leanCoP 2.0 and ileanCoP 1.2: High Performance Lean Theorem Proving in Classical and Intuitionistic Logic (System Descriptions)

Jens Otten

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

Abstract. leanCoP is a very compact theorem prover for classical first-order logic, based on the connection (tableau) calculus and implemented in Prolog. leanCoP 2.0 enhances leanCoP 1.0 by adding regularity, lemmata, and a technique for restricting backtracking. It also provides a definitional translation into clausal form and integrates "Prolog technology" into a lean theorem prover. ileanCoP is a compact theorem prover for intuitionistic first-order logic and based on the clausal connection calculus for intuitionistic logic. ileanCoP 1.2 extends the classical prover leanCoP 2.0 by adding prefixes and a prefix unification algorithm. We present details of both implementations and evaluate their performance.

1 Introduction

Connection calculi, such as the connection calculus [3, 4], the connection tableau calculus [8, 9], and the model elimination calculus [10], are well known for their goal-oriented proof search. Several implementations that are based on these calculi have been developed, for example KoMeT [5], METEOR [1], PTTP [22], SETHEO [7], and leanCoP [16].

leanCoP is an automated theorem prover for classical first-order logic. It is a very compact Prolog implementation of the connection calculus. leanCoP 2.0 enhances leanCoP 1.0 [16] by adding regularity, lemmata, and a technique to restrict backtracking [15]. In contrast to leanCoP 1.0, the input clauses are stored in Prolog's, database, which makes it possible to use Prolog's built-in indexing mechanism. Furthermore leanCoP 2.0 provides a definitional translation into clausal form and uses a fixed strategy scheduling.

ileanCoP is an automated theorem prover for intuitionistic first-order logic and is based on the clausal connection calculus for intuitionistic logic [14]. It extends leanCoP by adding a prefix to each literal and a prefix unification algorithm [17]. ileanCoP 1.2 enhances ileanCoP 1.0 by integrating the new inference rules and techniques of leanCoP 2.0.

Details of the architecture and the implementation of both leanCoP 2.0 and ileanCoP 1.2 are presented in Section 2 and Section 3, respectively. We also present performance results on the TPTP library and the MPTP challenge.

2 leanCoP 2.0 for Classical Logic

We first describe the new search techniques of leanCoP 2.0 and provide details of the source code, before presenting some performance results.

2.1 Architecture

leanCoP [16] is based on the clausal connection (tableau) calculus [4, 9]. A derivation for a formula in clausal form is generated by first applying the start rule and then repeatedly applying the reduction or the extension rule. In each inference step a connection is identified along an active path. Iterative deepening on the length of the active path is performed. The following techniques are the main improvements of leanCoP 2.0 compared to leanCoP 1.0 [16]:

1. *Lean Prolog technology*: For all input clauses C and `L`$\in C$ of the given formula the fact `lit(L,C1,Grnd)` is stored in Prolog's database, where `C1`$=C\backslash\{$`L`$\}$ is a list and `Grnd` is `g` iff C is ground and `n` otherwise. Atoms are represented by Prolog atoms, negation is represented by "`-`". This new technique integrates the main advantage of the "Prolog technology" approach [22] by using Prolog's fast indexing mechanism to quickly find connections [15].
2. *Controlled iterative deepening*: Iterative deepening is aborted if the current limit for the length of the active path is not exceeded during the proof search.
3. *Definitional clausal form translation*: Formulae that are not in clausal form are translated into clausal form by introducing definitions for certain subformulae. In contrast to approaches for saturation-based calculi, our definitional translation is designed to work well with connection calculi [15].
4. *Regularity*: The proof search is restricted to proofs where no literal occurs more than once in the (currently investigated) active path [9].
5. *Lemmata*: A branch with a literal L that has already been closed can be reused to close other branches (below/to the right of L) that contain L [9].
6. *Restricted backtracking*: Once the application of the reduction or extension rule has successfully closed a branch by an appropriate connection, all alternative connections are cut off (on backtracking). Furthermore, alternative start clauses of the start rule are cut off (on backtracking). This new technique improves performance significantly, in particular for formulae containing many axioms, e.g. equality axioms [15].
7. *Strategy scheduling*: A list of settings is used to control the proof search (see below). The core Prolog prover is consecutively invoked by a shell script with different settings.

The minimal source code of the core prover is shown in Figure 1. It is invoked with `prove(1,S)` where `S` is a list of settings. The predicate succeeds iff there is a connection proof for the clauses stored in Prolog's database.[1] The full source code of the core prover and of the definitional clausal form translation is available on the leanCoP website. A detailed explanation of the source code, the underlying calculus and the new proof search techniques can be found in [15].

[1] Sound unification has to be switched on. In ECLiPSe Prolog this is done with `set_flag(occur_check,on)`.

```
prove(I,S) :- \+member(scut,S) -> prove([-(#)],[],I,[],S) ;
    lit(#,C,_) -> prove(C,[-(#)],I,[],S).
prove(I,S) :- member(comp(L),S), I=L -> prove(1,[]) ;
    (member(comp(_),S);retract(p)) -> J is I+1, prove(J,S).
prove([],_,_,_,_).
prove([L|C],P,I,Q,S) :- \+ (member(A,[L|C]), member(B,P),
    A==B), (-N=L;-L=N) -> ( member(D,Q), L==D ;
    member(E,P), unify_with_occurs_check(E,N) ; lit(N,F,H),
    (H=g -> true ; length(P,K), K<I -> true ;
    \+p -> assert(p), fail), prove(F,[L|P],I,Q,S) ),
    (member(cut,S) -> ! ; true), prove(C,P,I,[L|Q],S).
```

Fig. 1. The source code of the leanCoP 2.0 core prover

The list `S` of settings can contain one or more of the following options:

1. `nodef/def`: The standard/definitional translation into clausal form is done. If none of these options is given the standard translation is used for the axioms whereas the definitional translation is used for the conjecture.
2. `conj`: The special literal `#` is added to the conjecture clauses in order to mark them as possible start clauses. If this option is not given the literal `#` is added to all positive clauses to mark them as possible start clauses.
3. `reo(I)`: After the clausal form translation the clauses are reordered `I` times using a simple perfect shuffle algorithm.
4. `scut`: Backtracking is restricted for alternative start clauses.
5. `cut`: Backtracking is restricted for alternative reduction/extension steps.
6. `comp(I)`: Restricted backtracking is switched off when iterative deepening exceeds the active path length `I`.

Note that the option `conj` is complete only for formulae with a provable conjecture, and `scut` as well as `cut` are complete only if used in combination with `comp(I)`. leanCoP 2.0 uses a fixed strategy scheduling that preserves completeness. The controlled iterative deepening yields a decision procedure for ground (e.g. propositional) formulae, and refutes some (invalid) first-order formulae.

2.2 Performance

leanCoP 2.0 was tested on all 3644 problems in non-clausal form (FOF division) of version 3.3.0 of the TPTP library [24]. Problems that do not have a conjecture are negated. These are exactly those problems that are either satisfiable or unsatisfiable. Equality is dealt with by adding the equality axioms. All tests were performed on a 3 GHz Xeon system running Linux 2.6 and ECLiPSe Prolog version 5.8. The time limit was set to 600 seconds.

In Table 1 the performance of leanCoP 2.0 is compared with the performance of lean*TAP* 2.3 (the first popular lean prover) [2], leanCoP 1.0 [16], SETHEO 3.3

(likely the fastest connection tableau prover so far)[2] [7], Otter 3.3 (which commonly serves as the standard benchmark) [11] and version "Dec-2007" of Prover9 (the successor of Otter)[3] [12]. The rating and the percentage of proved problems for some rating intervals are given. FNE, FEQ and PEQ are problems without, with and containing only equality, respectively. Furthermore, the number of proved problems for each domain (see [24]) that contains at least ten problems is shown. The number of problems that are refuted, result in a time out or error, e.g. stack overflow or empty set-of-support, are listed in the last three lines.

Table 1. Performance of leanCoP 2.0 on the TPTP library

	leanTAP	leanCoP 1.0	SETHEO	Otter	leanCoP 2.0	Prover9
proved	375	1004	1192	1310	**1638**	1677
[%]	10%	28%	33%	36%	**45%**	46%
0s to 1s	351	787	864	987	**1124**	1281
1s to 10s	12	84	205	183	**123**	197
10s to 100s	11	74	62	106	**193**	141
100s to 600s	1	59	61	34	**198**	58
rating 0.0	194	397	435	455	**450**	464
rating >0.0	181	607	757	855	**1188**	1213
rating 1.0	0	0	2	7	**13**	8
0.00...0.24	22.8 %	56.2 %	63.9 %	72.2 %	**71.7 %**	72.8 %
0.25...0.49	5.9 %	26.0 %	34.2 %	39.7 %	**48.2 %**	69.9 %
0.50...0.74	2.2 %	7.1 %	8.5 %	3.0 %	**35.9 %**	28.2 %
0.75...1.00	0.4 %	0.0 %	1.5 %	0.7%	**11.2 %**	2.5 %
FNE	290	448	467	475	**491**	525
FEQ	85	556	725	835	**1147**	1152
PEQ	12	13	13	47	**30**	71
AGT	0	17	17	16	**29**	17
ALG	11	13	17	60	**32**	83
CSR	0	1	3	3	**2**	27
GEO	23	143	159	160	**171**	171
GRA	0	4	6	5	**6**	1
KRS	16	70	89	106	**105**	103
LCL	3	26	32	18	**24**	48
MED	0	0	1	5	**7**	1
MGT	11	31	41	54	**45**	62
NLP	4	3	7	6	**13**	13
NUM	1	35	59	31	**70**	49
PUZ	2	5	6	6	**6**	6
SET	25	170	197	229	**339**	276
SEU	5	125	124	149	**296**	259
SWC	14	14	65	84	**87**	101
SWV	55	142	154	157	**177**	183
SYN	201	199	205	210	**217**	267
refuted	0	1	27	0	33	0
time out	2979	2538	2134	700	1949	668
error	290	101	291	1634	24	1299

[2] For SETHEO the options -dr -reg -st were used, which showed the best performance.
[3] The most recent version "2008-04A" of Prover9 has a significant lower performance.

leanCoP 2.0 proves significantly more problems of the TPTP library than, e.g., Otter or SETHEO. The biggest improvement is made for problems with equality and problems that are rated as difficult, i.e. that have a high rating.

The MPTP challenge is a set of problems from the Mizar library translated into first-order logic [25]. The results of leanCoP 2.0 on the 252 "chainy" problems, in which irrelevant axioms and lemmata are *not* excluded, are shown in Table 2. leanCoP 2.0 deals with equality by adding the equality axioms. The resulting formulae contain up to 1700 axioms. Again the results are compared with lean*TAP* 2.3, Otter 3.3, SETHEO 3.3, leanCoP 1.0 and version "Dec-2007" of Prover9. All tests were performed on a 3 GHz Xeon system running Linux. leanCoP 2.0 solves significant more problem than all other listed systems.

Table 2. Performance of leanCoP 2.0 on the MPTP challenge (chainy)

	lean*TAP*	SETHEO	Otter	leanCoP 1.0	Prover9	leanCoP 2.0
proved	0	27	29	33	52	**88**
[%]	0%	11%	12%	13%	21%	**35%**
0s to 1s	0	15	17	14	33	**38**
1s to 10s	0	5	7	3	8	**21**
10s to 100s	0	6	5	11	6	**23**
100s to 300s	0	1	0	5	5	**6**
time out	252	225	150	219	101	164
error	0	0	73	0	99	0

leanCoP 2.0 participated in the FOF division of CASC-21, the CADE system competition. It solved more problems than four other (classical) provers, including Otter, and won the "Best Newcomer" award [23]. It solved four problems that the winning prover did not solve within the time limit of 360 seconds.

3 ileanCoP 1.2 for Intuitionistic Logic

We first provide details of the architecture and the source code of ileanCoP 1.2, before presenting performance results.

3.1 Architecture

ileanCoP is based on the clausal connection calculus for intuitionistic first-order logic [14]. It uses the classical search engine of leanCoP and an additional prefix unification algorithm [17] to unify the prefixes of the literals in every connection. This ensures that the characteristics of intuitionistic logic are respected and the given formula is intuitionistically valid (see also [6, 27, 28]).

ileanCoP 1.2 integrates all additional inference rules and search techniques of leanCoP 2.0 mentioned in Section 2.1. Like for the classical prover the input clauses are stored in Prolog's database: the fact `lit(L:Pre,C1,Grnd)` is stored

for all input clauses C and literals L$\in C$, where Pre is the prefix of L, C1$=C\backslash\{$L$\}$ is a list of literals, Grnd is g iff C is ground and n otherwise.

The main part of the minimal source code is shown in Figure 2. The underlined text was added to the source code of leanCoP 2.0 in Figure 1; no other changes were done. The prefix unification algorithm (check_addco and prefix_unify) requires another 26 lines of Prolog code (see [17, 13] for details). The full source code is available on the leanCoP website.

```
prove(I,S) :- ( \+member(scut,S) ->
    prove([(-(#)):(-[])],[],I,[],[Z,T],S) ;
    lit((#):_,G:C,_) -> prove(C,[(-(#)):(-[])],I,[],[Z,R],S),
    append(R,G,T) ), check_addco(T), prefix_unify(Z).
prove(I,S) :- member(comp(L),S), I=L -> prove(1,[]) ;
    (member(comp(_),S);retract(p)) -> J is I+1, prove(J,S).
prove([],_,_,_,[[],[]],_).
prove([L:U|C],P,I,Q,[Z,T],S) :- \+ (member(A,[L:U|C]), member(B,P),
    A==B), (-N=L;-L=N) -> ( member(D,Q), L:U==D, X=[], O=[] ;
    member(E:V,P), unify_with_occurs_check(E,N),
    \+ \+ prefix_unify([U=V]), X=[U=V], O=[] ;
    lit(N:V,M:F,H), \+ \+ prefix_unify([U=V]),
    (H=g -> true ; length(P,K), K<I -> true ;
    \+p -> assert(p), fail), prove(F,[L:U|P],I,Q,[W,R],S),
    X=[U=V|W], append(R,M,O) ), (member(cut,S) -> ! ; true),
    prove(C,P,I,[L:U|Q],[Y,J],S), append(X,Y,Z), append(J,O,T).
```

Fig. 2. The source code of the ileanCoP 1.2 core prover

Like the classical prover it is invoked with prove(1,S) where S is a list of settings (see Section 2.1). The predicate succeeds iff there is an intuitionistic connection proof for the clauses stored in Prolog's database.

3.2 Performance

ileanCoP 1.2 was tested on all 3644 problems in non-clausal form (FOF division) of version 3.3.0 of the TPTP library [24]. Formulae F that do not have a conjecture are translated to $F \Rightarrow \bot$. Equality is dealt with by adding the equality axioms. All tests were performed on a 3 GHz Xeon system running Linux 2.6 and ECLiPSe Prolog version 5.8. The time limit was set to 600 seconds.

In Table 3 the performance of ileanCoP 1.2 on the TPTP library is compared with all currently existing systems for intuitionistic first-order logic: JProver [21], the Prolog and C versions of ft [20], ileanTAP [13], ileanSeP[4] and ileanCoP 1.0 [14].[5] The figures for domains that contain at least 10 proved problems are shown.

[4] See http://www.leancop.de/ileansep/.

[5] The intuitionistic version of the Gandalf prover [26] is not included since it is neither complete nor sound (see the website of the ILTP library [19]).

Table 3. Performance of leanCoP 1.2 on the TPTP library

	JProver 11-2005	leanTAP 1.17	ft 1.23 (C)	ft 1.23 (Prolog)	leanSeP 1.0	leanCoP 1.0	leanCoP 1.2
proved	186	255	262	278	303	733	**1127**
[%]	5%	7%	7%	8%	8%	20%	**31%**
0s to 1s	171	248	258	246	208	543	**750**
1s to 10s	6	3	2	27	52	72	**80**
10s to 100s	6	1	2	0	29	73	**96**
100s to 600s	3	3	0	5	14	45	**201**
rating 0.0	147	142	156	174	160	331	**397**
rating >0.0	39	113	106	104	143	402	**730**
0.00...0.24	13.1 %	15.3 %	16.1 %	17.0 %	17.1 %	40.5 %	**54.5 %**
0.25...0.49	0.4 %	4.5 %	5.0 %	5.8 %	9.1 %	20.9 %	**34.1 %**
0.50...0.74	0.0 %	1.2 %	0.2 %	0.0 %	0.0 %	4.0 %	**20.2 %**
0.75...1.00	0.0 %	0.3 %	0.2 %	0.0 %	0.0 %	0.0 %	**2.3 %**
FNE	180	175	194	209	178	312	**371**
FEQ	6	80	68	69	125	421	**756**
PEQ	5	6	4	2	1	8	**19**
AGT	0	0	2	2	5	13	**18**
ALG	4	6	2	0	0	7	**15**
GEO	1	8	7	29	36	132	**154**
KRS	33	19	26	26	18	42	**94**
LCL	0	1	1	0	0	22	**22**
MGT	7	6	7	9	0	24	**30**
NLP	7	11	7	7	3	3	**11**
NUM	0	2	1	0	1	30	**58**
SET	18	32	29	24	32	120	**222**
SEU	2	7	10	10	10	83	**200**
SWV	1	51	46	54	89	135	**162**
SYN	108	106	115	110	105	107	**121**
refuted	4	4	15	0	4	73	71
time out	2931	3344	852	2993	3161	2732	2355
error	523	41	2515	373	176	106	91

Table 4. Performance of leanCoP 1.2 on the MPTP challenge (chainy)

	JProver 11-2005	leanTAP 1.17	ft 1.23 (Prolog)	ft 1.23 (C)	leanSeP 1.0	leanCoP 1.0	leanCoP 1.2
proved	0	0	0	1	2	19	**61**
[%]	0%	0%	0%	<1%	<1%	8%	**24%**
0s to 1s	0	0	0	1	0	4	**24**
1s to 10s	0	0	0	0	0	6	**20**
10s to 100s	0	0	0	0	2	4	**14**
100s to 300s	0	0	0	0	0	5	**3**
time out	235	252	58	9	250	233	191
error	17	0	194	242	0	0	0

The results of ileanCoP 1.2 on the 252 "chainy" problems of the MPTP challenge (see Section 2.2) are shown in Table 4.

ileanCoP 1.2 proves significantly more problems of the TPTP library and the MPTP challenge than any of the other systems. It even proves more problems of the AGT and NUM domain and of the MPTP challenge than Prover9, though intuitionistic logic is considered more difficult than classical logic and not all of those problems that are classical theorems are valid in intuitionistic logic.

4 Conclusion

We have presented leanCoP 2.0 and ileanCoP 1.2, theorem provers for classical and intuitionistic first-order logic. leanCoP 2.0 uses a few selected (well-known and new) empirical successful techniques for pruning the search space in connection calculi. By adding a prefix unification algorithm it is turned into the intuitionistic prover ileanCoP 1.2. Performance of both provers is in particular good for problems that contain equality or a large number of axioms. ileanCoP 1.2 achieves a performance that can even compete with some classical provers.

randoCoP [18] is an extension of leanCoP 2.0 and randomly reorders the axioms and literals of the given formula. It compensates for the drawback of restricted backtracking, which might narrow the search space too much. Repeatedly applying this reordering technique improves performance significantly.

Future work include the integration of further proof search techniques and the adaption to other non-classical logics, like some first-order modal logics [6].

The complete source code of leanCoP 2.0 and ileanCoP 1.2 together with more information is available at `http://www.leancop.de`.

Acknowledgements. Thomas Raths contributed to the development of both leanCoP 2.0 and ileanCoP 1.2 by running comprehensive benchmark tests on several problem libraries. Geoff Sutcliffe kindly provided the SETHEO system.

References

1. Astrachan, O., Loveland, D.: METEORs: High Performance Theorem Provers Using Model Elimination. In: Bledsoe, W. W., Boyer, S. (eds.) *Automated Reasoning: Essays in Honor of Woody Bledsoe*, pp. 31–60. Kluwer, Amsterdam (1991).
2. Beckert, B., Posegga, J.: lean*TAP*: Lean Tableau-Based Theorem Proving. In: Bundy, A. (ed.) *CADE-12*. LNAI, vol. 814, pp. 793–797. Springer, Heidelberg (1994).
3. Bibel, W.: Matings in Matrices. *Commun. ACM* 26, 844–852 (1983).
4. Bibel, W.: *Automated Theorem Proving*. Vieweg, Wiesbaden (1987).
5. Bibel, W., Brüning, S., Egly, U., Rath, T.: KoMeT. In: Bundy, A. (ed.) *CADE-12*. LNAI, vol. 814, pp. 783–787. Springer, Heidelberg (1994).
6. Kreitz, C., Otten, J.: Connection-based Theorem Proving in Classical and Non-classical Logics. *Journal of Universal Computer Science* 5, 88–112 (1999).
7. Letz, R., Schumann, J., Bayerl, S., Bibel, W.: SETHEO: A High-Performance Theorem Prover. *Journal of Automated Reasoning* 8, 183–212 (1992).

8. Letz, R., Mayr, K., Goller, C.: Controlled Integration of the Cut Rule into Connection Tableaux Calculi. *Journal of Automated Reasoning* 13, 297–337 (1994).
9. Letz, R., Stenz, G.: Model Elimination and Connection Tableau Procedures. In: Robinson, A., Voronkov, A. (eds.) *Handbook of Automated Reasoning*, pp. 2015–2114. Elsevier, Amsterdam (2001).
10. Loveland, D.: Mechanical Theorem-Proving by Model Elimination. *Journal of the ACM* 15, 236–251 (1968).
11. McCune, W.: Otter 3.0 Reference Manual and Guide. Technical report ANL-94/6, Argonne National Laboratory (1994).
12. McCune, W.: Release of Prover9. *Mile High Conference on Quasigroups, Loops and Nonassociative Systems*, Technical report, Denver (2005).
13. Otten, J.: ileanTAP: An Intuitionistic Theorem Prover. In: Galmiche, D. (ed.) *TABLEAUX '97*. LNAI, vol. 1227, pp. 307–312. Springer, Heidelberg (1997).
14. Otten, J.: Clausal Connection-Based Theorem Proving in Intuitionistic First-Order Logic. In: Beckert, B. (ed.) *TABLEAUX 2005*. LNAI, vol. 3702, pp. 245–261. Springer, Heidelberg (2005).
15. Otten, J.: Restricting Backtracking in Connection Calculi. Technical report, Institut für Informatik, University of Potsdam (2008).
16. Otten, J., Bibel, W.: leanCoP: Lean Connection-based Theorem Proving. *Journal of Symbolic Computation* 36, 139–161 (2003).
17. Otten, J., Kreitz, C.: T-String-Unification: Unifying Prefixes in Non-classical Proof Methods. In: Miglioli, P., Moscato, U., Mundici, D., Ornaghi, M. (eds.) *TABLEAUX '96*. LNAI, vol. 1071, pp. 244–260. Springer, Heidelberg (1996).
18. Raths, T., Otten, J.: randoCoP: Randomizing the Proof Search Order in the Connection Calculus. Technical report, Institut für Informatik, University of Potsdam (2008).
19. Raths, T., Otten, J., Kreitz, C.: The ILTP Problem Library for Intuitionistic Logic. *Journal of Automated Reasoning* 38, 261–271 (2007).
20. Sahlin, D., Franzen, T., Haridi, S.: An Intuitionistic Predicate Logic Theorem Prover. *Journal of Logic and Computation* 2, 619–656 (1992).
21. Schmitt, S., Lorigo, L., Kreitz, C., Nogin, A.: JProver: Integrating Connection-based Theorem Proving into Interactive Proof Assistants. In: Goré, R., Leitsch, A., Nipkow, T. (eds.) *IJCAR 2001*. LNAI, vol. 2083, pp. 421–426. Springer, Heidelberg (2001).
22. Stickel, M.: A Prolog Technology Theorem Prover: Implementation by an Extended Prolog Compiler. *Journal of Automated Reasoning* 4, 353–380 (1988).
23. Sutcliffe, G.: The CADE-21 Automated Theorem Proving System Competition. *AI Communications* 21, 71–81 (2008).
24. Sutcliffe, G., Suttner, C.: The TPTP Problem Library. *Journal of Automated Reasoning* 21: 177–203 (1998).
25. Urban, J.: MPTP 0.2: Design, Implementation, and Initial Experiments. *Journal of Automated Reasoning* 37, 21–43 (2006).
26. Tammet, T.: A Resolution Theorem Prover for Intuitionistic Logic. In: McRobbie, M., Slaney, J. (eds.) *CADE-13*. LNAI, vol. 1104, pp. 2–16. Springer, Heidelberg (1996).
27. Waaler, A.: Connections in Nonclassical Logics. In: Robinson, A., Voronkov, A. (eds.) *Handbook of Automated Reasoning*, pp. 1487–1578. Elsevier, Amsterdam (2001).
28. Wallen, L.: *Automated Deduction in Nonclassical Logics*. MIT Press, Cambridge (1990).