Using classical theorem-proving techniques for approximate reasoning

Stefan Brüning
FG Intellektik, TH Darmstadt
Alexanderstraße 10
D-64283 Darmstadt
stebr@intellektik.informatik.th-darmstadt.de

Torsten Schaub
IRISA
Campus de Beaulieu
F-35042 Rennes Cedex
torsten@irisa.fr

Abstract

We propose an approach to approximate reasoning based on well-known theorem-proving techniques. Unlike other approaches, our approach takes into account the interplay of knowledge bases and queries and thus allows for query-sensitive approximate reasoning. We demonstrate that our approach deals extremely well with the examples found in the literature. This reveals that conventional theorem-proving techniques can account for approximate reasoning.

1 Introduction

Reasoning processes are usually formalized in logical systems like first-order logic. But even though this logical approach offers several indispensable benefits, problem solving in logical systems is usually intractable. For instance, query-answering in propositional logic is co-NP-complete [5].

Recently, a couple of approaches to approximate reasoning in logical systems have been proposed in order to overcome this gap. In [12] a method for *knowledge compilation* is proposed. The idea is to compile a knowledge base (KB) represented as a propositional formula into a logically stronger and a logically weaker Horn-formula for which query-answering is polynomial [4]. [2] extend the idea of *limited inference* found in [7] by proposing a stronger and a weaker version of propositional entailment. The idea is to restrict classical satisfiability to a certain subset of the language. This leads to an extended notion of interpretations in which some propositions and their negation may be both true or both false. Both approaches offer a *complete* and a *sound* approximation of propositional entailment. However, both methods cannot be directly mapped onto existing automated theorem provers. In the first case, one needs special-purpose algorithms for compilation. Moreover, this approach might be to rigid whenever we are faced with rapidly changing KBs. In the second case, one has to deal with non-classical entailment relations requiring non-standard proof procedures. In addition, it is still unclear how one has to partition the language for determining the non-classical entailment relations.

In what follows, we introduce an approach to approximate reasoning that relies on existing theorem-proving techniques. The idea is to take an existing calculus for propositional (or even first-order) logic which can be decomposed into a "fast" and a "slower" part; thereby the "fast" (and possibly incomplete) submethod is enriched by the "slower" one such that the entire method is complete. This enrichment should allow for a stepwise approximation that finally meets the entire (and complete) method. A prime candidate for this purpose is the combination of *unit-resulting resolution* (URR) [13] and *reasoning by cases* (CASE). Together, both methods provide a sound and complete calculus for propositional and first-order logic. The "fast" part is taken up by URR. Roughly speaking, URR is a restriction of hyper-resolutions to those yielding unit clauses only [11]. Notably, URR allows for polynomial query-answering beyond the class of Horn-formulas. Moreover, URR constitutes a special purpose but high-speed reasoning mode in high-performance theorem provers like OTTER [9]. The "slower" part is played by CASE. As regards approximate reasoning, our intended purpose of CASE is (roughly) to enable URR whenever the latter is inapplicable. In such a case, CASE splits clauses into several ones that then allow for URR. In our approach, the control of such CASE-steps is accomplished by so-called *reachability relations*, which also constitute a well-known concept in automated theorem proving. These relations provide means for accessing the part of a KB relevant to a given query. This approach takes into account the

interplay of KBs and queries and thus allows for query-sensitive approximate reasoning.

2 Background

We consider propositional formulas in conjunctive normal form. In this case, a KB can be represented as a set of clauses, where a clause is a disjunction of literals. We verify whether a query q is entailed by a KB Σ by checking whether the set of clauses representing $\Sigma \wedge \neg q$ is unsatisfiable.

URR restricts derivable clauses to *unit clauses* (ie. clauses with a single literal) and the *empty clause* $\Box$. Let L^d denote the negation of the literal L.

Definition 2.1 *Let $C = \{L_1, \ldots, L_n\}$ and $C_1 = \{L_1^d\}, \ldots, C_m = \{L_m^d\}$ be clauses. Define*

$$\mathsf{URR}(C, C_1, \ldots, C_m) = \begin{cases} \{L_m\} & \text{if } n = m+1 \\ \Box & \text{if } n = m \end{cases}$$

For a set of clauses S, define $\mathsf{URR}^0(S) = S$ and $\mathsf{URR}^{i+1}(S) = \mathsf{URR}^i(S) \cup \{C \mid C = \mathsf{URR}(C_1, \ldots, C_n),\ C_i \in \mathsf{URR}^i(S) \text{ for } 1 \leq i \leq n\}$. Then, a clause C is derivable from S by URR iff $C \in \bigcup_i \mathsf{URR}^i(S)$.

A query q is entailed by a KB Σ if $\Box$ is derivable by URR from the clauses representing $\Sigma \wedge \neg q$. Consider the KB consisting of clauses (1) to (4) along with the query D resulting in clause (5):

$$(1)\ \{\neg A, B\}, \quad (2)\ \{A, B, \neg C\},$$
$$(3)\ \{\neg B, \neg C, D\},\ (4)\ \{C\},\ (5)\ \{\neg D\}$$

The sole possibility to start with is to perform URR with clauses (3), (4), and (5), resulting in the new unit clause (6) $\{\neg B\}$. Further URR involving clause (1) and (6) yields (7) $\{\neg A\}$. Now, the empty clause is derivable via (2), (4), (6), and (7). Hence, clauses (1) to (5) are unsatisfiable and so D is entailed by clauses (1) to (4).

A set of clauses can be shown to be unsatisfiable by URR iff it contains an unsatisfiable subset of clauses which is *renamable-Horn* (ie. switching signs of literals in a suitable way yields a Horn-clause set [8]). In the propositional case, the satisfiability of such a set S can be decided in $O(n^2)$ time and $O(n)$ space, where n is the number of literals occurring in S.

But URR relies on the existence of unit-clauses. So, we might sometimes be enforced to reason by cases before applying URR. In fact, we obtain a complete inference mechanism by adding the concept of case-analysis [3]:

Definition 2.2 *Let $S = \{C_1, \ldots, C_m\}$ be a clause set and let $\langle S_i \rangle_{i \in I}$ be a family of clause sets such that $\bigcup_{i \in I} S_i \models S$ and $S_i = \{C_{i_1}, \ldots, C_{i_m}\}$ with $C_{i_j} \subseteq C_j$ for $1 \leq j \leq n$ and $i \in I$. Then, S is satisfiable if S_i is satisfiable for some $i \in I$.*

Observe that each case S_i is strictly smaller (in the number of occurring literals) than S. In fact, this definition is a very general one and thus admits various instantiations (see Section 3 and 4).

Consider the clause set obtained by replacing clause (4) by $(4')\{C, E\}$ in the above example. Now, URR is not applicable. However, we may generate two cases by splitting clause $(4')$. In the first case, we delete literal E in $(4')$, which allows for deriving the empty clause by URR, as shown above. The second case, obtained by deleting C in $(4')$, however yields a satisfiable set of clauses which is perceivable after further CASE-steps. This implies that the original set of clauses is satisfiable. Hence D is not entailed by the KB consisting of $(1), (2), (3)$, and $(4')$.

The task of approximation can also be seen as identifying the relevant information for (dis)proving queries. The notion of relevance relies on the interplay between KBs and queries. In our approach this is accomplished by means of a *reachability relation*[1] that allows for measuring the inferential distance between queries and clauses in the KB.

Definition 2.3 (Reachability) *Let S be a clause set and let L be a literal occurring in a clause C of S. The set of literals reachable from L is defined inductively as follows:*[2]

1. *Any complementary literal L^d in a clause $C' \in S$ is reachable from L if $C \neq C'$.*
2. *If $E = \{K, L_1, \ldots, L_n\}$ is a clause of S and K is reachable from L, then any literal L_i^d contained in a clause $E' \in S$ is reachable from L if $E \neq E'$, where $1 \leq i \leq n$. We call K an* ancestor *of each L_i^d.*

Roughly, a literal K is reachable from a literal L if the derivation of K might contribute to a (resolution based) derivation of L.

For approximate reasoning, we proceed by defining two concepts for measuring the relevance of literals occurring in a KB to putative queries. That is, we define the notions of *distance* and *weight* of a literal K to another literal L. Both measures are

[1] Reachability was introduced in [10] in a modified way.

[2] Recall that L^d denotes the negation of the literal L.

incrementally computed for each literal while parsing a KB. In this way, we can easily compute both measures for each query in turn.

Definition 2.4 *Let S be a clause set and let L and K be distinct literals occurring in S such that K is reachable from L.*
The distance of K to L, $d(L,K)$, is defined as the minimal length of a sequence $K_1,\ldots,K_n$ of literals such that $K_1^d = L$, $K_n = K$ and K_i is an ancestor of K_{i+1} for $1 \leq i < n$. We set $d(L,L) = 0$ and $d(L,K) = \infty$ if K is not reachable from L.
The weight of K to L, $w(L,K)$, is defined as the number of sequences of different[3] literals $K_1,\ldots,K_n$ such that $K_1^d = L$, $K_n = K$ and K_i is an ancestor of K_{i+1} for $1 \leq i < n$. We set $w(L,L) = \infty$ and $w(L,K) = 0$ if K is not reachable from L.

In the above example, we obtain $d(\neg D, C) = 2$ from the minimal sequence D, C. Accordingly, we know that while proving $\neg D$, we might have to use clause (4) after 2 derivation steps. Thus, if the distance of a literal K to a literal L is small, K is likely to constitute an important part of a proof of L. We have $w(\neg D, C) = 3$ in the above example. This tells us that there are three different derivations starting with the goal $\neg D$ using clause (4). Thus, if the weight of K to L is high, the chance of using K in a proof of L is also high.

Given two clauses c_1 and c_2, the weight of c_2 to c_1 is defined as the *maximal weight* of a literal in c_2 to a literal in c_1. Similarly, the distance of c_2 to c_1 is defined as the *minimal distance* of a literal in c_2 to a literal in c_1.

3 Correct approximate reasoning

We introduce a correct but incomplete method for approximate reasoning. That is, if a query can be answered approximately, then it is also entailed by the given KB but not vice versa. The idea is to process a set of clauses representing the KB and the negated query by URR as long as possible. If the resulting set of clauses contains the empty clause, it is unsatisfiable, and we are done. If not, we perform reasoning by cases via CASE and then return to URR. The (possibly exponential) number of CASE-inferences is bound and thus serves as a parameter for an approximation. This leads to an overall complexity of $O(n^2 \times p)$, where n is the number of literals in the KB and p is the number of permissible CASE-inferences. Since p is fixed, our method in polynomial in n.

Clearly, we have to make precise how cases are generated. For the purpose of correct (but incomplete) approximate reasoning, it is sufficient to select a clause $C = \{L_1,\ldots,L_n\}$ from the actual clause set S and to generate the cases $S_i = (S \setminus C) \cup \{L_i\}$. The unsatisfiability of S then follows from the unsatisfiability of the S_i.

While testing this approach on the examples in the literature, the most striking observation is that they can be solved without reasoning by cases. This substantiates the use of URR as the fundamental inference mechanism. Its power helps to minimize the need for reasoning by cases for many practical problems. As a first example, consider the KB Σ represented by clauses (1) to (7). In [6], Σ is used to demonstrate that the compiled Horn-KB for correct approximate reasoning may be of exponential size, which in turn leads to exponential query-answering relative to the original KB. In our approach, however, any query q to Σ is solvable in polynomial time provided the clausal representation of $\Sigma \wedge \neg q$ is renamable-Horn (note, that Σ itself is renamable-Horn). In this case, we can answer any query by pure URR without any case-analysis. Consider how the query ($CompSci \wedge ReadsDennet \wedge ReadsKosslyn \rightarrow CogSci$) represented by clause (5) to (8) (labeled by Q) is answered in our approach by means of the URR-steps (9) to (12):

(1) KB $\{\neg CompSci, \neg Phil, \neg Psych, CogSci\}$
(2) KB $\{\neg ReadsMcCarthy, CompSci, CogSci\}$
(3) KB $\{\neg ReadsDennet, Phil, CogSci\}$
(4) KB $\{\neg ReadsKosslyn, Psych, CogSci\}$
(5) Q $\{CompSci\}$
(6) Q $\{ReadsDennet\}$
(7) Q $\{ReadsKosslyn\}$
(8) Q $\{\neg CogSci\}$

(9) URR(4, 7, 8) $\{Psych\}$
(10) URR(3, 6, 8) $\{Phil\}$
(11) URR(1, 8, 9, 10) $\{\neg CompSci\}$
(12) URR(5, 11) $\square$

The next KB given by clause (1) to (8) along with the query ($cow \rightarrow molar\text{-}teeth$) given in (9) and (10) are taken from [2]. Again, this clause set can be decided employing URR only even though it is non-Horn:

(1) KB $\{\neg cow, grass\text{-}eater\}$
(2) KB $\{\neg dog, carnivore\}$
(3) KB $\{\neg grass\text{-}eater, \neg canine\text{-}teeth\}$
(4) KB $\{\neg grass\text{-}eater, mammal\}$
(5) KB $\{\neg carnivore, mammal\}$
(6) KB $\{\neg mammal, canine\text{-}teeth, molar\text{-}teeth\}$
(7) KB $\{\neg mammal, vertebrate\}$
(8) KB $\{\neg vertebrate, animal\}$

[3] Note that we do not have to consider sequences containing multiple occurrences of a literal. A derivation corresponding to such a sequence can always be pruned.

(9) Q $\{cow\}$
(10) Q $\{\neg molar\text{-}teeth\}$

(11) URR$(1, 9)$ $\{grass\text{-}eater\}$
(12) URR$(3, 11)$ $\{\neg canine\text{-}teeth\}$
(13) URR$(4, 11)$ $\{mammal\}$
(14) URR$(6, 12, 13)$ $\{molar\text{-}teeth\}$
(15) URR$(10, 14)$ $\square$

In [2], a certain subset of the language, namely $\{cow, grass\text{-}eater, mammal, canine\text{-}teeth, molar\text{-}teeth\}$, has to be determined (in a non-obvious manner) in order to yield the same result.

In general, the selection of a clause for reasoning by cases is the point where we have to look for relevant information. In fact, we pick clauses for case-analysis in a "query-oriented manner". That is, we select clauses which seem to be relevant for solving a query by choosing clauses that probably contribute to a successful derivation. This is accomplished by means of distance and weight: A clause is selected for case-analysis if its distance to a clause of the query is low and its corresponding weight is high.

Consider for instance the following KB containing train connections. An implication $A \rightarrow B$ means that there is a train connection from A to B.

(1) KB $\{\neg Rome, Pisa\}$
(2) KB $\{\neg Paris, Rome\}$
(3) KB $\{\neg Berlin, Paris\}$
(4) KB $\{\neg Warsaw, Berlin\}$
(5) KB $\{\neg Warsaw, Moskow\}$

Suppose that we have the query

$$\{\{Paris, Berlin\}, \{\neg Rome, \neg Pisa\}\}$$

asking whether there is a train connection from *Paris* or *Berlin* to *Rome* and *Pisa*. This query is derivable from the first three clauses whereas the two last clauses are of no importance. Initially, URR is not applicable and therefore case-analysis has to be performed. Now we may split any of the above clauses. But splitting clause (5) is totally useless since it does not contribute to a successful derivation. This is recognized by the concept of reachability. No literal in clause (5) is reachable. Thus the distance of clause (5) to a clause of the query is ∞. Correspondingly the weight of clause (5) to a query-clause is 0. Since the clauses most relevant to a query are the clauses encoding the query (the distance of such clauses is equal to 0, and the weight equals ∞), the clause $\{Paris, Berlin\}$ is (besides $\{\neg Rome, \neg Pisa\}$) the best choice for case-analysis. This yields two clause sets, one containing the clause $\{Paris\}$ and another containing $\{Berlin\}$. Both sets can be shown to be unsatisfiable by URR. Thus, the original clause set is unsatisfiable and the query is provable by means of a single case-analysis.

4 Complete approximate reasoning

We describe a complete but incorrect method for approximate reasoning. That is, if a query cannot be answered approximately, then it is neither entailed by the given KB. Here, we are thus interested in disproving queries in an approximate way. The basic idea is as follows. First, case-analysis is performed on a set of clauses representing the KB and the negated query. This yields a set of cases. Afterwards the satisfiability of one (or several) such cases is tested. If one of these cases is satisfiable, the original clause set is satisfiable and the query is disproved. Again, our method is parameterized by the number of CASE-inferences p, which in turn leads to a complexity of $O(n^2 \times p)$ being polynomial in the number of literals in the KB, n.

First of all, let us make precise how such cases are generated. For complete approximate reasoning, it suffices to turn sets of clauses into certain sets of Horn-clauses. This offers several advantages. First, Horn-clauses are a prime candidate for fast URR. Second, we can decide the satisfiability of a set of Horn-clauses purely by URR, without any case-analysis. We generate Horn-cases by deleting all but one positive literal from each non-Horn-clause. In this way, a Horn-case consists of maximal non-empty Horn-clauses. Since the disjunction of all such Horn-cases is equivalent[4] to the original set of clauses, our approach meets propositional entailment whenever we consider the entire set of Horn-cases.

The possibly exponential number of Horn-cases in the worst case is controlled by the parameter p limiting the number of CASE-inferences. The major problem is to identify the Horn-cases most relevant for disproving the given query. We address this problem by means of the notion of reachability and its induced measures. Recall that we are looking for satisfiable cases. We thus try to turn the KB into a case *not* entailing the query. Candidates for such cases are generated by removing positive literals from non-Horn-clauses which are (possibly) relevant for deriving the query.[5] In terms of our measures, we select literals with either a minimal distance or a maximal weight.

[4] This is a corollary to Theorem 3 in [6].

[5] Without loss of generality, we assume that a query is a set of unit clauses.

Consider the following KB taken from [2]:

(1) KB $\{\neg person, child, youngster, adult, senior\}$
(2) KB $\{\neg youngster, student, worker\}$
(3) KB $\{\neg adult, student, worker, unemployed\}$
(4) KB $\{\neg senior, pensioner, worker\}$
(5) KB $\{\neg student, child, youngster, adult\}$
(6) KB $\{\neg pensioner, senior\}$
(7) KB $\{\neg pensioner, \neg student\}$
(8) KB $\{\neg pensioner, \neg worker\}$

The goal is to show that the query ($child \rightarrow pensioner$) is not derivable from this set of clauses. Accordingly, we have to show that the clause set S containing (1) to (8) and $\{child\}$ and $\{\neg pensioner\}$ is satisfiable. To this end, we have to generate a set of Horn-clauses by deleting positive literals relevant to the query. We proceed again in a "query-oriented manner". First, consider the query-clause $\{child\}$. Since no literal in S is reachable from *child*, no literal in S is relevant to this part of the query. Therefore, $\{child\}$ does not influence the deletion of literals in S. This is different for the second query-clause, $\{\neg pensioner\}$. The literal most relevant to this part of the query is obviously *pensioner* in (4); its distance to $\neg pensioner$ is 1 rendering it a prime candidate for deletion. Then, turning (4) into a Horn-clause by removing the literal *pensioner* yields $\{\neg senior, worker\}$. Now, there is no way left for deriving *pensioner*. Since no clause apart from (4) allows for deriving *pensioner*, we thus may remove positive literals (to a single one) from the remaining clauses in an arbitrary way. In any case, the resulting set of Horn-clauses will be satisfiable and so the query is disproved.

5 Discussion

In [12], a KB Σ is translated into two Horn-KBs Σ_{glb} and Σ_{lub} guaranteeing correct but incomplete and complete but incorrect inferences wrt Σ. Σ_{glb} is referred to as the *greatest lower bound* (GLB) of Σ since it is required to be a Horn-formula with a maximum set of models $\mathcal{M}(\Sigma_{glb}) \subseteq \mathcal{M}(\Sigma)$. Accordingly, the *lowest upper bound* (LUB) of Σ, Σ_{lub}, is a Horn-formula with a minimum set of models $\mathcal{M}(\Sigma_{lub}) \supseteq \mathcal{M}(\Sigma)$. The computation of such Horn-approximations is NP-hard. Hence the idea is to compute them "off-line" in order to allow for polynomial "on-line" approximations wrt Σ_{glb} and Σ_{lub}. In general, there are many GLBs whereas there is only one LUB of Σ. Also, in general, the size of Σ_{glb} is linear whereas the size of Σ_{lub} is exponential in the size of Σ.

Let us first deal with the correct (but incomplete) case. The exponential size of LUBs is a serious drawback: First, it leads to exponential query-answering relative to the original KB. Second, it may even be impossible to store the resulting LUB. This difficulty applies even to genuine examples like the one given in Section 3 on a person's reading habits. In this example, the LUB is of exponential size as shown in [6]. In contrast, we have seen that our methods behaves polynomially in this example if the clausal representation of the KB and the negation of the query is renamable-Horn. In general, we control the exponential factor by limiting the number of CASE-inferences. So in the unlimited case, our method corresponds to propositional entailment, which in turn may require exponentially many CASE-inferences. Finally, finding a LUB is a problem that cannot be parallelized [1] while case-analysis is (in general) perfectly suited for parallelization.

In the complete (but incorrect) case, our approach resembles the one taken in [6]. In both approaches, the original KB is approximated by regarding some of its Horn-variants. That is, a case of the KB corresponds to one of its lower bounds. So the difference rests on how these Horn-variants are chosen. In [6], all approximations are compared in an anytime manner. Initially, an arbitrary lower bound LB_1 is selected. Then, it is tested whether another lower bound LB_2 is entailed by LB_1. If $\mathrm{LB}_1 \models \mathrm{LB}_2$, then LB_1 is replaced by LB_2. This continues as long as no weaker lower bound can be found. Apart from the fact that comparing two lower bounds is NP-hard, this approach suffers from the problem that many lower bounds are not even comparable, ie. neither $\mathrm{LB}_1 \models \mathrm{LB}_2$ nor $\mathrm{LB}_2 \models \mathrm{LB}_1$ holds. Recall that this results in multiple GLBs. Moreover, it is then extremely difficult to select the right GLB being appropriately "weak" for each query in turn. Since we pursue an "on-line" approach, we cannot afford the aforementioned difficulties. Therefore, we leave it to our reachability measures to select the Horn-cases most relevant to the given query. The price we pay for avoiding computationally expensive comparisons of approximations is that the number of Horn-cases may exceed the number of GLBs. However, our approach is (in general) extremely flexible in adjusting the selection of approximations wrt to the given query.

Now, let us turn to the approach proposed in [2] which relies on two extended notions of interpre-

tations (on literals[6]), *S-1-* and *S-3-interpretations*. For a subset S of the underlying alphabet L, an *S-1*-interpretation maps every letter in $L \setminus S$ and its negation into *false*; and an *S-3*-interpretation does not map both a letter in $L \setminus S$ and its negation into *false*. The letters in S are treated in the standard way. The two notions of entailment, $\models^1_S$ and $\models^3_S$, are defined in the usual way. It is instructive to verify that $\models^1_S$ is complete but incorrect and that $\models^3_S$ is correct but incomplete. Both entailment relations are parameterized by S and coincide with standard entailment in the case of $S = L$. For a KB Σ, this approach has in both cases an overall complexity of $O(|\Sigma| \times |S| \times 2^{|S|})$ [2].

As in our approach, the exponential factor in the approximation is controlled by the parameter of the approximation. In both approaches, this allows for a stepwise convergence to standard entailment. Apart from its appealing duality, we note that the approach of [2] is semantically well-founded. The extremely problematic point is the choice of S, the letters treated in the standard way. In fact, there are no means for determining an S appropriate for answering a particular query. In our approach, we solve a similar problem by reachability relations providing means for accessing the part of a KB relevant for answering a given query. This technique should also be applicable for choosing S. But even without reachability relations, we can identify classes of formulas for which our method is polynomial, which is not the case in [2]. This is due to the fact that our basic inference mechanism, URR is much more powerful than the ones corresponding to $\models^1_S$ and $\models^3_S$, in the case of $S = \emptyset$.

6 Conclusion

We have proposed an approach to approximate reasoning which relies on the combination of unit-resulting resolution and reasoning by cases; thereby taking advantage of well-known theorem-proving techniques. This has revealed that conventional theorem-proving techniques can indeed account for approximate reasoning. We have shown how approximate reasoning can be controlled by means of reachability relations — another well-known theorem-proving technique — and their induced measures. This approach takes into account the interplay of KBs and queries and thus allows for query-sensitive approximate reasoning. Our method is orthogonal to the ones found in the literature inasmuch it is neither compiling KBs as [12] nor restricting satisfiability to certain subsets of the language as [2]. In particular, we have demonstrated that our method performs very well on the examples found in the literature. However, it remains future work to test ours as well as other approaches on large-scaled KBs.

[6] That is, a standard interpretation maps each letter and its negation into opposite truth-values.

Acknowledgments

The first author is supported by DFG grant no. Bi 228/6-2. The second author is supported by CEC grant no. ERB4001GT922433.

References

[1] M. Cadoli. Semantical and computational aspects of horn approximations. In *Proc. of IJCAI*, 39–45, 1993.

[2] M. Cadoli and M. Schaerf. Approximate Entailment. In *Proc. of the Conf. of the Ital. Ass. for AI*, 68–77. Springer, 1991.

[3] C. Chang. The decomposition principle for theorem proving systems. In *Allerton Conf. on Circuit and System Theory*, 20–28, U. of Illinois, 1972.

[4] W. P. Dowling and J. P. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Jour. of Logic Programming*, 1:267–284, 1984.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[6] H. Kautz and B. Selman. Forming Concepts for Fast Inference. In *Proc. of AAAI*, 1992.

[7] H. Levesque. Logic and the complexity of reasoning. *Jour. of Phil. Logic*, 17:355–389, 1988.

[8] H. R. Lewis. Renaming a Set of Clauses as a Horn Set. *Jour. of the ACM*, 25(1):134–135, 1978.

[9] W. McCune. Otter users' guide. Argonne Nat'l Laboratories, Argonne, 1988.

[10] G. Neugebauer. From Horn Clauses to First Order Logic: A Graceful Ascent. AIDA–92–21, TH Darmstadt, 1992.

[11] J. A. Robinson. Automatic deduction with hyper-resolution. *Jour. of Computer Math.*, 1:227–234, 1965.

[12] B. Selman and H. Kautz. Knowlede Compilation Using Horn Approximations. In *Proc. of AAAI*, 1991.

[13] L. Wos, R. Overbeek, E. Lusk, and J. Boyle. *Automated Reasoning, Introduction and Applications.* Prentice-Hall, 1984.