Evaluating Intuitionistic Automatic Theorem Provers

Thomas Raths

Institut für Informatik, University of Potsdam
August-Bebel-Str. 89, 14482 Potsdam-Babelsberg, Germany
raths@cs.uni-potsdam.de

Abstract. Intuitionistic automated theorem proving systems have been benchmarked on a large problem collection, the ILTP-Library [Raths *et al.*, 2005], which contains a few previously used formula collections, and part of the TPTP-Library [Sutcliff *et al.*, 1994]. The latter proves to be appropriate not only for classical but also for intuitionistic benchmarking. So-called lean theorem provers turn out to be not only compact but also very efficient compared to complex provers. We describe the theorem provers, test formulas, and present the results.

1 Introduction

Automated theorem proving (ATP) systems are important tools in interactive proof assistants like MetaPRL [Hickey *et al.*, 2003], Nuprl and Coq, for they can prove conceptually simple but technically complex subgoals, and thus release the user from tedious routine work. In practice however, there is still only limited use of theorem provers, because a lot of formulas cannot be solved in a reasonable amount of time. When working with a proof assistant, it makes a great difference whether a subgoal can be proved within a few seconds or several minuts (or much longer). Thus, it is worth striving for more efficient proof procedures. Benchmarking ATP systems can give some hints which proof methods and implementations are promising, and what can be optimised.

The presented tests focus on intuitionistic theorem provers. Intuitionistic (constructive) proofs are needed in program verification and synthesis. Since they contain much more information than classical proofs, they are much harder to find. Regarding the sequent calculi, classical and intuitionistic logic differ in the restrictions of the order in which the inference rules are applied. While in classical first-order logic only the order of the quantifier rules is restricted, in intuitionistic first-order logic also the order of the so-called *special* rules is constrained. The special rules decompose subformulas containing negation, implication, and the universal quantifier. Because of the interaction of these two constraints, the search space in intuitionistic first-order logic is very large, which means intuitionistic proving is very hard. In fact, we found only five intuitionistic first-order ATP systems. We also tested two purely propositional intuitiontistic provers.

We have estimated the time efficiency of these theorem provers by measuring the time needed for proving or refuting the formulas. There are also other criteria determining the usefulness of the ATP systems, such as space efficiency, correctness, completeness. For program verification and synthesis it is also desirable to supply the proof assistant with a sequent proof in order to extract a program.

Another criterion is the ease of handle and to adapt the theorem prover to the respective application. Among the tested ATP systems there are also so-called "lean"-provers consisting of only a few lines of code. In contrast to complex provers they are easy to adapt, are transparent, can be verified formally to be correct and complete, and are suited for demonstrations in teaching.

For benchmarking we compiled a comprehensive problem collection, the Intuitionistic Logic Theorem Proving (ILTP) Library v1.0 [Raths *et al.*, 2005]. This platform contains three small formula sets that have been previously used for testing intuitionistic ATP systems, and part of the TPTP (Thousands of Problems for Theorem Provers) Library v2.7.0 [Sutcliff *et al.*, 1994]. The latter is a large, regularly updated and extended collection of problems from various domains (e.g. algebra, set theory, software creation). Originally intended for evaluating classical ATP systems, the TPTP Library can also be used for intuitionistic benchmarking since classical and intuitionistic logic share the same syntax. This idea had already been expressed by others, but comprehensive intuitionistic benchmarking using the TPTP Library was probably done here for the first time.

In the next sections we describe the ATP systems, the test formulas and conditions, present the results and draw some conclusions.

2 The Provers

Seven intuitionistic ATP systems have been evaluated: ft [Sahlin *et al.*, 1992], ileanTAP [Otten, 1997], ileanSeP[1], ileanCoP [Otten, 2005], JProver [Schmitt *et al.*, 2001], LJT [Dyckhoff, 1992], STRIP [Larchey-Wendling *et al.*, 2001]. Table 1 classifies them according to their proof procedure (tableau-/connection-based), the logic they are being applied to (first-order/propositional), their size (lean/complex), and whether clausal form transformation is performed. In the following, the ATP systems are described very briefly. The reader is assumed to be familiar with automatic theorem proving, and is referred to the cited papers.

Probably the first intuitionistic first-order theorem prover is ft [Sahlin *et al.*, 1992]. ft uses the single-conclusioned sequent calculus, applies the inference rules backwards (analytically), and introduces free variables when applying the quantifier rules. Because of the non-invertibility of some rules in the intuitionistic sequent calculus, backtracking is necessary. ft tries to control the arising large search space by sophisticated heuristics. For propositional logic, a decision procedure using the contraction-free sequent calculus [Dyckhoff, 1992] is provided. There is a C and a Prolog version of ft.[2]

[1] available at `http://www.leancop.de/ileansep/`

[2] In the Prolog version of ft the decision procedure must be called explicitly.

proof procedure	intuitionistic logic	
	first-order	propositional
tableau-based	ft, ileanTAP, ileanSeP	LJT, STRIP
connection-based	ileanCoP, JProver	

size	clausal form transformation	
	with	without
lean	ileanCoP	ileanTAP, ileanSeP
complex		LJT, STRIP, ft, JProver

Table 1. Classification of the provers according to their proof procedure, logic, size, and whether they apply clausal form transformation

ileanSeP, ileanTAP, and ileanCoP are compact and very efficient ATP systems. They consist of only a few lines of Prolog Code, and are thus easily to handle and to adapt to the respective application. Due to their modular design they can be extended to deal with other non-classical logics. ileanSeP [1] uses the single-conclusioned sequent calculus with free variables and analytic proof search (similar to ft), but applies skolemization instead of heuristics.

ileanTAP [Otten, 1997] first tries to prove the formula classically by semantic tableau with free variables. After finding a tableau proof, it aims to unify so-called prefixes of those literals closing the branches. A prefix of a literal is a description of its position in the formula tree. The prefix unification ensures the particular restrictions in intuitionistic logic.

ileanCoP [Otten, 2005] works in a similar way as ileanTAP, but uses a connection-based method to prove the formula classically in the first step. To keep the proof procedure compact, it first transforms the formula into clausal form. There is also a version that implements the so-called Prolog technology: ileanCoP_pt represents the matrix corresponding to the formula in the Prolog database so that it can be accessed very fast.

JProver [Schmitt *et al.*, 2001] is a theorem prover for classical and intuitionistic first-order logic based on a connection method for non-clausal form and prefix unification [Otten und Kreitz, 1996]. In contrast to ileanCoP, clausal form transformation is not necessary, but the multiplicities of the submatrices are increased globally instead of locally. The found matrix proof is converted into a sequent proof that is used by the interactive proof assistants MetaPRL [Hickey *et al.*, 2003], Nuprl and Coq. JProver is implemented in OCaml.

All the theorem provers described above can decide a fragment of intuitionistic first-order logic. ft also decides whole propositional logic. LJT [Dyckhoff, 1992] and STRIP [Larchey-Wendling *et al.*, 2001] are purely propositional intuitionistic ATP systems, which use the contraction-free sequent calculus, apply analytic proof procedures and provide decision procedures.

3 Test Formulas and Conditions

We used the 1445 problems from the ILTP-Library v1.0 [Raths *et al.*, 2005]. These problems are divided into two sets: the TPTP problem set contains 1337 problems from the TPTP-Library v2.7.0 [Sutcliff *et al.*, 1994], and the ILTP problem set contains 108 problems from three previously used formula collections.

Each problem is associated with its *status*: `Theorem`, `Non-Theorem`, `Unknown`. Whereas in the ILTP problem set the status of every problem was known in advance, in the TPTP problem set the intuitionistic status of each problem has been determined by the ATP systems themselves assuming them to be correct. There is also a difficulty *rating* for each problem stating the portion of current state-of-the-art ATP systems that fail to solve the problem in a reasonable time. For example, a rating of 0.0 means that all state-of-the-art provers solve the problem, and a rating of 1.0 says that none of the provers could solve it within a certain time. [3] To specify the state-of-the-art ATP systems, we selected four first-order and one purely propositional prover which have solved the highest number of problems: ft (C-version), JProver, leanTAP, leanCoP, and STRIP.

3.1 The TPTP Problem Set

The TPTP-Library v2.7.0 is a collection of about 7000 problems from various domains for classical benchmarking. It can in principle also be used for intuitionistic benchmarking, as intuitionistic and classical logic have the same syntax. The TPTP-Library contains problems in clausal form (CNF), and in non-clausal form ("first-order form", FOF). Formulas in clausal form are intuitionistically invalid, and therefore not interesting for intuitionistic reasoning.[4]

From the 1745 FOF formulas 408 are classically invalid. These problems are not interesting as well, since they could be refuted intuitionistically by a classical ATP system. So we used the remaining 1337 formulas, of which 1279 are classical theorems, and 58 are not known to be classically valid or invalid.

3.2 The ILTP Problem Set

The ILTP problem set consists of 108 formulas of which 90 are intuitionistically valid, and 18 classically but not intuitionistically valid. The formulas were taken from three small previously used benchmark collections.

The first collection contains 39 intuitionistically valid first-order formulas that were used to test ft [Sahlin *et al.*, 1992]. The formulas `ft3.1` to `ft3.5` were left out as these problems (Pelletier 39 to 45) already exist in the TPTP problem set.

The second collection contains 36 propositional formulas from Dyckhoff's benchmark collection[5]. There are six classes, which are formulated in a intuitionistically valid and a classically but not intuitionistically valid variant. From each variant, problems of size 2, 6, and 10 were selected.

The third collection contains 33 intuitionistically valid formulas (21 first-order, 12 propositional) used to test JProver [Schmitt *et al.*, 2001]. Three formulas (`barber`, `fv1`, `fv3`) were omitted because they are already classically invalid, or cannot be represented in pure first-order logic.

[3] We gave the provers 300 seconds to solve a problem (see 3.3). The rating can be 1.0 while the status is `Theorem` or `Non-Theorem`, when solving takes more than 300 s.

[4] Already $p \vee \neg p$ does not hold intuitionistically. Also a transformation into clausal form does not preserve intuitionistic validity, because applied implications such as $\neg(A \wedge B) \rightarrow (\neg A \vee \neg B)$ and $\neg\neg A \rightarrow A$ are intuitionistically invalid.

[5] See `http://www.dcs.st-and.ac.uk/~rd/logic/marks.html`

3.3 Test Conditions

The benchmarks have been performed on a Xeon 3 Ghz under Linux 2.4.22. The ATP systems had 300 seconds to solve a problem. A problem was regarded as "solved", i.e. proved or refuted, when the ATP system put out a corresponding message. Table 2 gives information about the used versions and compilers for each tested ATP system.[6]

prover	ft		ileanSeP	ileanTAP	ileanCoP	LJT	STRIP	JProver
version	1.23		1.0	1.17	1.0	1.0	1.0	MetaPRL-CVS-2004.03.07
	C-version	Prolog-vs.						
language	C	Prolog					C	OCaml
compiler	gcc v.3.3.1	ECLiPSe 5.7 #33 (using flag "nodbgcomp.")					(provided binary)	Ocaml v3.06, OMake v0.7.9

Table 2. Versions and compilers of the tested ATP systems

4 Results

Table 3 shows the overall performance of the tested intuitionistic ATP systems on the ILTP-Library v1.0. ileanCoP_pt and ileanCoP solved the highest number of problems. Far behind ft, ileanTAP, JProver, and ileanSeP follow. ft's C-version does better than its Prolog-version. Regarding only the number of proved formulas, ileanTAP comes between the two ft versions. Of the purely propositional provers, STRIP performs better than LJT.

All the theorem provers yield error messages for a few formulas. ileanCoP and ileanCoP_pt put out `Stack overflow` when clausal form transformation generates too large formulas. The same error message occurs with all tableau-based provers when managing the open proof branches needs too much stack. The C-version of ft reports further memory limits. JProver shows an error in its prefix unification module in certain cases.

None of the tested ATP systems turned out to be incorrect. That means, from the ILTP problem set only intuitionistically valid formulas have been proved, and only intuitionistically invalid formulas have been refuted. For the TPTP problem set we must give a weaker statement, since the status of these formulas was not known in advance but was determined by the ATP systems themselves: The formulas that have been proved (refuted) were not refuted (proved) by any other ATP system. We have also checked JProver's proof conversion (in a separate test): Each generated sequent proof was understood by the interactive proof environment MetaPRL.

For the TPTP problem set, table 4 gives some information about the complexity behaviour of the ATP systems, and their performance on problems of

[6] For testing JProver, we chose a version that is integrated in MetaPRL in order to use its term operations and communication facilities. We slightly adapted MetaPRL's first-order theory such that type information is not needed to represent and prove the formulas. Furthermore, the proof conversion was switched off during benchmarking.

	provers								
	first-order							propositional	
	ft		ileanSeP	ileanTAP	ileanCoP	ileanCoP_pt	JProver	LJT	STRIP
	Prolog	C							
solved	199	226	156	184	318	323	162	55	60
proved	173	198	153	181	265	270	160	31	34
TPTP	99	112	87	113	188	193	94	5	5
ILTP	74	86	66	68	77	77	66	26	29
refuted	26	28	3	3	53	53	2	24	26
>300s	1192	673	1212	1201	1064	1050	1267	7	4
avg. time*	0.03	<0.01	1.96	0.01	1.50	0.44	0.70	<0.01	0.05
errors**	54	546	77	60	63	72	16	4	2

Table 3. Overall performance

* average time in s, computed only for formulas proved by all first-order provers, or propos. provers, resp. (7.4% of all formulas, or 42.4% of all propos. formulas, resp.)

** mainly `stack overflow`; other errors:

ft (C): `segmentation violation` (104 formulas), `memory allocation error` (3)
JProver: `Invalid_argument JProver bug (Jtunify module)` (16)

which the only predicate is equality. All theorem provers solve the most problems already within the first seconds. Giving the provers more time, the number of additionally proved formulas decreases exponentially. This is because with increasing formula size the search space and thus the time needed for proving increase exponentially.

ft and JProver show a strong decline. This implies that these provers will not solve much more problems, if more time, a faster system or compiler would be available, or code optimizations would be carried out. ileanCoP has a softer decline, meaning that it has the potential to solve more comlex problems.

	provers								
	first-order							proposit.	
	ft		ileanSeP	ileanTAP	ileanCoP	ileanCoP_pt	JProver	LJT	STRIP
	Prolog	C							
solved	112	125	90	116	234	239	96	17	17
proved	99	112	87	113	188	193	94	5	5
0 - 1 s	93	109	73	107	146	155	78	5	5
1 - 10 s	5	1	8	2	16	12	9	0	0
10 - 100 s	1	2	4	2	17	9	7	0	0
100 - 200 s	0	0	2	0	1	12	0	0	0
200 - 300 s	0	0	0	2	8	5	0	0	0
pure equal.	2	2	1	6	6	7	2	0	0
refuted	13	13	3	3	46	46	2	12	12

Table 4. Performance on the TPTP Problem Set

Table 5 shows the number of proved problems from each domain of the TPTP problem set. In the domains MGT (management), SET (set theory), and GEO (geometry) of which the problems are rated middle or high, ileanCoP performs far best. In NLP (natural language processing) ileanCoP does quite bad because clausal transformation produces too large formulas. In SYN (syntactic), which contains rather constructed and partly very simple problems, ft does best.

Beweiser	AGT	ALG	COM	GEO	GRP	HAL	LCL	MGT	NLP	PUZ	SET	SWC	SWV	SYN	total
ft Prolog	2	0	1	0	0	0	0	7	7	0	15	0	1	66	99
ft C	2	0	2	0	0	0	1	7	7	0	21	0	1	71	112
ileanSeP	5	0	0	1	0	0	0	0	3	0	19	0	1	58	87
ileanTAP	0	6	0	1	0	0	1	6	11	0	26	0	1	61	113
ileanCoP	14	5	1	6	0	0	1	21	3	2	72	0	1	62	188
ileanCoP_pt	14	6	1	7	0	0	1	24	3	2	73	0	1	61	193
JProver	0	1	0	0	0	0	1	9	7	0	14	0	1	61	94
LJT	0	0	0	0	0	0	0	0	0	0	0	0	0	5	5
STRIP	0	0	0	0	0	0	0	0	0	0	0	0	0	5	5

Table 5. Number of proved formulas for each domain of the TPTP Problem Set

5 Conclusions

The tested intuitionistic theorem provers, benchmark formulas, and results have been presented. ileanCoP has solved the highest number of problems, and is probably the most efficient intuitionistic first-order ATP system at present. This demonstrates that lean provers are not only compact but also powerful.

The connection method for clausal form seems to be more efficient than the sequent- or tableau proof methods. However, clausal form transformation often generates a very large formula that is hard to prove, and makes a conversion from the matrix proof to a sequent proof difficult. JProver applies a connection method for non-clausal form. But this procedure increases the multiplicities of the submatrices globally, which strongly inflates the matrix and results in a perfomance below that of the tableau provers. One is working on a connection method for non-clausal form that increases the multiplicities dynamically.

From the tested provers, JProver is currently the only one that is used by interactive proof assistants. We will extend some of the other ATP systems such that they yield a sequent proof and can be applied by interactive systems as well.

Besides improving the proof procedures, we also want to build ATP systems that sequentially try various proof procedures and heuristics (perhaps a few seconds each) during the overall available time. This idea seems to be promising since all tested provers solve most problems within the first seconds. A simple example is the combination of ileanCoP and ileanTAP, giving 150 seconds each, which proves 14 formulas more than ileanCoP within 300 seconds.

The tests helped compiling the ILTP-Library by determing the status and rating of the problems. The ILTP-Library is intended as an open platform for intuitionistic theorem proving, and shall be extended by problems which occur e.g. in current ATP literature, or while using interactive proof development systems.

Acknowledgements

The author would like to thank Jens Otten for many helpful suggestions and fruitful discussions.

References

[Beckert und Posegga, 1995] Bernhard Beckert and Joachim Posegga. leanTAP: Lean Tableau-based Deduction. *Journal of Automated Reasoning*, vol. 15, no. 3, pp. 339-358, 1995. http://i12www.ira.uka.de/leantap/

[Dyckhoff, 1992] Roy Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. In *Journal of Symbolic Logic*, 57, pp. 795-807, 1992. http://www.dcs.st-and.ac.uk/~rd/logic/nonmac/LJT.pl.html

[Hickey *et al.*, 2003] Jason Hickey, Aleksey Nogin, Robert L. Constable, Brian. E. Aydemir, Eli Barzilay, Yegor Bryukhov, Richard Eaton, Adam Granicz, Alexei Kopylov, Christoph Kreitz, Vladimir N. Krupski, L. Lorigo, Stephan. Schmitt, C. Witty and Xin Yu. MetaPRL - A Modular Logical Environment. In David Basin and Burkhart Wolff (Eds.) *Proceedings of the 16th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2003)*, vol. 2758 of *Lecture Notes in Computer Science*, pp. 287-303. Springer-Verlag, 2003. http://www.metaprl.org/

[Larchey-Wendling *et al.*, 2001] Dominique Larchey-Wendling, Daniel Méry and Didier Galmiche. STRIP: Structural Sharing for Efficient Proof-search. In *IJCAR 2001*. http://www.loria.ft/~larchey/STRIP/

[Otten, 1997] Jens Otten. ileanTAP: An Intuitionistic Theorem Prover. In D. Galmiche (ed.) In *Proceedings of the International Conference on Tableaux and Related Systems (TABLEAUX'97)*, LNAI 1227, S. 307-312, Springer Verlag, 1997. http://www.leancop.de/ileantap/

[Otten, 2005] Jens Otten. *Clausal Connection-Based Theorem Proving in Intuitionistic First-Order Logic*. Technical Report, University of Potsdam, to appear, 2005. http://www.leancop.de/ileancop.html

[Otten und Kreitz, 1996] Jens Otten and Christoph Kreitz. A uniform proof procedure for classical and non-classical logics. In *KI-96: Advances in Artificial Intelligence*, LNAI 1137, pp. 307-319, Springer Verlag, 1996.

[Raths *et al.*, 2005] Thomas Raths, Jens Otten and Christoph Kreitz. The ILTP Library: Benchmarking Automated Theorem Provers for Intuitionistic Logic. In *Proceedings of the International Conference on Tableaux and Related Systems (TABLEAUX'05)*, to appear 2005. http://www.cs.uni-potsdam.de/ti/iltp/

[Sahlin *et al.*, 1992] Dan Sahlin, Torkel Franzén and Seif Haridi. An Intuitionistic Predicate Logic Theorem Prover.In *Journal of Logic and Computation*, vol. 2, no. 5, pp. 619-656, 1992. http://www.sm.luth.se/~torkel/eget/ftinfo.html
Benchmarks: http://www.sm.luth.se/~torkel/eget/ft/ft1.23/benchmarks.pred

[Schmitt *et al.*, 2001] Stephan Schmitt, Lori Lorigo, Christoph Kreitz, Aleksey Nogin. JProver: Integrating Connection-based Theorem Proving into Interactive Proof Assistants. In R. Gore, A. Leitsch, T. Nipkow (eds.) *International Joint Conference on Automatic Reasoning (IJCAR 2001)*, LNAI 2083, pp. 421-426, Springer Verlag, 2001. http://cvs.metaprl.org:12000/cvsweb/metaprl/refiner/reflib/, files: jall.ml, jtunify.ml, jlogic_sig.ml Benchmarks: http://cvs.metaprl.org:12000/cvsweb/metaprl/theories/itt/jprover_tests.ml

[Sutcliff *et al.*, 1994] Geoff Sutcliffe, Christian B. Suttner and Theodor Yemenis. The TPTP Problem Library. In Alan Bundy (ed.) *Automated Deduction - CADE-12*, LNAI 814, pp. 252-266, Springer Verlag, 1994. http://www.cs.miami.edu/~tptp/