2000 12th IEEE Internationals Conference on Tools with Artificial Intelligence

Proceedings

12th IEEE Internationals Conference on Tools with Artificial Intelligence

ICTAI 2000

November 13–15 2000
Vancouver, British Columbia, Canada

Sponsored by:
IEEE Computer Society

IEEE COMPUTER SOCIETY

Los Alamitos, California
Washington • Brussels • Tokyo

IEEE Computer Society Order Number PR00909
ISBN 0-7695-0909-6
ISBN 0-7695-0910-X (case)
ISBN 0-7695-0911-8 (microfiche)
ISSN 1082-3409

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 714 821 8380
Fax: + 1 714 821 4641
http://computer.org/
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

Editorial production by Frances M. Titsworth

Cover art production by Joseph Daigle/Studio Productions

Printed in the United States of America by The Printing House

IEEE
COMPUTER
SOCIETY

Proceedings

12th IEEE Internationals Conference on Tools with Artificial Intelligence

ICTAI 2000

General Chairs' Foreword

On behalf of the ICTAI-2000 chairs and committees, we welcome you to the 12th International IEEE Conference on Tools with Artificial Intelligence (ICTAI-2000). We hope that this meeting to be prolific and challenging for your present and future research activities.

The ICTAI series always provides a unique opportunity to researchers, scientists and practitioners to share knowledge and scientific achievements in the important research field of Artificial Intelligence and present tools (architectures, languages and algorithms) for solving scientific and engineering problems.

The ICTAI-2000 topics of interest include: Neural Networks, Machine Learning, Software Engineering and AI, Intelligent Information Retrieval, Optimization, Constraints Satisfaction, Multimedia, Knowledge- based Systems, Scheduling, Planning, AI applications, etc.

The success of this year's meeting is due to hard work and voluntary efforts of many people. From the post of the ICTAI General Chairs we would like to express our sincere thanks to all these individuals, but especially we would like to thank the Program Chairman, Babak Hamidzadeh for his enormous efforts to put a nice and high quality program for the 12th ICTAI-2000. We also express our great thanks and appreciation to Nik Bourbakis, the founder and Steering Chair of the ICTAI series for many years, for his infinite support at all levels.

Finally, we would like to acknowledge our appreciation of the IEEE Computer Society and the IEEE CS Virtual Intelligence Task Force for the sponsorship and assistance for the ICTAI-2000 proposal, hotel arrangements, the proceedings production, and for the continuous support of our efforts.

Once again welcome to ICTAI-2000.

C. Koutsougeras
F.Golshani
ICTAI – 2000 General Chairs

Message from the Program Chair

The Year 2000 IEEE International Conference on Tools with Artificial Intelligence continues its commitment to excellence in representing research and practice in artificial intelligence. This conference has a strong tradition in disseminating innovations in artificial intelligence and its applications, to a wide range of participants from academic, government and industrial organizations.

This year we received very high-quality papers in areas such as:

- Intelligent Agent Architectures
- Machine Learning
- Data Mining
- Artificial Neural Networks and Genetic Algorithms
- Robotics
- Planning and Scheduling
- Interactive Multimedia
- Geographic Information Systems
- Constraint Satisfaction and Optimization
- Fuzzy Logic
- Software Engineering
- Vision and Image Processing

Many papers demonstrated implemented systems with strong analytical and empirical analyses. This year, we received a total of 112 paper submissions by authors from around the world. The majority of papers received three reviews. All papers received at least two reviews.

Of the submitted papers, 34 were accepted as regular papers, for presentation and publication. The acceptance rate for regular papers was approximately 30%. Another 36 papers were accepted as short papers. Each regular paper can be up to 8 proceedings pages in length. Short papers can be up to 4 proceedings pages in length.

Many individuals have contributed to the technical program and to the conference as a whole. I would like to thank all authors and participants for taking part in the conference. My sincere gratitude goes to the members of the program committee for handling the reviewing of the papers, particularly those who handled their assigned papers within the tight time constraints. I would also like to thank all the external reviewers who helped us with reviewing the large number of papers. The large number of strong submissions this year indicates the superb job that the general chairs and the program vice chairs performed in promoting the conference.

I am very grateful to Professor Nikolaos Bourbakis, the Chair of the Steering Committee, for his valuable guidance and vision throughout the conference organization and preparation. The conference and I owe a great deal of appreciation to Mr. Alireza Shankaie, the Conference Publicity Chair and Local Arrangements Chair, for his tireless efforts and for doing much more than his conference titles seem to convey. Finally, many thanks go to Ms. Frances Titsworth and other members of the staff at the IEEE Computer Society for their help and patience throughout the conference and proceedings preparations.

Babak Hamidzadeh
Program Chair

Vice Chairs

Program Committee

Suryanarayana Sripada, *Boeing Corporation, USA*
Sven Koenig, *Carnegie Mellon University, USA*
Milind Tambe, *University of Southern California, USA*
Nevin Zhang, *University of Science and Technology, Hong Kong*
Aditya K. Ghose, *Simon Fraser University, Canada*
Vincent Tam, *National University of Singapore, Singapore*
Jonathan Lee, *National Central University, Taiwan*
Nickolaos Avouris, *University of Patras, Greece*
Patrick O. Bobbie, *Florida A&M University, USA*
Athman Bouguettaya, *Purdue University & QUT, USA*
Basilis Boutsinas, *University of Patras, Greece*
Jason Chen, *National Chiao-Tung University, Taiwan*
Bruce D'Ambrosio, *Oregon State University, USA*
Timo Honkela, *University of Art and Design, Finland*
Richard D. Hull, *Merck Research Laboratories, USA*
Dimitris Kalles, *Computer Technology Institute, Greece*
Rudolf Kruse, *OVG-Universitaet Magdeburg, Germany*
Chengwen Liu, *DePaul University, USA*
Bamshad Mobasher, *DePaul University, USA*
Masoud Mohammadian, *University of Canberra, Australia*
Chilukuri K. Mohan, *Syracuse University, USA*
Kinji Mori, *Tokyo Institute of Technology, Japan*
Dan O'Leary, *University of Southern California, USA*
Robert G. Reynolds, *Wayne State University, USA*
Isidore Rigoutsos, *IBM Thomas J. Watson Research Center, USA*
Yi Shang, *University of Missouri, USA*
Robert P. Simon, *George Mason University, USA*
Brian M. Slator, *North Dakota State University, USA*
Vassilis Tsotras, *University of California at Riverside, USA*
Eugene Wallingford, *University of Northern Iowa, USA*
John Yen, *Texas A&M University, USA*
Du Zhang, *California State University, USA*
Zhongfei Zhang, State University of New York, USA

Table of Contents

12th IEEE International Conference on Tools with Artificial Intelligence

Session A7: Constraint Satisfaction and Optimization

Session B7: Logic and Reasoning

Session C7: Machine Learning

Session A1

Logic and Reasoning

Multi-Resolution on Compressed Sets of Clauses

Philippe Chatalic and Laurent Simon
Laboratoire de Recherche en Informatique,
U.M.R. CNRS 8623, Université Paris-Sud,
91405 Orsay Cedex, France,
{chatalic, simon}@lri.fr

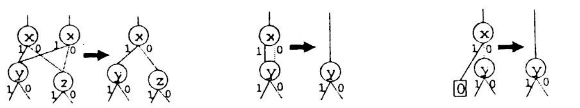
Abstract

This paper presents a system based on new operators for handling sets of propositional clauses represented by means of ZBDDs. The high compression power of such data structures allows efficient encodings of structured instances. A specialized operator for the distribution of sets of clauses is introduced and used for performing multi-resolution on clause sets. Cut eliminations between sets of clauses of exponential size may then be performed using polynomial size data structures. The ZRES *system, a new implementation of the Davis-Putnam procedure of 1960, solves two hard problems for resolution, that are currently out of the scope of the best SAT provers.*

1 Introduction

Recent years have seen a lot of work using propositional logic as a framework for knowledge representation and problem solving. The poor expressive power of propositional logic is counterbalanced by its simplicity which allows simple, but efficient, provers to be constructed. In particular, the SAT problem has been the focus of much interest, ranging from refinement and optimization of complete procedures [22] to the introduction of new incomplete methods [17]. Hard random instances have been identified [14] and reference problems have been gathered in benchmarks, subject to program competitions (e.g. [7, 18]). If its central place in complexity theory is probably part of the motivation of such work, the resolution of real world problems is also an attractive challenge for SAT. In the area of complete methods, most current SAT provers are based on the Davis, Logeman and Loveland procedure (DLL) [11]. One reason of success of such solvers is their ability to limit memory usage. This is due to the fact that they essentially make choices, resulting in simplifications and unit propagations. The difficulty comes from the number of possible choices which is exponential. The role of heuristics is then central. However, heuristics which are efficient on random problems are seldom appropriate for structured instances, on which dynamic heuristics based on learning techniques give better results [22]. Other problems, such as prime implicants/implicates generation or knowledge base compilation, often handle large sets of clauses. Then, the difficulty rather comes from the size of such sets that may grow exponentially. Whatever the goal, one is rapidly faced with combinatorial explosion. On structured instances, the only way to deal with such growth seems to be to take the structure of the available information into account. If heuristics can prune the search space efficiently, savings in space may be achieved by means of *compression* algorithms.

This paper focuses on data structures used for encoding sets of clauses and on associated operators. We are particularly interested in instances with a special structure. Our feeling is that such instances should show some regularities through the encoding. An appropriate data structure should be able to take advantage of such regularities to handle a more compact representation of the encoded formula. For cnf formulæ, such regularities may correspond to subsets of literals that appear in many different clauses. We propose to use a data structure allowing the factorization of common subsets of literals. *Tries* structures [5] enable the factorization of clauses beginning with the same sequence of literals. They remain the state-of-the-art data structures for subsumption checking [22]. We further generalize this idea to factorize simultaneously ends of clauses. In practice, we use a variant of BDDs [2], called ZBDDs [13], to store sets of clauses by means of their characteristic functions. We show that large sets of clauses corresponding to structured instances may be efficiently compressed in that way. Moreover, such structures are also well suited to subsumption checking, and all set operations can be realized as operations on ZBDDs. We introduce a new operator for performing multiple resolutions on sets of clauses represented by ZBDDs, in a single step. We use this operator in an implementation of the original Davis and Putnam algorithm [4] (DP as opposed to DLL).



a.Node sharing *b*.Node elimination *c*.Z-elimination

Figure 1. Reductions types in BDDs

The next section briefly recalls the basic principles of BDDs and ZBDDs structures. It introduces our technique to encode sets of clauses by means of ZBDDs, as well as new operators. In the third section we show that expressing cut-elimination using these operators makes it possible to perform multi-resolution on sets of clauses. We illustrate its use in a new implementation of DP, called ZRES. The last sections are devoted to empirical studies. Two classes of hard instances for resolution, namely the Pigeon Hole problem and the Urquhart problem, are tested and compression capabilities of ZBDDs are evaluated.

2 Representing sets of clauses with BDDs

2.1 Encoding boolean functions with BDDs

A BDD [2, 16] is a directed acyclic graph with labeled nodes, a unique source node, and such that each node is either a sink node or an internal node (denoted by $\Delta(x, n_1, n_2)$) having two children (n_1, n_2) and a *label* x. BDDs encoding boolean functions have only two sinks, labeled 1 and 0, and their internal nodes are labeled with boolean variables $\{x, y, \ldots\}$. The classical semantics interprets sinks 1 and 0 as *true* and *false*, and any internal node $n = \Delta(x, n_1, n_2)$ as the function $f =$ *if* x *then* f_1 *else* f_2, where f_1 and f_2 are the respective interpretations of n_1 and n_2. Each path from the source node to the sink 1 of a BDD thus corresponds to a model of the encoded formula. Given an ordering over variables, Ordered BDDs (OBDDs) require the label of any node to be smaller than the labels of its children. OBDDs can thus be viewed as binary trees encoding the Shannon decomposition [16] of the initial formula. To optimize memory usage, additional reduction rules may be used. Reduced OBDDs (ROBDDs) require the graph not to contain any isomorphic subgraphs (*node sharing* rule, fig 1-a) or any useless node of the form $\Delta(x, n, n)$, that do not care about its label value (*node elimination* rule, fig 1-b). ZBDDs [13] replace *node elimination* by *Z-elimination* (fig. 1-c), for which useless nodes are those of the form $\Delta(x, 0, n)$ which default interpretation is *false*. In the following, without further precisions, we simply use the term BDD in place of ROBDD.

The main advantage of using a BDD for encoding a propositional formula is that it can describe very large sets of models in a very compact way. However this requires computing the Shannon normal form of the formula, which may be very expensive if the formula is in conjunctive normal form (cnf). From the SAT point of view, since the BDD encodes the set of all models, this is as difficult as counting them, which is #P-complete.

2.2 Encoding sets of clauses with ZBDDs

To benefit from the high compression capabilities of BDDs while avoiding the computation of Shannon normal forms, we propose an alternative encoding and use a ZBDD to encode the set of clauses of a cnf formula. We still use two sink nodes 1 and 0 but internal nodes are labeled with literals (instead of variables). Intuitively, each path, from the source node to the 1 sink, represents a clause containing all the literals labeling the parent nodes of 1-arcs of this path. One may notice that in [1, 16], ZBDDs are also used as data structures for encoding sequences of literals corresponding to sets of prime implicants. However all these approaches first compute the ROBDD of the initial formula, and use a property of the Shannon decomposition (the *decomposition theorem*) to derive the set of prime implicants. Our approach is different since the ZBDD is directly constructed from the original cnf. In the following, we assume that, given an initial ordering $x_1 < \ldots < x_n$ on the set of variables, we use for such ZBDDs the extended literal ordering corresponding to $x_1 < \neg x_1 < \ldots < x_n < \neg x_n$. Given a set of clauses S, we denote by f_S the formula corresponding to the conjunction of all clauses of S. Encoding a set of clauses via a ZBDD essentially helps to factorize common beginnings and ends of clauses. For instance, with the initial variable ordering $x < y < z$, the set $S_1 = \{x \vee y \vee \neg z, \neg x \vee \neg y, \neg y \vee \neg z, z\}$ is represented[1] by the ZBDD of figure 2.*a*.

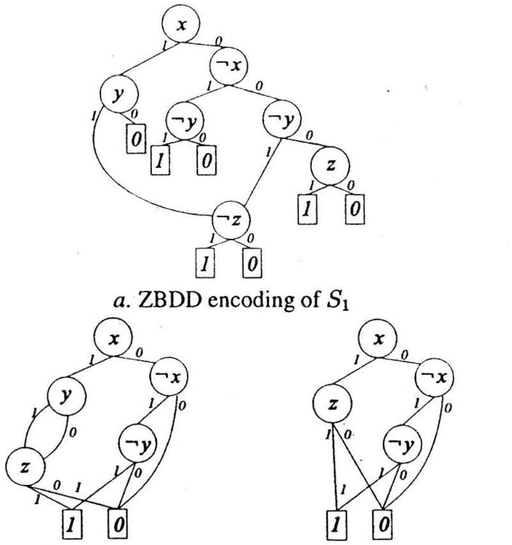
The semantics we use interprets such ZBDDs as sets of clauses. This may be formalized by:

- $[\![0]\!] = \emptyset$ (the empty set of clause)
- $[\![1]\!] = \{\square\}$ (the set reduced to the empty clause)
- $[\![\Delta(l, n_1, n_2)]\!] = \{l \vee [\![n_1]\!]\} \cup [\![n_2]\!]$, where $\{l \vee [\![n_1]\!]\}$ denotes the set of clauses obtained by adding the literal l to each of the clauses of $[\![n_1]\!]$.

Definition 1 *Let* $nc(\Delta)$ *(resp.* $nl(\Delta)$*) be defined by:*
- $nc(0){=}0$, $nc(1){=}1$, $nc(\Delta(l, A, B)){=}nc(A){+}nc(B)$
- $nl(0){=}0$, $nl(1){=}0$, $nl(\Delta(l, A, B)){=}nc(A){+}nl(A){+}nl(B)$

With our semantics, it can be easily shown that $nc(\Delta)$ (resp. $nl(\Delta)$) compute respectively the number of clauses

[1] Sink nodes 1 and 0 have been duplicated for a better readability.



a. ZBDD encoding of S_1

b. S_2 before node elimination   *c*. after node elimination

Figure 2. ZBDD encoding of sets of clauses

and the number of literals of $[\![\Delta]\!]$. This properties can be proven using the Z-elimination rule semantics.

Theorem 1 *Let C be a set of clauses corresponding to a formula f_C, the number of internal nodes of the ZBDD Δ encoding f_C is always at most equal to the total number of literals of C, $nl(\Delta)$.*

Interpreting ZBDDs as sets of clauses has a significant impact on the encoding, one of which is the easy detection of some kinds of subsumed clauses. Let us consider the set $S_2 = \{\neg x \vee \neg y, x \vee y \vee z, x \vee z, \neg x \vee \neg y \vee z\}$. The corresponding ZBDD is represented on figure 2-b. Note that the clause $\neg x \vee \neg y \vee z$, which is subsumed by the clause $\neg x \vee \neg y$, cannot be explicitely represented. More generally, any clause c_1 of length $n + k$, such that the n first literals correspond to another clause c_2 cannot be explicitly represented. But, since c_1 is subsumed by c_2, it can be simply removed. Another simplification may also be performed on the node labelled by y. The path going through its 1-arc corresponds to the clause $x \vee y \vee z$ while the path going through its 0-arc corresponds to the clause $x \vee z$, which clearly subsumes the previous one. The ZBDD obtained after elimination of the y node (fig. 2-c) thus corresponds to an equivalent set of clauses. More generally, any node of the form $\Delta(x, A, A)$ corresponds to the set of clauses $\{x \vee [\![A]\!]\} \cup [\![A]\!]$. So, each clause of the first set is subsumed by a clause of the second set. With our semantics, it is thus possible to take advantage of both the Z-elimination rule and the ROBDD node-elimination rule. Another easy simplification is the elimination of tautologies. A ZBDD of the form $\Delta(x, \Delta(\neg x, A, B), C)$ corresponds to the union of three sets of clauses : $\{x \vee \{\neg x \vee [\![A]\!]\}\} \cup \{x \vee [\![B]\!]\} \cup [\![C]\!]$. However, clauses containing both x and $\neg x$ are tautologies. A simple way to eliminate them is to apply the *tautology elimination rule*, which systematically replaces nodes of the form $\Delta(x, \Delta(\neg x, A, B), C)$ by $\Delta(x, B, C)$. In the following, we assume that all considered ZBDDs are constructed by systematically applying the appropriate reduction rules.

2.3 Operations on clause sets

Minato has shown in [13] that basic set operations (union, intersection,...) may be realized as operations on ZBDDs. We here introduce new recursive operators that take advantage of our special semantics, but that could not be used with usual semantics of ZBDDs. In the following, literals are denoted by $\{l, m, \ldots\}$. Proofs of most properties are done by induction on the sum of the size (nl) of the sets of clauses corresponding to operators' arguments[2]. We use the following lemma:

Lemma 1 *Let $A = \Delta(l, A_1, A_2)$ and $B = \Delta(m, B_1, B_2)$ be two ZBDDs. If $l < m$, then no clause of $[\![B]\!]$ contains the literal l and no clause of $\{l \vee [\![A_1]\!]\}$ subsumes a clause of $[\![A_2]\!]$.*

We now introduce a first binary operator on ZBDD, that is used in the definition of the next operators:

Definition 2 (Subsumed difference) *The $\setminus\!\!\setminus$ operator is defined by:*

$T_1 :\ 0 \setminus\!\!\setminus A = 0$

T_{2a}: $1 \setminus\!\!\setminus 1 = 0$ T_{2b}: *if* $A \neq 1$, $1 \setminus\!\!\setminus A = 1$

T_{3a}: $\Delta(l, A_1, A_2) \setminus\!\!\setminus 0 = \Delta(l, A_1, A_2)$

T_{3b}: $\Delta(l, A_1, A_2) \setminus\!\!\setminus 1 = 0$

$R_1 :\ (l > m)\ \Delta(l, A_1, A_2) \setminus\!\!\setminus \Delta(m, B_1, B_2) = \Delta(l, A_1, A_2) \setminus\!\!\setminus B_2$

$R_2 :\ (l < m)\ \Delta(l, A_1, A_2) \setminus\!\!\setminus \Delta(m, B_1, B_2) = \Delta(l,\ A_1 \setminus\!\!\setminus \Delta(m, B_1, B_2),\ A_2 \setminus\!\!\setminus \Delta(m, B_1, B_2))$

$R_3 :\ \Delta(l, A_1, A_2) \setminus\!\!\setminus \Delta(l, B_1, B_2) = \Delta(l,\ (A_1 \setminus\!\!\setminus B_1) \setminus\!\!\setminus B_2,\ (A_2 \setminus\!\!\setminus B_2))$

Theorem 2 *Given two sets of clauses S_1 and S_2, let us denote by $S_1 \mathbin{\widehat{\setminus\!\!\setminus}} S_2$ the set of clauses obtained by removing from S_1 all the clauses that are subsumed by some clause of S_2. Let A and B be two ZBDDs encoding sets of clauses, then $[\![A \setminus\!\!\setminus B]\!] = [\![A]\!] \mathbin{\widehat{\setminus\!\!\setminus}} [\![B]\!]$.*

[2]Full proofs are available from the authors.