SAT ソルバーを用いた命題論理問題の説明と具体的問題に対する計算機実験

文理学部情報科学科
5419045 高林 秀

2021 年 11 月 1 日

概要

本稿は、今年度論理と計算 2 における課題学習として「命題論理の説明」及び「SAT ソルバーを使用した、その具体的問題の解決を行う計算機実験」を行うものである。本稿の冒頭〜中盤では関係理論の説明を行い、終盤ではその理論を利用して、実際に具体的な問題を SAT ソルバーを使用して解答する。なお、本演習には SAT ソルバーとして clasp を使用した。

1　目的

本稿は、今年度論理と計算 2 における課題学習として、SAT ソルバーを用いた命題論理による宣言的問題解決を通じ，命題論理に関する学修内容を振り返ることを目的とする。

本稿は大まかに次のように構成される。

1. 計算理論説明
   （a）命題論理における解釈とモデル、その他関連する事項について
   （b）SAT 問題とはなにか
   （c）DPLL アルゴリズムの解説
2. 計算機実験
   （a）N 人の女王
   （b）グラフ頂点の彩色
3. 各問に関する考察
4. まとめ
5. 巻末資料

2　計算理論説明

この章では、今回の計算機実験に使用した各計算理論の解説を行う。

2.1 命題論理とは

まず命題論理とはなにか説明する。命題論理という言葉の意味はデジタル大辞泉に次のように書かれている。

> 記号論理学の基礎的部門。個々の命題を結合する「かつ」「または」「ならば」「でない」などの関係を、論理記号を用いて論理積（＞）・論理和（＜）・含意（→）・否定（～）などにより記号化して演算形式に表し、複合された命題を研究する学問。命題計算。

そもそも命題は、数学では「真偽の判断の対象となる文章または式」であり、論理学においては「判断を言葉で示したもので真または偽」という性質を持つもの、という意味である。したがって、命題論理とは、命題同士の関係性を論理記号を使用して記号化し、演算できるようにしたものということだ。

■結合子　今説明したように、命題論理では命題同士の性質、関係性を扱う。それを説明する上で「結合子（論理記号）」と呼ばれるものが定義されている。

記号	訳	意味
$\wedge$	連言	プログラミングではよく and、&&として扱われる。p かつ q
$\vee$	選言	プログラミングでは or, ||。p または q
$\neg$	否定	プログラミングでは not, !。p ではない
$\Rightarrow, \supset$	含意	～ならばの意味で使われる。直感的には「p が真であるとき、必ず q は真である」
$\Leftrightarrow, \equiv$	同値	「p は q である」が true のとき、もしくはその時点に限り true であるとき。p と q は同
$\top$	トートロジー (恒真)	後述するトートロジーを示す記号
$\bot$	恒偽 (矛盾)	後述する恒偽を示す記号
(補足)$\veebar, \oplus$	排他的論理和	NAND と呼ばれるもの。

■命題文～原子文, 複合文　命題論理は次の要素から構成される。

- 文、命題文 (sentence) ※これは命題論理式とも言う。
  - 原子文 (atomic formula)：これ以上分解することができない命題。最も単純な文。いわゆる 1 つの命題であり、以下の例のようにそれぞれ固有の記号で示すことができる。
    （例）p：「動物はいつか死ぬ」,q：「人間は動物である」,z：「人間はいつか死ぬ」
  - 複合文 (complex sentence)：文同士を「結合子」で連結した文。命題同士の関係性を結合子を利用して連結し、新たな文を作ることができる。
    （例）$p \wedge q \Rightarrow z$：「動物はいつか死ぬ」かつ「人間は動物である」ならば「人間はいつか死ぬ」。

まとめると、命題文には原子文や複合文と呼ばれる区分けが存在し、原子文は「真 (true)、偽 (false)、それ以上分解できない命題（記号）」であり、複合文は「命題文同士を結合子で連結した新たな命題文」ということになる。

なお、true,false と呼ばれる命題の真偽を示すこれらの記号は論理定数と呼ばれる。原子文の例で示した命題文を各記号に置き換えたもの p, q, z は命題記号ないしは命題変数と呼ばれる。

■命題文の構文　これらの要素を組み合わせて作られる命題文は、使用する結合子によって以下に区分けされる。

- 否定文：結合子 $\neg$ で連結されている複合文。
- 連言文：結合子 $\wedge$ で連結されている複合文。
- 選言文：結合子 $\vee$ で連結されている複合文。
- 含意文：結合子 $\Rightarrow$ で連結されている複合文。
- 同値文：結合子 $\Leftrightarrow$ で連結されている複合文。

とくに含意文 $\alpha \Rightarrow \beta$ については α を前提、β を帰結と呼ぶ。また、原子文とその否定文 $p, q, \neg z$ をひとくくりにしてリテラルと呼ぶ。

■知識ベース　また、命題文の集合において、各文を結合子 $\wedge$ で連結したものを知識ベースと呼ぶことがある。（例：KB[KnowledgeBase] = {S1, S2, S3} $\Rightarrow$ S1 $\wedge$ S2 $\wedge$ S3)

■真理値表 (trueth table)　命題文の真偽は先に上げた論理定数 true, false で示す。複合文の真偽を表にまとめて示したものを真理値表と呼ぶ。先に上げた各構文の真理値は、真理値表を用いて次のように定義されている。

定義 1. 複合文の真理値は以下のとおりである。

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

上記真理値表より、含意文に関して重要なことを以下に列挙する。

- 前提 α が false の場合、帰結 β の真理値に関係なく、式として true になる。
- 前提 α と帰結 β の間に因果関係や関連性は要求されない。すなわち、前提文と帰結文がかけ離れた話題であっても、前提が真ならば含意文としては真となる。（例：p：のび太は人間である $\Rightarrow$ q：スネ夫は金持ちのボンボンである、の p, q に直接的な関連性はないが、前提の p が true であるので、文としては true（正しい）となる。）

ここまで、基本用語の説明を行った。以降は、命題論理における「解釈、モデル、伴意関係」とはなにか、加えて SAT 問題や DPLL アルゴリズムについて説明する。

2.2　解釈について

解釈という言葉の意味は日本国語大辞典に次のように書かれている (一部抜粋)。

> 語句や物事の意味、内容などを説明すること。解き明かすこと。また、その解説。
> 物事、特に表現されたものを、自分の経験や判断力などによって理解すること。

法令の意味を明確にして、その内容が動かないように定めること。

命題論理における「解釈」とは、これらの意味に関係なく次のように定義する。

定義 2. 命題文 α に現れる原子文（または命題記号）を $\alpha_1, \alpha_2, \alpha_3, ..., \alpha_n$ とする。このとき、各原子文 $\alpha_{i(1\leq i\leq n)}$ に対する真理値（*truem false*）の割当を「α の解釈」と呼ぶ。

命題記号 p, q, r が定義されているとする。このとき、命題文「$p \wedge q \vee r$」の解釈は次のものが考えられる。

$$\{p = true, q = true, r = true\}, \{p = true, q = true, r = false\},$$
$$\{p = false, q = true, r = true\}, \{p = false, q = true, r = false\},$$
$$\{p = true, q = false, r = true\}, \{p = true, q = false, r = false\},$$
$$\{p = false, q = false, r = true\}, \{p = false, q = false, r = false\}$$

そしてその組み合わせの数は $2^3 = 8$ より 8 個である。一般化すると、「n 個の原子文がある時、解釈の個数は 2^n個」である。

ただし、こうズラズラ書いていては非常に煩雑であるので、以下のように true が割り当てられている命題記号だけ抜き出して記述する略記法も定義されている。上記例の場合は以下のようになる。

$$\{p, q, r\}, \{p, q\}, \{q, r\}, \{q\}, \{p, r\}, \{p\}, \{r\}$$

すなわち、解釈とは命題記号に対する true,false の割当パターンのことである。この解釈のうち、命題文 α の真理値を true にする解釈が、次に示す「モデル」と呼ばれる。

2.3 モデルについて

モデルという言葉の意味には次のようなものが挙げられるだろう。

- ある事柄の手本や見本
- ある事象について、様々な要素とそれらの相互関係を定式化したもの
- 機械学習モデルなど、データ解析の手法のこと。

命題論理における「モデル」とは先に示したとおり、「命題文の真理値を true にする解釈」のことである。

定義 3. 命題文 α の真理値を真 *(true)* にする解釈 I を α のモデルと呼ぶ。

繰り返すが、命題文を true にする解釈をモデルと呼ぶ、ただそれだけである。

真理値表では、各行がそれぞれ一つの解釈となっており、全体の命題文を真にする行（解釈）がモデルということだ。

2.3.1 トートロジーと充足可能

任意の命題文の真理値表を作成すると、すべての解釈で true になる命題文がでてくることがある。この様な命題文ないしは命題論理式を「トートロジー (tautology)」と呼ぶことがある。トートロジーとは日本語で「恒真」、すなわち常に真である、ということを意味する。以下、`https://qiita.com/kimunny/items/195f45154b6cc6a2940b` より引用する。

命題論理において、パラメータの命題の値にかかわらず、常に真になる論理式をトートロジー、あるいは常真式と呼びます。

以下にトートロジーになる命題論理式の例を示す。これらは後述する「命題論理式の標準形」に変形する際に必要となる考え方である。

$\alpha \Rightarrow \alpha$: 排中律
$\alpha \vee \neg\alpha$: 二重否定
$(\alpha \Rightarrow \beta) \Leftrightarrow (\neg\alpha \vee \beta)$：含意記号の定義
$(\alpha \Leftrightarrow \beta) \Leftrightarrow (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$：同値記号の定義
$(\alpha \Rightarrow \beta) \Leftrightarrow (\neg\beta \Rightarrow \neg\alpha)$：対偶
$\alpha \wedge \beta \Leftrightarrow \beta \wedge \alpha$: 連言交換率
$\alpha \vee \beta \Leftrightarrow \beta \vee \alpha$: 選言交換率
$\neg(\alpha \wedge \beta) \Leftrightarrow \neg\alpha \vee \beta$: ド・モルガンの法則
$\neg(\alpha \vee \beta) \Leftrightarrow \neg\alpha \wedge \neg\beta$: ド・モルガンの法則
$\alpha \wedge (\beta \wedge \gamma) \Leftrightarrow (\alpha \wedge \beta) \wedge \gamma$: 連言結合律
$\alpha \vee (\beta \vee \gamma) \Leftrightarrow (\alpha \vee \beta) \vee \gamma$: 選言結合律
$\alpha \wedge (\beta \vee \gamma) \Leftrightarrow (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$：連言の選言への分配率
$\alpha \vee (\beta \wedge \gamma) \Leftrightarrow (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$：選言の連言への分配率

また、真理値表を書くとトートロジーであることを確認することができる。

α	β	γ	α	$\vee$	$(\beta \wedge \gamma)$	$\Leftrightarrow$	$(\alpha \vee \beta)$	$\wedge$	$(\alpha \vee \gamma)$
false	false	false	false	false	false	true	false	false	false
false	false	true	false	false	false	true	false	false	true
false	true	false	false	false	false	true	true	false	false
false	true	true	false	true	true	true	true	true	true
true	false	false	true	true	false	true	true	true	true
true	false	true	true	true	false	true	true	true	true
true	true	false	true	true	false	true	true	true	true
true	true	true	true	true	true	true	true	true	true

α	β	$\neg$	$(\alpha \wedge \beta)$	$\Leftrightarrow$	$\neg\alpha$	$\vee$	$\neg\beta$
false	false	true	false	true	true	true	true
false	true	true	false	true	true	true	false
true	false	true	false	true	false	true	true
true	true	false	true	true	false	false	false

これとは反対に、どの解釈でも常に偽になる命題論理式を「矛盾 (contradictory well-formed formula)」または「恒偽」と呼ぶ。

また、真にも偽にもなりうる命題論理式を「整合式 (consistent well-formed formula)」または「充足可能」と呼ぶ。

これらの言葉を使うと、先に述べた論理定数である true と false はそれぞれ、「トートロジー, 恒偽」であり、任意の命題変数 p は true,false どちらも取り得るため充足可能である、と言える。整理すると以下。

- 論理定数 true:トートロジーである。
- 論理定数 false:恒偽である。
- 任意の命題変数 p:充足可能である。

すなわち、ある命題文、命題論理式の真理値表を作成するということは「その命題文がトートロジーであるか、それとも恒偽であるか、それとも充足可能であるか」を調査、判定していることになる。

2.3.2 命題論理式の標準形

以降の章で登場する SAT ソルバーを扱う際に必要となる知識である「命題論理式の標準形」について説明する。ここまで示したように、ある文を命題論理式で表すにはいくつかのパターンが考えられる。これらをより制限された文に制限することで、単純な文とすることが求められる。そこで登場するのが命題論理式の標準形と呼ばれるものである。

本稿では以下 2 つの標準形について扱う。

- 選言標準形 (disjunctive normal form)
- 連言標準形 (（conjunctive normal form) ←後述する SAT ソルバーで利用する。

上記の 2 つを総称してここでは標準形と言う。さて、標準形は文を制限すると述べたが具体的にどのように制限するのか。標準形では論理式中で使用できる結合子が限られている。以下は使用できる結合子の一覧である。

- 連言 $\wedge$
- 選言 $\vee$
- 否定 $\neg$

この 3 つのみの結合子を利用して、論理式を記述する。以下、標準形の定義を示す。

定義 4. 選言標準形：すべての選言文を構成する各要素（選言肢）がリテラルの連言である選言文は、選言標準形をしている、という。

「リテラルの連言である選言文」とは、リテラルの連言記号 $\wedge$ を論理式を変形して $\vee$ で連結したものである。例えば、$(p \wedge q) \vee (r \wedge \neg s \wedge t) \vee (q \wedge r)$ などのように、各命題文を結合子 $\vee$ で連結している文の事を指す。

定義 5. 連言標準形：すべての選言文を構成する各要素（選言肢）がリテラルの選言である連言文は、連言標準形をしている、という。

選言標準形と同様の考え方で、リテラルの選言記号 $\vee$ を論理式を変形して $\wedge$ で連結したものである。例えば、$(p \vee q) \wedge (r \vee \neg s \vee t) \wedge (q \vee r)$ などのように、各命題文を結合子 $\wedge$ で連結している文の事を指す。

■標準形への変形　すべての命題文は標準形に変形することができる。具体的には、与えられた命題論理式に対し次のような操作を実行し、求めたい標準形に変形する。

- 同値記号の削除：トートロジーの例で示した「$(\alpha \Leftrightarrow \beta) \Leftrightarrow (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$」により同値記号を削

除する。

- 含意記号の削除：トートロジーの例で示した「$(\alpha \Rightarrow \beta) \Leftrightarrow (\neg\alpha \vee \beta)$」により含意記号を削除する。
- 否定記号の除去と移動：ド・モルガンの法則や、二重否定を用いて論理式中の否定記号を削除・移動する。
- 選言・連言記号の移動：選言・連言の交換率「$\alpha \vee \beta \Leftrightarrow \beta\alpha$」・「$\alpha \wedge \beta \Leftrightarrow \wedge\alpha$」、や選言・連言の分配率「(略)」を利用して、求める標準系の記号を、そうでない記号より内側へ移動させる。

2.4 伴意関係 (Entailment)

命題論理における伴意関係（別名：論理的帰結）とは以下のような意味をもつ。以下 Wikipedia の `https://ja.wikipedia.org/wiki/%E8%AB%96%E7%90%86%E7%9A%84%E5%B8%B0%E7%B5%90` より引用する。

> 論理的帰結（ろんりてききけつ、伴意、英: logical consequence, entailment）は、論理学における最も基本的な概念であり、複数の文（または命題）の集合と 1 つの文（命題）の間が「〜だから、当然〜」という繋がり方をする関係を指す。

すなわち、「命題文の集合」と「命題文」の関係のことを示している。この伴意関係を式で示したものを「伴意式」と呼ぶ。このとき、命題文の集合のことを「理論」と呼ぶ。加えて、この「理論」は先に示した知識ベース「KB」のことであり、各命題文を連言記号で連結したもの「連言文」と同じである。

■伴意式　一般的に伴意式は次のような表記をする。

> $G \models \alpha$
> 意味：「理論 G を文（命題文)αが伴意する」＝理論 G からαが論理的に導出できる。

これは、理論 G の解釈が true であるとき、α も true となることを意味している。すなわち、G が true のとき、α は常に true になるということだ。よって、含意文で示すと $G \Rightarrow \alpha$がトートロジー となる。つまり伴意関係とは、G が true（G の全要素が true）であるとき、α が false になることはありえない、トートロジーであると言っているのである。したがって、命題文が伴意関係にあるか否かはトートロジーであることを確認すればよいことになる。すなわち、含意文のトートロジー判定に還元していることになる。

例を挙げる。$G = \{p, q, (p \wedge q) \Rightarrow r\}$ としたとき、$G \models r$ または、G を展開して、$\{p, q, (p \wedge q) \Rightarrow r\} \models r$ といったように表記する。このとき、$\{p, q, (p \wedge q) \Rightarrow r\}$ は、$(p \wedge q \wedge (p \wedge q) \Rightarrow r)$ と同じであるので、真理値表では以下のように示すことができる。

p	q	r	$p \wedge q$	$p \wedge q \Leftrightarrow r$	$p \wedge q \wedge (p \wedge q \Rightarrow r)$
false	false	false	false	true	false
false	false	true	false	true	false
false	true	false	false	true	false
false	true	true	false	true	false
true	false	false	false	true	false
true	false	true	false	true	false
true	true	false	true	false	false
true	ture	true	true	true	true

以上より、G が true であるとき、r も true となっていることが確認できる。したがって、伴意式「$G \models r$」は成立する（伴意関係にある）。

2.5 SAT 問題とは

前章では、命題論理における解釈とモデルや伴意関係とはなにか、加えてトートロジーや恒偽、充足可能とはなにかについて説明してきた。この章では、先に示した「充足可能」についてより詳細に扱う。

初めに繰り返しになるが、「充足可能」とは「真にも偽にもなりうる命題論理式」のことであることは示した。ある命題論理式が充足可能であるか否かを判定するのが、SAT 問題[*1]（充足可能性問題）である。より厳密には以下のように言われる※引用元 `https://ja.wikipedia.org/wiki/%E5%85%85%E8%B6%B3%E5%8F%AF%E8%83%BD%E6%80%A7%E5%95%8F%E9%A1%8C`。

> 充足可能性問題（じゅうそくかのうせいもんだい、satisfiability problem, SAT）は、一つの命題論理式が与えられたとき、それに含まれる変数の値を偽 (False) あるいは真 (True) にうまく定めることによって全体の値を' 真' にできるか、という問題をいう。

すなわち、与えられた命題論理式にモデルが存在するかを判定する問題ということになる。SAT 問題は、命題論理式が以下のどちらに属するか決定する問題と言える。

- モデルが存在する→充足可能
- モデルが存在しない→恒偽

これを利用して、背理法を使用して命題論理式にモデルが存在しないことを証明することもできる。

余談だが、この SAT 問題は「NP 完全[*2]」であることが最初に証明された問題でもあることが知られている。

では、実際どのように命題論理式にモデルが存在するか否かを求めるのかについて述べる。SAT 問題は、与えられた命題論理式が充足可能かどうか言えれば良い。したがってまず思いつく手法としては先に示した「真理値表を作成する」という方法が思いつくだろう。しかし、真理値表はすべての命題変数に真理値を割り当て

[*1] ・SAT : satisfiability problem の頭 3 文字
・SAT 問題 : Boolean Satisfiability Testing

[*2] NP-complete problem : クラス NP に属する決定問題かつクラス NP に属する任意の問題から多項式時間還元可能な問題のこと。詳細な説明は本稿では扱わないが下記に参考 URL を示す。

- うさぎでもわかる P vs NP 問題 : `https://www.momoyama-usagi.com/entry/info-p-np`

た後、解釈を見ることができる。したがって、論理式中の命題変数の個数が多ければ多いほど、充足可能かどうか判定するのに非常に時間がかかる。そこで、後述する SAT ソルバーと呼ばれるものが存在する。この SAT ソルバーは、単位伝搬や DPLL アルゴリズムなど様々な仕組みによって、効率よく命題論理式が充足可能かどうか判定することができる。単位伝搬や DPLL アルゴリズムについての説明は後述する。まずは、この SAT ソルバーを用いた問題解決システムである「SAT 型システム」について紹介する。

■SAT 型システム　与えられた問題を SAT 符号化し、SAT ソルバーを使用して解くシステムを SAT 型システムと言う。SAT 型システムが生まれた背景としては、問題ごとに特化（適した）アルゴリズムを作成するのではなく、SAT 符号化して SAT ソルバーに解いてもらおうとする考え方があった。問題ごとに特化（適した）アルゴリズムを作成するのは手間と時間がかかることがある。しかし、問題を SAT 符号化すれば、SAT ソルバーに入力するだけで良いので、手間と時間がかからなくなる。他の問題に対しても、SAT ソルバーに対する入力を変更しさえすればよいので、前者よりもはるかに効率が良い。

近年、SAT ソルバーの解を求める性能が飛躍的に向上したこともあり、SAT 型システムは以下のような分野において成功を収めている。

- 自動テストパターン生成
- ソフトウェア検証
- 解集合プログラミング
- Linux パッケージマネージャー（DNF）の依存性解決

この他にも多種多様な分野で SAT 型システムは成功を収めている。他の事例については `https://www.jstage.jst.go.jp/article/jssst/35/4/35_72/_pdf` を参照いただきたい。

2.5.1　SAT の基本アルゴリズム

■木構造化　さて、先に示したとおり命題論理式が true になる解釈を調べる（モデルが存在するか否かを調べる）のが SAT 問題である。問題を解くには各命題変数に true,false の 2 通りの真理値を割り当てていくことになる。よって、命題変数をノード、真理値の割り当てを分岐として捉えることで「二分木」を作成することが可能になる。

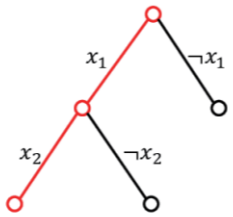

図 1　SAT の木構造化

※引用元：`https://jssst-ppl.org/workshop/2017/slides/ppl2017_c4_soh.pdf`

このように捉えることによって、木構造の特徴である「枝刈り」や「探索打ち切り」を行うことができる。SAT ソルバーはこのように、真理値割当中に命題論理式が true にならないと確定した時点で「探索打ち切り」や「分岐の削減」を行うことで効率を高めている。SAT 問題においては、モデルの有無のみが問われるので、

探索中に 1 つのモデルが存在すればその時点で探索を終了すれば良いことになる。したがって、命題変数への真理値割当の際に命題論理式が true にならないことが確定した時点で探索を打ち切り、次の真理値割当へと進めば良い。その方法が後述する DPLL アルゴリズムの章で触れる「早期停止」や「節学習」、「単位伝搬」と呼ばれる方法であり、それらを組み合わせた SAT アルゴリズムが「DPLL」と呼ばれるアルゴリズムである。

■CNF 式　さて、先に示したように SAT 問題を解くには SAT ソルバーに入力できるよう、形式を整える必要がある。具体的に言うと、SAT 問題は先に示した、連言標準形でつくられる CNF 式によって与えられる。CNF 式とは、「複数の節の論理積、すなわち連言」である。ここで、「節」という単語が登場する。節とは、複数のリテラルの論理和、すなわち選言である。

先に示したように、連言標準形は「$(p \vee q) \wedge (r \vee \neg s \vee t) \wedge (q \vee r)$」のような形をしている。各命題文同士を結合子 $\wedge$ で連結し、1 つの小さな命題文※（）の中では、$\vee$ を使用してリテラルが連結されているのが分かるだろう。CNF 式はいってしまえば、連言標準形で記載された命題論理式を「1,2,3」のような数字記号に変形して、一定の法則で記述したものである。一般に、この一定の法則には「DIMACS CNF」と呼ばれるフォーマットが使用される。

```
p cnf 3 4      ; 命題変数の数␣節の数
1 2 3 0        ; p1 ∨ p2 ∨ p3
-1 -2 0        ; ¬p1 ∨ ¬p2
-1 -3 0        ; ¬p1 ∨ ¬p3
-2 -3 0        ; ¬p2 ∨ ¬p3
```

図 2　DIMACS CNF

※引用元：https://jssst-ppl.org/workshop/2017/slides/ppl2017_c4_soh.pdf

見ての通り、先頭行にある「p cnf」はおまじないみたいなもので、その隣りにある「3 4」の部分がそれぞれ「命題変数の個数」と「節の個数」を示している。続く 2 行目以降に、命題論理式の節を記載していくのだが、このとき否定記号を「マイナス −」で示す。また、各命題変数には固有の数字記号が与えられる。命題論理式の記述は、先頭行で示した節の個数分だけ記載する。

2.6　DPLL アルゴリズム

ここまで、SAT 問題とは何かまた、SAT システムについてや SAT ソルバーに入力を渡すための CNF 式について説明した。この章では、SAT ソルバーが実際にどの様なアルゴリズムで充足可能か否か判定するのかを見ていく。

SAT ソルバーが使用するアルゴリズムとして有名なものが以下に列挙するアルゴリズムである。

- DPLL(Davis-Putnam-Logemann-Loveland)
- CDCL(Conflict Driven Clause Learning)
- 変数ヒューリスティック VSIDS
- ２リテラルウォッチ

上記以外にも複数個存在する。本稿では DPLL について説明する。

DPLL は 1962 年に開発されたアルゴリズムで、開発者の名前をとって DPLL と呼ばれている。DPLL は一言で言うと「二分木の深さ優先探索と単位伝搬を組み合わせたアルゴリズム」である。

まず、前章でも述べたが命題論理式の各命題変数に対する真理値の割当行為は二分木として捉えることができる。繰り返しになるが、この様に捉えることで、木構造の特徴である「枝刈り」や「探索打ち切り」を行うことができる。S 真理値割当中に命題論理式が true にならないと確定した時点で「探索打ち切り」や「分岐の削減」を行うことで効率を高める。SAT 問題においては、モデルの有無のみが問われるので、探索中に 1 つのモデルが存在すればその時点で探索を終了すれば良いことになる。したがって、命題変数への真理値割当の際に命題論理式が true にならないことが確定した時点で探索を打ち切り、次の真理値割当へと進めば良い。

■単位伝搬　ここではまず「単位伝搬」について説明する。単位伝搬とは「単位節の伝搬」のことを意味している。単位節とは「論理式中に 1 つのみ除いて、ほか全てのリテラルに false が割り当てられているような節」のことを言う。例えば、「$B = false$」となっている時、節「$B \vee \neg C$」は B のみに false が割り当てられ、C に対する真理値割当は未確定の状態である。したがって、1 つのリテラルを除いて、他のリテラルすべて（今回は B）に false が割り当てられているので、単位節となる。

SAT 問題では充足可能か否かを判定する。したがって、命題論理式が充足可能であるためには、ある真理値の割当において、すべての節が true にならなければならない。したがって、単位節に出現する「真理値未割り当てのリテラル」というのは絶対に true でなければならない。なぜなら、例でも示したとおり単位節は、未割り当てのリテラル以外、全てのリテラルに false が割り当てられているからだ。したがって、残り 1 つのリテラルは true にならなければ、節として true とすることができない。

この性質を利用すると、単位節中のリテラルは計算せずとも自動的に真理値が確定する。先程の例を上げると、「$B = false$」であるので、残りのリテラルである「$\neg C$」は true でなけれななならない。つまり、「$C = false$」であることが確定する。

また、例えば他の節で「$C \vee A$」があったとしよう。ここまで「$B = false, C = false$」である。したがって節「$C \vee A$」は単位節であり、「$A = true$」が確定する。このように一つの単位節に対する真理値の割り当てが、自動的に別の単位節を生み出す可能性がある。単位伝搬とはこのことである。なので付け加えるならば「単位（節）伝搬」と呼ぶほうが表現上適切かもしれない。

■単位伝搬とバックトラック　すべての単位節中のリテラルに対し、最初に真理値を割り当てる、すると下図のように、恒偽（矛盾）となる節が出現する。此の様な場合、それまで行ってきた真理値の割り当てが不適切であったということになるので、バックトラックをして他の選択肢、すなわち次の解釈を求める。

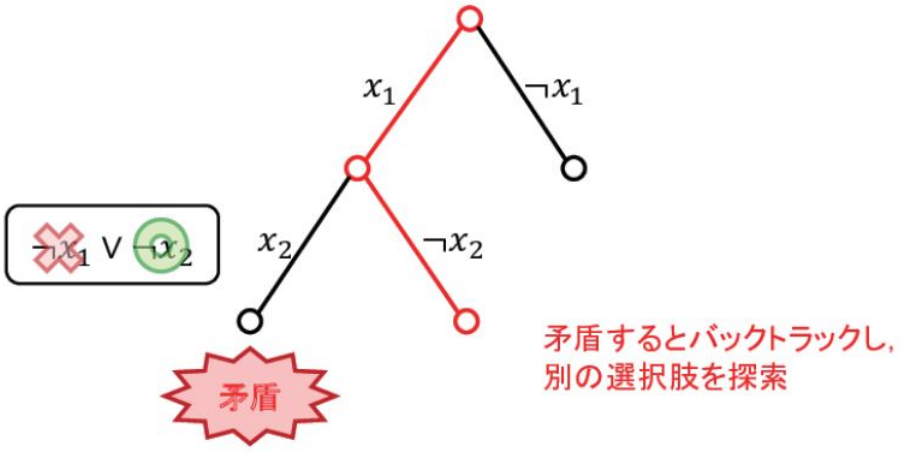

図 3　探索の打ち切り

※引用元：https://jssst-ppl.org/workshop/2017/slides/ppl2017_c4_soh.pdf

単位伝搬によって、自動的に定まる真理値を見つけ、恒偽となる節を導くことにより探索木の深さを低減することができる。したがって、全体の計算時間短縮にもつなげることができる。

■早期停止・純粋記号ヒューリスティックス　単位伝搬という方法以外にも、いくつか真理値割り当て前に、真理値を決定できる場合がある。その一つが、「早期停止」と呼ばれる現象である。

早期停止とは、「部分的に完成しているモデルに対し、その真偽を検出することで、すべての命題変数に真理値を割り当てる前に充足可能か否か決定できる場合がある」ということである。繰り返しになるが、充足可能であるためには命題論理式中のすべての節が true でなければならない。SAT 問題における命題論理式は連言標準形であるので、節中の命題変数は選言記号 $\vee$ で連結されている。したがって、少なくとも 1 つのリテラルが true であれば、その節は true とすることができる。例を上げると、「$\alpha \vee \beta \vee \gamma$」このような節があった時に、$\{\alpha = false, \beta = true, \gamma = false\}$（単位節）であればこの節は true になる事ができる。このような節が、連言記号 $\wedge$ で連結されているので、1 つでも false の節が発生した場合、その解釈は false となるので、充足不可能となる。

まとめると、早期停止とはわざわざすべての変数に真理値の割り当てを行わなくとも、割り当て中に false となる節が発生した場合、その後に続く節の真理値に関係なく充足不可能である、と決定できてしまうということだ。

純粋記号ヒューリスティックとは、「純粋な記号に対する真理値割当において、true が確定している節は無視する」というものである。「純粋な記号」とは、「すべての節で同じ符号を持つ命題記号」のことで、例えば、$(A \vee \neg B) \wedge (\neg B \vee C) \wedge (A \vee C)$ という命題論理式における「A, B」のことである。この例の場合、命題変数 A は正のリテラルとして（¬ がなし）、B は負のリテラルとして（¬ があり）各節中で使用されている。反対に C は、節ごとに符号（¬ の有無）が異なるので純粋記号とは見なさない。

この純粋記号に対して true が割り当てられている節は、その節が false になることはないという性質を利用して、すべての命題変数に真理値を割り当てる前に充足可能か否か判定できるのが純粋記号ヒューリスティックである。これは、実際に前に示した例に真理値を割り当てる (A=true, ¬B=true) と分かるが、純粋記号に

true を割り当てるとその節自体が自動的に true となりその節を無視することができる。なぜなら、節中の最低 1 つのリテラルが true になっている時点でその節は true であるからだ。したがって、残りの真理値未割当のリテラルがどうなろうと節が true であることは変わらない。

最後に、DPLL のアルゴリズムの概略を示す。ここまで紹介した、単位伝搬、早期停止、純粋記号ヒューリスティック、二分木の深さ優先探索を組み合わせて、このアルゴリズムは動作している。

以下にプログラム言語 scala の文法に従った、DPLL の疑似コードを示す。

```
   def solve(clauses: Set[Clause], assignment: Map[Bool,Boolean]): Boolean = {
 2:    val a = unitPropagation(clauses, assignment)
 3:    if (clauses に部分真理値割当 a を適用した結果，空節が含まれている)
 4:      false
 5:    else if (clauses に部分真理値割当 a を適用した結果，全節が恒真である)
 6:      true
 7:    else {
 8:      val p = select(clauses, a)
 9:      solve(clauses, a + (p -> false)) || solve(clauses, a + (p -> true))
10:    }
11:  }
```

※引用元：https://tamura70.gitlab.io/lect-proplogic/org/proplogic-sat.html

各単語を説明する。clauses は「節集合」であり、assignment は真理値割当である。この 2 つを引数にする。この関数は、clauses が充足可能であるとき true を返却し、充足不能であるときは false を返却する。コード 2 行目において、単位伝搬による、必然的な真理値の割当を求めている。3~~4~~ 行目にて、早期停止の考え方に従い、求めた真理値割当で true になる節が無い場合、すなわち節が空であるとき、関数は false を返却し即座に終了する。充足不可能ということだ。5~~6~~ 行目にて、暫定的な真理値割当で全ての節がトートロジーであるとき、この関数は true を返却し即座に終了する。充足可能ということだ。8 行目にて、先に示した純粋記号ヒューリスティックの考え方にしたがって、真理値割り当てにおいて true が確定している節を無視し、9 行目にて 1 つのリテラルに真理値を割り当て、節が充足可能か否か調べていく。

3 計算機実験

3.1 実験準備

3.1.1 実験環境

今回の実験は仮想マシン上で clasp のバイナリをダウンロードして行った。下記に実験時の環境を示す。

- ホスト OS：Window10 Home 20H2
- 仮想 OS：Ubuntu 20.04.2 LTS
- CPU：Intel(R)Core(TM)i7-9700K @ 3.6GHz
- GPU：Nvidia Geforce RTX2070 OC @ 8GB
- ホスト RAM：16GB

- 仮想 RAM：4GB

SAT ソルバー clasp を利用するには、先に示した CNF 形式のファイルを作成し、以下のコマンドを入力することで実行することができる。

```
$ ./clasp [オプション] cmf ファイル名
```

オプションの部分に 0 を入れると、すべての解を示し、何もなければ一番最初に見つけた解を表示する。

3.1.2 問題 1:N 人の女王

配布資料中に、Processing のプログラムが「nQueen.pde」として以下の関数が用意されている。

- バックトラック法を用いて nQueen を解く関数
- clasp への入力ファイルを作成する関数

1. この問題に対する SAT 符号化を詳細に説明せよ。
2. N の大きさを様々に変えながら，バックトラック法で解いた場合と SAT ソルバーで解いた場合とでの実行時間を比較・考察しなさい.

実験の手法としては以下の通りとする。

1. nQueen.pde 内の setup() 関数にある変数 N の値を任意の値に設定する。
2. まず、makeCNF 関数を呼び出し N 人の場合の cnf ファイルを自動生成する。※生成後のファイル名「queen_N.cnf」
3. 上記コマンドで、SAT ソルバーに生成した cnf ファイルを入力し、実行する。時間表記は秒
4. nQueen.pde 内の setup() 関数内にて、関数 nqueen を呼びだし、実行する。時間表記はミリ秒。
5. 両者の実行時間を比較、考察する。

3.1.3 問題 2:グラフ頂点の彩色問題

配布資料中の「GraphColoring」フォルダに、「都道府県の隣接関係」を表すグラフの頂点彩色問題の CNF ファイルが用意されている。

1. この問題に対する SAT 符号化を詳細に説明せよ。
2. 関東地方を対象に、いくつの塗分け方法があるか調べなさい。
3. 47 都道府県を対象とした色塗りの例を一つ示しなさい。
   (a)（例）長野県：青色, 神奈川県：赤色、のように、どの都道府県をどの色で塗るのかを具体的に示すこと。

3.2 各問に対する解答・考察

3.2.1 問題 1:N 人の女王

■実行結果 初めに SAT ソルバーを用いた実行時間と、バックトラック法で解いた場合の実行時間を比較する。

N=5 としたときは以下のようになった。

- SAT ソルバーの場合
  - 総実行時間：0.000 秒
  - CPU 実行時間：0.000 秒
  - 結果：充足可能
- バックトラック法の場合
  - 総実行時間：0.000 秒
  - 結果：充足可能

N=10 としたときは以下のようになった。

- SAT ソルバーの場合
  - 総実行時間：0.002 秒
  - CPU 実行時間：0.002 秒
  - 結果：充足可能
- バックトラック法の場合
  - 総実行時間：0.001 秒
  - 結果：充足可能

N=20 としたときは以下のようになった。

- SAT ソルバーの場合
  - 総実行時間：0.698 秒
  - CPU 実行時間：0.697 秒
  - 結果：充足可能
- バックトラック法の場合
  - 総実行時間：0.053 秒
  - 結果：充足可能

N=30 としたときは以下のようになった。

- SAT ソルバーの場合
  - 総実行時間：2.227 秒
  - CPU 実行時間：2.223 秒
  - 結果：充足可能
- バックトラック法の場合

  - 総実行時間：5.075 秒
  - 結果：充足可能

N=40 としたときの結果を載せる。

- SAT ソルバーの場合
  - 総実行時間：1.336 秒
  - CPU 実行時間：1.335 秒
  - 結果：充足可能
- バックトラック法の場合
  - 総実行時間：測定不可

■考察　N=20 までは、SAT ソルバーより、バックトラック法の方が実行時間が速かったが N=30 以上から、SAT ソルバーの方が 2 倍ほど実行時間が速いことが分かる。加えて、N=40 の場合では SAT ソルバーは約 1.3 秒ほどなのに対し、バックトラック法では数分間待機しても計算が終了せずに結果が得られなかった。

まず、N=20 までの場合について考察する。結果をみても分かるとおり N=20 までは、SAT ソルバーよりバックトラック法の方が速く計算できていることが分かる。これは、SAT ソルバーが利用しているアルゴリズムの場合、理論説明でも示したとおり純粋記号ヒューリスティックや、単位伝搬等の複数の処理を経ているため、N の値が小さい時では反対に効率が悪くなっているものと考えられる。N=20 程度の場合では、単位伝搬等の処理を利用している SAT ソルバーよりも、それらの処理を介さないバックトラック法の方が有利であると考えられる。

次に、N=30 の場合について考察する。ここから SAT ソルバーの方が、バックトラック法に対して 2 倍程度の差をつけて速くなっている。これは、N=20 の場合と逆の現象で、SAT ソルバーが利用する単位伝搬や、純粋記号ヒューリスティック等の処理を介する方が、それらを介さないバックトラック法よりも適していることを意味するだろう。N の値が大きくなればなるほど、命題変数の数、節の数を増えることになる。なので、バックトラック法では単位伝搬等の探索の打ち切りを行える処理を介さないので SAT ソルバーよりも大きく実行時間に遅れを取っていると思われる。

上記の考察を裏付ける結果として、N=40 の場合の結果をみると分かるが、ここまでくるともはやバックトラック法では計算が追いついていない。それに対して、SAT ソルバーは N=30 の場合よりも若干速度を速めるまでになっている。これは、単位伝搬や純粋記号ヒューリスティックなどの処理を介することで、命題変数に真理値を割り当てるまえに充足可能かどうか判定できるので、それらを利用しないバックトラック法よりも、速度が上がっていると考えられる。なお、N=20 での命題変数の数と、節数はそれぞれ、400 と 12560 であり、N=40 の場合では 1600 と 103520 である。

以上を総合すると、バックトラック法で処理する場合は命題変数の個数が 400 個程度、節数が 12000 個程度であれば SAT ソルバーよりも速く処理できると考えられる。しかし、命題変数の個数が 1600 個程度、節数が 102000 個にまで達すると、バックトラック法では計算が追いつかない。したがって、SAT ソルバーは命題変数や節の数が非常に大きい場合に適していると言えるだろう。

3.2.2　問題 2:グラフ頂点の彩色問題

4　まとめ

5　巻末資料

本稿で使用した画像、プログラムコード等はすべて以下のリンク先に掲載している。必要に応じてご覧頂きたい。

- GoogleDrive:https://drive.google.com/drive/folders/1kOW_1KPUw_kBznaMWjge7HaBI7FoRAoq?usp=sharing
- GitHub:https://github.com/tsyu12345/logical_and_calculating_LectureCode/tree/master/No5