First-Order Theorem Provers: the Next Generation
Extended Abstract

Andrei Voronkov
University of Manchester

Abstract

We briefly overview past and present of resolution theorem proving and will try to predict how resolution theorem proving will develop in the next several years. Our presentation will be centered around two main motives: efficiency and usefulness for existing and future potential applications.

From 1993 the author of this paper participates in the development and implementation of the theorem prover Vampire, see [Riazanov & Voronkov 2002*a*]. The paper is an attempt to look into the future of first-order theorem provers. It is based on the author's experience with the development, implementation, and applications of Vampire. Since the paper is to be presented at a description logics workshop, we will sometimes refer to reasoning in description logics.

The philosophy of automated reasoning in first-order logic is different from that of reasoning in description logics. First-order logic is undecidable, so first-order theorem proving does not pay much attention to termination. There are decidable classes of first-order formulas but formulas of these classes do not arise often in applications, unless these are classes of formulas obtained by translation from modal or description logics. Due to undecidability, very short formulas may turn out to be extremely complex, while very long ones rather easy. Sometimes first-order provers find proofs consisting of several thousand steps in a few seconds, but sometimes it takes hours to find a ten-step proof. The theory of first-order reasoning is centered around the completeness theorems while in practice completeness is usually not an issue due to the intrinsic complexity of first-order reasoning.

Historically the first description logics turned out to be decidable, and investigation of decidability, complexity, and expressive power remains to be one of the main issues in description logic reasoning. In expressive description logics one has to balance on a narrow margin between decidability and undecidability. When it comes to designing description logic languages, the description logic community tries to stay within the decidable fragments, even when this excludes a class of applications.

Although the two fields seems to stay apart, nevertheless they can learn from one another, especially when it comes to extensions. For example, some connections between the first-order and description logic reasoning were recently considered in [Schmidt 1998, Voronkov 2001, Baader & Tobies 2001].

Introduction. First-order theorem provers have several recognized application areas. These are software and hardware verification, theorem proving in mathematics, first-order knowledge bases, reasoning in nonclassical logics, retrieval of software components, program synthesis etc.

prover	affiliation	references
	General-purpose provers	
Otter	Argonne National Laboratory	[McCune 1994]
Setheo	Munich University of Technology	[Moser, Ibens, Letz, Steinbach, Goller, Schumann & Mayr 1997]
Spass	Max-Planck Institute	[Weidenbach, Gaede & Rock 1996]
Vampire	University of Manchester	[Riazanov & Voronkov 2002*a*]
Gandalf	Tallinn University of Technology	[Tammet 1997]
SCOTT	Australian National University	[Slaney, Lusk & McCune 1994]
E	Munich University of Technology	[Schulz 2001]
Snark	SRI	[Stickel, Waldinger & Chaudhri 2001]
Bliksem	Max-Planck Institute	[de Nivelle 2000]

Figure 1: First-order theorem provers

Among the most well-known achievements so far is their use as subsystems of interactive provers, such as HOL, for verification, for example verification of real arithmetic in Intel processors and solution of several open problems mathematics (mainly in algebra), for example the Robbins problem. First-order theorem provers are in practice used much more than it is reported in the literature, since many companies do not disclose proprietary use of the provers.

Since the early work on automated theorem proving, an impressive progress has been made in first-order theorem proving. This progress is due to several factors described below:

1. Development of *theory*, both of the saturation-based theorem proving with redundancy, see [Bachmair & Ganzinger 2001, Nieuwenhuis & Rubio 2001], and of the tableau and model elimination proof-search, see [Hähnle 2001, Letz & Stenz 2001, Degtyarev & Voronkov 2001].

2. Development of *implementation techniques* (term indexing, new algorithms) [Graf 1996, Sekar, Ramakrishnan & Voronkov 2001, Nieuwenhuis, Hillenbrand, Riazanov & Voronkov 2001].

3. Growing *experience*, including the experiments with applications and the development of TPTP [Sutcliffe & Suttner 1998].

Systems. The main currently supported systems are given in Figure 1. Among these systems, Setheo is based on model elimination, while other provers are based on resolution. In the rest of this paper we will overview resolution theorem provers only.

Resolution. In the early papers, e.g., [Robinson 1965, Robinson & Wos 1969] resolution is defined as a logical system consisting of several inference rules operating on clauses, for example, *resolution* and *paramodulation*:

$$\frac{A \lor C \quad \neg A' \lor D}{(C \lor D)\mathrm{mgu}(A, A')}\ ; \qquad \frac{s = t \lor C \quad L[s'] \lor D}{(C \lor L[t] \lor D)\mathrm{mgu}(s, s')}\ .$$

where mgu denotes the most general unifier. Since these rules are *local*, i.e., their applicability is identified only by a small number of clauses and the result of a rule application does not change the previously generated clauses, resolution is implemented using *saturation algorithms*.

Saturation algorithms. These algorithms operate on a set of clauses S, initially the set of input clauses. Roughly speaking, they are based on the following loop.

1. Apply inferences to clauses in S, adding to S the conclusions of these inferences.
2. If the empty clause $\square$ is proved, terminate with success. If no inference rule is applicable, terminate with failure.

The set S is the current *search space*. However, a naive application of this loop would not solve even some of the problems considered trivial by the modern theorem provers due to the fast growth of the search space. However, even in the first paper on resolution [Robinson 1965] it was noted that some clauses can be removed from the search space without losing completeness.

Redundancy. More precisely, it has been observed that clauses *subsumed* by other clauses can be removed and thus can be regarded as *redundant*. Later, several other notions of redundancy have been discovered. For example, [Brand 1975] proved that the function reflexivity inferences and paramodulation into a variable are redundant. In general, there are two kinds of redundancy in resolution theorem proving: *redundant inferences* and *redundant clauses*. Currently, many useful notions of redundancy are based on *simplification orders* introduced in [Knuth & Bendix 1970]. These orders are orders $\succ$ on terms which can be extended to literals and clauses. An example of an inference which is redundant due to the orderings restrictions is a paramodulation inference

$$\frac{s = t \vee C \quad L[s'] \vee D}{(C \vee L[t] \vee D)\theta}\ ,$$

for which $t\theta \succ s\theta$. In the 1970s–1980s many notions of redundant inferences and clauses have been investigated. In the 1990s a general theory of redundancies was described in [Bachmair & Ganzinger 1994*a*]. Nearly all state-of-the-art resolution theorem provers are based on this theory. Resolution with redundancy is based on the following saturation algorithm.

1. Apply all *nonredundant* inferences to clauses in S, adding to S those conclusions of these inferences *that are nonredundant*.
2. If the empty clause $\square$ is proved, terminate with success. If no inference rule is applicable, terminate with failure.
3. Remove all sequents that *become redundant* due to the addition of these conclusions of inferences.

In addition to the standard inference rule, these algorithms also operate with *simplifications*. An inference is called a *simplification* if it makes at least one clause in S redundant.

Given clause algorithms. In the modern provers inference selection is mainly done via the *clause selection.* There are two main concretizations of the saturation algorithm based on the clause selection: the Otter algorithm [Lusk 1992, McCune 1994] and the Discount algorithm [Denzinger, Kronenburg & Schulz 1997]. These algorithms are described and analyzed in more detail in [Riazanov & Voronkov 2002*b*].

Search space size and term indexing. Even the proof-search in the inference systems with redundancy creates enormously large such spaces. For example, storing 10^6 clauses is not unusual. Of these clauses 10^5 can participate in inferences. Some operations on clauses are very expensive if implemented naively. For example, subsumption on multi-literal clauses is NP-complete and must ideally be performed between each pair of clauses. It is difficult to imagine an implementation able to perform 10^{12} operations in reasonable time.

In the state-of-the-art theorem provers all expensive operations are implemented using *term indexing* [Graf 1996, Sekar et al. 2001]. The problem of term indexing can be formulated abstractly as follows. Given a set L of *indexed terms or clauses*, a binary relation R over terms or clauses (called the *retrieval condition*) and a term or clause t (called the *query term or clause*), identify the subset M of L that consists of the terms or clauses l such that $R(l, t)$ holds. A typical retrieval condition used in theorem proving is *subsumption*: retrieve all clauses in L subsumed by t. In order to support rapid retrieval of candidate clauses, we need to process the indexed set into a data structure called the *index*. Modern theorem provers maintain several indexes to support expensive operations. For example, Vampire [Riazanov & Voronkov 2002*a*] uses *flatterms* in constant memory for storing temporary clauses, *code trees* [Voronkov 1995] for forward subsumption, *code trees with precompiled ordering constraints* for forward simplification by unit equalities, *perfectly shared terms* for storing clauses, *shared terms with renaming lists* for backward simplification by unit equalities, *path index with compiled database joins* for backward subsumption and some other indexes.

Progress in first-order reasoning. Since the first implementation of first-order provers, there has been a considerable progress in solving difficult problems. This progress is due to the development of theory and techniques described above. For example, the provers of the 1970s could not be efficient for equality reasoning because of the unrestricted paramodulation and function reflexivity axioms. The provers of the 1980s would be much faster because of the restrictions on paramodulation and the use of simplification orderings. The provers of the early 1990s paid a special attention to term indexing, and of the late 1990s used the general theory of redundancy, including literal selection functions. Every 10 years, there are problems which are solved by *several orders of magnitude faster* than by the previous generations of theorem provers.

Efficient theorem proving in future. Theorem proving is a very hard problem. The next generation of theorem provers will incorporate new theory, data structures, algorithms, and implementation techniques to solve the problem which has been characterized by the ANL automatic reasoning group as *controlling redundancy in large search spaces* [Lusk 1992].

It is unreasonable to expect the future theorem provers to be mush faster on all possible problems. However, if we can increase performance of provers by *several orders of magnitude* for a *large number of problems coming from applications*, many of these problems will be routinely solved, thus saving time for application developers. The development of next generation provers will require:

1. development of *new theory*;
2. addition of *new features*;
3. development of new *algorithms and datastructures*;
4. understanding how the theory developed so far can be efficiently implemented *on top of existing architectures* of theorem provers;

This development is impossible without considerable implementation efforts and a large number of experiments.

The near future. Let me make some predictions about possible directions in theorem proving that will be exploited in the next several years. These predictions are based on the analysis of possible applications of first-order provers. Among the possible applications of theorem provers in the new future are verification, assistance to interactive proof assistants, and semantic web. These applications require development of new features and also increased performance. By improving the existing algorithms and datastructures, we can improve performance on pure first-order problems for some benchmarks by maybe an order of magnitude. With this improved performance many problems will still be beyond the capabilities of provers.

How can we improve their performance by *several* orders of magnitude? And more importantly, how to improve their performance for practical applications? Several possibilities are sketched below.

Built-in theories. Already in the case of equality it was noted that the naive addition of equality axioms is too inefficient. Development of methods of equality reasoning resulted in an impressive progress of theorem provers. There are other important theories which arise in many applications. Development of specialized reasoning methods for these theories will be a very central problem in theorem proving. This problem is different from the problem of designing decision procedures for theories such as Presburger arithmetic because relations and functions used in these theories may be intertwined with relations and functions from other theories and arbitrary quantifiers. Procedures for the combination of decision and unification algorithms can be of some help [Nelson & Oppen 1979, Shostak 1984, Baader & Schulz 1996, Rueß & Shankar 2001].

Among the most important built-in procedures are the following:

- AC (the theory of associative and commutative functions). These axioms occur in axiomatizations very often. There are many results related to building-in AC in theorem provers but term indexing modulo AC is still in its infancy. Associativity and commutativity were built in the EQP theorem prover [McCune 1997] with a considerable success but without term indexing. Term indexing modulo AC was considered in [Bachmair, Chen & Ramakrishnan 1993] but only for a very special case.
- Theories of transitive relations and orders [Bachmair & Ganzinger 1994*b*].

- Presburger arithmetic.

As a first step towards building-in important theories one can consider creation of libraries of axioms/theorems about commonly used data types. An example of such a project is the Standard Upper Ontology [IEEE SUO-KIF project 2002]. One can also enhance theorem provers by means of specifying built-in theories, for example, by constraints or by additional inference rules.

Inductively defined types. In many applications of interactive proof assistants one has to deal with inductively defined types. The proof assistants such as Isabelle [Paulson 2002], HOL [Gordon & Melham 1993], COQ [The Coq Development Team 2001], Twelf [Pfenning & Schuermann 1998] have facilities to define data types and functions inductively. First-order theorem provers have no such facilities. The work on building-in inductively defined types can be developed in the following directions:

- Inductive definitions of data types;
- Inductive definitions of functions on these data types;
- Limited forms of inductive reasoning.

Finite domain reasoning. In applications there are often either variables ranging over finite domains or finite relations which can be given by an explicit enumeration of all tuples in them. One can axiomatize membership in a finite domain or a finite relation in first-order logic, but the use of such axioms in a first-order prover would create a large number of axioms. More efficient ways of reasoning with finite domains or relations are required.

Nonstandard quantifiers. A problem similar to the finite domain reasoning is the problem of dealing with quantifiers specifying restrictions on the number of elements satisfying a relation, for example, $\exists^{\leq n}$ (there exists at most n). Such quantifiers are familiar to the description logic community. One can translate formulas with such quantifiers into first-order logic with equality but the translation will create formulas of a prohibitive size. It is interesting to develop special ways of reasoning with such quantifiers, although it is not an easy task.

Working with large axiomatizations. Very often theorem provers must work with large axiomatizations containing many irrelevant axioms. This is typical for applications such as reasoning with ontologies and assisting proof assistants. Recognition of irrelevant axioms is one of the most important problems in theorem proving. Many of the axioms are definitions of relations and functions in various forms. Recognition of the most typical definitions and efficient work with them play a major role in the future provers. The first steps in this direction are reported in [Ganzinger, Nieuwenhuis & Nivela 2001, Afshordel, Hillenbrand & Weidenbach 2001, Degtyarev, Nieuwenhuis & Voronkov 2002].

Proof checking. Surprisingly, the modern theorem provers are not good at producing proofs. Some of them print detailed resolution and paramodulation proofs but none gives a proof of the preprocessing steps, such as skolemization. Without proof-checking one cannot use provers in verification and in assisting proof assistants. We expect that in the near future proof-checking components will be added to all major first-order provers. We conjecture that the main approaches to proof-checking will be similar to those used in proof-carrying code [Necula & Lee 2000], so that a proof is built using a number of inference rules, and foundational proof-carrying code [Appel 2001], so that a proof will given in a system whose language is rich enough to prove even the preprocessing steps, such as HOL.

Other features. There are other features required of the next generation first-order theorem provers. For example, for the naive users who do not know (and usually do not want to know) much about theorem proving, theorem provers must have a strong auto-mode which will try to select automatically a strategy or strategies best suited for solving the given problem. For more advanced users, there should be options to specify term orderings, literal selection, and clause selection.

References

Afshordel, B., Hillenbrand, T. & Weidenbach, C. [2001], First-order atom definitions extended, *in* R. Nieuwenhuis & A. Voronkov, eds, 'LPAR 2001. Eighth International Conference on Logic for Programming, Artificial Intelligence and Reasoning', Vol. 2050 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, Havana, Cuba, pp. 309–319.

Appel, A. [2001], Foundational proof-carrying code, *in* 'Proc. 16th Annual IEEE Symposium on Logic in Computer Science, LICS 2001', pp. 247–258.

Baader, F. & Schulz, K. [1996], 'Unification in the union of disjoint equational theories: Combining decision procedures', *Journal of Symbolic Computations* **21**, 211–243.

Baader, F. & Tobies, S. [2001], The inverse method implements the automata approach for modal satisfiability, *in* R. Goré, A. Leitsch & T. Nipkow, eds, 'First International Joint Conference on Automated Reasoning, IJCAR 2001', Vol. 2083 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 92–106.

Bachmair, L., Chen, T. & Ramakrishnan, I. [1993], Associative-commutative discrimination nets, *in* M.-C. Gaudel & J.-P. Jouannaud, eds, 'Proceedings of the 4th International Joint Conference on Theory and Practice of Software Development (TAPSOFT)', Vol. 668 of *Lecture Notes in Computer Science*, Springer Verlag, Orsay, France, pp. 61–74.

Bachmair, L. & Ganzinger, H. [1994*a*], 'Rewrite-based equational theorem proving with selection and simplification', *Journal of Logic and Computation* **4**(3), 217–247.

Bachmair, L. & Ganzinger, H. [1994*b*], Rewrite techniques for transitive relations, *in* 'Proc. IEEE Conference on Logic in Computer Science (LICS)', IEEE Computer Society Press.

Bachmair, L. & Ganzinger, H. [2001], Resolution theorem proving, *in* A. Robinson & A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 2, pp. 19–99.

Brand, D. [1975], 'Proving theorems with the modification method', *SIAM Journal of Computing* **4**, 412–430.

de Nivelle, H. [2000], *Bliksem 1.10 User's Manual*, MPI für Informatik, Saarbrücken.

Degtyarev, A., Nieuwenhuis, R. & Voronkov, A. [2002], ‘Stratified resolution’, *Journal of Symbolic Computations* pp. 1–24. to appear.

Degtyarev, A. & Voronkov, A. [2001], Equality reasoning in sequent-based calculi, *in* A. Robinson & A. Voronkov, eds, ‘Handbook of Automated Reasoning’, Vol. I, Elsevier Science, chapter 10, pp. 611–706.

Denzinger, J., Kronenburg, M. & Schulz, S. [1997], ‘DISCOUNT — a distributed and learning equational prover’, *Journal of Automated Reasoning* **18**(2), 189–198.
URL: *citeseer.nj.nec.com/denzinger96discount.html*

Ganzinger, H., Nieuwenhuis, R. & Nivela, P. [2001], *The Saturate system*, Max-Planck Institute für Informatik.
URL: *http://www.mpi-sb.mpg.de/SATURATE/Saturate.html*

Gordon, M. & Melham, T. [1993], *Introduction to HOL*, Cambridge University Press.

Graf, P. [1996], *Term Indexing*, Vol. 1053 of *Lecture Notes in Computer Science*, Springer Verlag.

Hähnle, R. [2001], Tableaux and related methods, *in* A. Robinson & A. Voronkov, eds, ‘Handbook of Automated Reasoning’, Vol. I, Elsevier Science, chapter 3, pp. 100–178.

IEEE SUO-KIF project [2002], ‘Standard upper ontology knowledge interchange format’.
URL: *http://suo.ieee.org/suo-kif.html*

Knuth, D. & Bendix, P. [1970], Simple word problems in universal algebras, *in* J. Leech, ed., ‘Computational Problems in Abstract Algebra’, Pergamon Press, Oxford, pp. 263–297.

Letz, R. & Stenz, G. [2001], Model elimination and connection tableau procedures, *in* A. Robinson & A. Voronkov, eds, ‘Handbook of Automated Reasoning’, Vol. II, Elsevier Science, chapter 28, pp. 2015–2114.

Lusk, E. [1992], Controlling redundancy in large search spaces: Argonne-style theorem proving through the years, *in* A. Voronkov, ed., ‘Logic Programming and Automated Reasoning. International Conference LPAR’92.’, Vol. 624 of *Lecture Notes in Artificial Intelligence*, St.Petersburg, Russia, pp. 96–106.

McCune, W. [1994], OTTER 3.0 reference manual and guide, Technical Report ANL-94/6, Argonne National Laboratory.

McCune, W. [1997], ‘Solution of the robbins problem’, *Journal of Automated Reasoning* **19**(3), 263–276.

Moser, M., Ibens, O., Letz, R., Steinbach, J., Goller, C., Schumann, J. & Mayr, K. [1997], ‘SETHEO and E-SETHEO—the CADE-13 systems’, *Journal of Automated Reasoning* **18**, 237–246.

Necula, G. & Lee, P. [2000], Proof generation in the Touchstone theorem prover, *in* D. McAllester, ed., ‘17th International Conference on Automated Deduction (CADE-17)’, Vol. 1831 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, Pittsburgh, pp. 25–44.

Nelson, G. & Oppen, D. [1979], 'Simplification by cooperating decision procedures', *ACM Transactions on Programming Languages and Systems* **1**(2), 245–257.

Nieuwenhuis, R., Hillenbrand, T., Riazanov, A. & Voronkov, A. [2001], On the evaluation of indexing techniques for theorem proving, *in* R. Goré, A. Leitsch & T. Nipkow, eds, 'First International Joint Conference on Automated Reasoning, IJCAR 2001', Vol. 2083 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 257–271.

Nieuwenhuis, R. & Rubio, A. [2001], Paramodulation-based theorem proving, *in* A. Robinson & A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 7, pp. 371–443.

Paulson, L. [2002], *The Isabelle Reference Manual*, Cambridge University.

Pfenning, F. & Schuermann, C. [1998], *Twelf User's Guide*.
URL: *www.twelf.org*

Riazanov, A. & Voronkov, A. [2002*a*], 'The design and implementation of Vampire', *AI Communications* pp. 1–25. To appear.

Riazanov, A. & Voronkov, A. [2002*b*], 'Limited resource strategy in resolution theorem proving', *Journal of Symbolic Computations* pp. 1–14. to appear.

Robinson, G. & Wos, L. [1969], Paramodulation and theorem-proving in first order theories with equality, *in* B. Meltzer & D. Michie, eds, 'Machine Intelligence', Vol. 4, Edinburgh University Press, pp. 135–150.

Robinson, J. [1965], 'A machine-oriented logic based on the resolution principle', *Journal of the Association for Computing Machinery* **12**(1), 23–41. Reprinted as [Robinson 1983].

Robinson, J. [1983], A machine oriented logic based on the resolution principle, *in* J. Siekmann & G. Wrightson, eds, 'Automation of Reasoning. Classical Papers on Computational Logic', Vol. 1, Springer Verlag, pp. 397–415. Reprinted from [Robinson 1965].

Rueß, H. & Shankar, N. [2001], Deconstructing Shostak, *in* 'Proc. 16th Annual IEEE Symposium on Logic in Computer Science, LICS 2001', pp. 19–28.

Schmidt, R. [1998], Resolution is a decision procedure for many propositional modal logics, *in* M. Kracht, M. de Rijke, H. Wansing & M. Zakharyaschev, eds, 'Advances in Modal Logic, I', Vol. 87 of *Lecture Notes*, CSLI Publications, Stanford, pp. 189–208.

Schulz, S. [2001], System abstract: E 0.61, *in* R. Goré, A. Leitsch & T. Nipkow, eds, 'First International Joint Conference on Automated Reasoning, IJCAR 2001', Vol. 2083 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, pp. 370–375.

Sekar, R., Ramakrishnan, I. & Voronkov, A. [2001], Term indexing, *in* A. Robinson & A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 26, pp. 1853–1964.

Shostak, R. [1984], 'Deciding combinations of theories', *Journal of the Association for Computing Machinery* **31**(1), 1–12.

Slaney, J., Lusk, E. & McCune, W. [1994], SCOTT: Semantically Constrained Otter (system description), *in* A. Bundy, ed., 'Automated Deduction—CADE-12. 12th International Conference on Automated Deduction', Vol. 814 of *Lecture Notes in Artificial Intelligence*, Nancy, France, pp. 764–768.

Stickel, M., Waldinger, R. & Chaudhri, V. [2001], *A Guide to Snark*, SRI.
URL: *http://www.ai.sri.com/snark/*

Sutcliffe, G. & Suttner, C. [1998], 'The TPTP problem library — CNF release v. 1.2.1', *Journal of Automated Reasoning* **21**(2).

Tammet, T. [1997], 'Gandalf', *Journal of Automated Reasoning* **18**(2), 199–204.

The Coq Development Team [2001], *The Coq Proof Assistant Reference Manual. Version 7.2*, INRIA.
URL: *http://coq.inria.fr/doc/main.html*

Voronkov, A. [1995], 'The anatomy of Vampire: Implementing bottom-up procedures with code trees', *Journal of Automated Reasoning* **15**(2), 237–265.

Voronkov, A. [2001], 'How to optimize proof-search in modal logics: new methods of proving redundancy criteria for sequent calculi', *ACM Transactions on Computational Logic* **2**(2), 182–215.

Weidenbach, C., Gaede, B. & Rock, G. [1996], SPASS & FLOTTER. Version 0.42, *in* M. McRobbie & J. Slaney, eds, 'Automated Deduction — CADE-13', Vol. 1104 of *Lecture Notes in Computer Science*, New Brunswick, NJ, USA, pp. 141–145.