2E4-OS-09a-3

最新SATソルバーへの充足不能コア抽出手法の実装

Implementation of Unsatisfiable Cores Extraction technique to Modern SATsolver

渡辺 大樹[*1] 鍋島 英知[*2]
Daiki Watanabe Hidetomo Nabeshima

[*1]山梨大学医学工学総合教育部コンピュータ・メディア工学専攻
Computer Science and Media Engineering, Department of Education Interdisciplinary Graduate School of Medicine and Engineering, University of Yamanashi

[*2]山梨大学大学院医学工学総合研究部
Department of Research Interdisciplinary Graduate School of Medicine and Engineering, University of Yamanashi

In this paper, we show the implementation of an unsatisfiable (UNSAT)core extraction algorithm to a modern SAT solver with a simplification technique during search process. UNSAT core extraction from propositional formula has a wide range of practical applications. There are two kinds of extraction approach of UNSAT cores: resolution-based and assumption-based approach. The former is more promising but more difficult to implement it to modern SAT solvers compared with the latter. In this work, we show the implementation of resolution-based approach proposed by Ryvchin and resolve trace minimization on memory proposed by Shacham. Furthermore, the implemented method allows to simplify clauses based on self-subsumptionin during search process.

1. はじめに

命題論理式の充足可能性判定問題である SAT 問題 (satisfiability testing) は主にハードウェア・ソフトウェア検証，プランニング，スケジューリングなどの様々な分野に応用されている．与えられた SAT 問題が充足不能であるとき，その原因を特定することはハードウェアやソフトウェア検証など幅広い分野において有用である．充足不能の原因の抽出手法はいくつか提案されているが [2]，最新の SAT ソルバーに効率よく実装することは容易ではない．本稿では，充足不能の証明に強い最新の SAT ソルバーに対して充足不能の原因抽出手法を実装する．また最新の SAT ソルバーでは，問題を高速に解くために，しばしば SAT 問題の簡単化手法が用いられる．本稿では，SAT ソルバーによる SAT 問題の簡単化に対応した充足不能コア抽出手法を示す．

充足不能コアの代表的な抽出手法には次の 2 つがある．

Resolution-based の充足不能コア抽出手法 [2]
反駁に至るまでの導出木を計算・保存しておき，そこから充足不能コアを抽出する．この手法は SAT ソルバーに組み込まなければならないため実装が複雑であるが，高速に充足不能コアを抽出できる可能性がある．

Assumption-based の充足不能コア抽出手法 [2]
assumption を真偽値割り当てで真が割り当てられていたリテラルとし，制約の伝播がその assumption と矛盾した場合，SAT 問題は充足不能なので矛盾に結びついた節を計算し，充足不能コアを抽出する．この手法は SAT ソルバーに組み込む必要がないため実装が簡単であるが，通常充足不能コアの抽出は高速ではない．

充足不能な SAT 問題から，充足不能な節集合である充足不能コアを高速に抽出するため，我々は，Shacham らによる導出過程をメモリ上に効率よく保持する手法 [3] を，入力節集合や学習節の簡単化手法を備えた最新の SAT ソルバーである **GlueMiniSat** [4] に実装した．GlueMiniSat とは 2011 年の SAT Competition アプリケーション部門の UNSAT 部門と SAT+UNSAT 部門でそれぞれ 1 位と 2 位を獲得した SAT ソルバーである．節の簡単化は最新 SAT ソルバーの重要な要素技術の 1 つであり，高速に充足不能コアを高速抽出するため必要不可欠な要素である．GlueMiniSat は充足不能な問題の証明に強く，充足不能コアの抽出手法を導入する意義が大きいといえる．また，GlueMiniSat では，学習節の簡単化手法 [5] が導入されているが，本稿では，その簡単化手法に対応した充足不能コアの抽出手法を示す．

本論文の構成は以下の通りである．まず，2 節において SAT 問題と SAT ソルバーについて解説する．3 節においては充足不能コアについて概説する．4 節では実装手法について解説する．5 節では実装手法の評価実験について示し，6 節では本稿をまとめ，今後の課題について述べる．

連絡先: 渡辺 大樹，山梨大学大学院医学工学総合研究部コンピュータ・メディア工学専攻，〒 400 - 8511 山梨県甲府市武田 4 - 3 - 11，E-mail : daiki@nabelab.org

2. SAT 問題と SAT ソルバー

SAT 問題とは，命題論理式の充足可能性判定問題である．SAT 問題は以下の例にあるような連言標準形 (Conjunctive Normal Form ; CNF) で与えられる．

$$(a \vee b \vee e) \wedge (\neg a \vee b) \wedge (\neg a \vee \neg c) \wedge .....$$

ここで a，b，c は命題変数である．命題変数は真または偽のいずれかの値をとる．命題変数そのもの，またはその否定をリテラルという．リテラル同士を選言 ($\vee$) で結んだ括弧の部分を節といい，節同士を連言 ($\wedge$) で結んだ命題論理式を連言標準形という．

連言標準形で与えられた命題論理式は節の連言でできているため，すべての節を真にするように命題変数の値を決められれば命題論理式を真にできる．節はリテラルの選言で与えられ

ているため，節内のリテラルのうち少なくとも 1 つを真にできればその節は真になる．すなわちすべての節を真にするためには，各節で少なくとも 1 つのリテラルを真にする必要がある．SAT 問題において，すべての節を真にすることができる命題変数の真偽値割り当てが存在するとき充足可能といい，そのような割り当てが存在しない場合，充足不能という．

このような SAT 問題を解くソフトウェアが SAT ソルバーである．以下に現在 SAT ソルバーの基本的なアルゴリズムとなっている CDCL 手続き [6, 7] について説明する．

CDCL 手続きでは，まず**単位節**と呼ばれる真偽値が割り当てられていない命題変数が 1 つだけの節を探す．単位節が存在した場合，その節を充足させるため，単位節の未割り当てのリテラルに真を割り当てる．このとき，他の単位節が存在した場合，単位節を充足させる割り当てを繰り返し行う．この動作を**単位伝播**という．単位節が存在しなければ，変数選択ヒューリスティクスという戦略に基づき変数を 1 つ選択し，割り当てを行う．このとき選択された変数を**決定変数**といい，決定変数の数を**決定レベル**と呼ぶ．もし，探索の過程において単位伝播により，あるリテラルを持つ単位節とそのリテラルの否定のリテラルを持つ単位節が発生 (矛盾) した場合，矛盾の原因を解析し，学習節と呼ばれるその矛盾を起こさないようにする節を学習し，適当な決定レベルまで**バックジャンプ**する．矛盾の原因が x と y に真を割り当てたことである場合，$\neg x \vee \neg y$ という原因の否定の節を学習する．最後に，学習節が単位節になるまでバックジャンプを行う．以上が CDCL 手続きの概要である．

多くの SAT ソルバーは，特定の条件で探索を打ち切り，探索をやり直す**リスタート戦略**を用いている．リスタート戦略では学習節は保持するため，同様の矛盾を繰り返すことはない．

学習節は矛盾の原因となった節に**融合法** [8] を用いることで得ることができる．融合法は，任意のリテラル x を含む節とその否定 $\neg x$ を含む節があったとき，それらのリテラル x，$\neg x$ を除いた他のリテラルによって構成される**融合節**を導出する手法である．融合法の定義を以下に示す．

$$\frac{x \vee a_1 \vee \cdots \vee a_n \quad \neg x \vee b_1 \vee \cdots \vee b_m}{a_1 \vee \cdots \vee a_n \vee b_1 \vee \cdots \vee b_m}$$

x_i，a_i，b_i はリテラルを表している．このような関係は **Resolution グラフ** という節または学習節を頂点に持つ非循環有向グラフによって表される．Resolution グラフは融合法に使われた節を親とし，導出される融合節を子に持つ．融合節を求めた結果が空節であった場合，融合法に使用された節は充足不能である．

3. 充足不能コア

充足不能コアとは充足不能な節集合であり，充足不能な CNF 式の中に存在する．充足不能コアは複数の**遷移節**と呼ばれる充足不能の原因の節とそれ以外の節からなる．充足不能な CNF 式を F，遷移節を c とすると以下のように表せる．

$$c \subseteq F \wedge F \setminus \{c\} \text{ is satisfiable}$$

4. 実装手法

充足不能コア抽出は高速に行う必要があるため，GlueMiniSat に充足不能コア抽出手法として **Resolution-based の充足不能コア抽出法** [2] を実装する．また充足不能コア抽出を高速化するため，充足不能コア抽出に必要な情報をメモリ上に保持する手法 [3] を用いる．上記の手法を以下に示す．

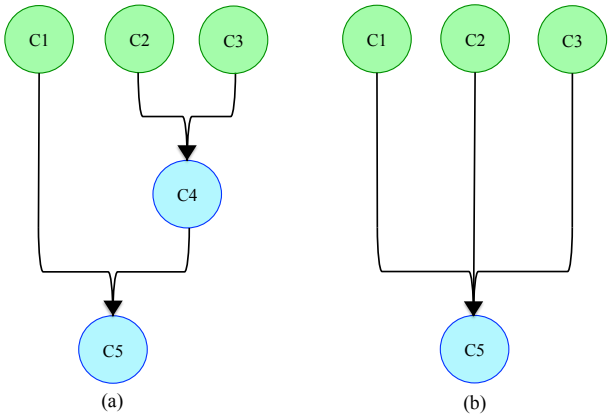

図 1: Resolution グラフと節含意グラフ

4.1 Resolution-based の充足不能コア抽出法 [2]

矛盾からの節学習により学習節を得るとき，矛盾の原因となった節集合に対して融合法を適用する．Resolution グラフ上で学習節を得る過程の例を図 1(a) に示す．ある矛盾が起こり，C_1，C_2，C_3 の節が矛盾の原因である場合を考える．まず C_2 と C_3 に融合法を用いる．この後，得られた融合節を C_4 とすると C_1 と C_4 に融合法を適用することで，C_5 という学習節を得る．C_5 を学習する流れを以下に示す．

$$\frac{\dfrac{C_2 \;\; C_3}{C_1 \;\; C_4}}{C_5}$$

Resolution-based の充足不能コア抽出法に必要な情報は学習節と矛盾の原因となった節の節集合であるため，図 1(b) に示すような C_4 を省いた**節含意グラフ** [9] を用いて融合法の情報を表す．

図では矛盾の原因である節が C_1，C_2，C_3 であり融合法を繰り返した結果，得られた学習節が C_4 であったことを表している．これらの情報は独自のデータベースを作成し，情報を保存する．各節には特有の節番号を与え，節番号によって節を表すことで高速にデータベースの作成を行える．独自のデータベースでは矛盾の原因となった節集合と学習された学習節の節番号を一組として持つ．親ノードを持たない節は CNF 式中の節であり，親ノードを持つ節は学習節である．SAT ソルバーが SAT 問題を解くとき，CNF 式の節集合に対し融合法を繰り返し用いることで，SAT 問題を解いていく．融合節として空節が導出された場合，CNF 式が充足不能であることが分かる．このとき導出された空節から，生成された節含意グラフのエッジを CNF 式中の節まで辿ることで充足不能コアを導出することができる．このような充足不能コア抽出手法を Resolution-based の充足不能コア抽出法という．

4.2 メモリ上での充足不能コア抽出 [3]

充足不能コアを抽出するために必要な節含意グラフは問題により膨大になる可能性がある．そのため，充足不能コア抽出器では節含意グラフをファイルに書き出し，充足不能コアを抽出する．ファイルへの書き出しや読み込みは処理が遅いため，

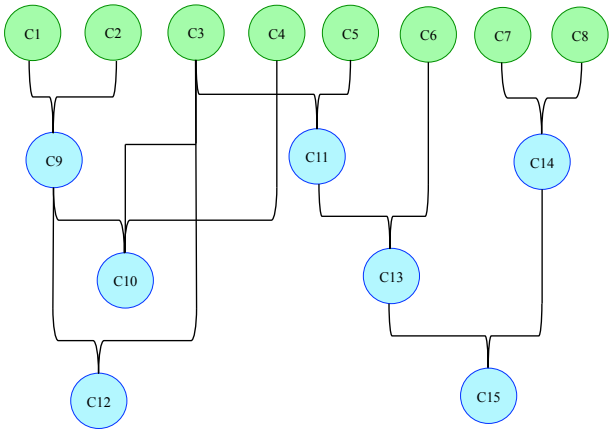

図 2: 節含意グラフ

メモリ上のみの処理にすることで充足不能コア抽出を高速化することができる．

節含意グラフは配列を用いて実装する．すべての節に特有の節番号を保持させることで各節の判別を可能にしている．また，学習節が発生した順に配列の末尾に追加していく．配列は生成した節数分の要素を持ち，配列の各要素に格納する情報は“ 節番号“ と ”その節を導出した節集合 ”である．このようにすることで節含意グラフを再現することができる．充足不能コアを抽出するためには節含意グラフを辿る必要がある．節含意グラフを辿るには学習節を導出した節を 1 つずつ調べ，その節が元問題の節でないならば調べている節番号を学習節にもつ要素を探さなければならない．我々の実装では，この探索に二分探索を用いることによって，節数を n とすると $O(\log_2 n)$ で探索可能である．メモリ上で充足不能コアを抽出するためには充足不能コアを抽出するための節含意グラフをメモリ上に保持する必要があるが，節含意グラフのサイズが膨大になる可能性を考慮すると，節含意グラフを縮約し，メモリ量の削減を行う必要がある．

SAT 問題を解く際，学習節を得ることで同じ矛盾を回避することができる．しかし，学習節は問題サイズに対し指数関数的に生成されるため，過度な量の学習節を保持することでメモリが埋め尽くされてしてしまい，処理速度が低下する．そこである評価尺度に基づき，今後の探索に有用と考えられる学習節を残し，そうでない学習節を削除するということを定期的に行う．節含意グラフを用いて充足不能コアを抽出する場合，学習節が削除されたとしても節含意グラフ からその学習節を削除することは出来ない．それは，削除された学習節が原因で別の学習節が生成されていた場合，新たに生成された学習節が空節を導出するために使用される可能性があるためである．そのため節含意グラフからノードを削除する場合，そのノードの子ノードの数を考慮する必要がある．削除された学習節が節含意グラフ上で子ノードを持たないとき，削除された学習節は空節の導出に使われないため，節含意グラフから削除することができる．よって節含意グラフから節を削除するためには，節ごとに直下の子ノードの個数を保持しておく必要がある．節が矛盾の原因になる度にその節の子ノードの数を 1 つ増やし，学習節が削除される度に原因となった節の子ノードの数を 1 つ減らすことで各節の子ノードの数をカウントすることができる．節含意グラフ縮小の例を図 2 のような節含意グラフが生成されている場合を例に説明する．

学習節の削除により C_9，C_{10}，C_{11}，C_{12} が削除されているとする．このとき子ノードを持つ C_9 と C_{11} は削除することができない．しかし，子ノードを持たない C_{10}，C_{12} は削除できる．このとき，C_{10}，C_{12} が削除されたことによって C_9 は子を持たなくなるため，C_9 も削除できる．

このように，ある節を削除したことにより他の節が削除できる可能性があるため，この処理はすべての節に対して行われる必要がある．上記のように処理することで節含意グラフを縮小し，使用メモリ量を削減する．

4.3 学習節の簡単化への対応

SAT ソルバーの高速化手法の 1 つに学習節の簡単化が挙げられる．これは，代表的な SAT ソルバーである MiniSat[10] に用いられている簡単化手法である．ある矛盾が起こったとき，融合法により学習節が得られる．このとき，得られた学習節と他の節に対して融合法を行うことにより，学習節の簡単化を行うことができる (Self-subsumption)．

ある節から $(\neg a \vee b \vee d \vee e)$ という節を学習したとする．$(\neg a \vee b \vee \neg d)$ のような節があり，a が真，b が偽であれば $\neg d$ が真となる．このとき，学習する節を $(\neg a \vee b \vee e)$ に縮小することで学習節の簡単化を行う．しかし，この学習節の簡単化を行うと得られた学習節の親ノードは矛盾の原因の節のみではなくなってしまう．そこで，この簡単化を行った場合，得られた学習節の親ノードに簡単化に用いられた節を追加する処理を追加することで，学習節の簡単化に対応することができる．

5. 評価実験

使用メモリ量と実行時間の平均を比較する評価実験を行った．比較対象は GlueMiniSat2.2.6(充足不能コアの抽出なし)，メモリの削減処理を行わない充足不能コア抽出，メモリの削減処理を行う充足不能コア抽出である．

実験環境を以下に示す．使用した SAT ソルバーは GlueMiniSat2.2.6 であり，使用した問題は SAT Competition 2011 MUS track の問題 200 問である．CPU は Core Duo 1.66GHz でメモリは 2GB である．また，制限時間は 600 秒 / 問で行った．充足不能コアの抽出を行わない GlueMiniSat2.2.6 を比較対象に挙げている理由は充足不能コア抽出にかかるオーバーヘッドを示すためであり，他の 2 つの比較対象はメモリの削減処理にかかるオーバーヘッドを示すためである．また，メモリの削減処理を行わない節含意グラフの作成を CIG ，メモリの削減処理を行う節含意グラフの作成を MinCIG として扱い，メモリの削減処理を行わない充足不能コア抽出を CIG+Core ，メモリの削減処理を行う充足不能コア抽出を MinCIG+Core として扱うこととする．

実験結果を表 2 にまとめる．表 2 より節含意グラフの作成にかかるオーバーヘッドは約 21 %であることが分かる．表 2 よりメモリ量の削減を行うことにより平均で約 3.5MB のメモリ量の削減をしていることが分かる．また，このメモリ量の削減処理にかかるオーバーヘッドは約 13 %であることが分かる．CIG や MinCIG と CIG+Core や MinCIG+Core の結果より，節含意グラフからの充足不能コア抽出は非常にコストが高いことが分かる．また，充足不能コアのサイズは元問題の約 39 %であった．

SAT ソルバーで問題を解くとき，問題により学習節の削除を全く行わずに充足可能性が判定できるときがある．そのため，使用メモリ量が平均で 3.5MB しか削減されなかった理由は，学習節が削除されなかったため節含意グラフの縮小処理が行われなかった問題が多数含まれていたことに起因すると考

表 1: 実験環境

比較対象	平均使用メモリ量 (MB)	平均実行時間 (秒)
GlueMiniSat2.2.6	36.57	4.95
CIG	68.25	5.99
MinCIG	64.78	6.07
CIG+Core	68.25	22.05
MinCIG+Core	64.78	22.14

えられる．今回使用した問題中でメモリの縮小処理が行われたのは 200 問中 72 問である．また，問題によりメモリの削減量にばらつきがあり，削減量が 50 %を越える問題も存在している．つまり学習節が多く，SAT ソルバーによる学習節の削除が多いほどメモリ量の削減処理は有効に働く可能性が高い．それは，節含意グラフから節を削除する際には学習節のデータベースから削除されている節しか削除することが出来ないためである．学習節の削除が多く行われる場合の多くは節含意グラフのサイズが大きくなるため，この削減手法は有用であると考える．

6. まとめと今後の課題

本稿では最新の SAT ソルバーである GlueMiniSat に充足不能コア抽出手法を実装した．充足不能コア抽出を高速化するため，SAT 問題から充足不能コアを抽出する際に保持する必要のある節含意グラフをメモリ上に保持し，メモリ量の削減をするために節含意グラフの縮小を行った．評価実験の結果，節含意グラフの縮小によるメモリの削減量，メモリ削減処理のコストから節含意グラフは有用であるといえる．

今後の課題として，充足不能コアは高速により小さな充足不能コアを抽出することを目的としているため，充足不能コアの縮小技術を導入することが挙げられる．また，節含意グラフからの充足不能コア抽出はコストが高いため，高速な充足不能コア抽出手法が必要である．

参考文献

[1] Joao Marques-Silva and Ines Lynce. On Improving MUS Extraction Algorithms. In Proceedings of the 14th international conference on Theory and application of satisfiability testing(SAT ’11). pp. 159-173, 2011.

[2] Vadim Ryvchin and Ofer Strichman. Faster Extraction of High-Level Minimal Unsatisfiable Core, In Proceedings of the 14th international conference on Theory and application of satisfiability testing(SAT ’11). pp. 174-187, 2011.

[3] O. Shacham and K. Yorav. On-the-fly Resolve Trace Minimization, Proceedings of the 44th annual Design Automation Conference. pp. 594-599, 2007.

[4] 鍋島英知，岩沼宏治，井上克巳．Glueminisat2.2.5: 単位伝搬を促す学習節の積極的獲得戦略に基づく高速 SAT ソルバー．コンピュータ ソフトウェア, Vol. 29, No. 4, pp. 146-160, 2012.

[5] Niklas Sörensson and Armin Biere. Minimizing learned clauses. Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing(SAT’09). 237-243, 2009.

[6] Roberto J. Bayardo and Robert Schrag. Using CSP Look-Back Techniques to Solve Real-World SAT Instances. . AAAI/IAAI. pp. 203-208, 1997.

[7] João P. Marques Silva and Karem A. Sakallah. GRASP: A Search Algorithm for Propositional Satisfiability. IEEE Trans. Computers. Vol. 48. No. 5. pp. 506-521. 1999.

[8] Martin Davis, George Logemann and Donald Loveland. A machine program for theorem proving. Communications of the ACM. pp. 394-397, 1962.

[9] Roman Gershman, Maya Koifman and Ofer Strichman. An Approach for Extracting a Small Unsatisfiable Core. J. on Formal Methods in System Design. pp. 1-27, 2008.

[10] Eén, N. and Sörensson, N. . An Extensible SAT-solver, SAT, Giunchiglia, E. and Tacchella, A. (eds.). Lecture Notes in Computer Science. Vol. 2919. pp. 502-518, 2003.