充足可能性ソルバ （SAT ソルバ） の原理

Hiromi ISHII

2024-03-10

Tsukuba Computer Mathematics Seminar 2024

自己紹介

自己紹介

- 石井大海（いしい ひろみ）
- 2018 年度 筑波大学数学専攻博士後期課程修了（照井研）
- 計算機合宿には 2014 年から参加
- 現職： Haskell 製大規模数値計算ベンチャー研究開発職
- 宣伝：今年 05/11, 12 に横浜でお芝居をするので興味のある方は是非観にきてください

充足可能性ソルバ （SAT ソルバ） の原理

本日の話題：充足可能性問題と SAT ソルバ

- 充足可能性問題：与えられた命題論理式が （古典的に） 充足可能かどうかを判定する問題
  - 古典命題論理式：命題変数 $P_1, ..., Q_1, ...,$ を $\wedge$（かつ）、$\vee$（または）、$\rightarrow$（ならば）、$\neg$（でない）で結んで得られる論理式
  - 古典的充足可能性 : 与えられた式を真とするような、 命題変数への真偽値
    - （真）または $\times$（偽）の割り当てが存在するか？
- 充足可能性（SATisfiability）を略して SAT と呼ぶ。
- NP- 完全：総当たりで解けるような任意の問題が SAT に帰着できる
  - 本来 SAT は判定問題だが、具体的な求解まで含めて SAT と呼ぶ事が多い
- 色々な問題が SAT（やその拡張である SMT ソルバ）で解け、実用上も重要

SAT で解ける問題の例：論理パズル

問 1 (三人の島民 [1])

常に嘘だけをいう嘘吐きと、 本当のことだけをいう正直者だけが住む島で、A, B, C 三人の島民に出会った。彼らのいうことには：

- A：「B と C はどちらも正直者だ」
- B：「A は嘘吐きで、C は正直者だ」

A, B, C はそれぞれ正直者か、嘘吐きか？

三人の島民：回答

- A, B, C を「A が正直者」「B が正直者」「C が正直者」を表す命題変数とする
- 情報を命題論理式に変換して $(1) \land (2)$ を充足する解を求めればよい：

$$A \longleftrightarrow B \land C \tag{1}$$

$$B \longleftrightarrow \neg A \land C \tag{2}$$

- 真理表を書いてみると、全員嘘吐きだとわかる。

A	B	C	(1)	(2)	$(1) \land (2)$
○	○	○	○	×	×
○	○	×	×	×	×
○	×	○	×	○	×
○	×	×	×	○	×

A	B	C	(1)	(2)	$(1) \land (2)$
×	○	○	×	○	×
×	○	×	○	×	×
×	×	○	○	×	×
×	×	×	○	○	○

SAT で解ける問題の例：数独

- 簡単な例として、4×4 の小さな数独の問題を SAT で解くことを考える。
- $i, j, k \leq 4$ に対し命題変数 $P_{ij}^{=k}$ を用意する（$d_{ij} = k$ というきもち）。
- 各マスには $1, 2, 3, 4$ のいずれかの数字一つを入れる。

d_{14}	d_{24}	d_{34}	d_{44}
d_{13}	d_{23}	d_{33}	d_{43}
d_{12}	d_{22}	d_{32}	d_{42}
d_{11}	d_{21}	d_{31}	d_{41}

$$\bigwedge_{i \leq 4} \bigwedge_{j \leq 4} \left\{ \left(P_{ij}^{=1} \vee P_{ij}^{=2} \vee P_{ij}^{=3} \vee P_{ij}^{=4} \right) \wedge \bigwedge_{k \leq 4} \bigwedge_{l \neq k} \left(P_{ij}^{=k} \longrightarrow \neg P_{ij}^{=l} \right) \right\} \tag{3}$$

SAT で解ける問題の例：数独　（続）

- 各行、各列、各 2×2 の小ブロックには各数字 $1, \ldots, 4$ が一つずつ入る。
  - 「各行」は次のように書ける（「各列」も i, j の役割を入れ換え同様）：

$$\bigwedge_{i \leq 4} \bigwedge_{k \leq 4} (P_{i1}^{=k} \vee P_{i2}^{=k} \vee P_{i3}^{=k} \vee P_{i4}^{=k}) \tag{4}$$

  - 演習問題：「一意性」の条件が要らない理由を考えてみよう。
  - 演習問題：「各ブロック」の条件を書き下してみよう。
- あとは盤面の情報を個別に $P_{ij}^{=k}$ で与えてやれば、個別の問題を SAT で解ける！

SAT で解ける問題の例：命題論理の定理自動証明

定理 2 (古典命題論理の完全性定理および健全性定理 (Gödel))

任意の古典命題論理式 φ に対し、φ がどんな真偽値の割り当てに対しても真であることと、φ が証明可能であることは同値である。

系 3

任意の古典命題論理式 φ が証明可能である必要十分条件は、$\neg\varphi$ が充足可能でないことである。

- $\neg\varphi$ が充足可能でない事がわかれば、φ が証明できたことになる！

SAT で解ける問題の例： SMT ソルバへ

- SAT だけでも強力だが、更に述語論理（$\forall$ とか $\exists$ とかを入れたやつ）に拡張した SMT（SAT Modulo Theories）ソルバの研究も盛んに行われている
  - 代表的なソルバ： Z3, CvC5, etc.
- SMT で扱える理論の例：線型計画法、整数計画問題、同値関係、配列、etc.
  - 数独なんかは配列の理論とかを使ったほうが綺麗
  - 工学的にもこれらはかなり重要
  - 定理証明系で人間が手で書きたくない補題の自動証明にも使える
- SMT は SAT ソルバを外部の理論ソルバと協業させこれらの問題を解く
  - SAT を如何に速く解くかが SMT の実装にも重要

SAT ソルバの原理

SAT は 「難しい」

- 論理パズルの解法で見たように、すべての命題変数の有り得る真偽値の割り当てを総当たりして、真理表をつくれば原理的には解ける
  - これは命題変数の数を n とすると 2^n 通りの場合分けが必要になる
  - もっと効率的な方法はないか？
- 実は「総当たりで解ける問題」は全て SAT に帰着できる（SAT は NP- 完全である）ことが知られている
  - $P \neq \mathrm{NP}$ 予想が正しければ、本質的に効率的な解法は存在しない
- 本質的に速くなくても実用的な問題に対し十分速い手法が研究されている
  - 以下ではその中でも主流な CDCL 法の原理を簡単に紹介（参考文献 [2–5]）

連言標準形 （CNF）

- SAT ソルバは連言標準形 (CNF) と呼ばれる形の論理式を扱うことが多い

定義 4

1. 命題変数 P_i およびその否定 $\neg P_i$ を合わせてリテラルと呼ぶ。 以下、リテラルを表すメタ変数を l, l_i などと書く。
2. 0 個以上のリテラルを 「または」 で繋いだもの $l_1 \lor \ldots \lor l_k$ を節と呼ぶ。以下、節を表すメタ変数を C, C_i などと書く。
3. 0 個以上の節を「かつ」で繋いだ $C_1 \land \ldots \land C_m$ の形の論理式を連言標準形（Conjunctive Normal Form, CNF）と呼ぶ。

注意：空の節は矛盾に、空の CNF は恒真に対応する。

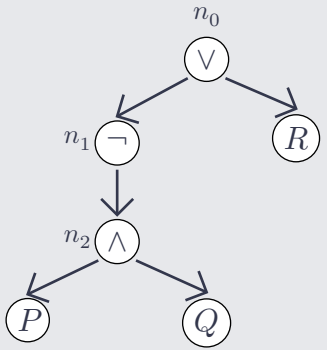
任意の命題論理式は CNF で表せる

定理 5 (命題論理式の CNF への変換)

任意の命題論理式 φ は同値な CNF φ_{CNF} を持つ。

- de Morgan 則と分配則を使って変換すればよいが、自乗のオーダーかかる
- 充足可能性だけを考えればよいのなら、結合子ごとに追加の命題変数 n_i を導入し、各節のサイズが高々 3 の CNF（3-CNF）に変換できる
- 論理式 $\neg(P \wedge Q) \vee (R \wedge S)$ の CNF への変換例：

$$
\begin{aligned}
&n_0 \wedge (\neg n_0 \vee n_1 \vee R) \wedge (\neg n_1 \vee n_0) \wedge (\neg R \vee n_0) \\
&\quad \wedge (\neg n_1 \vee \neg n_2) \wedge (n_1 \vee n_2) \\
&\quad \wedge (\neg n_2 \vee P) \wedge (\neg n_2 \vee Q) \wedge (\neg P \vee \neg Q \vee n_2)
\end{aligned}
$$



- 演習問題：他の結合子の変換規則を考えてみよう。

DPLL: CDCL の祖先

- CDCL は Davis–Putnam–Logemann–Loveland (DPLL) を基礎としている
- 基本的な考え方： CNF の節やリテラルを削れるまで削ってから場合分け
  - 充足された節は削除できる $\rightarrow$ 節が一つもなくなれば充足可能！
  - 偽なリテラルがあれば、節から削除できる $\rightarrow$ 空な節ができたら充足不能！
- 以下の規則に従って CNF を変形する：

(a) 単位節伝播：リテラル一つだけの節 $\{l\}$ があれば $l \equiv \top$ とし $\neg l$ を削除

(b) 純リテラル：肯定または否定のみ出現するリテラル l があれば $l \equiv \top$ としそのリテラルを含む節を削除（コスト面から実装しない事が多い）

(c) Decide：他規則が適用できないならリテラルを一つ選び場合分け

- 真偽を仮決めし継続。失敗したら巻き戻し (Backtrack) 否定で確定し継続

DPLL の動作例

- $(\neg P_1 \vee P_2) \wedge (P_3 \vee P_4) \wedge (\neg P_5 \vee \neg P_6) \wedge (\neg P_2 \vee \neg P_5 \vee P_6)$ を DPLL で解いてみよう（[2] の例）。
- 今後は CNF しか出て来ないので、$(\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6)$ と略記する

$$
\begin{aligned}
& (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6) && \mid \\
\xrightarrow{\text{(Decide 1)}} \ & (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6) && \mid 1^{\mathrm{d}} \\
\xrightarrow{\text{(UnitProp 2)}} \ & (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6) && \mid 1^{\mathrm{d}}\ 2 \\
\xrightarrow{\text{(Decide 3)}} \ & (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6) && \mid 1^{\mathrm{d}}\ 2\ 3^{\mathrm{d}} \\
\xrightarrow{\text{(Decide 5)}} \ & (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6) && \mid 1^{\mathrm{d}}\ 2\ 3^{\mathrm{d}}\ 5^{\mathrm{d}} \\
\xrightarrow{\text{(UnitProp 6)}} \ & (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \underline{\bar{6}})\ (\bar{2}, \bar{5}, 6) && \mid 1^{\mathrm{d}}\ 2\ 3^{\mathrm{d}}\ 5^{\mathrm{d}}\ \underline{\bar{6}} \\
\xrightarrow{\text{(Backtrack)}} \ & (\bar{1}, 2)\ (3, 4)\ (\bar{5}, \bar{6})\ (\bar{2}, \bar{5}, 6) && \mid 1^{\mathrm{d}}\ 2\ 3^{\mathrm{d}}\ \bar{5}
\end{aligned}
$$

DPLL の動作例 2

$$
\begin{array}{rll}
 & (\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & | \\
\text{–(Decide [1-4])}\longrightarrow & (\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & | \ \ldots\ 4^{\mathrm{d}} \\
\text{–(UnitProp 5)}\longrightarrow & (\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & | \ \ldots\ 4^{\mathrm{d}}\ 5 \\
\text{–(UnitProp }\bar{6}\text{)}\longrightarrow & (\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & | \ \ldots\ 4^{\mathrm{d}}\ 5\ \bar{6} \\
\text{–(UnitProp [7-11])}\longrightarrow & (\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & | \ \ldots\ 4^{\mathrm{d}}\ \ldots\ 11 \\
\text{–(UnitProp 12)}\longrightarrow & (\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},\underline{\underline{12}})\ (\overline{10},\bar{2},\overline{\underline{\underline{12}}}) & | \ \ldots\ 4^{\mathrm{d}}\ \ldots\ 11\ \underline{\underline{12}} \\
\text{–(Backtrack)}\longrightarrow & (\bar{1},\bar{2},\bar{3},\bar{4},5)\ (\bar{3},\bar{4},\bar{6})\ (\bar{5},6,\bar{1},7)\ (\bar{7},8)\ (\bar{2},\bar{7},9)\ (\bar{8},\bar{9},10)\ (\overline{10},11)\ (\overline{11},12)\ (\overline{10},\bar{2},\overline{12}) & | \ 1^{\mathrm{d}}\ 2^{\mathrm{d}}\ 3^{\mathrm{d}}\ \bar{4}
\end{array}
$$

CDCL: Conflict-Driven Clause Learning

- 単純総当たりより DPLL は賢いが、Backtrack で得られる情報が少ない
  - 折角 Backtrack でいったん真偽値が確定しても、更に Backtrack すればその情報は消えてしまう
- Backtrack は一段階ずつしか戻らない
  - 場合分けが入れ子になっている場合、実は直近の場合分けよりもっと早くに矛盾が起きていても延々と一段階ずつ Backtrack してしまう
- 矛盾からに過程を辿っていって矛盾の原因となった「理由」を見付け。学習節として CNF に追加してそこまで一気に巻き戻そう (Backjump) ！
  - これが矛盾からの節学習 (Conflict-Driven Clause Learning, CDCL)
  - 学習節は元の CNF からの帰結となるようにする

CDCL：含意グラフと学習節の生成

CDCL：更なる最適化手法

現代の CDCL では以下のようなテクニックで効率化が図られている：

- 監視リテラル：単位節または矛盾節の検出方法の効率化
  - 観察：単位節・矛盾節確定するのは、リテラル数が 2 以下になったときだけ
  - そこで、節ごとに最大 2 つのリテラルを監視リテラルとして指定
  - 監視リテラルの真偽が確定したときだけ、その節の状態を確かめて単位節・矛盾節の検出を行えばよい！
- 変数選択ヒューリスティクス：場合分けするリテラルには任意性がある
  - どのリテラルを選ぶかで性能が大きく変わる
  - 時間減衰する優先度をつかう VSIDS など変数選択の指標が提案されている
  - 選択の指標に関しては、例えば Liang らの論文 [6] が詳しい

CDCL：更なる最適化手法　（続）

- リスタート：履歴の破棄
  - 辿った履歴によって簡単にとけたり、逆に永遠に解けなかったりする
  - そこで矛盾が起きたら一定間隔で状態を破棄し、ゼロから解きなおす
  - 学習節は捨てず、間隔はだんだん長くするので、完全性は担保される
  - リスタート間隔のヒューリスティクスは Wu らの論文 [7] が詳しい
- 忘却：増えすぎた節の枝刈りをする
  - 矛盾のたびに学習節が増えるので、節数が爆発しがち
  - そこで、適切な指標の下で優先度の低い学習節を定期的に削除
  - 削除するのは学習節だけなので、完全性に影響しない

まとめ

まとめ

- Matome here

参考文献

[1] レイモンド・スマリヤン , “スマリヤンの決定不能の論理パズル ゲーデルの定理と様相論理 ,” 白揚社 , 2008.

[2] Aleksandar Zeljić, "CS357: Advanced Topics in Computer Science". Course Material. Stanford University. 2019. https://web.stanford.edu/class/cs357/.

[3] 鍋島秀和 and 宋剛秀 , “高速 SAT ソルバーの原理 ,” 人工知能 , pp. 68–68, vol. 25, no. 1, 2010, https://www.jstage.jst.go.jp/article/jjsai/25/1/25_68/_article/-char/ja/.

[4] 岩沼宏治 and 鍋島秀和 , “SMT : 個別理論を取り扱う SAT 技術 ,” 人工知能 , pp. 86–86, vol. 25, no. 1, 2010, https://www.jstage.jst.go.jp/article/jjsai/25/1/25_86/_article/-char/ja/.

[5] Enuba Torlak, “A Modern SAT Solver”. Course Material. 2003. https://courses.cs.washington.edu/courses/cse507/17wi/lectures/L02.pdf.

[6] J. H. Liang et al., “Understanding VSIDS Branching Heuristics in Conflict-Driven Clause-Learning SAT Solvers,” in *Hardware and Software: Verification and Testing*, N. Piterman, Eds., Cham: Springer International Publishing, November 2015, pp. 225–225.

[7] H. Wu, “Randomization and Restart Strategies,” *Master's Thesis, University of Waterloo*, pp. 1–1, 2006.