The_Joint_EasyCrypt-F*-CryptoVerif_School_2014

3. F*

F* is a new ML-like functional programming language designed with program verification in mind. It has a powerful refinement type-checker that discharges verification conditions using the Z3 SMT solver. F* has been successfully used to verify nearly 50,000 lines of code, ranging from cryptographic protocol implementations to web browser extensions, and from cloud-hosted web applications to key parts of the F* compiler itself. The newest version on F* erases to both F# and OCaml, on which it is based. It also compiles securely to JavaScript, enabling safe interoperability with arbitrary, untrusted JavaScript libraries.

The F* lectures were given by Antoine Delignat-Lavaud, Cédric Fournet, and Nikhil Swamy, and comprised (a) a high-level introduction into proving programs correct with F*; (b) an overview of how F* deals with state and other effects; (c) an overview of type-based verification at scale and miTLS, a verified reference implementation of TLS; and (d) a more in-depth illustration of how to use types for modular verification of cryptographic code.

The F* exercise sessions included (a) many basic examples: proving correctness of a simple model of access control and of simple recursive functions on numbers (factorial, fibonacci) and lists (mapping, finding, and sorting), and proving termination for Ackermann and tail recursive functions, (b) several cryptographic examples: cryptographic capabilities for accessing files, secure RPC using MACs, and verified encryption padding; and (c) a case-study on formalizing the metatheory of the simply-typed λ-calculus, illustrating the use of F* as a proof assistant. The exercise sessions were prepared and run jointly by: Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Cătălin Hriţcu, Chantal Keller, Markulf Kohlweiss, Aseem Rastogi, Pierre-Yves Strub, and Nikhil Swamy.

4. CRYPTOVERIF

CryptoVerif is an automatic protocol prover sound in the computational model of cryptography. It can prove secrecy and correspondences (e.g. authentication). The generated proofs are by sequences of games, as used by cryptographers. CryptoVerif was successfully used for security proofs of various cryptographic schemes and protocols, including Kerberos, OEKE, and the SSH transport layer protocol.

The CryptoVerif lectures and exercise sessions were run by Bruno Blanchet, the main designer and developer of CryptoVerif. The lectures covered (a) the process calculus used by CryptoVerif for expressing games; (b) representing security assumptions on primitives as indistinguishability properties; (c) syntactic transformations; and (d) generating protocol implementations from CryptoVerif specifications. Two illustrative examples were used during the lectures: encrypt-then-MAC and full domain hash (FDH) signatures. In the exercise sessions the students used CryptoVerif to prove (a) various security notions for encrypt-then-MAC (IND-CPA, IND-CCA2, INT-CTXT); (b) the security (IND-CPA) of the Bellare and Rogaway 1993 encryption scheme; and (c) authentication for the fixed version of the Woo-Lam shared-key protocol.

ACKNOWLEDGMENTS

The school was organized by Anna Bednarik, Karthikeyan Bhargavan, Cătălin Hriţcu, and Pierre-Yves Strub, together with a group of student volunteers: Evmorfia-Iro Bartzia, Benjamin Beurdouche, Antoine Delignat-Lavaud, and Jean Karim Zinzindohoué. This school was financially supported by the Prosecco team at Inria Paris-Rocquencourt with funds from the ERC Starting Grant CRYSP. The EPSRC CryptoForma network on formal methods and cryptography sponsored a number of grants for UK attendees. The MSR-Inria Joint Centre sponsored the social dinner. Cryptosense organized and sponsored the reception.

Formal verification of E-voting: solutions and challenges

Véronique Cortier, CNRS - Loria, France

In the last ten years, electronic voting has been used in an ever growing number of elections. There are many reasons for this development. First, some election modes require a mechanized way for counting since the number of questions or the number of choices is too large for a manual counting. Electronic voting also allows one to vote from home, possibly avoiding long travels. Sometimes, it simply follows the trend of using Internet in our daily life.

Of course, the use of electronic voting raises many questions: How to make sure my vote will be counted? Does my vote remain private? How to trust the final result? Is it possible to buy votes on the Internet? While some countries such as Estonia, France, or Norway use electronic voting for legally binding elections, some other countries have banned e-voting at least for some time. This is the case for example of the United Kingdom, Netherlands, or Germany [Barrat et al. 2012]. Electronic voting is used nevertheless and is likely to be used in most developed countries worldwide in non legally binding elections like the election of many representative committees (administration councils, scientific councils, etc.). Some of these elections may actually involve a large number of voters (more than a million) and may have important issues for example when it comes to elect the leader of a political party.

There is therefore an urgent need to offer secure, reliable, and trustworthy e-voting systems. The research community in logics and program verification has already a long tradition in developing analysis techniques for security protocols. E-voting protocols raise however new research challenges.

1. PROPERTIES

What is a good e-voting system? This a wide question and we focus here only on security properties the design of the system should offer. We will not discuss implementation or usability issues.

1.1. Privacy properties

A first family of properties is related to *privacy*. Any voting system should ensure that how voters voted is not leaked to anyone. To this end, most voting systems encrypt the votes on the voter's computer with the public key of the election, before sending it to the server. Note however that it is very difficult to protect against compromised computers that may leak the choice of the voter to a malicious third party. One possible countermeasure is to use pre-received code sheets [Chaum 2001] to hide the actual choice from the computer. Other currently studied solutions make use of (not necessarily trusted) external secure devices that perform parts of the computation [Lipmaa 2014].

Protecting vote privacy is necessary but not always sufficient from a security point of view. Ideally, a voter should not be able to show how he voted even if he is willing to. This is to avoid vote buying or coercion: if a voter can prove how he voted, then he may be coerced to vote in a certain way or he may sell his vote. Systems resistant to such attacks are called *receipt-free* or *coercion resistant*.

Another dimension of privacy is *everlasting privacy*. Will my vote remain secret in twenty years? It could be very embarrassing if our votes could be revealed years later. Most voting systems rely on cryptography which itself relies on computationally difficult problems. It is likely that in twenty years the keys in use nowadays will be broken, either due to faster computers, or algorithms improvements, or the discovery of some implementation flaws, and possibly a combination of those. It is therefore important for e-voting systems to also be robust on the long term, even when keys are eventually lost.

1.2. Verifiability properties

Anyone should be able to check that the final announced result does count all the votes casted by the voters. This *end-to-end verifiability* property may be divided in several sub-properties.

Individual verifiability ensures that each voter can check that his ballot appears on the ballot box. A natural way to achieve individual verifiability is to allow the ballot box to be publicly available on a website. But even if the ballot box is not available to the voters, some public version of the ballot box may suffice for a voter to check that his ballot has been successfully carried to the box. An example of such a case is a voting system based on Pedersen commitments [Cuvelier et al. 2013], that aims at everlasting privacy.

Then the announced result should correspond to the ballots on the box. This property is often referred to as *universal verifiability*. Typically, talliers produce a (zero-knowledge) proof of correct decryption. To preserve anonymity while offering verifiability, the tally is either computed after shuffling the ballots through mixnets or using homorphic encryption (e.g. El Gamal), which allows anyone to combine the votes before decryption.

Lastly, it should be possible to check that only legitimate voters have voted, otherwise the voting system may be subject to ballot stuffing. This is called *eligibility verifiability*.

Verifiability ensures that the entire process can be watched. Not only it is a very desirable property, but it also alleviates the task of auditing the implementation. There is no need anymore to trust the entire code run on the election system. The idea is that if something goes wrong it will be eventually noticed. It is even better if the faulty behaviour can be identified. If something went wrong, who should be blamed? This notion of *accountability* has been developed for example in [Küsters et al. 2012] and is very useful in the context of voting protocols.

2. EXISTING E-VOTING SYSTEMS

We briefly discuss some of the main voting systems in use and we present one of them in more details, Helios.

2.1. Brief survey on existing e-voting systems

The *Estonian voting system* is a rather simple system: voters encrypt their votes and sign the corresponding ballot, relying on the fact that Estonian citizens have an electronic ID card that contains a signing key. A voter may later query back his ballot to check that it is part of the box. But there is no verifiability: voters cannot check that their vote is actually counted nor that the result corresponds to the bulletin board. A precise description of the system and its vulnerabilities can be found in [Springall et al. 2014].

The *protocol used in Norway* is developed by Scytl [Gjøsteen 2010; doc 2013]. Once they have voted, voters receive a code by SMS, which can be used to check that their ballot belong to the box. Voters may still not verify the announced result. However, this protocol offers some "proxy-verifiability": external (approved) auditors can check the consistency of the ballot box and can check the proofs produced by the tallier when decrypting the ballots.

Helios [Adida et al. 2009] is an academic protocol developed by Ben Adida [Adida 2008], based on a protocol proposed by Cramers *et al* [Cramer et al. 1997]. It is used to elect the IACR board (International Association for Cryptologic Research) since 2010 [IACR]. The ballot box is public so that anyone can check that his ballot is present. The final result is also verifiable, based on either ElGamal encryption or mixnet tallying (both options are available). Helios therefore offers both ballot privacy, individual, and universal verifiability. A variant of Helios, Belenios [Cortier et al. 2013], further guarantees eligibility verifiability. Note however that neither Helios nor Belenios are receipt-free: these systems should be used in low-coercion environments.

Civitas [Clarkson et al. 2008] is one of the only protocols (if not the only one) to offer both verifiability and coercion-resistance. When a voter is under coercion, he may vote as prescribed using a fake credential. Once the coercer has left, the voter may vote again, with a valid credential. A coercer may not distinguish a fake credential from a valid one. Then invalid ballots are discarded after some shuffling, to guarantee that a coercer can still not notice any difference. Anyone can check that only invalid ballots have been rejected. However, to our knowledge, this protocol has not been used in real elections due to some practical issues. In particular, getting rid of invalid ballots requires a computation whose complexity is quadratic in the number of submitted ballots.

We have focused here on purely electronic voting systems, where voters cast their votes using the Internet. There are also a variety of hybrid systems where voters vote at a polling station and are offered some means to check that their votes have been counted. Such systems include for example Scantegrity [Chaum et al. 2008] and Prêt-à-voter [Ryan et al. 2009].

2.2. The Helios protocol in more details

An e-voting system should guarantee both privacy (no one know how I voted) and verifiability (I can check that my vote has been counted). These two properties are conflicting and the design of an e-voting system requires a good balance between confidentiality and verifiability. We explain how these two properties can be matched through the example of the Helios protocol [Adida et al. 2009]. In Helios, the ballot box is public (typically a web page) and anyone can access to it. Let pk denotes the public key of the election. The corresponding decryption key sk is split among several authorities: all the authorities need to collaborate to decrypt a message. For the sake of robustness, it is actually better to use *threshold decryption*: only k out of n authorities are needed to decrypt. This avoids to cancel a whole election because some authority has lost its key.

For simplicity, we assume here a referendum election where voters vote for 0 or 1. Assume Alice wishes to vote v_A. She simply encrypts her vote with the public key of the election, yielding the message $\{v_A\}_{pk}$. She also appends a zero-knowledge proof ZKP_A that v_A is a valid vote, that is $v_A = 0$ or $v_A = 1$. This is to avoid that Alice picks a wrong voting option. We will see later why it is important. Then Alice simply sends her ballot $\{v_A\}_{pk}, ZKP_A$ to the ballot box. Since the ballot box is public, she can check that her ballot is present on the ballot box, among other ballots as illustrated above.

$$\begin{array}{c|l} \multicolumn{2}{c}{\text{Ballot Box}} \\ \text{Alice} & \{v_A\}_{pk}, ZKP_A \\ \text{Bob} & \{v_B\}_{pk}, ZKP_B \\ \text{Charlie} & \{v_C\}_{pk}, ZKP_C \end{array}$$

The encryption scheme used in Helios is El Gamal. The tally phase makes use of the homomorphic property of El Gamal: The multiplication of the encryption of the votes yields the encryption of the sum of the votes.

$$\prod_{i=1}^{n} \{v_i\}_{pk} = \{\sum_{i=1}^{n} v_i\}_{pk}$$

We can observe here why voters must prove that they vote either 0 or 1. If they could submit an arbitrary message, they could modify arbitrarily the result by sending the encryption of k where k is a positive integer if they wish to increase the score and k is a negative integer if they wish to decrease the score.

Once $\{\sum_{i=1}^{n} v_i\}_{pk}$ is computed (which can be performed by anyone), the decryption authorities simply need to decrypt this single message, which preserves voter's privacy: the link between ballots and votes is never leaked. The authorities also provide a (zero-knowledge) proof of correct tabulation. Helios is therefore verifiable: any voter can check that his ballot belongs to the ballot box and everyone can check that the result has been correctly computed.

Helios is actually not fully private as it is subject to ballot-copying [Cortier and Smyth 2011]. A voter can copy a ballot on the board and send it as his own ballot. For simplicity, let's consider an election with three voters, Alice, Bob, and Charlie where Charlie is a dishonest voter. Assume that Alice and Bob have already voted.

$$\begin{array}{c|l} \multicolumn{2}{c}{\text{Ballot Box}} \\ \text{Alice} & \{v_A\}_{pk}, ZKP_A \\ \text{Bob} & \{v_B\}_{pk}, ZKP_B \end{array}$$

Then Charlie can simply copy Alice's ballot $\{v_A\}_{pk}, ZKP_A$ and send it to the ballot box, pretending it is his ballot.

$$\begin{array}{c|l} \multicolumn{2}{c}{\text{Ballot Box}} \\ \text{Alice} & \{v_A\}_{pk}, ZKP_A \\ \text{Bob} & \{v_B\}_{pk}, ZKP_B \\ \text{Charlie} & \{v_A\}_{pk}, ZKP_A \end{array}$$

The tally reveals the result of the election $r = 2v_A + v_B$. Charlie can then easily deduce Alice's vote: if $r \leq 1$ then $v_A = 0$ otherwise $v_A = 1$.

This attack can be fixed by weeding or rejected duplicated ballots. [Bernhard et al. 2012] rigorously shows that this is indeed sufficient.

3. SECURITY ANALYSIS

As for more traditional security protocols, the design of e-voting protocols is difficult and error-prone. These systems need a precise modelling and a rigorous analysis. The first task consists in formally defining security properties.

3.1. Properties

For vote privacy, we need to express that an adversary does not learn how a voter voted. However, a voting system does leak some information since the final result depends on the votes. For example, in the extreme case of unanimity, there is no privacy left. One common way of defining vote privacy is to require that an adversary cannot distinguish from the scenario where Alice is voting v_1 and Bob is voting v_2 from the converse scenario where Alice is voting v_2 and Bob is voting v_1:

$$\mathsf{Voter}(A, v_1) \mid \mathsf{Voter}(B, v_2) \approx \mathsf{Voter}(A, v_2) \mid \mathsf{Voter}(B, v_1)$$

This property has been formalized in symbolic models [Delaune et al. 2009], where primitives are abstracted by terms, as well as in cryptographic models. Cryptographic models offer three main types of definitions

— **Game-based definitions.** Game-based definitions are usually convenient for proving security since they are well tailored to proof by reduction to security assumptions (such as integer factorisation). The cryptographic counter-part of the privacy definition stated above has been proposed in [Benaloh and Yung 1986]. This definition does not capture all the information an attacker can get. For example, in the case of an approval voting, it may be the case that an attacker may distinguish when Alice is voting 0 and Bob is voting 2 from the scenario where both Alice and Bob are voting 1. Therefore several more general game-based definitions of privacy have been recently proposed (eg [Küsters et al. 2010; Bernhard et al. 2011; Bernhard et al. 2012; Chase et al. 2013; Cuvelier et al. 2013; Bernhard and Smyth 2013]). Surprisingly, game-based definitions are difficult to get right, which explains the number of definitions.
— **Entropy-based definitions.** Another approach intends to capture all kind of information leaked by a voting protocol. This is the case of entropy-based privacy definitions [Coney et al. 2005; Bernhard et al. 2012]. These definitions capture several sources of privacy leakage: leakage from the protocol itself but also from the votes distribution, or the result function: telling only the name of the elected candidate is clearly less leaky than providing the entire set of votes. Entropy-based notions are therefore powerful but also much harder to prove.
— **Ideal functionality.** Finally, privacy can be defined through an ideal voting system (also called ideal functionality) and proving that the actual voting system does not provide anymore information than its ideal version [Groth 2004].

As explained earlier, vote privacy is actually a weak form of privacy. For elections with important issues, voting schemes should be receipt-free or even coercion-resistant: an adversary should not be able to detect when a voter under coercion still votes for the candidate of his choice instead of voting as indicated by the coercer. In symbolic models, receipt-freeness as well as coercion-resistance can also be expressed through equivalence properties [Delaune et al. 2009; Backes et al. 2008]. Intuitively, these definitions state that an adversary should not be able to distinguish between the case where the voter votes exactly as requested by the coercer from the case where the voter uses a strategy to vote as he wishes, possibly revoting when he is not under coercion anymore.

Surprisingly, verifiability has deserved less attention than privacy. Individual, universal, and eligibility verifiability have been defined in symbolic models [Kremer et al. 2010]. This work also shows how these definitions depend on each other. Cryptographic models usually define verifiability in a more global way such as in [Juels et al. 2010; Cortier et al. 2013] where it is simply stated that the result should include at least the votes of all honest voters.

3.2. Formal analysis

Once the desired properties are defined, a first approach is to prove security by hand. This has been done for the Helios protocol for example, both in symbolic models [Cortier and Smyth 2013] and in cryptographic ones [Bernhard et al. 2011; Bernhard et al. 2012]. However, these proofs are rather long and difficult to check. We therefore focus here on how to automate, at least partially, security proofs of e-voting protocols. Proofs in computational models are difficult to automate. In contrast, several mature tools exist for the symbolic analysis of security protocols. For example, ProVerif [Blanchet 2005; Blanchet et al. 2005], Scyther [Cremers 2008], as well as Avispa [Armando et al. 2005] have been successfully applied to many protocols of the literature, including widely deployed ones such as TLS [Bhargavan et al. 2008] or Single-Sign-On [Armando et al. 2008]. However, these tools do not apply well to e-voting systems.

A first reason lies in the properties that need to be analysed. Standard security properties such as authentication or confidentiality are expressed as trace properties: for any execution trace of the protocol, the secret data a remains inaccessible (confidentiality); or for any execution trace, whenever Alice finishes a session supposedly with Bob then Bob has indeed initiated a session with Alice (authentication). Many variants can be defined as trace properties. In contrast, privacy properties are stated as indistinguishability properties, formally expressed through behavioural equivalence properties (e.g. observational equivalence, trace equivalence, or may-testing equivalence [Abadi and Gordon 1997]). Only the ProVerif tool can handle such equivalence properties. It actually proves a stronger notion, called diff-equivalence [Blanchet et al. 2005], that is unfortunately too strong when applied to voting protocols. The ProSwapper tool [Smyth] may help ProVerif in some cases. Note that equivalence properties are not only useful in the context of e-voting, they are also used for other privacy properties such as untraceability [Brusó et al. 2010] or anonymity [Cheval et al. 2013; Arapinis et al. 2014]. Therefore, some prototypes tools dedicated to equivalence have been recently developed: AKISS [Chadha et al. 2012], APTE [Cheval 2014], and SPEC [Dawson and Tiu 2010].

However, all these tools and techniques face a second challenge: e-voting systems make a wide use of not so standard cryptographic primitives. For example, the FOO protocol [Fujioka et al. 1992] relies on blind signatures: voters have their blinded ballots signed by the registration authority and then remove the blinding factor. This property can be formalised as follows [Delaune et al. 2009]:

$$\mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x, z), y), z) = \mathsf{sign}(x, y)$$

This equation is already out of reach of the APTE and SPEC but this protocol can actually by analysed by Akiss [Chadha et al. 2012]. Consider now the Helios protocol presented earlier: combining encrypted votes yields the encryption of the sum of the votes. This can be reflected by the following equation

$$\mathsf{enc}(pk; v_1) * \mathsf{enc}(pk; v_2) = \mathsf{enc}(pk; v_1 + v_2)$$

where $*$ and $+$ are associative and commutative (AC) operators. AC operators are currently out of reach of any existing tool for analysing security protocols. Things get even worse with the Norwegian protocol that uses an ad-hoc construction. In this protocol, voters encrypt their votes with public key $\mathsf{pk}(a_1)$ using ElGamal encryption. The ballot box then re-encrypts a blinded version of the ballot with a secret key a_2 such that the receipt generator can now decrypt the ballot with $a_3 = a_1 + a_2$, without having access to the vote. This is reflected by the following symbolic equations [Cortier and Wiedling 2012]:

$$\mathsf{renc}(\mathsf{enc}(x_{plain}, x_{rand}, \mathsf{pk}(x_{sk})), y_{sk}) = \mathsf{enc}(x_{plain}, x_{rand}, \mathsf{pk}(x_{sk} + y_{sk}))$$

$$\mathsf{dec}(\mathsf{blind}(\mathsf{enc}(x_{plain}, x_{rand}, \mathsf{pk}(x_{sk})), x_{blind}), x_{sk}) = \mathsf{blind}(x_{plain}, x_{blind})$$

Coming up with resolution procedures that handle such complex equational theories is a challenging research goal. In particular, we need new techniques for handling AC operators, larger theories, and for combining theories.

Challenges

In the previous section, we have already discussed two main challenges: to evaluate e-voting systems, formal analysis techniques have to cope with both equivalence properties and complex equational theories. No automatic prover can do both for the moment.

It will be probably difficult if not impossible to get decidability results for both equivalence properties and wide classes of equivalence properties. One way to circumvent decidability issues is to design good over-approximations. Defining over-approximations is somewhat easy in the case of trace-based properties: if a protocol P is "simplified" into P' then the security of P' implies the security of P as soon as the execution traces of P' include all execution traces of P. Typical approximations consist in removing nonce freshness (the same nonce may be generated twice) or providing more knowledge to the attacker. However, this reasoning no longer works in the case of equivalence properties. If P is simplified into P' and Q into Q', the equivalence $P' \approx Q'$ does not imply $P \approx Q$. It is therefore necessary to design new approximations, sound w.r.t. equivalence properties. One approach could be to use type systems [Barthe et al. 2014], which proved successful to handle indistinguishability properties as well as a relatively large class of cryptographic schemes.

Surprisingly, privacy has deserved much more attention than verifiability, both in terms of definitions and analysis, while both properties are equally desirable and important for e-voting systems. One explanation is that verifiability seems easier to define. Yet, it remains to formally define verifiability and check the definition against most existing protocols. From a verification point of view, a second step is to study how to automate verifiability analysis. Verifiability can be classified as a trace-based property, for which many techniques exist. However, verifiability requires to count: "the result contains exactly once each vote of honest voters, plus at most k dishonest votes", where k is the number of corrupted voters, that is the number of voters under the control of the attacker. It is unclear how existing tools can cope with verifiability.

Finally, e-voting systems also question the security models that are usually considered for protocols. In particular, it is not always realistic to trust the computer's voters since it may host worms for example. To control the computer voter's behaviour, some e-voting systems make use of external devices (e.g. secure tokens as in banking applications), code sheets, or out-of-band channels (e.g. SMS). Of course, the cost of each of these techniques differ and corresponding precise security models still need to be defined. The threat scenario also differs depending on the election. A challenging question is how to define a voting systems, or a family of voting systems, that can (provably) cope with each threat scenario.

Acknowledgments. I would like to thank Matteo Maffei for his careful reading and his numerous and helpful suggestions.

The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no 258865, project ProSecure.

REFERENCES

M. Abadi and A. D. Gordon. 1997. A Calculus for Cryptographic Protocols: The Spi Calculus. In *Proc. of the 4th ACM Conference on Computer and Communications Security (CCS'97)*. ACM Press, 36–47.

Ben Adida. 2008. Helios: Web-based Open-Audit Voting. In *USENIX Security'08: 17th USENIX Security Symposium*. USENIX Association, 335–348.

Ben Adida, Olivier de Marneffe, Oliver Pereira, and Jean-Jacques Quisquater. 2009. Electing a university president using open-audit voting: Analysis of real-world use of Helios. In *Proceedings of the 2009 conference on Electronic voting technology / workshop on trustworthy elections*.

M. Arapinis, L. Mancini, E. Ritter, and M. Ryan. 2014. Privacy through pseudonymity in mobile telephony systems. In *21st Annual Network and Distributed System Security Symposium (NDSS'14)*.

A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. 2005. The AVISPA Tool for the automated validation of internet security protocols and applications. In *17th International Conference on Computer Aided Verification, CAV'2005 (Lecture Notes in Computer Science)*, Vol. 3576. Springer, 281–285.

Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, and Llanos Tobarra Abad. 2008. Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps. In *Proceedings of the 6th ACM Workshop on Formal Methods in Security Engineering (FMSE 2008)*. 1–10.

Michael Backes, Catalin Hritcu, and Matteo Maffei. 2008. Automated Verification of Electronic Voting Protocols in the Applied Pi-calculus. In *Proceedings of 21st IEEE Symposium on Computer Security Foundations (CSF 2008),*. IEEE, 195–209.

Jordi Barrat, Esteve, Ben Goldsmith, and John Turner. 2012. *International Experience with E-Voting*. Technical Report. Norwegian E-Vote Project.

Gilles Barthe, Cédric Fournet, Benjamin Grégoire, Pierre-Yves Strub, Nikhil Swamy, and Santiago Zanella Béguelin. 2014. Probabilistic Relational Verification for Cryptographic Implementations. In *41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'14)*. 193–206.

Josh Cohen Benaloh and Moti Yung. 1986. Distributing the Power of a Government to Enhance the Privacy of Voters (Extended Abstract). In *Proc. 5th Annual ACM Symposium on Principles of Distributed Computing (PODC'86)*. ACM, 52–62.

David Bernhard, Véronique Cortier, Olivier Pereira, Ben Smyth, and Bogdan Warinschi. 2011. Adapting Helios for provable ballot secrecy. In *Proceedings of the 16th European Symposium on Research in Computer Security (ESORICS'11) (Lecture Notes in Computer Science)*, Vol. 6879.

David Bernhard, Véronique Cortier, Olivier Pereira, and Bogdan Warinschi. 2012. Measuring Vote Privacy, Revisited. In *19th ACM Conference on Computer and Communications Security (CCS'12)*. ACM.

David Bernhard, Olivier Pereira, and Bogdan Warinschi. 2012. How Not to Prove Yourself: Pitfalls of the Fiat-Shamir Heuristic and Applications to Helios. In *Advances in Cryptology - (ASIACRYPT 2012)*. 626–643.

David Bernhard and Ben Smyth. 2013. Ballot privacy and ballot independence coincide. In *Proceedings of the 18th European Symposium on Research in Computer Security (ESORICS'13) (Lecture Notes in Computer Science)*, Springer (Ed.).

Karthikeyan Bhargavan, Ricardo Corin, Cédric Fournet, and Eugen Zalinescu. 2008. Cryptographically Verified Implementations for TLS. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS'08)*. 459–468.

Bruno Blanchet. 2005. An Automatic Security Protocol Verifier based on Resolution Theorem Proving (invited tutorial). In *20th International Conference on Automated Deduction (CADE-20)*.

Bruno Blanchet, Martín Abadi, and Cédric Fournet. 2005. Automated Verification of Selected Equivalences for Security Protocols. In *20th IEEE Symposium on Logic in Computer Science (LICS 2005)*. IEEE Computer Society, 331–340.

Mayla Brusó, Konstantinos Chatzikokolakis, and Jerry den Hartog. 2010. Formal verification of privacy for RFID systems. In *CSF'10: 23rd Computer Security Foundations Symposium*. IEEE Computer Society, 75–88.

Rohit Chadha, Ştefan Ciobâcă, and Steve Kremer. 2012. Automated verification of equivalence properties of cryptographic protocols. In *21th European Symposium on Programming (ESOP'12) (LNCS)*. Springer. To appear.

Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Sarah Meiklejohn. 2013. Verifiable Elections That Scale for Free. In *16th International Conference on Practice and Theory in Public-Key Cryptography (PKC 2013) (Lecture Notes in Computer Science)*, Vol. 7778. Springer, 479–496.

David Chaum. 2001. Sure vote: Technical overview. In *Proceedings of the Workshop on Trustworthy Elections (WOTE 01)*.

David Chaum, Aleks Essex, Richard Carback, Jeremy Clark, Stefan Popoveniuc, Alan Sherman, and Poorvi Vora. 2008. Scantegrity: End-to-End Voter-Verifiable Optical-Scan Voting. *IEEE Security and Privacy* 6, 3 (2008), 40–46.

Vincent Cheval. 2014. APTE: an Algorithm for Proving Trace Equivalence. In *Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14) (Lecture Notes in Computer Science)*. Springer.

Vincent Cheval, Véronique Cortier, and Antoine Plet. 2013. Lengths may break privacy – or how to check for equivalences with length. In *Proceedings of the 25th International Conference on Computer Aided Verification (CAV'13) (Lecture Notes in Computer Science)*, Vol. 8043. Springer, 708–723.

Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. 2008. Civitas: Toward a Secure Voting System. In *Proc. IEEE Symposium on Security and Privacy*. 354–368.

Lillie Coney, Joseph L. Hall, Poorvi L. Vora, and David Wagner. 2005. Towards a Privacy Measurement Criterion for Voting Systems. In *In National Conference on Digital Government Research*.

Véronique Cortier, David Galindo, Stephane Glondu, and Malika Izabachene. 2013. A generic construction for voting correctness at minimum cost - Application to Helios. Cryptology ePrint Archive, Report 2013/177. (2013).

Véronique Cortier and Ben Smyth. 2011. Attacking and fixing Helios: An analysis of ballot secrecy. In *24th Computer Security Foundations Symposium (CSF'11)*. IEEE Computer Society.

Véronique Cortier and Ben Smyth. 2013. Attacking and fixing Helios: An analysis of ballot secrecy. *Journal of Computer Security* 21, 1 (2013), 89–148. DOI:http://dx.doi.org/10.3233/JCS-2012-0458

Véronique Cortier and Cyrille Wiedling. 2012. A formal analysis of the Norwegian E-voting protocol. In *Proceedings of the 1st International Conference on Principles of Security and Trust (POST'12) (Lecture Notes in Computer Science)*, Vol. 7215. Springer, 109–128. DOI:http://dx.doi.org/10.1007/978-3-642-28641-4_7

Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. 1997. A Secure and Optimally Efficient Multi-Authority Election Scheme. In *International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT'97)*. 103–118.

Cas Cremers. 2008. The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols. In *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, USA, Proc. (Lecture Notes in Computer Science)*, Vol. 5123/2008. Springer, 414–418. DOI:http://dx.doi.org/10.1007/978-3-540-70545-1_38

Edouard Cuvelier, Olivier Pereira, and Thomas Peters. 2013. Election Verifiability or Ballot Privacy: Do We Need to Choose?. In *Proceedings of the 18th European Symposium on Research in Computer Security (ESORICS'13) (Lecture Notes in Computer Science)*.

Jeremy Dawson and Alwen Tiu. 2010. Automating open bisimulation checking for the spi-calculus. In *proceedings of IEEE Computer Security Foundations Symposium (CSF 2010)*.

Stéphanie Delaune, Steve Kremer, and Mark D. Ryan. 2009. Verifying privacy-type properties of electronic voting protocols. *Journal of Computer Security* 17, 4 (2009), 435–487.

doc 2013. Documentations of the code used for the 2013 Parliamentary election in Norway. https://brukerveiledning.valg.no/Dokumentasjon/Dokumentasjon/Forms/AllItems.aspx. (2013).

Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. 1992. A Practical Secret Voting Scheme for Large Scale Elections. In *AUSCRYPT'92: Workshop on the Theory and Application of Cryptographic Techniques (LNCS)*. Springer.

Kristian Gjøsteen. 2010. Analysis of an internet voting protocol. Cryptology ePrint Archive, Report 2010/380. (2010). http://eprint.iacr.org/.

Jens Groth. 2004. Evaluating Security of Voting Schemes in the Universal Composability Framework. In *2nd Int. Conference in Applied Cryptography and Network Security (ACNS 2004) (Lecture Notes in Computer Science)*, Vol. 3089. Springer, 46–60.

IACR. International association for cryptologic research. Elections page at http://www. iacr.org/elections/.

Ari Juels, Dario Catalano, and Markus Jakobsson. 2010. Coercion-Resistant Electronic Elections. In *Towards Trustworthy Elections: New Directions in Electronic Voting*. LNCS, Vol. 6000. Springer, 37–63.

Steve Kremer, Mark D. Ryan, and Ben Smyth. 2010. Election verifiability in electronic voting protocols. In *15th European Symposium on Research in Computer Security (ESORICS'10) (LNCS)*. Springer.

Ralf Küsters, Tomasz Truderung, and Andreas Vogt. 2010. A Game-Based Definition of Coercion-Resistance and its Applications. In *CSF'10: 23rd IEEE Computer Security Foundations Symposium*. IEEE Computer Society, 122–136.

R. Küsters, T. Truderung, and A. Vogt. 2012. Clash Attacks on the Verifiability of E-Voting Systems. In *33rd IEEE Symposium on Security and Privacy (S&P 2012)*. IEEE Computer Society, 395–409.

Helger Lipmaa. 2014. A Simple Cast-as-Intended E-Voting Protocol by Using Secure Smart Cards. Cryptology ePrint Archive, Report 2014/348. (2014). http://eprint.iacr.org/.

P.Y.A. Ryan, D. Bismark, J. Heather, S. Schneider, and Z. Xia. 2009. The Pret a Voter Verifiable Election System. *IEEE Transactions on Information Forensics and Security* (2009).

Ben Smyth. ProSwapper. http://www.bensmyth.com/proswapper.php.

Drew Springall, Travis Finkenauer, Zakir Durumeric, Jason Kitcat, Harri Hursti, Margaret MacAlpine, and J. Alex Halderman. 2014. Security Analysis of the Estonian Internet Voting System. In *Proc. 21st ACM Conference on Computer and Communications Security (CCS'14)*.

SEMANTICS COLUMN

MICHAEL MISLOVE, Tulane University
mwm@math.tulane.edu

One of the goals of this column is to highlight breakthroughs so the semantics community can follow them as they unfold. This quarter's column concerns just such a breakthrough – the amazing relationship between algebraic topology, a mainstay of mathematics, and type theory, a fundamental component of logic and computer science, that has emerged through Voevodsky's univalent foundation for mathematics. Working out the details of this connection was the subject of a recent year-long program at IAS that produced the HoTT book referenced in the column itself.

On the mathematical side, there is homotopy theory. Given maps $f, g: X \to Y$ between topological spaces X and Y, a *homotopy* from f to g is a mapping $F: X \times [0,1] \to Y$ satisfying $F(-,0) = f$ and $F(-,1) = g$. The spaces X and Y are then *homotopic* if there are maps $\phi: X \to Y$ and $\psi: Y \to X$ so that the compositions in either order are homotopic to the appropriate identity map.

On the logical side, there is type theory. Originally devised by Russell as a way to avoid paradoxes like the one that bears his name, Church also used a simple type hierarchy to avoid the Kleene-Rosser paradox, and gave us the simply-typed lambda calculus.

These two seemingly disparate areas were brought together by Voevodsky's univalent foundations for mathematics, or, as is pointed out in the column itself, Voevodsky's work "clinched the connection" that was already emerging in work by others.

Every undergraduate mathematics major learns about homotopy theory when the *fundamental group* of a space is introduced: In the case of a path-connected space X, it is the set of homotopy equivalence classes of maps $f: \mathbb{S}^1 \to X$ from the circle to X. This construction is due to Poincaré [2], and was first applied in the study of Riemann surfaces. Čech [1] generalized this construction to the family of *homotopy groups* $\pi_n(X), n \geq 1$ of a space X. $\pi_1(X)$ is the fundamental group, and $\pi_n(X)$ is the family of maps from the n-sphere $\mathbb{S}^n$ to X. This family of abelian groups reflects structural properties of the space. Because homeomorphic spaces have isomorphic homotopy groups, the groups can tell when two spaces are not homeomorphic. For example, X is *contractible* if $\pi_1(X)$ is the one-point group, so $\mathbb{R}^n$ is contractible, while the n-sphere $\mathbb{S}^n$ is not.

Even though they appear in the foundations of mathematics, most mathematics students would not have encountered types or type theory – the Zermelo-Frankel Axioms make no mention of them. While the von Neumann hierarchy, which accounts for all ZF-sets because of the Foundation Axiom, is a type hierarchy, it is usually called the cumulative hierarchy, with no mention of type theory at all.

Of course, computer science students encounter types in their courses on programming languages, where they play a crucial role. And the seminal work of Dana Scott on Church's un(i)typed lambda calculus gave us domain theory, which is a fundamental tool for semantics.

Proof assistants have been under development for many years, with applications to a wide range of areas. The use of proof assistants to devise constructive proofs of mathematical results has opened the door to many unexpected results, one of which is the recent work on homotopy type theory.

Clearly then, there is ample reason for the semantics column to include an account of the recent developments relating homotopy theory, type theory and constructive mathematics. Steve Awodey and Bob Harper, two of the researchers who are pushing these developments forward, offer us their insights into how this relationship was discovered, and what might lie ahead.

REFERENCES

1. Čech, E., Höherdimensionale Homotopiegruppen, Verhandlungen des Internationalen Mathematikerkongress, Zürich (Orell Fssli) (1932).

2. Poincaré, H., Analysis situs. Journal de l'École Polytechnique **1** (1895), pp. 1–123.

Homotopy Type Theory: Unified Foundations of Mathematics and Computation[1]

Steve Awodey
Carnegie Mellon University

Robert Harper
Carnegie Mellon University

Homotopy type theory is a recently-developed unification of previously disparate frameworks, which can serve to advance the project of formalizing and mechanizing mathematics. One framework is based on a computational conception of the type of a construction, the other is based on a homotopical conception of the homotopy type of a space. The computational notion of type has its origins in Brouwer's program of intuitionism, and Church's λ-calculus, both of which sought to ground mathematics in computation (one would say "algorithm" these days). The homotopical notion comes from Grothendieck's late conception of homotopy types of spaces as represented by ∞-groupoids [Grothendieck 1983]. The computational perspective was developed most fully by Per Martin-Löf, leading in particular to his Intuitionistic Theory of Types [Martin-Löf and Sambin 1984], on which the formal system of homotopy type theory is based. The connection to homotopy theory was first hinted at in the groupoid interpretation of Hofmann and Streicher [Hofmann and Streicher 1994; 1995].[2] It was then made explicit by several researchers, roughly simultaneously.[3] The connection was clinched by Voevodsky's introduction of the *univalence axiom*, which is motivated by the homotopical interpretation, and which relates type equality to homotopy equivalence [Kapulkin et al. 2012; Awodey et al. 2013].

Constructive foundations are often regarded as incompatible with classical mathematics. By contrast, the framework of homotopy type theory is fully compatible with classical mathematics, and indeed allows for a classical conception of proposition, as well as a conception of set that is compatible with such principles as the axiom of

[1]Thanks to Daniel Grayson, Michael Mislove, and Vladimir Voevodsky for helpful comments on an earlier draft. Of course, the authors alone are still responsible for any errors or misstatements.

[2]The importance of equality of elements of a type in constructive mathematics was also emphasized by Bishop [Bishop and Bridges 1985]. Quotient types, which were introduced in NuPRL as a further development of Bishop's and Martin-Löf's ideas [Constable, *et al.* 1985], may be seen as a particular case of this connection.

[3]Awodey and Warren [Awodey and Warren 2009] showed that the basic system of Martin-Löf type theory can be interpreted in a Quillen model category (an abstract framework for doing homotopy theory); Lumsdaine [Lumsdaine 2009] and van den Berg and Garner [van den Berg and Garner 2011] showed that every type in the system has the structure of an ω-category (a structure closely related to that of an ∞-groupoid); Gambino and Garner [Gambino and Garner 2008] showed that the type theory itself supports a weak factorization system (the basic building block of a Quillen model structure); and both Streicher [Streicher 2006] and Voevodsky [Voevodsky 2006] proposed interpretations into the category of simplicial sets, using ideas from homotopy theory.

choice. The key to achieving this unification is to avoid postulating generally certain reasoning principles, such as the decidability of every type, although these may still be postulated "locally", for example the decidability of every *proposition*. It is notable that these same reasoning principles are also those that are usually avoided in constructive foundations, opening the door to the unification of the constructive (computational) and homotopic (spatial) interpretations of types, the implications of which are only just beginning to be understood. Moreover, by not insisting on these principles globally, it is possible to consider a far richer notion of type than has previously been considered in the computational approach, namely one in which types are abstract spaces that may have non-trivial higher-dimensional structure, like the n-spheres for all $n \geq 0$. In conventional foundations, such as axiomatic set theory, these objects are presented as structured sets representing certain conceptions of space, such as topological spaces. Here, instead, such higher-dimensional objects arise "synthetically" in much the way that lines and triangles in Euclid's geometry are primitive abstract objects, rather than being comprised of analytic point-sets. This provides a new perspective on some familiar constructions in homotopy theory, such as the homotopy groups of a space [Univalent Foundations Program 2013; Licata and Shulman 2013; Licata and Brunerie 2013] and the construction of so-called Eilenberg-MacLane spaces [Licata and Finster 2014], with specified homotopy groups. Moreover, new proofs of some standard results have a distinctively "logical" flavor, in combination with more "geometric" and "topological" elements.

What is it that makes this new unification possible? Although it may be too early to formulate a single, deep unifying principle, it is possible to make a few observations that will give the reader a sense of its inevitability. First, all of the constructions of Intuitionistic Type Theory, including especially the identity type, are homotopy invariant, in the sense that type families and mappings between types inherently respect identifications (paths, homotopies, or deformations). Moreover, the formation of indexed products and sums of types, which correspond to analogous constructions on spaces, respect the homotopically motivated notion of equivelence of types,corresponding to the homotopy equivalence of spaces. This invariance essentially follows from the basic fact that Martin-Löf's ingenious concept of the *identity type* corresponds to the path space of a space, and since everything in the formal system respects identity, everything in the interpretation respects homotopy, which is determined by identfication along paths. Second, a characteristic feature of both intuitionistic type theory and homotopy theory is an emphasis on *structure* over *property*. Under the propositions-as-types conception of intuitionistic logic,[4] types express propositions, and objects of the type are proofs of those propositions in the form of mathematical constructions that provide evidence for their truth. A similar emphasis can be discerned in abstract homotopy theory, in which, *e.g.*, paths (homotopies) may be seen as evidence for the "identification" of two points, and similarly for paths between the corresponding values of two functions. Two points are not merely "equal", as a property, but rather are identified by a (not necessarily unique) deformation, construction, or procedure. This approach extends to higher dimensions, in that one may speak of the identifications of two (parallel) identifications, at all higher dimensions. Such a structure of a hierarchy of identifications via path connectedness is found in standard settings for homotopy theory such as simplicial sets, cubical sets, and globular sets, all of which stress the role of cells as identifications.

We thus already see an analogy between the constructively motivated concept of *proof relevance*, in which proofs are mathematical objects classified by a type, and the homotopically motivated distinction between structure and property. An important ad-

[4]See, *e.g.*, [Howard 1980] for its original formulation.

vantage resulting from proof relevance is that it naturally supports a comprehensive approach to mechanized mathematics in which computer systems, such as Coq [Coq] and Agda [Agda], can be used to verify the correctness of mathematical arguments, of either a classical, set-theoretic form, or a constructive, type-theoretic form. In either case the proof of a theorem constitutes a formal mathematical object whose validity can be independently checked, avoiding the need to rely on the correctness of the proof checker itself. Once a proof has been obtained, others can not only check its formal correctness by the usual means, but can also submit the proof to another checker, to ensure that it is valid according to the rules of homotopy type theory. This approach to verification is fundamentally the same as that proposed by de Bruijn in the Automath system [Automath], albeit applied to a language with richer foundational commitments than were required there. It should be contrasted with the approach of systems such as NuPRL [NuPRL] or HOL [HOL ; HOL-Light], that rely on a small trusted code base to ensure the validity of proofs.

The idea of identifications of points in a space along a continuous path, and of higher identifications of paths as homotopies, etc., leads to Voevodsky's conception of a hierarchy of *homotopy levels*, or *h-levels* for short, which is definable within type theory. Whereas the usual hierarchy of size is determined by type universes or large cardinals in type theory or set theory, the hierarchy of h-levels is based instead on the internal structure of types. Roughly speaking, the lowest level consists of the types that have at most one element, up to path-connectedness; these are called *propositions*, and they correspond to the (empty or) contractible spaces. The next level, called *sets*, consists of those types whose identity types are themselves propositions — two elements of a set are "equal in at most one way". After that come the types whose identity types are sets; these are the *groupoids*. And so on, with the types at level $n + 1$ being those whose identity types have level n, for all $n \geq 0$. Just the recognition that this hierarchy of h-levels is present in the system of all types has been a huge advance in our understanding of type theory; previously, it was simply a mystery that some types were fully determined by their elements, while others seemed to behave as though they had some further structure. The construction of quotient types, for example, is now greatly simplified when one knows that the equivalence relation being factored out is a family of propositions, and not of "higher-dimensional" types. For another example, for types A and B that are propositions, the relevant notion of equivalence is *logical equivalence*, represented by the type $A \leftrightarrow B$. For sets, the relevant notion is *isomorphism* $A \cong B$, and for groupoids, there is the notion of *(categorical) equivalence* $A \simeq B$. Each of these concepts results by specializing the single, uniform notion of *equivalence of types* $A \simeq B$ (also due to Voevodsky) to the respective cases of propositions, sets, and groupoids.

In this setting, Voevodsky's univalence axiom can be stated as the assertion that the type $A \simeq B$ of all equivalences between two types A and B is itself equivalent to their identity type,

$$(A \simeq B) \;\simeq\; \mathsf{Id}(A, B). \tag{UA}$$

Thus in particular, logically equivalent propositions will be identified, as in the original, extensional type theory of Church [Church 1940]. Isomorphic sets, too, will be identified "up to homotopy", *i.e.*, by paths between them in the universe of all types, and similarly for equivalent groupoids, and equivalent types in general. Note that this stipulation also serves to specify the otherwise underdetermined identity type $\mathsf{Id}(A, B)$.

This is not really the place for a systematic introduction (for that, see [Univalent Foundations Program 2013]), but a brief example may serve to convey a bit of the flavor of the new approach, especially the distinctive intermingling of logical and homotopical ideas. As is the case in conventional Martin-Löf type theory, the basic types of booleans $\mathbb{B}$ and natural numbers $\mathbb{N}$ can have at most one identification between any

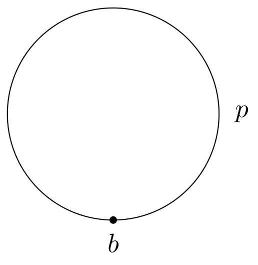


Fig. 1. The 1-sphere S^1

two elements; that is, given say $n, m : \mathbb{N}$ and $p, q : \mathsf{Id}_{\mathbb{N}}(n, m)$ in the identity type of n and m, we always have some $\alpha : \mathsf{Id}_{(\mathsf{Id}_{\mathbb{N}}(n,m))}(p, q)$ identifying p and q. In this sense, there is no real information in the type $\mathsf{Id}_{\mathbb{N}}(n, m)$, apart from whether or not it is inhabited. Such types with at most one identification between any two elements are called "sets". Any types that can be constructed from $\mathbb{B}$, $\mathbb{N}$, or any other sets, by means of the usual type constructors of dependent sum $\Sigma_{x:A}\, B(x)$ and dependent product $\Pi_{x:A}\, B(x)$ (which include $A \times B$ and $A \to B$ as special cases) are also sets, and the same is true for the identity types $\mathsf{Id}_A(a, a')$ for $a, a' : A$, for any set A.

An example of a type that is not a set is the circle (or "1-sphere") S^1, which has a base point $b : S^1$ and a generating loop $p : \mathsf{Id}_{S^1}(b, b)$. There are then many different self-identifications, which may be labelled

$$\mathsf{refl}(b),\ p,\ p \cdot p,\ \ldots : \mathsf{Id}_{S^1}(b, b).$$

Here $\mathsf{refl}(b)$ is the trivial identification, *i.e.*, the canonical witness to the reflexivity of identity. There is also the identification p, which is different from $\mathsf{refl}(b)$ in the sense that $\mathsf{Id}_{(\mathsf{Id}_{S^1}(b,b))}(\mathsf{refl}(b), p)$ is empty. We can think of p homotopically as the continuous "path" that goes once around the circle.

By the (function witnessing the) transitivity of equality,

$$(-) \cdot (-) : \mathsf{Id}_{S^1}(a, b) \times \mathsf{Id}_{S^1}(b, c) \longrightarrow \mathsf{Id}_{S^1}(a, c),$$

there are also the "paths" $p \cdot p$, $p \cdot p \cdot p$, ... And by symmetry,

$$(-)^{-1} : \mathsf{Id}_{S^1}(a, b) \longrightarrow \mathsf{Id}_{S^1}(b, a),$$

there are similarly the paths p^{-1}, $p^{-1} \cdot p^{-1}, \ldots : \mathsf{Id}_{S^1}(b, b)$. Although S^1 is therefore not a set, it can be shown that $\mathsf{Id}_{S^1}(b, b)$ is one; that is, the types $\mathsf{Id}_{(\mathsf{Id}_{S^1}(b,b))}(x, y)$ are either inhabited (by reflexivities) or empty, depending on whether or not $x = y$, for all $x, y : \mathsf{Id}_{S^1}(b, b)$. Indeed, one can show that $\mathsf{Id}_{S^1}(b, b) \cong \mathbb{Z}$, *i.e.*, the fundamental group of the type S^1 is the integers, as it should be (see [Licata and Shulman 2013] for the details). The proof of this uses the univalence axiom, together with the specification of S^1 as a new kind of *higher* inductive type, generalizing the usual inductive specification of the natural numbers and similar structures. For S^1, the inductive specification essentially says that S^1 is "the type freely generated by the base point $b : S^1$ and the loop $p : \mathsf{Id}_{S^1}(b, b)$", in the same sense that the usual inductive specification of $\mathbb{N}$ says that it is the type freely generated by $0 : \mathbb{N}$ and the successor function $s : \mathbb{N} \longrightarrow \mathbb{N}$.

Another type that is not a set is the universe $\mathcal{U}$ of all (small) types. According to the Univalence Axiom, identifications between types $A, B : \mathcal{U}$ correspond to equivalences $A \simeq B$, which as we said above are generalized type isomorphisms. In fact, as already stated, if A and B themselves are sets, then an equivalence between them is just an isomorphism in the usual sense: a pair of maps back and forth that compose to the respective identity mapppings. Now the booleans $\mathbb{B}$, for example, have two different

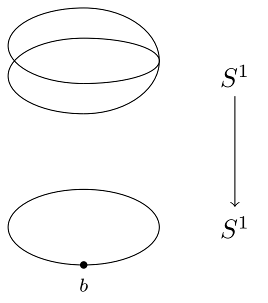


Fig. 2. The twisted double cover of S^1

isomorphisms $\mathbb{B} \cong \mathbb{B}$, namely the identity and the operation of "negation" $\neg : \mathbb{B} \longrightarrow \mathbb{B}$, which swaps the truth values $0, 1 : \mathbb{B}$. Thus by univalence there are two distinct identifications in $\mathsf{Id}_{\mathcal{U}}(\mathbb{B}, \mathbb{B})$, corresponding to these distinct isomorphisms, and so $\mathcal{U}$ is not a set, but a "higher-dimensional" type, like S^1.

Now observe that by the basic recursive property of S^1 as "the type freely generated by a point with a loop on it", there is a map

$$\mathsf{rec}(\mathbb{B}, n) : S^1 \longrightarrow \mathcal{U},$$

determined by sending the base point $b : S^1$ to the booleans $\mathbb{B} : \mathcal{U}$ and the generating loop $p : \mathsf{Id}_{S^1}(b, b)$ to (the loop corresponding under univalence to) negation, say $n : \mathsf{Id}_{\mathcal{U}}(\mathbb{B}, \mathbb{B})$. As a type of the form $S^1 \longrightarrow \mathcal{U}$, this $\mathsf{rec}(\mathbb{B}, n)$ is thus a family of types over S^1, sometimes called a "dependent type" and written

$$x : S^1 \vdash \mathsf{rec}(\mathbb{B}, n)(x).$$

Homotopically, such a type-family is interpreted as a "fibration" $E \longrightarrow S^1$, where the total space E is just the sum type $\Sigma_{x:S^1} \mathsf{rec}(\mathbb{B}, n)(x)$, equipped with its usual indexing projection. In the present case, the "fiber" is then the type $\mathsf{rec}(\mathbb{B}, n)(b) = \mathbb{B}$, and the action on (elements of) $\mathbb{B}$ induced by the path $n : \mathsf{Id}_{S^1}(b, b)$ in the base is exactly the operation of negation $\neg : \mathbb{B} \longrightarrow \mathbb{B}$. Thus, from a homotopical point of view, we have constructed the "twisted double cover" of the circle (see Figure 2). This construction from homotopy theory is closely related to the celebrated Hopf fibration which, among other things, can be used to compute some of the higher homotopy groups of the spheres S^2 and S^3. Indeed, one can construct the Hopf fibration in homotopy type theory in much the same way as the foregoing example, using univalence, negation, winding around the circle, and other constructions derived from combinations of logical, type-theoretic, and homotopical ideas (see [Univalent Foundations Program 2013], §8.5).

We can now say in a bit more detail how the univalent framework of homotopy type theory subsumes and extends the classical, set-theoretic framework for doing mathematics, by making use of the hierarchy of h-levels, which includes sets within a broader framework of homotopy types. At the bottom level, the propositions (the types having at most one element, up to higher identification) correspond to conventional, proof-irrelevant propositions; whether we also assert the law of excluded middle in the form that every such proposition is either inhabited or empty is a further, consistent assumption that may be made if classical logic is desired. Next, the sets (for which equality is a proposition that is taken to be "self-evident" or "proof-irrelevant") correspond to the usual sets, but now without any commitment to choice principles, or whether membership is a boolean proposition. Those further principles can still be consistently taken as axioms if needed, but they are not required, even with the introduction of infinite sets such as the type of natural numbers. Voevodsky's new insight, which plays such an important role in homotopy type theory, is that, besides the familiar concepts of proposition (classically formalized in predicate calculus) and set (classically given by the Zermelo-Fraenkel axioms), there is an infinite hierarchy of further dimensions extending beyond just these two. The groupoids (the next h-level above the sets), such as our example S^1, are the natural setting for systems of set-theoretic structures, such as groups and rings, that one may wish to regard as identified up to isomorphism. Because two groups, say, can be isomorphic in many different ways, however, the evidence for an identification is not a trivial proposition, but consists in the mutually inverse pair of homomorpisms, *i.e.*, the isomorphism, that warrant it. Here we see explicitly how proof-relevance (from constructivism) and the "property-structure" distinction (from homotopy theory) coincide.

In this way we can now distinguish, within Martin-Löf type theory, an infinite hierarchy of different "homotopical dimensions" that were not fully recognized previously, despite such models as Hofmann and Streicher's two-dimensional groupoid interpretation [Hofmann and Streicher 1994] that strongly hinted at the importance of higher dimensions of structure. Type theory was, of course, originally conceived as a foundation for constructive mathematics, in which all constructions, including proofs of propositions, have direct computational meaning in accordance with Brouwer's original program. This fundamental connection with computation has proved enormously influential in computer science, in particular in the theory of programming languages and the foundations of mechanized proof. Homotopy type theory makes essential use of the concept of proof relevance, which is so central to the constructive program, and emphasizes a notion of abstract types that is familiar from the theory of programming languages (*e.g.*, the identity type is itself an abstraction, rather than being encoded in terms of a concrete definition of homotopy). The grand challenge as of this writing is to extend the computational interpretation to the univalence axiom, and therewith to the full hierarchy of h-levels, providing a computational meaning for, say, mappings among higher-dimensional structures such as the spheres and toruses of arbitrary dimension. Recent advances, such as the landmark development of a constructively valid model using cubical sets [Bezem et al. 2014], strongly suggest that such a unification will be achieved in the near future. The potential implications for computer science are only beginning to be explored [Angiuli et al. 2014].

Perhaps the most important application of the unification of classical and constructive mathematics is the possibility of applying systems of mechanized proof verification to broad swaths of classical mathematics that were previously formalizable only via elaborate coding into set theory, and only in systems based on classical logic, which generally lack the benefits resulting from the computational interpretation of constructive systems (*e.g.*, the generation of independently verifiable proof certificates). The direct formalization of everything from quotient sets to cohomology simplifies and streamlines the formalization of even advanced mathematics, and has the potential to eventually make formal verification into a practical tool for the everyday mathematician. Interestingly, this practical development makes the logical foundations of mathematics finally relevant to the actual practice of mathematics, rather than being just a theoretical possibility. The result may be a new "post-Gödel" attitude toward foundations; for when their only interest was theoretical, the phenomenon of incompleteness seemed to lessen the importance of logical foundations in principle. But with the actual practical benefits of formalization (increased rigor and certainty, ease of remote collaboration, accumulation of results), the theoretical incompleteness phenomenon diminishes in importance, and logical foundations can become a useful addition to the toolbox of the working mathematician.[5]

It is a curious fact, made all the more interesting by the above-mentioned developments, that two of the most successful systems for mechanized proof, NuPRL [Constable, *et al.* 1985] and Coq [Coq], are both based on constructive type theory. Why ought that be the case? Homotopy type theory may provide a clue in the importance of proof-relevance, and the associated distinction between property and structure, in both constructive mathematics and homotopy theory. The univalent approach of homotopy type theory exploits the axiomatic freedom provided by constructive mathematics, allowing it to rely far less on elaborate encodings which impede the process of formalization required to admit machine-checked proof. This experience parallels the development of high-level (abstract) programming languages that provide a synthetic concept of computation, rather than one based on low-level machine models such as the Turing machine or Random-Access Machine. Thus we find that whether we are discussing mechanized mathematical proof or verified computer programming, Church's λ-calculus emerges as a central concept. Perhaps this explains why constructive mathematics and mechanized proof are so tightly linked. That they should also be entwined with homotopy theory — one of the most abstract, geometrical, and rarified areas of modern mathematics — is an intriguing and challenging fact inviting further investigation.

REFERENCES

The Agda Proof and Programming System. http://wiki.portal.chalmers.se/agda/pmwiki.php

Carlo Angiuli, Edward Morehouse, Daniel R. Licata, and Robert Harper. 2014. Homotopical patch theory. In *Proceedings of the 19th ACM SIGPLAN international conference on Functional programming, Gothenburg, Sweden, September 1-3, 2014*. 243–256.

The Automath Archive. http://www.win.tue.nl/automath.

S. Awodey, A. Pelayo, and M.A. Warren. 2013. Voevodsky's univalence uxiom in homotopy type theory. *Notices of the Amer. Mathem. Soc.* 60(08) (2013), 1164–1167.

S. Awodey and M.A. Warren. 2009. Homotopy-theoretic models of identity types. *Math. Proc. Camb. Phil. Soc.* 146 (2009), 45–55.

Marc Bezem, Thierry Coquand, and Simon Huber. 2014. A model of type theory in cubical sets. (March 2014). (Unpublished Manuscript).

Errett Bishop and Douglas Bridges. 1985. *Constructive Analysis*. Number 279 in Grundlehren der mathematischen Wissenschaften: A Series of Comprehensive Studies in Mathematics. Springer. (Revision of first edition by Bishop published in 1967.).

A. Church. 1940. A formulation of the simple theory of types. *Journal of Symbolic Logic* 5 (1940), 56–68.

Robert L. Constable, *et al.* 1985. *Implementing Mathematics with the NuPRL Proof Development System*. Prentice Hall.

The Coq Proof Assistant. http://coq.inria.fr.

Nicola Gambino and Richard Garner. 2008. The identity type weak factorisation system. *Theoret. Comput. Sci.* 409, 1 (2008), 94–109. DOI:http://dx.doi.org/10.1016/j.tcs.2008.08.030

A. Grothendieck. 1983. Pursuing stacks. (1983). Unpublished manuscript.

Martin Hofmann and Thomas Streicher. 1994. The Groupoid Model Refutes Uniqueness of Identity Proofs. In *LICS*. 208–212.

M. Hofmann and T. Streicher. 1995. The groupoid model of type theory. In *Twenty-five years of constructive type theory*, G. Sambin and J. Smith (Eds.). Oxford University Press.

The HOL System. http://www.cl.cam.ac.uk/research/hvg/HOL.

The HOL Light Theorem Prover. http://www.cl.cam.ac.uk/~jrh13/hol-light/index.html.

W. A. Howard. 1980. The formulae-as-types notion of construction. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*. Academic Press, 479–491.

[5]See the code section of http://homotopytypetheory.org for links to shared repositories of mechanized mathematics in homotopy type theory using the Coq and Agda systems. Readers are invited to contribute!

C. Kapulkin, P. LeFanu Lumsdaine, and V. Voevodsky. 2012. Univalence in Simplicial Sets. *arXiv* 1203.2553 (2012).

Daniel R. Licata and Guillaume Brunerie. 2013. $\Pi_n(S^n)$ in Homotopy Type Theory. In *Certified Programs and Proofs - Third International Conference, CPP 2013, Melbourne, VIC, Australia, December 11-13, 2013, Proceedings*. 1–16.

Daniel R. Licata and Eric Finster. 2014. Eilenberg-MacLane spaces in homotopy type theory. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*. 66.

Daniel R. Licata and Michael Shulman. 2013. Calculating the Fundamental Group of the Circle in Homotopy Type Theory. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '13)*. IEEE Computer Society, Washington, DC, USA, 223–232.

P. Lumsdaine. 2009. Weak ω-Categories from Intensional Type Theory. In *Typed Lambda Calculi and Applications (LNCS)*, P.-L. Curien (Ed.). Springer, 172–187.

Per Martin-Löf and Giovanni Sambin. 1984. *Intuitionistic type theory*. Vol. 17. Bibliopolis Naples.

The NuPRL Proof Development System. http://nuprl.org.

T. Streicher. 2006. Identity types vs. weak omega-groupoids: some ideas, some problems. (2006). Talk given in Uppsala at the meeting on "Identity Types: Topological and Categorical Structure".

The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. http://homotopytypetheory.org/book, Institute for Advanced Study.

B. van den Berg and R. Garner. 2011. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society* 102, 3 (2011), 370–394.

V. Voevodsky. 2006. A very short note on the homotopy λ-calculus. (2006). Unpublished note.

Verification Column

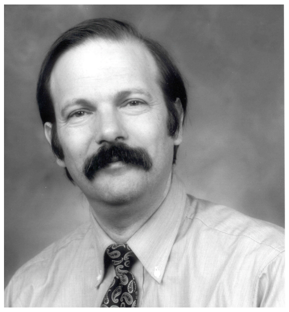
NEHA RUNGTA, SGT Inc./NASA Ames Research Center
neha.s.rungta@nasa.gov

A commonly used definition of verification is that it consists of techniques that formally prove a particular system is correct with respect to certain properties. Given the complexity of modern day systems such verification techniques are not decidable for most systems and often intractable for others. However, verification is much more than proving a system is a correct with respect to a property. In my opinion, verification encompasses a large set of activities associated with establishing confidence about the expected behaviors of the system in some specified environment. Building this confidence is key to having reliable and trustworthy systems. Our modern world overridden with technology includes not only traditional hardware and software systems, but, other emerging technologies, such as web-applications, hand-held mobile devices, wearable technologies such as smart watches, self-driving cars, unmanned aerial vehicles commonly known as drones, and many others. Insufficient verification effort can have varying degrees of impact ranging from a bad user-experience to financial losses, disruption of services, and in some extreme cases loss of human lives. An important challenge in order to push forth the verification effort is to recognize, develop, and better utilize activities that are grounded in logic and other computation models and can enable us to build confidence about the correctness of the system rather than prove its correctness.

In this column I hope to present innovations in various verification techniques based on model checking, symbolic evaluation, theorem proving, and others. I would like to have articles on activities that may not typically be labeled as verification but are widely used in industry and government organizations to build confidence about the correctness of the system. Another important aspect I hope to bring forth in this column is the application of verification techniques in other domains such biological systems, multi-agent systems, financial markets, and others. If you would like to contribute to this column please write to me. I very much look forward to hearing from our readers on ideas, suggestions, and feedback to help shape this column.

SURVEY

Theory in Practice for System Design and Verification



Rajeev Alur
Univ. of Pennsylvania

Thomas A. Henzinger
IST Austria

Moshe Y. Vardi
Rice University

1. INTRODUCTION

Methodology and tools for assisting developers in building high-confidence hardware and software at a reasonable cost has been one of the central themes in computer science since its inception. The formal methods research on this problem has focused on two complimentary goals: to provide mathematical abstractions to manage the complexity of the design and to develop analysis tools to check that the implementation works correctly as intended. Achieving these goals has proved to be extremely challenging for two reasons. First, the scale and complexity of systems being designed remains a moving target as computers have transformed from special-purpose and stand-alone number-crunching processors to networked devices interacting with the physical world. Second, once formalized, the computational problem of verifying that a system meets its specification is undecidable in the general case and has intractable complexity even in special cases.

Advances in theoretical foundations for system design have contributed in addressing these challenges partially by identifying logics and automata for specification and modeling, and by developing efficient data structures and algorithms for analysis. In Section 2, we discuss real-world impact of these advances in design automation for hardware, software, and embedded control systems. Critical to each success story was an initial demonstration of a compelling match between the capability of a research prototype and an industrial need, followed by sustained research on improving the scalability of the tools. It is worth noting that research on this topic has been recognized by multiple notable awards in the last twenty years. These include: A.M. Turing Award to Pnueli for temporal logic in 1996 and to Clarke, Emerson, and Sifakis for model checking in 2007, Paris Kanellakis Theory and Practice Award to Bryant, Clarke, Emerson, and McMillan for symbolic model checking in 1998, to Holzmannn, Kurshan, Vardi, and Wolper for automata-theoretic model checking in 2005, and to Brayton for logic synthesis in 2006, and ACM Software System Award to the model checker SPIN in 2001, to the Boyer-Moore theorem prover in 2005, and to Statemate in 2007.

2. SUCCESS STORIES

In this section, we describe a few illustrative examples of how ideas originating in academic research and rooted in theoretical foundations have matured into tools and methodologies used in industry and other communities.

2.1. Constraint Solvers

The propositional satisfiability problem (SAT) is known to be NP-complete and understanding its structure has been a central theme in complexity theory for the past forty years. A parallel thread of research in the verification community has been the development of efficient solvers for SAT. Modern SAT solvers are capable of solving instances with many thousands of variables due to sustained innovations in core algorithms, data structures, decision heuristics, and performance tuning by exploiting the architecture of contemporary processors [12].

A more challenging form of constraint satisfaction problem is to determine the truth of a logical formula built from propositional as well as other types of variables. For example, in *linear real arithmetic*, the input formula consists of propositional and real variables, logical connectives, and linear arithmetic operations. Theoretical understanding of general ways of combining distinct decision procedures (in case of linear real arithmetic, integrating the solver for propositional satisfiability with the solver for checking consistency of conjunctions of linear inequalities) has paved the way for the so-called SMT (Satisfiability Modulo Theories) solvers that can now solve constraint satisfaction problems over a rich set of types.

Contemporary analysis and verification tools vary widely in terms of source languages, verification methodology, and the degree of automation, but they all rely on repeatedly invoking a SAT or an SMT solver for core computational tasks such as checking validity of a verification condition and automatically generating a candidate invariant (see [3] for an introduction to decision procedures and program verification). Due to their impressive scalability and maturity, SAT and SMT solvers are also used in many other contexts such as planning and optimization (see [7] and `smt-lib.org`).

2.2. Hardware Design Automation

The key to managing the complexity of modern VLSI circuits has been the introduction of industry-standard abstractions such as RTL (Register Transfer Level) and Hardware Description Languages (such as Verilog and VHDL). The challenge in electronic design automation then is to automatically map descriptions in such high-level abstractions to a low-level circuit for fabrication while ensuring semantic correctness and satisfying performance objectives related to area, power, and timing.

The significant advances in electronic design automation have originated in academic tools. Prominent examples include: (1) SPICE for accurate and fast simulation of large-scale circuits, (2) SIS for translating state machines to optimized netlists, and (3) Espresso for minimizing the number of gates in a circuit. These tools rely on a variety of algorithmic techniques such as algebraic rewriting, heuristics for multiobjective optimization, efficient techniques for simulation of differential equations, and equivalence checking for finite-state machines.

The work on circuit simulation and logic synthesis in 1980s resulted in founding of Cadence and Synopsis, which are still the two leading companies in EDA (Electronic Design Automation). More significantly, EDA tools are used universally within the semiconductor industry, and the contemporary computing infrastructure would not exist without these advances in hardware design automation (see [13] for a survey of the synergy between the academic research and EDA industry).

2.3. Temporal Logic Model Checking

Temporal logic offers a natural way of formally expressing requirements concerning safety (avoidance of undesirable states) and liveness (eventual satisfaction of goals) properties of reactive systems—systems that interact with their environment via inputs and outputs in an ongoing manner. Model checking, introduced in early 1980s, is the problem of algorithmically checking that a finite-state abstraction of a system satisfies its temporal-logic specification.

Model checking has been a topic of extensive theoretical research for the past thirty years. Key theoretical advances include symbolic algorithms based on the data structure of BDDs (binary decision diagrams), an understanding of the expressiveness and complexity of different variants of temporal logics, automata over infinite strings with applications to decision procedures for temporal logics, reduction strategies for limiting the search through the state-space of concurrent state-machines, and techniques for automatic abstraction and refinement. Early research prototypes such as Cospan, Murphi, SMV, and SPIN demonstrated how these theoretical ideas can lead to efficient tools, and were successful in finding hard-to-find logical bugs in multiprocessor coordination protocols and distributed algorithms.

In hardware design, it is now a common practice to augment the design with assertions or monitors as correctness specifications. The specification language PSL (IEEE 1850 Standard Property Specification Language) is rooted in temporal logic, and supported by commercial simulation tools (see also the emerging standard SVA (System Verilog Assertions). Companies such as Intel and Motorola have in-house verification groups that routinely use model checkers to debug challenging designs such as cache coherence protocols and pipelined microprocessor architectures. There are also new companies focused primarily on tools and consulting for formal verification such as Jasper (`jasper-da.com`) and Oski (`oskitechnology.com`).

We refer the reader to [5] for a technical introduction to model checking, and the 2009 ACM Turing Award lecture for an overview of its impact [4].

2.4. Software Analysis

The software verification problem is to check whether a program meets a correctness specification. While this problem is undecidable, a number of algorithmic analysis techniques have been developed to solve this problem approximately (in the sense that the tool is not guaranteed to give the correct answer on every input instance).

In *static analysis*, a set of facts of a particular pattern relating program variables are derived at every program location by propagation of constraints along the control-flow graph of the program. Theoretical research on abstract interpretation, constraint simplification, and algorithms for inter-procedural analysis has contributed to scalable tools such as Astrée (used by Airbus to check absence of floating point errors in avionics software) [6] and PreFix/PreFast (used by Microsoft to ensure absence of buffer overflow errors in Windows operating system). Companies such as Grammatech (`grammatech.com`) and Coverity (`coverity.com`) originated from academic research, and have developed industrial-strength static analysis tools [2].

In *software model checking*, automatic abstraction and static analysis are used to derive a finite-state abstraction of a program, which is then subjected to exhaustive state-space exploration using symbolic techniques developed for model checking. There have been prominent successes of this approach recently: the SDV (Static Device Verifier) tool is able to certify conformance of code for device drivers to the Windows API usage rules [1]; the C code running on NASA's robotic vehicle Curiosity that successfully landed on Mars in August 2012 was extensively debugged using formal analysis tools [9]; and the F-SOFT model checker is used in NEC on a regular basis to find bugs in millions of lines of C/C++ code [10].

The objective of *software testing* is to select a *representative* set of inputs for executing the code, and of *dynamic analysis* is to infer as much information as possible about the program behavior based on observed executions. For these classical software engineering problems, new algorithmic techniques have been recently developed based on the use of constraint solvers and symbolic execution. Recently, the testing tool SAGE based on symbolic execution was credited to have found roughly one third of all the security vulnerabilities during the development of Microsoft's Windows 7 software [8]. A promising new direction for software analysis combines constraint-based static approach with the execution-based dynamic approach.

2.5. Formal Models for Cyber-Physical Systems

Cyber-physical systems are networked computing devices interacting with the physical world. Model-based design is emerging as a promising approach for developing this new class of complex systems in a principled manner, and the foundations of this methodology lie in cross-fertilization of ideas from mathematical modeling and algorithmic analysis.

Examples of modeling frameworks include Statecharts for visual and structured modeling of reactive systems, Esterel for simplifying design abstractions based on synchrony hypothesis, timed automata for integrating timing constraints in state machines, hybrid automata for integrating discrete behavior with continuous-time models of dynamical systems, and Ptolemy for unifying heterogenous models of interaction. Examples of analysis techniques include algorithms for estimation of worst-case execution time and scheduling of computational resources, synthesis of code from models subject to resource constraints, transformation and composition of models, verification algorithms based on computing finite-state abstractions of timed and hybrid systems, symbolic analysis of dynamical systems, and metric-based notions of abstraction and refinement for hybrid systems. See [11] for an introduction to model-based design and analysis of cyber-physical systems.

Model-based design and analysis is slowly being adapted by industry for design of embedded control software in domains such as avionics, automotive, and medical devices. Mathworks, the leading tool vendor in this sector (see `mathworks.com`), now supports modeling using notations such as Statecharts and hybrid automata, schedulability analysis, and test-case generation using symbolic techniques (Simulink Design Verifier). Companies such as TTTech (`tttech.com`), Reactive Systems (`reactive-systems.com`), and Uppaal (`uppaal.com`) originated from academic research, and market tools for formal modeling and analysis. It is also worth noting the adoption of concepts and tools from formal methods in disciplines such as control theory and systems engineering, both in research and undergraduate education.

3. FUTURE DIRECTIONS

Realizing the full potential of emerging computing platforms requires that advances in processing and communication technology are matched by advances in tools for designing complex software systems and ensuring their safe and reliable operation. Thus, the research goal of formal approaches to system design and analysis is as relevant and as challenging today as it was fifty years ago, and the success stories discussed in the previous section suggest that theory can be effective in this pursuit. Another lesson is that the path to a successful technology transfer in this domain has been typically long: small steps in advancing the scalability of tools collectively contribute towards impressive results over decades. This calls for continued research in core areas of formal methods such as identification of analyzable design abstractions, analysis algorithms, and scalability of tools.

We conclude this report by listing some new avenues for research.

— **Synthesis:** With maturing of verification technology that can check the conformance of an implementation to its specification, it is natural to focus on synthesis—automatic derivation of implementation from specifications, both to improve programmer productivity and to integrate verification with design so that bugs are found in early stages. There is a growing research on this topic, for instance, on synthesizing finite-state controllers from temporal logic specifications, on automatic completion of partial programs based on user-supplied assertions, and synthesis of programs from examples by exploiting the domain-specific knowledge. New theoretical approaches that combine logical deduction with machine learning can offer scalable computational solutions for synthesis.
— **Concurrency:** While the methodology for design and verification of sequential programs is well understood, despite many proposals for programming languages and verification techniques for concurrent programs, developing concurrent systems remains a difficult and error-prone task. Emerging multiprocessor and multicore architectures offer enormous computational power, but exploiting this parallelism efficiently and correctly is challenging due to complex memory models for shared data. The emergence of data-centers and cloud computing again offers exciting opportunities for concurrent computation, but need new programming abstractions to ensure data consistency and fault tolerance. Research in formal methods can potentially have significant impact on programming abstractions and languages for concurrent systems.
— **Probabilistic and Quantitative Models:** Traditionally, models and techniques used for establishing correctness and for evaluating performance have been disjoint. A promising new direction in formal methods research these days is the development of probabilistic models, with associated tools for quantitative evaluation of system performance along with correctness. Concurrent software, distributed protocols, and resource allocation for cloud computing, are potential application domains for such work.
— **Beyond Worst-Case Complexity:** Classical computation theory focuses on establishing the worst-case complexity of problems. For verification problems, such estimates always indicate intractability, and yet, some of the modern SAT and SMT solvers work well on many instances in this context. Theoretical tools for estimating complexity on *real-world* instances of problems thus can provide useful insights into the structure of these problems.

REFERENCES

T. Ball, V. Levin, and S. K. Rajamani. A decade of software model checking with SLAM. *Commun. ACM*, 54(7):68–76, 2011.

A. Bessey, K. Block, B. Chelf, A. Chou, B. Fulton, S. Hallem, C.-H. Gros, A. Kamsky, S. McPeak, and D. R. Engler. A few billion lines of code later: using static analysis to find bugs in the real world. *Commun. ACM*, 53(2):66–75, 2010.

A. R. Bradley and Z. Manna. *The calculus of computation - decision procedures with applications to verification*. Springer, 2007.

E. M. Clarke, E. A. Emerson, and J. Sifakis. Model checking: algorithmic verification and debugging. *Commun. ACM*, 52(11):74–84, 2009.

E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, 2000.

P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, and X. Rival. Why does Astrée scale up? *Formal Methods in System Design*, 35(3):229–264, 2009.

L. de Moura and N. Bjørner. Satisfiability Modulo Theories: introduction and applications. *Commun. ACM*, 54(9):69–77, 2011.

P. Godefroid, M. Y. Levin, and D. A. Molnar. SAGE: whitebox fuzzing for security testing. *Commun. ACM*, 55(3):40–44, 2012.

G. J. Holzmann. Landing a spacecraft on Mars. *IEEE Software*, 30(2):83–86, 2013.

F. Ivancic, G. Balakrishnan, A. Gupta, S. Sankaranarayanan, N. Maeda, H. Tokuoka, T. Imoto, and Y. Miyazaki. DC2: A framework for scalable, scope-bounded software verification. In *Proc. 26th IEEE/ACM Intl. Conf. on Automated Software Engineering*, pages 133–142, 2011.

E. A. Lee and S. A. Seshia. *Introduction to Embedded Systems, A Cyber-Physical Systems Approach*. 2011.

S. Malik and L. Zhang. Boolean satisfiability: from theoretical hardness to practical success. *Commun. ACM*, 52(8):76–82, 2009.

A. L. Sangiovanni-Vincentelli. The tides of EDA. *IEEE Design & Test of Computers*, 20(6):59–75, 2003.

CALLS

COMPUTABILITY IN EUROPE 2015: Evolving Computability

Call for Papers
June 29 - July 3, 2015
Bucharest, Romania
`http://fmi.unibuc.ro/CiE2015/`

GENERAL. CiE 2015 is the 11-th conference organized by CiE (Computability in Europe), a European association of mathematicians, logicians, computer scientists, philosophers, physicists and others interested in new developments in computability and their underlying significance for the real world. Evolution of the universe, and us within it, invite a parallel evolution in understanding. The CiE agenda - fundamental and engaged - targets the extracting and developing of computational models basic to current challenges. From the origins of life, to the understanding of human mentality, to the characterizing of quantum randomness - computability theoretic questions arise in many guises. The CiE community, this coming year meeting for the first time in Bucharest, carries forward the search for coherence, depth and new thinking across this rich and vital field of research.

SUBMISSIONS. We are looking for fundamental and theoretical submissions. In line with other conferences in this series, CiE 2015 has a broad scope and provides a forum for the discussion of theoretical and practical issues in Computability with an emphasis on new paradigms of computation and the development of their mathematical theory. We particularly invite papers that build bridges between different parts of the research community.

DATES.

— Submission Deadline for LNCS: 11 January 2015
— Notification of authors: 9 March 2015
— Deadline for final revisions: 6 April 2015

For submission instructions consult
`http://fmi.unibuc.ro/CiE2015/submission.html`

30TH ANNUAL ACM/IEEE SYMPOSIUM ON LOGIC IN COMPUTER SCIENCE (LICS 2015)

Call for Papers
July 6-10, 2015
Kyoto, Japan
`http://lics.rwth-aachen.de/lics15/`
(colocated with ICALP 2015)

AIMS. The LICS Symposium is an annual international forum on theoretical and practical topics in computer science that relate to logic, broadly construed. We invite submissions on topics that fit under that rubric. Suggested, but not exclusive, topics of interest include: automata theory, automated deduction, categorical models and logics, concurrency and distributed computation, constraint programming, constructive mathematics, database theory, decision procedures, description logics, domain theory, finite model theory, formal aspects of program analysis, formal methods, foundations of computability, higher-order logic, lambda and combinatory calculi, linear logic, logic in artificial intelligence, logic programming, logical aspects of bioinformatics, logical aspects of computational complexity, logical aspects of quantum computation, logical frameworks, logics of programs, modal and temporal logics, model checking, probabilistic systems, process calculi, programming language semantics, proof theory, real-time systems, reasoning about security and privacy, rewriting, type systems and type theory, and verification.

INSTRUCTIONS. Authors are required to submit a paper title and a short abstract of about 100 words in advance of submitting the extended abstract of the paper. The exact deadline time on these dates is given by anywhere on earth (AoE).

— Title and Short Abstracts Due: January 12, 2015
— Extended Abstracts Due: January 19, 2015
— Author Feedback/Rebuttal Period: March 12-16, 2015
— Author Notification: March 30, 2015
— Final Versions Due for Proceedings: April 27, 2015

Deadlines are firm; late submissions will not be considered. All submissions will be electronic via https://www.easychair.org/conferences/?conf=lics2015. Every extended abstract must be submitted in the IEEE Proceedings 2-column 10pt format and may not be longer than 12 pages, including references. LaTeX style files are available from the website.

CONFERENCE CHAIR. Masahito Hasegawa, RIMS, Kyoto U.

PROGRAM COMMITTEE CHAIR. Catuscia Palamidessi, INRIA & E. Polytechnique

WORKSHOP CHAIR. Patricia Bouyer-Decitre, CNRS & ENS Cachan

GENERAL CHAIR. Luke Ong, U. Oxford

SHORT PRESENTATIONS. A session of short presentations, intended for descriptions of student research, works in progress, and other brief communications, is planned. These abstracts will not be published. Dates and guidelines will be posted on the conference website.

KLEENE AWARD FOR BEST STUDENT PAPER. An award in honor of the late Stephen C. Kleene will be given for the best student paper(s), as judged by the program committee.

SPECIAL ISSUES. Full versions of up to three accepted papers, to be selected by the program committee, will be invited for submission to the Journal of the ACM. Additional selected papers will be invited to a special issue of Logical Methods in Computer Science.

SPONSORSHIP. The symposium is sponsored by ACM SIGLOG and the IEEE Technical Committee on Mathematical Foundations of Computing, in cooperation with the Association for Symbolic Logic and the European Association for Theoretical Computer Science.

2ND INTERNATIONAL COMPETITION ON RUNTIME VERIFICATION (CRV 2015)

Call for Participation
September 22-25, 2015
Vienna, Austria
`http://rv2015.conf.tuwien.ac.at`

AIMS. CRV-2015 will draw attention to the invaluable effort of software developers and researchers who contribute in this field by providing the community with new or updated tools, libraries and frameworks for the instrumentation and runtime verification of software. The main goal of CRV 2015 is to compare tools for runtime verification. We invite and encourage the participation with benchmarks and tools for the competition.

TRACKS. The competition will consist of three main tracks based on the input language used:

— Track on monitoring Java programs (online monitoring).
— Track on monitoring C programs (online monitoring).
— Track on monitoring of traces (offline monitoring).

The competition will follow three phases:

— Benchmarks/Specification collection phase - the participants are invited to submit their benchmarks (C or Java programs and/or traces). The organizers will collect them in a common repository (publicly available). The participants will then train their tools using the shared benchmarks.
— Monitor collection phase - the participants are invited to submit their monitors. The participants with the tools/monitors that meet the qualification requirements will be qualified for the evaluation phase.
— Evaluation phase - the qualified tools will be evaluated on the submitted benchmarks and they will be ranked using different criteria (i.e., memory utilization, CPU utilization, ...). The final results will be presented at the RV 2015 conference.

ENQUIRIES. Please direct any enquiries to the competition co-organizers (crv15.chairs@imag.fr)

— Ylies Falcone (Universite Joseph Fourier, France)
— Dejan Nickovic (AIT Austrian Institute of Technology GmbH, Austria)
— Giles Reger (University of Manchester, UK)
— Daniel Thoma (University of Luebeck, Germany)

CRV-2015 Jury. The CSRV Jury will include a representative for each participating team and the competition chairs. The Jury will be consulted at each stage of the competition to ensure that the rules set by the competition chairs are fair and reasonable.

IMPORTANT DATES.

— January 15, 2015: Declaration of intent (email: crv15.chairs@imag.fr)
— March 1, 2015 Submission deadline for benchmark programs and the properties to be monitored
— March 15, 2015 Tool training starts by participants
— May 15, 2015 Monitor submission
— June 15, 2015 Notifications
— At RV 2015 Presentation of results

4TH ANNUAL MEETING OF THE REASONING CLUB MEETING

`http://www.kent.ac.uk/secl/philosophy/jw/reasoning/club/`
`http://www.maths.manchester.ac.uk/news-and-events/events/fourth-reasoning-club-conf/`
School of Mathematics, Manchester University
March 30th-31st, 2015

TALKS. The Keynote Speakers are Richard Booth (Luxembourg), Leon Horsten (Bristol), Federico Luzzi (Aberdeen) and Sara Uckelman (Durham). In addition it is planned to have ten 40 minute contributed talks by Ph.D.students and early Postdocs, for whom grants will be available to cover the cost of accommodation and subsistence.

SUBMISSIONS. If you would like to give a talk please attach a short abstract when returning your registration form. The deadline for abstracts is 15th January 2015. Please see http://www.maths.manchester.ac.uk/news-and-events/events/fourth-reasoning-club-conf/ for information on abstract submission and (free) registration.

ORGANIZERS. Jeff Paris & Alena Vencovska

27TH INTERNATIONAL CONFERENCE ON COMPUTER-AIDED VERIFICATION (CAV 2015)

Call for Papers and CAV Award
July 18-24 2015
San Francisco, California
`http://i-cav.org/2015/`

AIMS AND SCOPE. CAV 2015 is the 27th in a series dedicated to the advancement of the theory and practice of computer-aided formal analysis methods for hardware and software systems. CAV considers it vital to continue spurring advances in hardware and software verification while expanding to new domains such as biological systems and computer security. The conference covers the spectrum from theoretical results to concrete applications, with an emphasis on practical verification tools and the algorithms and techniques that are needed for their implementation. The proceedings of the conference will be published in the Springer LNCS series. A selection of papers will be invited to a special issue of Formal Methods in System Design and the Journal of the ACM.

TOPICS. Topics of interest include but are not limited to:

— Algorithms and tools for verifying models and implementations
— Hardware verification techniques
— Deductive, compositional, and abstraction techniques for verification
— Program analysis and software verification
— Verification methods for parallel and concurrent hardware/software systems
— Testing and run-time analysis based on verification technology
— Applications and case studies in verification
— Decision procedures and solvers for verification
— Mathematical and logical foundations of practical verification tools
— Verification in industrial practice
— Algorithms and tools for system synthesis
— Hybrid systems and embedded systems verification
— Verification techniques for security
— Formal models and methods for biological systems

DEADLINES.

— Abstract submission: January 30, 2015
— Paper submission (firm): February 6, 2015
— Author feedback/rebuttal period: March 23-26, 2015
— Notification of acceptance/rejection: April 17, 2015
— Final version due: May 1, 2015

CALL FOR CAV AWARD NOMINATIONS. The CAV award is given annually at the CAV conference for fundamental contributions to the field of Computer-Aided Verification. The award comes with a cash prize of $10,000 shared equally among recipients. Nominations should be submitted by e-mail to a member of the CAV Award committee.

PC CHAIRS. Daniel Kroening, University of Oxford, UK. Corina Pasareanu, Carnegie Mellon Silicon Valley/NASA Ames, USA.

WORKSHOP CHAIR. Dirk Beyer, University of Passau, Germany

LOCAL ORGANIZATION CHAIR. Temesghen Kahsai, Carnegie Mellon Silicon Valley/NASA Ames, USA.

CAV AWARD COMMITTEE. Moshe Vardi (Chair), Rice University; Ahmed Bouajjani, Univ. Paris Diderot (Paris 7); Tom Ball , Microsoft Research; Kim G. Larsen, Aalborg University

TTL 2015 - 4th INT'L CONF ON TOOLS FOR TEACHING LOGIC

Call for Papers
June 9-12, 2015, Rennes, France
`http://ttl2015.irisa.fr/`

TOPICS. Topics that fit the interests of Tools for Teaching Logic include (but are not limited to): teaching logic in sciences and humanities; teaching logic at different levels of instruction (secondary education, university level, and postgraduate); didactic software; facing some difficulties concerning what to teach; international postgraduate programs; resources and challenges for e-Learning Logic; teaching Argumentation Theory, Critical Thinking and Informal Logic; teaching specific topics, such as Modal Logic, Algebraic Logic, Knowledge Representation, Model Theory, Philosophy of Logic, and others; dissemination of logic courseware and logic textbooks; teaching Logic Thinking.

IMPORTANT DATES.

— Paper submission: 30 Jan 2015;
— Notification: 1 Mar 2015;
— Final camera-ready due: 29 Mar 2015

CONTINUITY, COMPUTABILITY, CONSTRUCTIVITY: FROM LOGIC TO ALGORITHMS 2014 POSTPROCEEDINGS

Call for Submissions

GENERAL. After a further year of successful work in the EU-IRSES project COMPUTAL and an excellent workshop in Ljubljana (Slovenia) in September this year, we are planning to publish a collection of papers dedicated to the meeting and the project in the JOURNAL OF LOGIC AND ANALYSIS. The issue should reflect progress made in Computable Analysis and related areas, not only work in the project. Submissions are welcome from all scientists and should be on topics in the spectrum from logic to algorithms including, but not limited to,

— Computable analysis
— Complexity of real number computations
— Computing with continuous data
— Domain theory and analysis

— Randomness and computable measure theory
— Models of computation with real numbers
— Realizability theory and analysis
— Reverse analysis
— Exact real number computation
— Program extraction in analysis

EDITORS. Andrej Bauer (Ljubljana, Slovenia) Ulrich Berger (Swansea, UK) Willem Fouche (Pretoria, South Africa) Dieter Spreen (Siegen, Germany & Pretoria, South Africa) Hideki Tsuiki (Kyoto, Japan) Martin Ziegler (Darmstadt, Germany)

DEADLINE FOR SUBMISSION. 31 January 2015 Please prepare your manuscript using the JLA class file jlogana.cls and the bibliography style file jloganal.bst which can be downloaded from http://logicandanalysis.org/latex/latexinstructions.html. For submissions go to the JLA webpage
http://logicandanalysis.org/index.php/jla/information/authors
and follow the instructions given there. In addition, important, When submitting to JLA, write CCC2014 POSTPROCEEDINGS in the Comments-for-the-Editor box. Send a separate copy of your submission to spreen@math.uni-siegen.de. And, if appropriate, identify one or more members of the Issue Editors mentioned above whose interests are closest to the subject matter of the paper in the mail.

23RD GOEDEL PRIZE

Call for Nominations
`http://www.sigact.org/Prizes/Godel/`

CONTEXT. The Goedel Prize for outstanding papers in the area of theoretical computer science is sponsored jointly by the European Association for Theoretical Computer Science (EATCS) and the Association for Computing Machinery, Special Interest Group on Algorithms and Computation Theory (ACM SIGACT). The award is presented annually, with the presentation taking place alternately at the International Colloquium on Automata, Languages, and Programming (ICALP) and the ACM Symposium on Theory of Computing (STOC). The 23rd Goedel Prize will be awarded at the 47th ACM Symposium on Theory of Computing, June, 2015 in Portland, Oregon. The Prize is named in honor of Kurt Goedel in recognition of his major contributions to mathematical logic and of his interest, discovered in a letter he wrote to John von Neumann shortly before von Neumann's death, in what has become the famous "P versus NP"question. The Prize includes an award of USD 5000.

AWARD COMMITEE. The winner of the Prize is selected by a committee of six members. The EATCS President and the SIGACT Chair each appoint three members to the committee, to serve staggered three-year terms. The committee is chaired alternately by representatives of EATCS and SIGACT. The 2015 Award Committee consists of Krzysztof Apt (CWI Amsterdam), Kurt Mehlhorn (Max Planck Institute), Joseph Mitchell (State University of New York at Stony Brook), Andrew Pitts (University of Cambridge), Madhu Sudan (Microsoft) and Eva Tardos (Cornell University).

ELIGIBILITY. The rules for the 2015 Prize are given below and they supersede any different interpretation of the generic rule to be found on websites of both SIGACT and EATCS. Any research paper or series of papers by a single author or by a team of authors is deemed eligible if (i) the paper was published in a recognized refereed journal no later than December 31, 2014; (ii) the main results were not published (in either preliminary or final form) in a journal or conference proceedings before January 1st,

2002. The research work nominated for the award should be in the area of theoretical computer science. The term ‘theoretical computer science’ is meant to encompass, but is not restricted to, research areas covered by ICALP and STOC. Nominations are encouraged from the broadest spectrum of the theoretical computer science community so as to ensure that potential award winning papers are not overlooked. The Award Committee shall have the ultimate authority to decide whether a particular paper is eligible for the Prize.

NOMINATIONS. Nominations for the award should be submitted by email to the Award Committee Chair Eva Tardos: eva.tardos@cornell.edu. Please make sure that the Subject line of all nominations and related messages begin with Goedel Prize 2015. To be considered, nominations for the 2015 Prize must be received by January 31, 2015.

MORE DETAILS. http://www.sigact.org/Prizes/Godel/

ACM SIGPLAN/SIGBED Conference on Languages, Compilers, Tools and Theory for Embedded Systems (LCTES 2015)

Call for Papers
June 18-19, 2015
Portland, Oregon
(part of the Federated Computing Research Conference 2015)
`http://lctes2015.lctes.org`

GENERAL. LCTES provides a link between the programming languages and embedded systems engineering communities. Researchers and developers in these areas are addressing many similar problems, but with different backgrounds and approaches. LCTES is intended to expose researchers and developers from either area to relevant work and interesting problems in the other area and provide a forum where they can interact.

SUBMISSIONS. LCTES 2015 solicits papers presenting original work on programming languages, compilers, tools, theory, and architectures that help in overcoming these challenges. Research papers on innovative techniques are welcome, as well as experience papers on insights obtained by experimenting with real-world systems and applications.

IMPORTANT DATES.

— Submission deadline: Feb. 15
— Notifications by: Apr. 1
— Camera-ready deadline: Apr. 11

SPECIAL ISSUE. A few of the best submissions to LCTES 2015 are planned to be invited for submission, with some revisions, to a special issue of the ACM Transactions on Embedded Computing Systems (TECS). The official publication date is the date the proceedings are made available in the ACM Digital Library. This date may be up to two weeks prior to the first day of your conference.

ORGANIZATION

— General Chair: Sam H. Noh, Hongik University, Republic of Korea
— Program Chairs: Sebastian Fischmesiter, University of Waterloo, Canada Jason Xue, City University of Hong Kong, China

CONFERENCE ON INTELLIGENT COMPUTER MATHEMATICS (CICM 2015)

Call for Papers
13-17 July 2015
Washington DC, USA
`http://cicm-conference.org/2015/cicm.php`

AIMS. Digital and computational solutions are becoming the prevalent means for the generation, communication, processing, storage and curation of mathematical information. Separate communities have developed to investigate and build computer based systems for computer algebra, automated deduction, and mathematical publishing as well as novel user interfaces. While all of these systems excel in their own right, their integration can lead to synergies offering significant added value. The Conference on Intelligent Computer Mathematics (CICM) offers a venue for discussing and developing solutions to the great challenges posed by the integration of these diverse areas.

HISTORY. CICM has been held annually as a joint meeting since 2008, co-locating related conferences and workshops to advance work in these subjects. Previous meetings have been held in Birmingham (UK 2008), Grand Bend (Canada 2009), Paris (France 2010), Bertinoro (Italy 2011), Bremen (Germany 2012), Bath (UK 2013), and Coimbra (Portugal 2014).

TRACKS.

— Calculemus (Symbolic Computation and Mechanised Reasoning), Chair: Jacques Carette
— DML (Digital Mathematical Libraries), Chair: Volker Sorge
— MKM (Mathematical Knowledge Management), Chair: Cezary Kaliszyk
— Systems and Data Chair: Florian Rabe

ORGANIZATION. Publicity chair is Serge Autexier. The local arrangements will be coordinated by the Local Arrangements Chairs, Bruce R. Miller (National Institute of Standards and Technology, USA) and Abdou Youssef (The George Washington University, Washington, D.C.), and the overall programme will be organized by the General Programme Chair, Manfred Kerber (U. Birmingham, UK). As in previous years, it is anticipated that there will be a number co-located workshops, including one to mentor doctoral students giving presentations. We also solicit for project descriptions and work-in-progress papers.

IMPORTANT DATES.

— Conference submissions:
  — Abstract submission deadline: 16 February 2015
  — Submission deadline: 23 February 2015
  — Reviews sent to authors: 6 April 2015
  — Rebuttals due: 9 April 2015
  — Notification of acceptance: 13 April 2015
  — Camera ready copies due: 27 April 2015
  — Conference: 13-17 July 2015
— Work-in-progress and Doctoral Programme submissions:
  — Submission deadline: (Doctoral: Abstract+CV) 4 May 2015
  — Notification of acceptance: 25 May 2015
  — Camera ready copies due: 1 June 2015

42ND INTERNATIONAL COLLOQUIUM ON AUTOMATA, LANGUAGES, AND PROGRAMMING (ICALP 2015)

Call for Papers
July 6-10, 2015
Kyoto, Japan
`http://www.kurims.kyoto-u.ac.jp/icalp2015/`

GENERAL. ICALP 2015 will co-locate with LICS 2015, the 30th ACM/IEEE Symposium on Logic in Computer Science. The ICALP 2015 conference chair is Kazuo Iwama (Kyoto University). ICALP is the main conference and annual meeting of the European Association for Theoretical Computer Science (EATCS). As usual, the main conference will be preceded and/or followed by a series of workshops.

IMPORTANT DATES.

— Submission deadline: Tuesday, 17 February 2015, 23:59 PST (Pacific Standard Time, UTC-8)
— Author notification: 15 April 2015
— Final manuscript due: 30 April 2015

Deadlines are firm; late submissions will not be considered.

PROCEEDINGS. ICALP proceedings are published in the Springer-Verlag ARCoSS (Advanced Research in Computing and Software Science) subseries of LNCS (Lecture Notes in Computer Science).

INVITED SPEAKERS. Ken Kawarabayashi, NII, Japan; Valerie King, University of Victoria, Canada; Thomas Moscibroda, MSR Asia, China; Anca Muscholl, Universite Bordeaux, France (Joint with LICS); Peter O'Hearn, Facebook, UK (Joint with LICS)

INVITED TUTORIAL SPEAKERS (JOINT WITH LICS). Piotr Indyk, MIT, USA; Andrew Pitts, University of Cambridge, UK; Geoffrey Smith, Florida International University, USA

MASTERCLASS SPEAKER. Ryuhei Uehara, JAIST, Japan

TOPICS. Papers presenting original research on all aspects of theoretical computer science are sought.

— Track A: Algorithms, Complexity and Games
— Track B: Logic, Semantics, Automata and Theory of Programming
— Track C: Foundations of Networked Computation: Models, Algorithms and Information Management

SUBMISSION GUIDELINES. Authors are invited to submit an extended abstract of no more than 12 pages, including references, in LNCS style presenting original research on the theory of Computer Science. All submissions will be electronic via the EasyChair page for the conference, with three tracks (A, B and C): https://easychair.org/conferences/?conf=icalp2015 Submissions should be made to the appropriate track of the conference. No prior publication or simultaneous submission to other publication outlets (either a conference or a journal) is allowed.

BEST PAPER AWARDS. As in previous editions of ICALP, there will be best paper and best student paper awards for each track of the conference. In order to be eligible for a best student paper award, a paper should be authored only by students and should be marked as such upon submission.

CHAIRS.

— Track A: Bettina Speckmann, TU Eindhoven, The Netherlands
— Track B: Naoki Kobayashi, The University of Tokyo, Japan
— Track C: Magnus M. Halldorsson, Reykjavik Univ, Iceland

JOURNAL OF LOGICAL AND ALGEBRAIC METHODS IN PROGRAMMING

Call for Papers
Special Issue on Automated Verification of Programs and Web Systems

SPECIAL ISSUE. This special issue of the Journal of Logical and Algebraic Methods in Programming (JLAMP) is devoted to the themes of the WWV and VPT workshop series on Automated Specification and Verification of Web Systems (WWV) and on Verification and Program Transformation (VPT). This is however an open call for papers. Both participants of the most recent editions of the WWV and VPT workshop series and others working on the themes of this special issue are hereby invited to submit a paper.

IMPORTANT DATES.

— Abstract submission: 25 February 2015
— Full paper submission: 15 March 2015
— Acceptance notification: 30 June 2015
— Final manuscript due: 25 July 2015
— Expected publication: Fall 2015

AIMS AND SCOPE. This special issue provides a forum for researchers working in the areas of verification, program transformation, software engineering, rule-based programming, formal methods, and Web-oriented research, to submit their papers on the Automated Verification of Programs and Web Systems. We solicit original papers on topics of either theoretical or applied interest.

SUBMISSION. We expect original articles (typically 20-30 pages; submission of larger papers will be evaluated depending on editorial constraints) that present high-quality contributions, which have not previously been published and that are also not simultaneously submitted for publication elsewhere. Each paper will undergo a thorough evaluation by at least three reviewers. All contributions must be written in English, must be submitted in PDF format and must comply with JLAMP's author instructions (the manuscripts should be prepared using Elsevier's elsart.cls LaTeX article class) which can be retrieved from the journal's homepage: http://www.journals.elsevier.com/journal-of-logical-and-algebraic-methods-in-programming/ Submissions are handled using the Elsevier Editorial System and can be uploaded via the aforementioned JLAMP homepage. In the submission process, the authors must select article type "WWVPT".

GUEST EDITORS. Maurice H. ter Beek, ISTI-CNR, Pisa, Italy; Alexei Lisitsa, University of Liverpool, UK; Andrei P. Nemytykh, Russian Academy of Sciences, Russia; Antonio Ravara, Universidade Nova de Lisboa, Portugal

ACTA INFORMATICA: SPECIAL ISSUE ON SYNTHESIS

Call for Papers
`http://www.easychair.org/smart-program/VSL2014/SYNT-cfp_specialissue.html`

SCOPE. This special issue is devoted to the scope of the Third Workshop on Synthesis, SYNT 2014 (see http://vsl2014.at/synt). SYNT 2014 was co-located with CAV in the scope of the Vienna Summer of Logic 2014 and was devoted to bringing together researchers from different research areas who work on the quickly growing field of synthesis. The special issue is open to all topics related to synthesis.

SUBMISSIONS. Submission to this special issue is completely open and not limited to participants of the SYNT 2014 workshop. We expect original articles (typically 15-30 pages), which present high-quality contributions that have not been previously published in a journal and are not concurrently submitted to any other peer reviewed venue. All submissions should include some theoretical contribution to the area of synthesis. Extended versions of contributions previously published in proceedings need to contain significant new material and should be accompanied by a short description of the extension.

DATES. Submissions are accepted starting in January 2015, using the "Submit Online" button of the journal's website:
http://www.springer.com/computer/theoretical+computer+science/journal/236
Submission deadline: 1st of March 2015

TOPOLOGY, ALGEBRA, AND CATEGORIES IN LOGIC (TACL 2015)

Call for Participation
School: 15 - 19 June 2015, University of Salerno (Italy)
Conference: 21 - 26 June 2015, Ischia Island (Italy)
`http://logica.dmi.unisa.it/tacl/`

PROGRAMME. The programme of the conference TACL 2015 will focus on three interconnecting mathematical themes central to the semantic study of logics and their applications: algebraic, categorical, and topological methods. This is the seventh conference in the series Topology, Algebra, and Categories in Logic (TACL). Earlier instalments of this conference have been organised in Tbilisi (2003), Barcelona (2005), Oxford (2007), Amsterdam (2009), Marseilles (2011), and Nashville (2013). Starting from 2013, the conference is preceded by a one-week school. This year the school will be held at the campus of the University of Salerno and will include four tutorials, each consisting of 1.5 hour lectures for five days.

IMPORTANT DATES. The website is now open for submissions and registration.

— Deadline for submissions 1 March 2015
— Notification of acceptance 30 March 2015
— Deadline for early registration (conference) 30 April 2015
— Deadline for registration (school) 30 April 2015
— School dates: 15 - 19 June 2015, University of Salerno (Italy)
— Conference dates: 21 - 26 June 2015, Ischia Island (Italy)

INVITED SPEAKERS. Olivia Caramello (Institut des Hautes Etudes Scientifiques), Agata Ciabattoni (Technische Universitaet Wien), Maria Manuel Clementino (Universidade de Coimbra), Emil Jerabek (Academy of Sciences of the Czech Republic), Andre Joyal (Universite du Quebec), Keith A. Kearnes (University of Colorado), Daniele Mundici (University of Florence), Paulo Oliva (Queen Mary University of London), Jorge Picado (Universidade de Coimbra), Michael Pinsker (University Paris Diderot)

SCHOOL LECTURERS. Guram Bezhanishvili (New Mexico State University), Brian Davey (La Trobe University), Ieke Moerdijk (Nijmegen University), Luke Ong (Oxford University)

THE 11TH INTERNATIONAL TBILISI SYMPOSIUM ON LANGUAGE, LOGIC AND COMPUTATION

Call for Papers
21-26 September 2015
Tbilisi, Georgia
`http://www.illc.uva.nl/Tbilisi/Tbilisi2015`

AIMS. The Eleventh International Tbilisi Symposium on Language, Logic and Computation will be held on 21-26 September 2015 in Tbilisi, Georgia. The Programme Committee invites submissions for contributions on all aspects of language, logic and computation. Work of an interdisciplinary nature is particularly welcome. Areas of interest include, but are not limited to:

- Algorithmic game theory
- Computational social choice
- Constructive, modal and algebraic logic
- Formal models of multiagent systems
- Historical linguistics, history of logic
- Information retrieval, query answer systems
- Language evolution and learnability
- Linguistic typology and semantic universals
- Logic, games, and formal pragmatics
- Logics for artificial intelligence
- Natural language syntax, semantics, and pragmatics
- Natural logic, inference and entailment in natural language
- Distributional and probabilistic models of information and meaning

SUBMISSIONS. Authors can submit an abstract of three pages (including references) at the EasyChair conference system here:
http://www.easychair.org/conferences/?conf=tbillc2015

PROGRAMME. The programme will include the following invited lectures and tutorials.

- Tutorials
  - Logic: Brunella Gerla (University of Insubria)
  - Language: Lisa Matthewson (University of British Columbia)
  - Computation: Joel Ouaknine (Oxford University)
- Invited Lectures
  - Rajesh Bhatt (University of Massachusetts)
  - Melvin Fitting (Graduate School and University Center of New York)
  - Helle Hansen (Delft University of Technology)
  - George Metcalfe (Bern University)
  - Sarah Murray (Cornell University)
  - Mehrnoosh Sadrzadeh (Queen Mary, University of London)

WORKSHOPS. There will also be a workshop on Automata and Coalgebra, organised by Helle Hansen and Alexandra Silva and a workshop on “How to make things happen in grammar: Encoding Obligatoriness”, organised by Rajesh Bhatt and Vincent Homer.

CHAIRS. Daniel Altshuler (Chair, Heinrich-Heine-University Duesseldorf); Luca Spada (Chair, ILLC, University of Amsterdam and University of Salerno)

PUBLICATION INFORMATION. Post-proceedings of the symposium will be published in the LNCS series of Springer.

IMPORTANT DATES.

— Submission deadline: 1 March 2015
— Notification: 1 May 2015
— Final abstracts due: 1 June 2015
— Registration deadline: 1 August 2015
— Symposium: September 21-26, 2015

COMPUTER SCIENCE LOGIC 2015 (CSL 2015)

Call for Papers
7-10 September 2015
Berlin, Germany
`http://logic.las.tu-berlin.de/csl2015/`

AIM AND SCOPE. Computer Science Logic (CSL) is the annual conference of the European Association for Computer Science Logic (EACSL). The conference is intended for computer scientists whose research activities involve logic, as well as for logicians working on issues significant for computer science.

LOCATION. The 24th EACSL Annual Conference on Computer Science Logic will be held at the Technical University Berlin from Monday, 7 September 2015 to Thurday, 10 September 2015.

LIST OF TOPICS OF INTEREST (NON EXHAUSTIVE).

- automated deduction and interactive theorem proving
- constructive mathematics and type theory
- equational logic and term rewriting
- automata and games, game semantics
- modal and temporal logic
- model checking
- decision procedures
- logical aspects of computational complexity
- finite model theory
- computational proof theory
- bounded arithmetic and propositional proof complexity
- logic programming and constraints
- lambda calculus and combinatory logic
- domain theory
- categorical logic and topological semantics
- database theory
- specification, extraction and transformation of programs
- logical aspects of quantum computing
- logical foundations of programming paradigms
- verification and program analysis
- linear logic
- higher-order logic
- nonmonotonic reasoning

IMPORTANT DATES.

— Abstract submission: 3 April 2015
— Paper Submission: 10 April 2015
— Paper Notification: 13 June 2015
— Conference: 7 - 10 September 2015

SUBMISSION. Authors are invited to submit papers of not more than 15 pages in LIPIcs style presenting work not previously published. Papers are to be submitted through Easychair. Submitted papers must be in English and must provide sufficient detail to allow the PC to assess the merits of the paper. Full proofs may appear in a technical appendix which will be read at the reviewers' discretion. Authors are strongly encouraged to include a well written introduction which is directed at all members of the program committee.

SATELLITE EVENTS.

— The 11th International Workshop on Fixed Points in Computer Science (FICS'15) will be held on 11 and 12 September 2015 as a co-located event of CSL15.
— YuriFest: we will celebrate Yuri Gurevichs 75th birthday with a symposium in his honour on 11 September 2015 as a co-located event of CSL15.
— The annual meeting of the GI Fachgruppe Logik will be organised at the Technical University Berlin in conjunction with CSL'15.

PC CHAIR. Stephan Kreutzer (Technical University Berlin)

ORGANISING COMMITTEE.

- Christoph Dittmann (Technical University Berlin)
- Viktor Engelmann (Technical University Berlin)
- Stephan Kreutzer (Technical University Berlin, Chair)
- Jana Pilz (Technical University Berlin)
- Roman Rabinovich (Technical University Berlin)
- Sebastian Siebertz (Technical University Berlin)

MATHEMATICAL FOUNDATIONS OF PROGRAMMING SEMANTICS XXXI (MFPS 2015)

Call for Papers
22-25 June 2015, Nijmegen, Netherlands
`http://events.cs.bham.ac.uk/mfps31/`

MFPS SERIES. MFPS conferences are dedicated to the areas of mathematics, logic, and computer science that are related to models of computation in general, and to semantics of programming languages in particular. This is a forum where researchers in mathematics and computer science can meet and exchange ideas. The participation of researchers in neighbouring areas is strongly encouraged. This edition of MFPS will be co-located with CALCO.

IMPORTANT DATES.

— Submission: April 3, 2015
— Notification: May 15, 2015
— Final version: May 29, 2015

INVITED SPEAKERS. Andrew Pitts, Thierry Coquand, Paul B. Levy, Guy McCusker, Sam Staton

INVITED TUTORIAL SPEAKERS. Matija Pretnar, Andrzej Murawski, Martin Escardo

SPECIAL SESSIONS. algebraic effects, game semantics, homotopy type theory, quantitative semantics

MORE INFO. For more information please consult the web page.

13TH INTERNATIONAL CONFERENCE ON LOGIC PROGRAMMING AND NON-MONOTONIC REASONING (LPNMR 2015)

Preliminary Call for Papers
Lexington, KY, USA
September 27-30, 2015
`http://lpnmr2015.mat.unical.it/`
(Collocated with the 4th Conference on Algorithmic Decision Theory 2015)

AIMS AND SCOPE. LPNMR 2015 is the thirteenth in the series of international meetings on logic programming and non-monotonic reasoning. LPNMR is a forum for exchanging ideas on declarative logic programming, non-monotonic reasoning, and knowledge representation. The aim of the conference is to facilitate interactions between researchers and practitioners interested in the design and implementation of logic-based programming languages and database systems, and those working in knowledge representation and nonmonotonic reasoning. LPNMR strives to encompass theoretical and experimental studies that have led or will lead to the construction of systems for declarative programming and knowledge representation, as well as their use in practical applications. This edition of LPNMR will feature several workshops, a special session dedicated to the 6th ASP Systems Competition, and will be collocated with the 4th Algorithmic Decision Theory Conference, ADT 2015. Joint LPNMR-ADT Doctoral Consortium will be a part of the program. Authors are invited to submit papers presenting original and unpublished research on all aspects of non-monotonic approaches in logic programming and knowledge representation. We invite submissions of both long and short papers.

TOPICS. Conference topics include, but are not limited to:

(1) Foundations of LPNMR Systems
(2) Implementation of LPNMR systems
(3) Applications of LPNMR

SUBMISSION. LPNMR 2015 welcomes submissions of long papers (13 pages) or short papers (6 pages) in the following categories:

— Technical papers
— System descriptions
— Application descriptions

The indicated number of pages includes title page, references and figures. All submissions will be peer-reviewed and accepted papers will appear in the conference proceedings published in the Springer-Verlag Lecture Notes in Artificial Intelligence (LNAI/LNCS) series. At least one author of each accepted paper is expected to register for the conference to present the work. The Program Committee chairs are planning to arrange for the best papers to be published in a special issue of a premiere journal in the field. LPNMR 2015 will not accept any paper which, at the time of submission, is under review or has already been published or accepted for publication in a journal or another conference. Authors are also required not to submit their papers elsewhere during LPNMR's review period. However, these restrictions do not apply to previous workshops with a limited audience and without archival proceedings.

ASSOCIATED WORKSHOPS. LPNMR 2015 will include specialized workshops to be held on September 27 prior to the main conference. Currently planned workshops include:

— Grounding, Transforming, and Modularizing Theories with Variables; Organizers: Marc Denecker, Tomi Janhunen
— Action Languages, Process Modeling, and Policy Reasoning; Organizer: Joohyung Lee
— Natural Language Processing and Automated Reasoning; Organizers: Marcello Balduccini, Ekaterina Ovchinnikova, Peter Schueller
— Learning and Nonmonotonic Reasoning; Organizers: Alessandra Russo and Alessandra Mileo

IMPORTANT DATES (TENTATIVE).

— Paper registration: April 13, 2015
— Paper submission: April 20, 2015
— Notification: June 1, 2015
— Final versions due: June 15, 2015

VENUE. Lexington is a medium size, pleasant and quiet university town. It is located in the heart of the so-called Bluegrass Region in Central Kentucky. The city is surrounded by beautiful horse farms on green pastures dotted with ponds and traditional architecture stables, and small race tracks, and bordered by white or black fences. The Horse Museum is as beautifully located as it is interesting. Overall, the city has a nice feel that mixes well old and new. The conference will be held in the Hilton Lexington Downtown hotel.

GENERAL CHAIR. Victor Marek, University of Kentucky, KY, USA

PROGRAM CHAIRS. Giovambattista Ianni, University of Calabria, Italy; Mirek Truszczynski, University of Kentucky, KY, USA

WORKSHOPS CHAIR. Yuliya Lierler, University of Nebraska at Omaha, NE, USA

PUBLICITY CHAIR. Francesco Calimeri, University of Calabria, Italy

CONTACT. lpnmr2015@mat.unical.it

31ST INTERNATIONAL CONFERENCE ON LOGIC PROGRAMMING (ICLP 2015)

Call for Papers
Cork, Ireland
August 31 - September 4, 2015
`http://booleconferences.ucc.ie/iclp2015`

HISTORY. Since the first conference held in Marseilles in 1982, ICLP has been the premier international conference for presenting research in logic programming. ICLP 2015 will be co-located with the 21st International Conference on Principles and Practice of Constraint Programming (CP 2015) and is part of "The Year of George Boole", a celebration of the life and work of George Boole who was born in 1815 and worked at the University College of Cork.

IMPORTANT DATES.

— Abstracts due: April 20, 2015
— Papers due: April 27, 2015
— Notification to authors: June 5, 2015
— Camera ready versions due: July 21, 2015
— Conference: August 31-September 4, 2015

CONFERENCE SCOPE. Contributions are sought in all areas of logic programming, including but not restricted to:

— Theory: Semantic Foundations, Formalisms, Nonmonotonic Reasoning, Knowledge Representation.
— Implementation: Compilation, Virtual Machines, Parallelism, Constraint Handling Rules and Tabling.
— Environments: Program Analysis, Transformation, Validation, Verification, Debugging, Profiling, Testing.
— Language Issues: Concurrency, Objects, Coordination, Mobility, Higher Order, Types, Modes, Assertions, Programming Techniques.
— Related Paradigms: Inductive and Coinductive Logic Programming, Constraint Logic Programming, Answer-Set Programming, SAT, Constraints, Computational Argumentation, Abductive Logic Programming, Functional Logic Programming.
— Applications: Databases, Data Integration and Federation, Software Engineering, Natural Language Processing, Web and Semantic Web, Agents, Artificial Intelligence, Bioinformatics, Social Networks and Social Choice.

In addition to the presentations of accepted papers, the technical program will include invited talks, advanced tutorials, the doctoral consortium, the Prolog contest and several workshops.

SUBMISSION DETAILS. There are two categories for submissions:

— Regular papers, including: (1) technical papers for describing technically sound, innovative ideas that can advance the state of logic programming; (2) application papers, with emphasis on impact on some application domains; (3) system and tool papers, with emphasis on novelty, practicality, usability and availability of the systems and tools described.
— Technical communications aimed at describing recent developments, new projects, and other materials not ready for publication as regular papers.

All regular papers and technical communications will be presented during the conference. All submissions must be written in English and describe original, previously unpublished research, and must not simultaneously be submitted for publication elsewhere. Regular papers must not exceed 12 pages plus bibliography: however the papers may include appendices beyond 12 pages. Technical communications must not exceed 10 pages. Submissions must be made in the TPLP format (see http://journals.cambridge.org/images/fileUpload/images/tlp_ifc_MAY2014.pdf) via the EasyChair submission system, available at
http://www.easychair.org/conferences/?conf=iclp2015.

PAPER PUBLICATION. All accepted regular papers will be published in the journal Theory and Practice of Logic Programming (TPLP), Cambridge University Press (CUP), in one or more special issues. In order to ensure the quality of the final version, papers may be subject to two rounds of refereeing (within the decision period). Accepted technical communications will be published in archival form. The program committee may also recommend papers submitted as regular to be published as technical communications.

ICLP 2015 ORGANIZATION.

- General Co-Chairs:
  Barry O'Sullivan, University College Cork, Ireland; Roland Yap,National University of Singapore.
- Program Co-Chairs:
  Thomas Eiter, TU Wien, Austria; Francesca Toni, Imperial College London, UK.
- Local Arrangements Co-Chairs: Barry O'Sullivan, University College Cork, Ireland; Ken Brown, University College Cork, Ireland.
- Workshops Chair: Mats Carlsson, SICS, Uppsala, Sweden.
- Doctoral Consortium Chairs: Marina De Vos, University of Bath, UK; Yuliya Lierler, University of Nebraska at Omaha, USA.
- LP/CP Programming Contest Chair: Neng-Fa Zhou, City University of New York, USA; Peter Stuckey, NICTA and the University of Melbourne, Australia.
- Publicity Chair: Ian Miguel, University of St Andrews, UK.

13TH INTERNATIONAL SYMPOSIUM ON AUTOMATED TECHNOLOGY FOR VERIFICATION AND ANALYSIS (ATVA 2015)

October 12-15, 2015
Shanghai, China
`http://atva2015.ios.ac.cn/`

BACKGROUND. The purpose of ATVA is to promote research on theoretical and practical aspects of automated analysis, verification and syn-thesis by providing a forum for interaction between the regional and the international research communities and industry in the field.

SCOPE. ATVA 2015 solicits high-quality submissions in areas related to the theory and practice of automated analysis and verification of hardware and software systems. Topics of interest include, but are not limited to:

- Formalisms for modeling hardware, software and embedded systems
- Specification and verification of finite-state, infinite-state and parameterized systems
- Program analysis and software verification
- Analysis and verification of hardware circuits, systems-on-chip and embedded systems
- Analysis of real-time, hybrid, priced/weighted and probabilistic systems
- Deductive, algorithmic, compositional, and abstraction refinement techniques for analysis and verification
- Analytical techniques for safety, security, and dependability
- Testing and runtime analysis based on verification technology
- Analysis and verification of parallel and concurrent hardware/software systems
- Verification in industrial practice
- Applications and case studies Theory papers should preferably be motivated by practical problems, and applications should be based on sound theory and should solve problems of practical interest.

IMPORTANT DATES.

- April 22, 2015 Abstract submission deadline (AOE)

— April 25, 2015 Paper submission deadline (AOE)
— May 5, 2015 Submission of workshop proposals
— Jun 8, 2015 Paper acceptance/rejection notification
— Jun 10, 2015 Announcement of the accepted papers
— July 5, 2015 Camera-ready copy deadline

GENERAL CHAIR. Jifeng He (East China Normal University, China)

PROGRAMME CHAIRS. Bernd Finkbeiner (Saarland University, Germany), Geguang Pu (East China Normal University, China), Lijun Zhang (Institute of Software, Chinese Academy of Sciences)

PUBLICITY CHAIRS. David N. Jansen (Radboud Universiteit, Netherlands), Huibiao Zhu (East China Normal University, China)

WORKSHOP CHAIR. Jun Sun (National University of Singapore, SG)

KEYNOTES. Dino Distefano (Queen Mary, University of London, UK), Joost-Pieter Katoen (RWTH Aachen University, Germany), Jay Strother Moore (University of Texas-Austin, USA)

SYMPOSIUM ON DEPENDABLE SOFTWARE ENGINEERING: THEORIES, TOOLS AND APPLICATIONS (SETTA 2015)

Call for Papers
November 4-6, 2015
Nanjing University
`http://cs.nju.edu.cn/setta/`

BACKGROUND AND OBJECTIVES. The aim of the symposium is to bring together international researchers and practitioners in the field of software technology. Its focus is on formal methods and advanced software technologies, especially for engineering complex, large-scale artefacts like cyber-physical systems, networks of things, enterprise systems, or cloud-based services. Contributions relating to formal methods or integrating them with software engineering, as well as papers advancing scalability or widening the scope of rigorous methods to new design goals are especially welcome. Being hosted in China, the symposium will also provide a platform for building up research collaborations between the rapidly growing Chinese computer science community and its international counterpart. The symposium will support this process through dedicated events and therefore welcomes both young researchers considering international collaboration in formal methods and established researchers looking for international cooperation and willing to attract new colleagues to the domain.

SUBMISSIONS. Authors are invited to submit papers on original research, industrial applications, or position papers proposing challenges in fundamental research and technology. The latter two types of submissions are expected to contribute to the development of formal methods either by substantiating the advantages of integrating formal methods into the development cycle or through delineating need for research by demonstrating weaknesses of existing technologies, especially when addressing new application domains. Submissions can take the form of either normal or short papers. Short papers can discuss ongoing research at an early stage, including PhD projects. Papers should be written in English. Regular Papers should not exceed 15 pages and Short Papers should not exceed 6 pages in LNCS format (see http://www.springer.de/comp/lncs/authors.html for details). The proceedings will be published as a volume in Springer's LNCS series. The authors of a selected subset of accepted papers will be invited to submit extended versions of their papers to appear in a special issue of the Formal Aspect Computing journal.

TOPICS.

— Requirements specification and analysis
— Formalisms for modeling, design and implementation
— Model checking, theorem proving, and decision procedures
— Scalable approaches to formal system analysis
— Formal approaches to simulation and testing
— Integration of formal methods into software engineering practice
— Contract-based engineering of components, systems, and systems of systems
— Formal and engineering aspects of software evolution and maintenance
— Parallel and multicore programming
— Embedded, real-time, hybrid, and cyber-physical systems
— Mixed-critical applications and systems
— Formal aspects of service-oriented and cloud computing
— Safety, reliability, robustness, and fault-tolerance
— Empirical analysis techniques and integration with formal methods
— Applications and industrial experience reports
— Tool integration

IMPORTANT DATES.

— June 12,2015 Abstracts
— June 19,2015 Submission of papers
— August 21,2015 Notification to authors
— September 4,2015 Camera-ready versions

KEYNOTE SPEAKERS. Sanjoy Baruah, University of North Carolina at Chapel Hill, USA; David Harel, Weizmann Institute of Science, Israel; Huimin Lin, Institute of Software, CAS, China

GENERAL CHAIR. Jian Lv, Nanjing University, China

PROGRAMME CO-CHAIRS. Xuandong Li, Nanjing University, China; Zhiming Liu, Birmingham City University, UK; Yi Wang, Uppsala University, Sweden

PUBLICITY CHAIRS. Jonathan Bowen, Birmingham City University, UK; Lijun Zhang, Institute of Software,Chinese Academy of Sciences, China

PUBLICATION CHAIR. Martin Fraenzle, University of Oldenburg, Germany

LOCAL ORGANIZATION CHAIR. Xin Chen, Nanjing University, China

NEW DOCTORAL PROGRAM ON LOGICAL METHODS IN COMPUTER SCIENCE (LogiCS)

`http://logic-cs.at/phd`
Funded Doctoral Positions in Computer Science

PROGRAM. TU Wien, TU Graz, and JKU Linz are seeking exceptionally talented and motivated students for their joint doctoral program LogiCS. The LogiCS doctoral college focuses on interdisciplinary research topics covering (i) computational logic, and applications of logic to (ii) databases and artificial intelligence as well as to (iii) computer-aided verification.

THE PROGRAM. LogiCS is a doctoral college focusing on logic and its applications in computer science. Successful applicants will work with and be supervised by leading researchers in the fields of computational logic, databases and knowledge representation, and computer-aided verification.

FACULTY MEMBERS. M. Baaz, A. Biere, R. Bloem, A. Ciabattoni, U. Egly, T. Eiter, C. Fermueller, R. Grosu, A. Leitsch, M. Ortiz, R. Pichler, S. Szeider, H. Tompits, H. Veith, G. Weissenbacher

POSITIONS AND FUNDING. We are looking for 1-2 doctoral students per faculty member, where 30% of the positions are reserved for highly qualified female candidates. The doctoral positions are funded for a period of 3 years according to the funding scheme of the Austrian Science Fund (details: http://www.fwf.ac.at/de/projects/personalkostensaetze.html) The funding can be extended for one additional year contingent on a placement at one of our international partner institutions.

HOW TO APPLY. Detailed information about the application process is available on the LogiCS web-page http://logic-cs.at/phd/ The applicants are expected to have completed an excellent diploma or master's degree in computer science, mathematics, or a related field. Candidates with comparable achievements will be considered on a case-by-case basis. Applications by the candidates need to be submitted electronically. Applications can be submitted at any time. Next screenings: March 1, 2015.

HIGHEST QUALITY OF LIFE. The Austrian cities Vienna, Graz, and Linz, located close to the Alps and surrounded by beautiful nature, provide an exceptionally high quality of life, with a vibrant cultural scene, numerous cultural events, world-famous historical sites, a large international community, a varied cuisine and famous coffee houses. For further information please contact: info@logic-cs.at

join today!

SIGLOG & ACM

siglog.acm.org www.acm.org

The **Special Interest Group on Logic and Computation** is the premier international community for the advancement of logic and computation, and formal methods in computer science, broadly defined.

The **Association for Computing Machinery** (ACM) is an educational and scientific computing society which works to advance computing as a science and a profession. Benefits include subscriptions to *Communications of the ACM*, *MemberNet*, *TechNews* and *CareerNews*, full and unlimited access to online courses and books, discounts on conferences and the option to subscribe to the ACM Digital Library.

❑ SIGLOG (ACM Member)	$ 25
❑ SIGLOG (ACM Student Member & Non-ACM Student Member)	$ 15
❑ SIGLOG (Non-ACM Member)	$ 25
❑ ACM Professional Membership ($99) & SIGLOG ($25)	$124
❑ ACM Professional Membership ($99) & SIGLOG ($25) & ACM Digital Library ($99)	$223
❑ ACM Student Membership ($19) & SIGLOG ($15)	$ 34

payment information

Name ____________________

ACM Member # ____________________

Mailing Address ____________________

____

City/State/Province ____________________

ZIP/Postal Code/Country ____________________

Email ____________________

Mobile Phone ____________________

Fax ____________________

Credit Card Type: ❑ AMEX ❑ VISA ❑ MC

Credit Card # ____________________

Exp. Date ____________________

Signature ____________________

Make check or money order payable to ACM, Inc

ACM accepts U.S. dollars or equivalent in foreign currency. Prices include surface delivery charge. Expedited Air Service, which is a partial air freight delivery service, is available outside North America. Contact ACM for more information.

Mailing List Restriction
ACM occasionally makes its mailing list available to computer-related organizations, educational institutions and sister societies. All email addresses remain strictly confidential. Check one of the following if you wish to restrict the use of your name:

❑ ACM announcements only
❑ ACM and other sister society announcements
❑ ACM subscription and renewal notices only

Questions? Contact:
ACM Headquarters
2 Penn Plaza, Suite 701
New York, NY 10121-0701
voice: 212-626-0500
fax: 212-944-1318
email: acmhelp@acm.org

Remit to:
ACM
General Post Office
P.O. Box 30777
New York, NY 10087-0777

SIGAPP

www.acm.org/joinsigs

Association for Computing Machinery

Advancing Computing as a Science & Profession