LEO-II - A Cooperative Automatic Theorem Prover for Classical Higher-Order Logic (System Description)

Christoph Benzmüller[1], Lawrence C. Paulson[2], Frank Theiss[1], and Arnaud Fietzke[3]

[1]Dep. of Computer Science, Saarland University, Saarbrücken, Germany
[2]Computer Laboratory, The University of Cambridge, UK
[3]Max Planck Institute for Informatics, Saarbrücken, Germany

Abstract. LEO-II is a standalone, resolution-based higher-order theorem prover designed for effective cooperation with specialist provers for natural fragments of higher-order logic. At present LEO-II can cooperate with the first-order automated theorem provers E, SPASS, and Vampire. The improved performance of LEO-II, especially in comparison to its predecessor LEO, is due to several novel features including the exploitation of term sharing and term indexing techniques, support for primitive equality reasoning, and improved heuristics at the calculus level. LEO-II is implemented in Objective Caml and its problem representation language is the new TPTP THF language.

1 Introduction and Motivation

Automatic theorem provers (ATPs) based on the resolution principle, such as Vampire [20], E [21], and SPASS [25], have reached a high degree of sophistication. They can often find long proofs even for problems having thousands of axioms. However, they are limited to first-order (FO) logic. Higher-order (HO) logic extends FO logic with lambda notation for functions and with function and predicate variables. It supports reasoning in set theory, using the obvious representation of sets by predicates. HO logic is a natural language for expressing mathematics, and it is also ideal for formal verification. Moving from FO to HO logic requires a more complicated proof calculus, but it often allows much simpler problem statements. HO logic's built-in support for functions and sets often leads to shorter proofs. Conversely, elementary identities (such as the distributive law for union and intersection) turn into difficult problems when expressed in FO form.

LEO-II is a standalone, resolution-based HO ATP that is designed for fruitful cooperation with specialist provers for fragments of HO logic. The idea is to combine the strengths of the different systems. On the other hand, LEO-II itself, as an external reasoner, is designed to support HO proof assistants such as Isabelle/HOL [18], HOL [13], and ΩMEGA [22] by efficiently automating subproblems and thereby reducing user effort.

1.1 Motivation for LEO-II

LEO-II is the successor of LEO [4], which was implemented in Allegro Common Lisp as a part of the ΩMEGA system and which unfortunately was not available as a standalone reasoner. LEO shared the basic data structures such as terms and clauses with ΩMEGA; these shared basic data structures were not designed for efficiency and their term indexing support was limited [15]. LEO's performance strongly improved after coupling it with the FO ATP Otter in the agent based ΩANTS framework [8]. This integration has subsequently been improved and Otter has been replaced by Bliksem and Vampire [9]. This cooperative theorem proving approach outperforms – modulo different problem representations – FO ATPs such as Vampire on problems about sets, relations, and functions as given in the TPTP SET domain (see Figure 1). The cooperative approach not only solves more problems in this domain; it also solves them more efficiently. This provides evidence for the following working hypothesis:

It is well known in AI that representation matters. Problem representation particularly matters in automated theorem proving: many proof problems can be effectively solved when they are initially represented in a natural way in HO logic and then tackled with a cooperative theorem proving approach in which a HO reasoner subsequently reduces them to a suitable fragment of HO logic in which they can be tackled by an effective specialist reasoner.

So far, we consider only FO logic as a target fragment. The general idea, however, is not limited to this; other fragments will be studied in future work. Examples include decidable fragments like monadic second-order logic and the guarded fragment. Decidability can probably produce useful feedback for improved heuristic control in the HO ATP.

2 Overview of LEO-II

The one year project "LEO-II: An Effective Higher-Order Theorem Prover" was funded by EPSRC at Cambridge University, UK under grant EP/D070511/1. In this project LEO-II has been (re-)implemented in Objective Caml 3.09 (approx. 12500 lines of code) as a standalone HO ATP. LEO-II can be easily installed, deployed and integrated with other reasoners and its sources are available from the LEO-II web-site at: `http://www.ags.uni-sb.de/~leo/`.

Proof search in LEO-II, which is based on the extensional HO resolution calculus [3] has been further improved, e.g., it now supports efficient (recursive) expansion of definitions and primitive equality reasoning (see Section 3). For cooperation with FO ATPs, LEO-II offers two different HO to FO mappings; further mappings can easily be added.

LEO-II provides efficient term data structures. It employs perfect term sharing and supports for HO term indexing. LEO-II also provides analysis tools for exploring its proof object, term graph and term index. This includes statistical analysis of the term graph (see Section 4).

In addition to a fully automatic mode, LEO-II also provides an interactive mode [6]. This mode supports debugging and inspection of the search space,

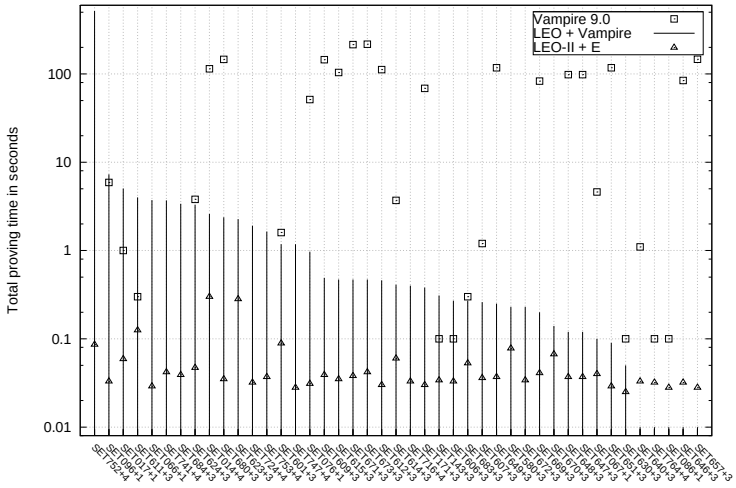


Fig. 1. The new LEO-II+E cooperation strongly outperforms the old LEO+Vampire cooperation on HO encodings of proof problems as given in the TPTP SET domain (these HO encodings are distributed with LEO-II). The worst approach is to tackle these problems in their original, less elegant FO encoding with Vampire.

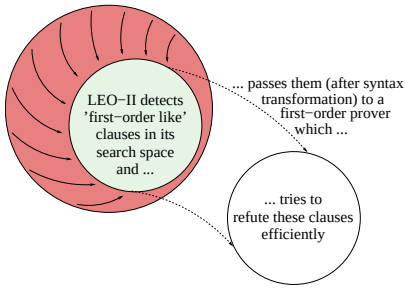
but also the tutoring of resolution based HO theorem proving to students. The interactive mode and the automatic mode can be interleaved.

LEO-II supports prefix-polymorphism; full polymorphism adds many non-trivial choice points to the already challenging search space of LEO-II [6] and is therefore future work. Moreover, LEO-II employs the new TPTP THF representation language [7].

At present, LEO-II cooperates with FO ATPs only in a sequential mode and not via the agent based architecture ΩANTS. Figure 1 shows that LEO-II's performance has nevertheless strongly improved (in our experiment version 0.95 of LEO-II was cooperating with E 0.99 "Singtom").

In the remainder we sketch LEO-II's cooperative proof search (Section 3) and its term sharing and term indexing (Section 4).

3 Cooperative Proof Search



LEO-II's clause set generally consists of HO clauses (dark-colored area), which are processed with LEO-II's calculus rules. Some of the clauses in LEO-II's search space additionally attain a special status (the ones in the light-colored area): they are in the domain of a transformation function to a particular fragment of HO logic, here FO logic. Light-colored clauses are available to both LEO-II's proof search and

to a specialist prover for the target fragment via the transformation function. Roughly speaking, currently all clauses that do not contain any λ-term and embedded predicate or proposition are light-colored, and our default FO transformation function employs Kerber's ideas [14]: it recursively translates every application $(p_{\alpha\to\beta}\, q_\alpha)$ into a term $@^{(\alpha\to\beta)\text{-}\alpha}(p,q)$, where $@^{(\alpha\to\beta)\text{-}\alpha}$ is a new function or predicate constant that encodes the types of its two arguments. LEO-II's extensional HO resolution approach, which enhances standard resolution proof search with specific extensionality rules, is explained in detail in [3] and [6]. It is well suited to subsequently generate more and more light-colored clauses from dark colored ones. In some proof problems (such as Examples 1 and 2 below) the light-colored area quickly collects enough information for constructing a refutation; however, LEO-II is often too weak to find this refutation on its own. In other proof problems (see Example 3 below) the refutation can be found only in the dark-colored area. This observation motivates a distributed architecture in which LEO-II dynamically cooperates with incremental specialist reasoners. At present, LEO-II sequentially launches a fresh call of the cooperating specialist prover every n iterations of its (standard) resolution proof search loop (currently $n = 10$).

Next, we discuss three proof problems from the domain of normal multimodal logics. For this domain, our cooperative approach yields elegant problem encodings and efficient solutions [5].

The encoding of multimodal logic in simple type theory is intuitive. Choose a base type — we choose ι — to denote the set of all possible worlds. Certain formulas of type $\iota \to o$ then correspond to multimodal logic expressions. The modal operators $\neg$, $\vee$, and $\Box_R$ become λ-terms of types $(\iota \to o) \to (\iota \to o)$, $(\iota \to o) \to (\iota \to o) \to (\iota \to o)$, and $(\iota \to \iota \to o) \to (\iota \to o) \to (\iota \to o)$ respectively. Note that $\neg$ forms the complement of a set of worlds, while $\vee$ forms the union of two such sets. $\Box_R$ explicitly abstracts over the accessibility relation R:

$$\begin{aligned}
\neg_{\,(\iota\to o)\to(\iota\to o)} &= \lambda A_{\iota\to o}.\lambda X_\iota.\neg A\, X\\
\vee_{\,(\iota\to o)\to(\iota\to o)\to(\iota\to o)} &= \lambda A_{\iota\to o}.\lambda B_{\iota\to o}.\lambda X_\iota.A\, X \vee B\, X\\
\Box_{R\,(\iota\to\iota\to o)\to(\iota\to o)\to(\iota\to o)} &= \lambda R_{\iota\to\iota\to o}.\lambda A_{\iota\to o}.\lambda X_\iota.\forall X_\iota.R\, X\, Y \Rightarrow A\, X
\end{aligned}$$

A multimodal logic formula $A_{\iota\to o}$ is valid iff for all possible worlds W_ι we have $W \in A$, that is, iff $A\, W$ holds: $\mathbf{valid} = \lambda A_{\iota\to o}.\forall W_\iota.A\, W$. Reflexivity and transitivity are defined as (we omit types) $\mathbf{refl} = \lambda R.\forall X.R\, X\, X$ and $\mathbf{trans} = \lambda R.\forall X.\forall Y.\forall Z.R\, X\, Y \wedge R\, Y\, Z \Rightarrow R\, X\, Z$. More details on this encoding, which models multimodal logic **K**, can be found in [5].

Example 1 (A simple equation between modal logic formulas).
$\forall R.\forall A.\forall B.(\Box_R\,(A \vee B)) = (\Box_R\,(B \vee A))$

Example 2 (The well known multimodal axioms $\Box_R\, A \Rightarrow A$ and $\Box_R\, A \Rightarrow \Box_R\, \Box_R\, A$ are equivalent to reflexivity and transitivity of the accessibility relation R).
$\forall R.(\forall A.\mathbf{valid}(\Box_R\, A \Rightarrow A) \wedge \mathbf{valid}(\Box_R\, A \Rightarrow \Box_R\, \Box_R\, A)) \Leftrightarrow (\mathbf{refl}(R) \wedge \mathbf{trans}(R))$

Example 3 (Axiom T is not valid).
$\neg\forall R.\forall A.(\mathbf{valid}(\Box_R\, A \Rightarrow A))$

After negating the statement in Example 1, recursive expansion of the definitions, and exhaustive clause normalization (with integrated functional and Boolean extensionality treatment), LEO-II generates ten light-colored clauses. Amongst them are $(b\,V) \vee (a\,V) \vee \neg((r\,w)\,V) \vee \neg((r\,w)\,U) \vee (b\,U) \vee (a\,U)$ and $\neg(a\,z) \vee \neg(b\,v)$ and $((r\,w)\,z) \vee ((r\,w)\,v)$ where upper case and lower case symbols denote variables and Skolem constants, respectively.[1] This light-colored clause set is immediately refutable by E, so that LEO-II does not even start its main proof loop. (The total proving time is $0.071s$ on a Linux notebook with 1.60GHz, 1GB memory.) Definition expansion and normalization does not always produce refutable light-colored clauses: When replacing the primitive equality $=$ in Example 1 by Leibniz equality $\lambda X, Y.\forall P_{\cdot}(P\,X) \Rightarrow (P\,Y)$, we obtain the dark-colored clauses $(p\,(\lambda X_\iota.\forall Y_\iota{}_{\cdot}\neg((r\,X)\,Y) \vee (a\,Y) \vee (b\,Y)))$ and $\neg(p\,(\lambda X_\iota.\forall Y_\iota{}_{\cdot}\neg((r\,X)\,Y) \vee (b\,Y) \vee (a\,Y)))$. LEO-II starts its proof search and applies resolution and extensional unification [3] to them which subsequently returns a refutable set of light-colored clauses as above. (The total proving time is $0.166s$.)

Example 2 is more challenging than Example 1. Proof search in LEO-II, however, is analogous with one crucial difference: definition expansion and normalization generates 70 clauses and E generates more than 20 000 clauses from them before finding the refutation. (The total proving time is $2.48s$.) This example illustrates the benefit of the cooperation, since LEO-II alone is not able to find this refutation in its search space: LEO-II generates too many clauses and gets stuck; moreover, LEO-II is still incomplete even for the FO fragment.

In Example 3, the refutation is found only in the dark-colored area. (The total proving time is $9.02s$.) Definition expansion and normalization generates the clauses $((R\,W)\,s^{A,W,R}) \vee (A\,W)$ and $\neg(A\,s^{A,W,R}) \vee (A\,W)$, where $s^{A,W,R} = (((s\,A)\,W)\,R)$ is a new Skolem term. The refutation employs only the former clause. LEO-II applies its primitive substitution rule [3, 6] to guess the instantiations $R \leftarrow \lambda X, Y.((M\,X)\,Y) \neq ((N\,X)\,Y)$ and $A \leftarrow \lambda X.(OX) \neq (PX)$ where M, N, O, P are new free variables. (Hence, LEO-II proposes to consider inequality for the the accessibility relation, which is not reflexive and does not satisfy axiom T.) Applying this instantiation leads to two unification constraints, i.e., negated equation literals in LEO-II, which both have flexible term heads. Proof search terminates since such flex-flex unification constraints are always solvable.

It is not obvious how the examples in this section can be represented in first-order logic. Therefore, these examples (and many others we are currently studying) are not included in the comparison in Figure 1.

LEO-II shows some promising first results on examples as presented in this paper, though it is not yet complete for simple type theory. Two known sources of incompleteness are LEO-II's pruning of unification problems at a preset unification depth (nested projections and imitations) and its limited support for

[1] To illustrate the use of FO TPTP we show how the latter clause is represented modulo our transformation function in FO TPTP syntax (with T encoding brackets):
$fof(leo\_II\_clause\_177, axiom,$
$(at\_io\_i(at\_iTioT\_i(sk1, sk4), sk13)|at\_io\_i(at\_iTioT\_i(sk1, sk4), sk11))).$

factorization modulo unification. In simple examples with Church numerals the former restriction, for example, prevents LEO-II from synthesizing Church numerals beyond this depth and because of the latter restriction LEO-II still cannot solve the famous Cantor theorem. Future work will address these aspects.

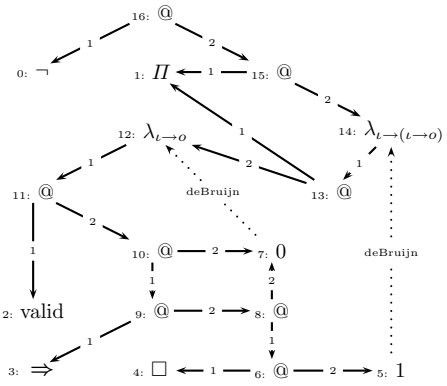
4 Term Sharing and Term Indexing

Term indexing techniques are widely used in major FO ATPs [20, 21, 25]. The indexing data structures store large numbers of terms and, for a given query term t, support the fast retrieval of terms from the index that satisfy a certain relation with t, such as matching, unifiability or syntactic equality [17]. Performance can be further enhanced by representation of terms in efficient data structures, such as shared terms, used for instance in E [21].

HO term indexing techniques are rarely addressed in the literature, which hampers the progress of systems in this field. An exception is Pientka [19].

LEO-II's implementation at term level is based on a perfectly shared term graph, i.e., syntactically equal terms are represented by a single instance. Ideas from FO term sharing are adapted to HO logic by (i) keeping indexed terms in $\beta\eta$ normal form (i.e., η short and β normal) and (ii) using de Bruijn indices [12] to allow λ-abstracted terms to be shared.

The resulting data structure represents terms in a directed acyclic graph (DAG). LEO-II supports the visualization of such term graphs. The graph to the right shows the (not heavily shared) term graph after initialization of Example 3.[2]



LEO-II also supports statistical analysis of its term graphs (Figure 2). Future work will investigate whether such information can be exploited for improving heuristic control.

Representation of terms in a shared graph naturally advances the performance of a number of operations, e.g., it allows fast lookup of all occurrences of syntactically equal terms or subterms, and it improves the performance of rewrite operations, such as global unfolding of definitions. Additionally, LEO-II employs a term indexing data structure, which is based on structural indexing methods from the FO domain [16, 23], as well as road sign techniques. Road signs are features of the data structure which guide

[2] To further visualize the evolution of the term graph during proof search, we modified LEO-II to output a snapshot of its state after each processing step. This data was used to create animations of dynamically changing term graphs during proof search. The video clips can be obtained at `http://www.ags.uni-sb.de/~leo/art.html`.

```
------------ The Termset Analysis ------------
[...]

Sharing rate: 17 nodes with 18 bindings
Average sharing rate (bindings per node):          1.05
Average term size:                                 6.58
Average number of supernodes:                      5.47
Average number of supernodes (symbols):            5.80
Average number of supernodes (nonprimitive terms): 4.50
Rate of term occurrences PST size / term size:     0.36
Rate of symbol occurrences PST size / term size:   0.39
Rate of bound occurrences PST size / term size:    0.57
------------ End Termset Analysis ------------
```

```
------------ The Termset Analysis ------------
[...]

Sharing rate: 2094 nodes with 3415 bindings
Average sharing rate (bindings per node):          1.63
Average term size:                                13.95
Average number of supernodes:                      9.47
Average number of supernodes (symbols):           28.83
Average number of supernodes (nonprimitive terms):  5.73
Rate of term occurrences PST size / term size:     0.24
Rate of symbol occurrences PST size / term size:   0.30
Rate of bound occurrences PST size / term size:    0.52
------------ End Termset Analysis ------------
```

Fig. 2. Statistical analysis of term sharing aspects in LEO-II after initialisation of Example 3 (left) and after the proof has been found (right).

operations based on graph traversal. They help to cut branches of the subgraph to be processed early and they are employed, e.g., in the construction of partial syntax trees [24] in which all branches with no occurrences of a given symbol or subterm are cut. This enables LEO-II to avoid potentially costly operations, such as occurs checks, and to speed up basic operations on terms, such as substitution.

Future work includes the development, comparison and evaluation of other termindexing techniques within LEO-II.

5 Conclusion

LEO-II, which replaces the previous LEO system, realizes a cooperative HO-FO theorem proving approach that shows some promising results in selected problem domains. It thus provides an interesting alternative to reasoning solely in FO logic and it also differs from other HO ATPs such as TPS [1] and OTTER-λ [2]. TPS is based on the mating method and does not cooperate with specialist FO ATPs. It is particularly strong in proving theorems which require selective expansion of definitions and goal directed instantiation of set variables. Examples of such theorems are presented in Bishop and Andrews [10] and Brown [11]. Many of these examples will require corresponding mechanisms in LEO-II to be proven automatically. The formalism of OTTER-λ is not simple type theory but λ-logic, which is a novel combination of λ-calculus and first-order logic.

Future work includes the experimentation with the LEO-II on case studies in verification and ontology reasoning.

References

1. P. B. Andrews and C. E. Brown. TPS: A hybrid automatic-interactive system for developing proofs. *J. Applied Logic*, 4(4):367–395, 2006.
2. M. Beeson. Mathematical induction in Otter-Lambda. *J. Autom. Reasoning*, 36(4):311–344, 2006.
3. C. Benzmüller. *Equality and Extensionality in Automated Higher-Order Theorem Proving*. PhD thesis, Saarlandes University, 1999.
4. C. Benzmüller and M. Kohlhase. LEO – a higher-order theorem prover. In *Proc. of 15th Conference on Automated Deduction*, number 1421 in LNAI, pages 139–143. Springer, 1998.

5. C. Benzmüller and L. Paulson. *Festschrift in Honour of Peter B. Andrews on his 70th Birthday*, chapter Exploring Properties of Normal Multimodal Logics in Simple Type Theory with LEO-II. IFCoLog, 2007. To appear.
6. C. Benzmüller, L. Paulson, F. Theiss, and A. Fietzke. Progress report on LEO-II – an automatic theorem prover for higher-order logic. In *TPHOLs 2007 Emerging Trends Proc.*, pages 33–48. Internal Report 364/07, Department of Computer Science, University Kaiserslautern, Germany, 2007.
7. C. Benzmüller, F. Rabe, and G. Sutcliffe. THF0 — the core TPTP language for classical higher-order logic. This Proc. (IJCAR 2008).
8. C. Benzmüller, V. Sorge, M. Jamnik, and M. Kerber. Experiments with an Agent-Oriented Reasoning System. In *In Proc. of KI 2001*, volume 2174 of *LNAI*, pages 409–424. Springer, 2001.
9. C. Benzmüller, V. Sorge, M. Jamnik, and M. Kerber. Combined reasoning by automated cooperation. *J. Applied Logic*, 2008. In print.
10. M. Bishop and P. B. Andrews. Selectively instantiating definitions. In Claude Kirchner and Hélène Kirchner, editors, *Proc. of the 15th International Conference on Automated Deduction*, volume 1421 of *LNAI*, pages 365–380. Springer, 1998.
11. C. E. Brown. Solving for Set Variables in Higher-Order Theorem Proving. In Andrei Voronkov, editor, *Proc. of the 18th International Conference on Automated Deduction*, volume 2392 of *LNAI*, pages 408–422. Springer, 2002.
12. N.G. de Bruijn. Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indag. Math.*, 34(5):381–392, 1972.
13. M.J. Gordon and T.F. Melham. *Introduction to HOL: A Theorem-Proving Environment for Higher-Order Logic*. Cambridge University Press, 1993.
14. M. Kerber. *On the Representation of Mathematical Concepts and their Translation into First-Order Logic*. PhD thesis, Univ. Kaiserslautern, Germany, 1992.
15. L. Klein. Indexing für Terme höherer Stufe. Master's thesis, Saarland Univ., 1997.
16. W. McCune. Experiments with discrimination-tree indexing and path indexing for term retrieval. *J. Automated Reasoning*, 9(2):147–167, 1992.
17. R. Nieuwenhuis, T. Hillenbrand, A. Riazanov, and A. Voronkov. On the evaluation of indexing techniques for theorem proving. In *In Proc. of International Joint Conference on Automated Reasoning*, volume 2083 of *LNAI*, pages 257–271. Springer, 2001.
18. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.
19. B. Pientka. Higher-order substitution tree indexing. In *Proc. of ICLP*, volume 2916 of *LNCS*, pages 377–391. Springer, 2003.
20. A. Riazanov and A. Voronkov. The design and implementation of Vampire. *AICOM*, 15(2-3):91–110, 2002.
21. S. Schulz. E – A Brainiac Theorem Prover. *J. AI Communications*, 15(2/3):111–126, 2002.
22. J. Siekmann, C. Benzmüller, and S. Autexier. Computer supported mathematics with OMEGA. *J. Applied Logic*, 4(4):533–559, 2006.
23. M. Stickel. The path-indexing method for indexing terms. Technical Report 473, Artificial Intelligence Center, SRI International, USA, 1989.
24. F. Theiss and C. Benzmüller. Term indexing for the LEO-II prover. In *Proc. of the 6th International Workshop on the Implementation of Logics*, volume 212 of *CEUR Workshop Proc.*, 2006.
25. C. Weidenbach et al. Spass version 2.0. In *Proc. of 18th Conference on Automated Deduction*, volume 2392 of *LNAI*, pages 275–279. Springer, 2002.