

1 SAT とは

SATとはSatisfiability(充足可能)の略であり、与えられた論理式が充足可能であるかを判定する問題を充足可能性問題(SAT問題、SAT)といいます。

SAT問題で対象とする論理式は和積標準形(Conjunctive normal form, CNF, 主乗法標準形とも)とよばれる論理式です。

1.1 論理式

論理式は論理変数からなります。論理変数は0(偽)または1(真)をとる変数です。論理変数 p は1変数からなる論理式です。

論理変数 p と q の論理和を $p \vee q$ であらわします。 p, q がともに0のときに $p \vee q$ が0となり、かつそのときに限ります。論理変数 p と q の論理積を $p \wedge q$ であらわします。 p, q がともに1のときに $p \wedge q$ が1となり、かつそのときに限ります。論理変数 p の否定を $\neg p$ とかきます。 p が1のとき $\neg p$ は0をとり、 p が0のとき $\neg p$ は1をとる。

論理変数に否定を表す論理記号 $\neg$ をつけた $\neg p$ も1変数からなる論理式です。論理式に否定を表す論理記号 $\neg$ をつけたもの、論理式の記号($\vee, \neg, \rightarrow$)で結合したのも論理式です。 $p \vee q, p \wedge q, p \rightarrow q$ は論理式になります。もっと複雑な $(p_1 \vee p_2 \vee p_3 \wedge p_4) \vee (p_1 \vee p_2 \wedge \neg p_3 \wedge p_4) \wedge ((\neg p_1 \vee p_3) \rightarrow p_5) \wedge \neg p_6 \vee (p_2 \vee p_4 \wedge \neg p_6) \wedge (\neg p_3 \wedge \neg p_5 \vee p_6)$ なども論理式になります。

$p \rightarrow q$ は $\neg p \vee q$ と同値なので $\rightarrow$ を含む論理式は $\neg$ と $\vee$ を使って書き換えることができます。 $(\neg p_1 \vee p_3) \rightarrow p_5$ は $(\neg p_1 \vee p_3) \rightarrow p_5 \equiv \neg(\neg p_1 \vee p_3) \vee p_5 \equiv (p_1 \wedge p_3) \vee p_5 \equiv (p_1 \vee p_5) \wedge (p_3 \vee p_5)$ となります。

論理式 P, Q に対しても論理変数と同様に論理式の論理和 $P \vee Q$ 、論理積 $P \wedge Q$ 、否定 $\neg P$ が論理変数の場合と同様に定義できます。

1.2 和積標準形

論理式についてざっくりと説明したので、次は和積標準形です。論理変数 x_1 に対して $x_1, \neg x_1$ をリテラルといいます。リテラルを $\vee$ で結んだものを節といいます。 $x_1 \vee x_2, x_1 \vee x_2 \vee \neg x_3$ などは節です。1つの節からなる論理式は和積標準形です。節の論理積も和積標準形です。例えば $P = (p_1 \vee p_2) \wedge (p_2 \vee \neg p_3) \wedge (p_1 \vee p_3) \wedge (\neg x_p \vee p_4)$ は和積標準形ですが $Q = (x_p \vee p_2) \vee (p_2 \vee \neg p_3) \wedge (p_1 \vee p_3) \wedge (\neg p_2 \vee p_4)$ は和積標準形ではありません。それは節が論理和 $\vee$ で結合されているからです。 $R = (x_p \vee p_2) \wedge (p_2 \wedge \neg p_3) \wedge (p_1 \vee p_3) \wedge (\neg p_2 \vee p_4)$ も和積標準形ではありません。2番目の節でリテラル p_2 と $\neg p_3$ が論理積 $\wedge$ で結合されているからです。つまり、 $P = P_1 \wedge P_2 \wedge \dots \wedge P_n$ において $P_1, P_2, \dots, P_n$ の各節がリテラルの論理和で構成されているとき P を和積標準形といいます。また、今後、和積標準形 $P = P_1 \wedge P_2 \wedge \dots \wedge P_n$ を $P = P_1 P_2 \dots P_n$ というように各節を結ぶ $\wedge$ は省略して書きます。なので $P = (p_1 \vee p_2) \wedge (p_2 \vee \neg p_3) \wedge (p_1 \vee p_3) \wedge (\neg x_p \vee p_4)$ とは書かずに $P = (p_1 \vee p_2)(p_2 \vee \neg p_3)(p_1 \vee p_3)(\neg x_p \vee p_4)$ と書きます。

任意の論理式は主乗法標準形に変形可能であることが知られています。 $P = P_1 \wedge P_2 \vee P_3$ は

$P = (P_1 \vee P_3) \wedge (P_2 \vee P_3)$ とできます。例として $P = (x_1 \vee x_2) \wedge (x_2 \wedge \neg x_3) \vee (x_1 \vee x_3) \wedge (\neg x_2 \vee x_4)$ では

$$\begin{aligned}
P &= (x_1 \vee x_2) \wedge (x_2 \wedge \neg x_3) \vee (x_1 \vee x_3) \wedge (\neg x_2 \vee x_4) \\
&\equiv ((x_1 \vee x_2) \wedge (x_2 \wedge \neg x_3) \wedge (\neg x_2 \vee x_4)) \vee (x_1 \vee x_3) \\
&\equiv ((x_1 \vee x_2) \vee (x_1 \vee x_3)) \wedge ((x_2 \wedge \neg x_3) \vee (x_1 \vee x_3)) \wedge ((\neg x_2 \vee x_4) \vee (x_1 \vee x_3)) \\
&\equiv (x_1 \vee x_2 \vee x_3) \wedge ((x_2 \wedge \neg x_3) \vee (x_1 \vee x_3)) \wedge (x_1 \vee \neg x_2 \vee x_3 \vee x_4) \\
&\equiv (x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee (x_1 \vee x_3)) \wedge (\neg x_3 \vee (x_1 \vee x_3)) \wedge (x_1 \vee \neg x_2 \vee x_3 \vee x_4) \\
&\equiv (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_3 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3 \vee x_4) \\
&\equiv (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3 \vee x_4) \\
&= (x_1 \vee x_2 \vee x_3)(x_1 \vee x_2 \vee x_3)(x_1 \vee \neg x_2 \vee x_3 \vee x_4)
\end{aligned}$$

となります。

人工知能における推論も論理式に変換できます。例えば推論 $p_1 \rightarrow p_2, p_3 \rightarrow p_4 \vdash p_5$ が妥当であることは論理式 $(p_1 \rightarrow p_2 \wedge p_3 \rightarrow p_4) \rightarrow p_5$ がトートロジーであることと同値です。また、論理式 $p \rightarrow q$ は $\neg p \vee q$ と同値な論理式であるので、 $p_1 \rightarrow p_2, p_3 \rightarrow p_4 \vdash p_5$ の妥当性は論理式

$$\begin{aligned}
(p_1 \rightarrow p_2 \wedge p_3 \rightarrow p_4) \rightarrow p_5 &\equiv \neg(p_1 \rightarrow p_2 \wedge p_3 \rightarrow p_4) \vee p_5 \\
&\equiv \neg((\neg p_1 \vee p_2) \wedge (\neg p_3 \vee p_4)) \vee p_5 \\
&\equiv (\neg(\neg p_1 \vee p_2) \vee \neg(\neg p_3 \vee p_4)) \vee p_5 \\
&\equiv (p_1 \wedge \neg p_2) \vee (p_3 \wedge \neg p_4) \vee p_5
\end{aligned}$$

となり、 $(p_1 \wedge \neg p_2) \vee (p_3 \wedge \neg p_4) \vee p_5$ がトートロジーであることと同値です。さらに $(p_1 \wedge \neg p_2) \vee (p_3 \wedge \neg p_4) \vee p_5$ がトートロジーであることは $(p_1 \wedge \neg p_2) \vee (p_3 \wedge \neg p_4) \vee p_5$ の否定が充足可能でないことと同値です。

$$\begin{aligned}
\neg((p_1 \wedge \neg p_2) \vee (p_3 \wedge \neg p_4) \vee p_5) &\equiv \neg(p_1 \wedge \neg p_2) \wedge \neg(p_3 \wedge \neg p_4) \wedge \neg p_5 \\
&\equiv (\neg p_1 \vee p_2) \wedge (p_3 \vee \neg p_4) \wedge \neg p_5
\end{aligned}$$

より、 $(p_1 \wedge \neg p_2) \vee (p_3 \wedge \neg p_4) \vee p_5$ がトートロジーであることは $(\neg p_1 \vee p_2) \wedge (p_3 \vee \neg p_4) \wedge \neg p_5$ が充足可能でないことと同値になります。よって $(\neg p_1 \vee p_2) \wedge (p_3 \vee \neg p_4) \wedge \neg p_5$ が充足可能であれば $(p_1 \wedge \neg p_2) \vee (p_3 \wedge \neg p_4) \vee p_5$ がトートロジーでないことになります。このような推論の妥当性は論理式の充足可能性問題と同値となり、上述の例だけでなく、一般的に推論の妥当性と論理式の充足可能性問題が同値であることが成り立ちます。論理式の充足可能性問題は人工知能への応用を含め重要な問題であり、後で説明する $P=NP$ のような計算機科学の基礎的かつ重要な問題なのです。

推論は論理式に置き換えられ、任意の論理式は和積標準形として書けることを述べました。なのでSAT問題では和積標準形の論理式のみを対象として、その論理式が充足可能かどうかを判定したいのです。

論理式 $P = (p_1 \vee \neg p_2 \vee \neg p_3) \wedge (p_1 \vee p_2 \vee p_4) \wedge (p_1 \vee \neg p_2 \vee p_3 \vee p_4)$ の場合 $p_1 = 1$ とすることで $P = 1$ になります。この場合 p_2, p_3, p_4 は任意の値でよいことになります。 $(p_1, p_2, p_3, p_4) = (1, 0, 0, 0)$ 、 $(p_1, p_2, p_3, p_4) = (1, 0, 0, 1)$ 、 $(p_1, p_2, p_3, p_4) = (1, 1, 1, 1)$ など P を充足するような論理変数への割り当てが複数あります。一般に P を充足するような論理変数への割り当ては1つとは限らないのです。ここでは論理式が1となるような変数への割り当てが存在するかどうかを判定したいので、割り当ては1つしかなくても複数あってもいいのです。要は割り当ての存在が知りたいのです。そしてこのような論理式が1となるような変数への割り当てが存在するとき、その論理式は充足可能であるといいます。さらに和積標準形の論理式 P の各節に含まれるリテラルの個数が高々3であるときを3-SAT問題といいます。

論理式の節に含まれるリテラルの個数を3にする方法について説明します。

まず節のリテラル数が1であるとき、その節と同値な論理式を以下のように構成できます。3-SATでは節に含まれるリテラルの個数が高々3としているので、節のリテラル数が1でも2でも3-SATの対象となる論理式なのですが、ちょうど3にできる方法を示します。節のリテラル数が1であるとき

$$x_1 = (p_1 \vee q_1)(p_1 \vee \neg q_1) = (p_1 \vee q_1 \vee q_2)(p_1 \vee q_1 \vee \neg q_2)(p_1 \vee \neg q_1 \vee q_2)(p_1 \vee \neg q_1 \vee \neg q_2)$$

とします。ここで q_1, q_2 は新たに導入した変数です。しかし、 q_1, q_2 の真偽によらず p_1 と $(p_1 \vee q_1 \vee q_2)(p_1 \vee q_1 \vee \neg q_2)(p_1 \vee \neg q_1 \vee q_2)(p_1 \vee \neg q_1 \vee \neg q_2)$ の真偽は一致します。次に節のリテラル数が2である論理式 $P = (p_1 \vee p_2)$ と同値な論理式は以下のように構成できます。

$$(p_1 \vee p_2) = (p_1 \vee p_2 \vee q)(p_1 \vee p_2 \vee \neg q)$$

ここでも q は新たに導入した変数ですが、 q の真偽によらず $(p_1 \vee p_2)$ と $(p_1 \vee p_2 \vee q)(p_1 \vee p_2 \vee \neg q)$ の真偽は一致します。以上から SAT 問題のインスタンスは節のリテラル数が3以上の主乗法標準形とできることがわかります。和積標準形の論理式のある節が4つ以上のリテラルを含んでいるとし、その節を $P = p_1 \vee p_2 \vee p_3 \cdots \vee p_m$ とします。 $p_i (1 \leq i \leq m)$ はリテラルであり、 $m > 3$ です。新しい変数 q_1 を用意し、論理式

$$P_1 = (p_1 \vee p_2 \vee q_1)(p_3 \vee p_4 \vee \cdots \vee \neg q_1)(\neg p_3 \vee q_1)(\neg p_4 \vee q_1) \cdots (\neg p_m \vee q_1)$$

をつくります。 $P = 0$ となるのはリテラル $p_i (1 \leq i \leq m)$ すべてが0のときで q_1 の値によらず $P_1 = 0$ です。 $P = 1$ となるのはリテラル $p_i (1 \leq i \leq m)$ のいずれかが1となるときです。リテラル $p_i (3 \leq i \leq m)$ のいずれかが1となり $P = 1$ となっているとき $q_1 = 1$ とすることで $P_1 = 1$ とできます。リテラル $p_i (3 \leq i \leq m)$ のすべてが0で $P = 1$ となっているときは p_1 または p_2 が1となるときです。このとき $q_1 = 0$ とすることで $P_1 = 1$ とできます。よって $p_i (1 \leq i \leq m)$ の少なくとも1つが1をとるときにのみ $P_1 = 1$ とできます。 P に含まれるリテラル数が m であるのに対して P_1 の各節に含まれるリテラル数は高々 $m-1$ となっています。この置き換えを各節に含まれるリテラル数が高々3になるまで繰り返すことで SAT 問題の論理式は3-SAT問題の論理式に置き換えることができます。この方法は Karp[5] によるものです。他には $P = p_1 \vee p_2 \vee p_3 \cdots \vee p_m$ に対して

$$P_1 = (p_1 \vee p_2 \vee q_1)(\neg q_1 \vee p_3 \vee q_2)(\neg q_2 \vee p_4 \vee q_3) \cdots (\neg q_{m-4} \vee p_{m-2} \vee q_{m-3})(\neg q_{m-3} \vee p_{m-1} \vee p_m)$$

を構成し、SAT 問題の論理式を3-SAT問題の論理式に置き換えることもできます[4]。ここで $q_1, q_2, \dots, q_{m-3}$ は新たに用意した変数です。

一方で論理式において各節のリテラル数が高々2であるものは2-SAT問題とよばれます。

1.3 NP 完全問題とは

ちょっと話題を変えてNP完全問題について説明します。計算複雑性において問題を”はい”または”いいえ”で答えられる問題(判定問題)に置き換え、その問題の複雑さを求めます。例えば重み付きグラフの最小全域木を求める問題は重みの和がある値以下になる全域木があるかという問題に置き換えられます。このように判定問題に置き換えた問題と、もと

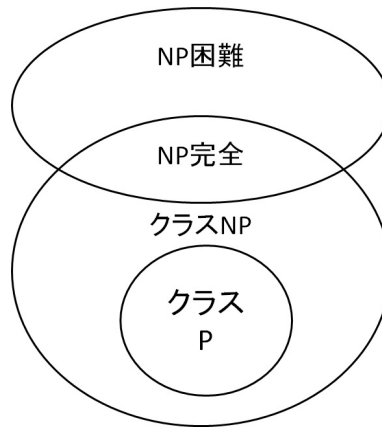


図1 計算クラスの模式図

の問題は同じ計算量であることが知られています。判定問題に対して問題の複雑さのクラスがあります。代表的なクラスはクラスPとクラスNPです。問題がクラスPに属するとはその問題のインスタンスを言語 L に直し、その言語のサイズを n とし、決定性のチューリングマシンが存在して n の多項式で抑えられるステップ数で L が受理されることをいいます。クラスPに属する代表的な問題として2-SAT, 最小全域木問題、2部グラフのマッチング問題、連立方程式などがあります。

一方で問題がクラスNPに属するとはその問題のインスタンスを言語 L に直し、その言語のサイズを n とし、このとき非決定性のチューリングマシンが存在して n の多項式で抑えられるステップ数で L が受理されることをいいます。この非決定性(Non-deterministic)と多項式時間(Polynomial time)の頭文字をとってクラスNPとよばれています。非決定性チューリングマシンは状態遷移の際に確率的な遷移を許すものです。この確率的な遷移を確率1と確率0にすることで、非決定性チューリングマシンは決定性チューリングマシンを模倣できます。よってクラスPはクラスNPに含まれることがわかります。クラスNPに属する任意の問題 L がクラスNPに属する問題 S に帰着可能であるとき、問題 S はNP完全な問題であるといえます。ここで帰着可能であるとは、 L の任意のインスタンスを I として、 I のサイズを m としたとき m の多項式で抑えられるサイズの S のインスタンスに置き換えられ、受理状態が同じになることをいいます。つまり、 S の問題が多項式で解けることは L のどの問題も多項式で解けることになるのです。これを $L \leq S$ と書きます。Cookによって $L \in \text{NP}$ ならば L はSAT問題に帰着可能であることが示されました[1]。さらにこのことから $P = \text{NP}$ であることとSAT問題がPに属することが必要十分であることが導かれます。つまり、SAT問題はNP完全であることが示された最初の問題なのです。クラスNPに属する任意の問題 L が問題 S に帰着可能であるとき、問題 S はNP困難な問題であるといえます。 $L \leq S$ であり、さらに S がNP完全であることを示すには $S \leq L$ を示せばよいのです。NP完全な問題として0-1整数計画問題、グラフの最大クリーク問題、ナップザック問題など多くの組合せ最適化問題が知られています[5]。そして3-SAT問題もNP完全な問題として知られている1つです。このことは次のようにしてわかります。3-SAT問題 $\leq$ SAT問題は3-SAT問題がSAT問題の特別な場合であることから明らかです。一方、SAT問題 $\leq$ 3-SAT問題も前節で述べたように示すことができます。

クラスP、クラスNP、NP完全、NP困難の関係を図1に示します。

前節で $P=NP$ 問題と名前だけ出しましたが、これはクラス P とクラス NP が同じクラスなのではないかというものです。同じクラスになるというのはクラス P とクラス NP も集合ですから、 $P \subset NP$ かつ $P \supset NP$ であるかということです。クラス P がクラス NP に含まれること($P \subset NP$)は先ほど述べました。なのでクラス NP がクラス P に含まれる $P \supset NP$ かどうかの問題となります。しかし今のところ $P \supset NP$ であることは証明も反証もされていません。 $P=NP$ かどうかは未解決問題の一つであり、クレイ研究所が100万ドルの懸賞金をかけたミレニアム懸賞問題の一つです。大方の予想は $P \neq NP$ です。

1.4 SAT を解くアルゴリズム

SAT問題は決定性のアルゴリズムで多項式時間で解けるかはわかりませんが、非決定性のアルゴリズムで多項式時間で解くことはできます。2-SAT問題に対して決定性のアルゴリズムで解けることを示しましょう。

$P = (p_1 \vee p_2)(p_3 \vee p_4) \cdots (p_{n-1} \vee p_n)$ とします。 $p_i (1 \leq i \leq n)$ はリテラルです。また、節の数を m とします。今、節の中には2つのリテラルしかないので p_1 が0になるように定めると $p_2=1$ と値が決定できます。すると p_2 と同じ変数をリテラルとして含む他の節($p_i \vee p_{i+1}$)のリテラルの値も決定できます。 p_2 と p_i が同じ変数のリテラルで $p_i = \neg p_2$ であれば $p_{i+1}=1$ と値を決定できます。以下同様に決定できる節を選びリテラルの値を決定していくことで真偽が決まった変数の系列が得られます。この系列に含まれるある変数が0と1を取らなければならないような矛盾が起きた場合は p_1 は1でなければなりません。すると $p_1=1$ と確定することで P より少ない節をもつ P' が得られます。もし上記のような矛盾が起きない場合はその系列で得られた値を割り当てることにより P より少ない節をもつ P' が得られます。いずれにしても P より少ない節をもつ P' が得られるのです。 P' を得るためには各節で p_1 が含まれるかを判断し、値を割り当てていく必要があるため $O(m)$ 時間かかります。これをすべての n 変数に対して行うのでこのアルゴリズムは $O(mn)$ 時間となり、多項式時間アルゴリズムが設計できるというわけです。以下に2-SATを解くアルゴリズムを以下に示します。

2-SATを解くアルゴリズム: 入力: 論理式 $P(P$ の各節は2つのリテラルからなる)

1. P の最初の節の最初のリテラルを x_1 とする。
2. 関数 f を呼び出す。
3. L_1 と L_0 の両方に含まれる変数が存在するならば矛盾しているので $x_1=1$ とし、 L_1 と L_0 を初期化し、関数 g を呼び出す。 L_1 と L_0 の両方に含まれる変数が存在しないならば、 P において L_1 と L_0 のリストに含まれる変数の値を固定して得られる論理式を P とし、5の処理を行う。
4. 3で g を呼び出した場合、 L_1 と L_0 の両方に含まれる変数が存在するならば $x_1=0, x_1=1$ のいずれの場合も矛盾することになる。よって充足不能であることを出力して終了する。 L_1 と L_0 の両方に含まれる変数が存在しないならば、 P において L_1 と L_0 のリストに含まれる変数の値を固定して得られる論理式を P とし、5の処理を行う。
5. 論理式 P が1ならば充足可能であることを出力して終了する。このとき、 L_1 と L_0 のリ

ストに格納されている変数が $P=1$ となるための割り当てである。論理式 P が 1 でないならば 1 の処理を行う。

関数 f :

1. P の各節で $x_1=0$ となることでその節が 1 となるものを除去し、その除去したものを P とする。
2. P の各節で $x_1=0$ となることでその節の変数が 1 とならなければならないものをリスト L_1 に保存する。
3. P の各節で $x_1=0$ となることでその節の変数が 0 とならなければならないものをリスト L_0 に保存する。

関数 g :

1. P の各節で $x_1=1$ となることでその節が 1 となるものを除去し、その除去したものを P とする。
2. P の各節で $x_1=1$ となることでその節の変数が 1 とならなければならないものをリスト L_1 に保存する。
3. P の各節で $x_1=1$ となることでその節の変数が 0 とならなければならないものをリスト L_0 に保存する。

具体的な 2-SAT 問題のインスタンスでこのアルゴリズムを確認しましょう。例として $P_0 = (p_1 \vee p_2)(p_1 \vee \neg p_3)(\neg p_1 \vee p_4)(\neg p_2 \vee p_3)(p_2 \vee p_4)(p_2 \vee p_5)(p_3 \vee \neg p_4)(p_3 \vee p_5)(p_4 \vee p_5)$ を考えます。 $p_1=0$ とすると $(p_1 \vee p_2)$ より $p_2=1$ 、 $(\neg p_2 \vee p_3)$ より $p_3=1$ が得られます。また、 $(p_1 \vee \neg p_3)$ より $p_3=0$ が得られます。これにより、この系列では $p_3=1$ かつ $p_3=0$ という矛盾が生じます。これにより P_0 が充足可能であるためには $p_1=1$ となります。すると

$$\begin{aligned} P_0 &= (p_1 \vee p_2)(p_1 \vee \neg p_3)(\neg p_1 \vee p_4)(\neg p_2 \vee p_3)(p_2 \vee p_4)(p_2 \vee p_5)(p_3 \vee \neg p_4)(p_3 \vee p_5)(p_4 \vee p_5) \\ &= (\neg p_1 \vee p_4)(\neg p_2 \vee p_3)(p_2 \vee p_4)(p_2 \vee p_5)(p_3 \vee \neg p_4)(p_3 \vee p_5)(p_4 \vee p_5) \end{aligned}$$

となり、さらに $(\neg p_1 \vee p_4)$ より $p_4=1$ 、 $(p_3 \vee \neg p_4)$ より $p_3=1$ を得ます。これにより、 $P_0 = (p_2 \vee p_5)$ となって $p_2=1$ とすれば P_0 が充足可能であることが示せます。また、 $P_1 = (p_1 \vee p_2)(p_1 \vee \neg p_3)(\neg p_1 \vee p_4)(\neg p_2 \vee p_3)(p_2 \vee p_4)(p_2 \vee p_5)(p_3 \vee \neg p_4)(\neg p_3 \vee p_5)(\neg p_4 \vee \neg p_5)$ では先ほどの P_0 と同様に $p_1=0$ とすると $(p_1 \vee p_2)$ より $p_2=1$ 、 $(\neg p_2 \vee p_3)$ より $p_3=1$ となります。また、 $(p_1 \vee \neg p_3)$ より $p_3=0$ が得られ矛盾し、 $p_1=1$ です。すると $p_3=p_4=1$ となります。さらに $(\neg p_3 \vee p_5)$ より $p_5=1$ 、 $(\neg p_4 \vee \neg p_5)$ より $p_5=0$ です。よって $p_1=1$ としても矛盾が得られるので、 P_1 は充足できないことが結論できます。上記のような決定性のアルゴリズムが 3-SAT では機能しないことは節に 3 つのリテラルがあることから明らかです。そこで、3-SAT を解くためのアルゴリズムとして Davis-Putnam のアルゴリズムと DPLL アルゴリズムが提案されました [2, 3]。次で Davis-Putnam のアルゴリズムと DPLL アルゴリズムについてみていくことにしましょう。

2 Davis-Putnam のアルゴリズムと DPLL アルゴリズム

まず、最初に Davis-Putnam のアルゴリズムと DPLL アルゴリズムに共通する手法について説明します。Davis-Putnam のアルゴリズム [2] は 1960 年に DPLL アルゴリズム [3] は 1962 年に発表されました。両方に共通する手法は次の 2 つです。

手法 1. Elimination of One-Literal Clauses

手法 2. Affirmative-Negative Rule

手法 1. Elimination of One-Literal Clauses とは 1 つのリテラルからなる節の値は決めて論理式から省くというものです。例として $P = (p_1 \vee p_2)(p_1 \vee \neg p_3)(\neg p_1 \vee p_4)(\neg p_2 \vee p_3)(p_2 \vee p_4)(p_2 \vee p_5)(\neg p_3 \vee \neg p_4)p_3(p_3 \vee p_5)(p_4 \vee p_5)$ を考えましょう。この P には後ろから 3 番目の節が p_3 のみです。 P が 1 となるためには $p_3 = 1$ でなくてはなりません。なので $p_3 = 1$ とすると $P \equiv (p_1 \vee p_2)p_1(\neg p_1 \vee p_4)(p_2 \vee p_4)(p_2 \vee p_5)\neg p_4(p_4 \vee p_5)$ になります。さらに k の論理式にも p_1 と $\neg p_4$ が 1 つのリテラルからなる節です。なので $p_1 = 1, p_4 = 0$ としなければ $P = 1$ にはなりません。まず $p_1 = 1$ としましょう。 $P \equiv p_4(p_2 \vee p_4)(p_2 \vee p_5)\neg p_4(p_4 \vee p_5)$ となります。ここで p_4 という 1 つのリテラルからなる節と $\neg p_4$ という 1 つのリテラルからなる節が現れました。なのでこの P は充足できないことがわかりました。このように 1 つのリテラルからなる節の値を決めて論理式から省くのが手法 1 です。

手法 2. Affirmative-Negative Rule とは論理式のある変数のリテラルが肯定か否定の一方のみで現れるときは値を決めて論理式から省くというものです。ある変数のリテラルが肯定のみのときはその変数の値を 1 とします。ある変数のリテラルが否定のみのときはその変数の値を 0 とします。 $P = (p_1 \vee p_2)(p_1 \vee \neg p_3)(p_1 \vee p_4)(\neg p_2 \vee \neg p_3)(p_2 \vee \neg p_4)(p_2 \vee \neg p_5)(p_3 \vee \neg p_4)(p_3 \vee p_5)(\neg p_4 \vee p_5)$ で考えてみましょう。この P には手法 1 は適用できません。そこで手法 2 です。 P の中でどの p_1 も $\neg$ がついていません。これが「ある変数のリテラルが肯定のみのとき」に該当します。そこで $p_1 = 1$ とします。すると $P \equiv (\neg p_2 \vee \neg p_3)(p_2 \vee \neg p_4)(p_2 \vee \neg p_5)(p_3 \vee \neg p_4)(p_3 \vee p_5)(\neg p_4 \vee p_5)$ とできます。すると今度、 p_4 のすべてに $\neg$ が付いています。これが「ある変数のリテラルが否定のみのとき」に該当します。そこで $p_4 = 0$ とします。すると $P \equiv (\neg p_2 \vee \neg p_3)(p_2 \vee \neg p_5)(p_3 \vee p_5)$ となります。このようにリテラルが肯定か否定の一方のみであることを利用して論理式から節を省くのが手法 2 です。これらの手法 1,2 は Davis-Putnam のアルゴリズムと DPLL アルゴリズムに共通するものです。

2.1 Davis-Putnam のアルゴリズム

Davis-Putnam のアルゴリズムでは手法 1,2 の他に手法 3 として Rule for Eliminating Atomic Formulas を採用している。これは論理式が $P = (P_1 \vee p)(P_2 \vee \neg p)P_3$ となっているとします。このとき、 p は論理変数、 P_1, P_2, P_3 は p を含まない論理式とします。 $P_1 = P_2 = 0$ のとき P は 0 となります。つまり、 p の値によらず P_1, P_2 の値によって P の値が決まります。 $P_1 = P_2 = 0$ のとき $P = 0$ より $P = (P_1 \vee p)(P_2 \vee \neg p)P_3$ は $P = (P_1 \vee P_2)P_3$ に置き換えられます。 $P = (P_1 \vee P_2)P_3$ に置き換えた後 $P_1 = P_2 = 1$ ならば p の値は 0 でも 1 でもよく。 $P_1 = 0, P_2 = 1$ ならば $p = 1$ とし、 $P_1 = 1, P_2 = 0$ ならば $p = 0$ とします。Davis-Putnam のアルゴリズム：

1. 与えられ論理式に手法 1 を適用する。適用した結果、手法 1 が適用できるならば 1 へ、

適用した結果、手法2が適用できるならば2へ 適用した結果、手法1も2も適用できなければ3へ進む。

2. 与えられ論理式に手法2を適用する。適用した結果、手法1が適用できるならば1へ、適用した結果手法2が適用できるならば2へ 適用した結果、手法1も2も適用できなければ3へ進む。
3. 与えられ論理式に手法3を適用する。 適用できない場合はある変数に着目して、手法3が適用できる論理式に変形する。適用した結果、手法1が適用できるならば1へ、適用した結果手法2が適用できるならば2へ 適用した結果、手法1も2も適用できなければ3へ進む。

上記のアルゴリズムの手順3で「適用できない場合はある変数に着目して、手法3が適用できる論理式に変形する。」とは以下のような変形を行うことです。 $(P_1 \vee p)(P_2 \vee p)(P_3 \vee \neg p)(P_4 \vee \neg p)P_5$ の場合を考える。 P_1, P_2, P_3, P_4, P_5 には p は含まれない。 $(P_1 \vee p)(P_2 \vee p)(P_3 \vee \neg p)(P_4 \vee \neg p)P_5 \equiv ((P_1 \wedge P_2) \vee p)((P_3 \wedge P_4) \vee \neg p)P_5$ と変形してそして手法3を使って $((P_1 \wedge P_2) \vee p)((P_3 \wedge P_4) \vee \neg p)P_5 = ((P_1 \wedge P_2) \vee (P_3 \wedge P_4))P_5$ とします。このアルゴリズムで論理式の節は単調に減少するので最終的には終了します。また、アルゴリズムの手順3である変数を適当に選んでいるのでうまく選べば多項式時間で終了します。

2.2 DPLL アルゴリズム

DPLL アルゴリズムは Davis-Putnam のアルゴリズムの手法3に替えて以下の手法4を利用します。

手法4. Splitting rule $P = (P_1 \vee p)(P_2 \vee \neg p)P_3$ で、 P_1, P_2, P_3 に変数 p は含まれないものとします。このとき $p=0$ として $P' = (P_1)P_3$ 、 $p=1$ として $P'' = (P_2)P_3$ をつくって $P' = 0$ になったら P'' の真偽を調べ $P'' = 0$ であったら $P = 0$ とするものです。DPLL アルゴリズム：

1. 与えられ論理式に手法1を適用する。適用した結果、手法1が適用できるならば1へ、適用した結果、手法2が適用できるならば2へ 適用した結果、手法1も2も適用できなければ3へ進む。
2. 与えられ論理式に手法2を適用する。適用した結果、手法1が適用できるならば1へ、適用した結果手法2が適用できるならば2へ 適用した結果、手法1も2も適用できなければ3へ進む。
3. 与えられ論理式に手法4を適用する。 適用した結果、手法1が適用できるならば1へ、適用した結果手法2が適用できるならば2へ 適用した結果、手法1も2も適用できなければ3へ進む。

上記のアルゴリズムの手順3で手法4は変数 p を適当に選んでいる。これを繰り返し適用することになると $P = 1$ となる割り当てを見つけるのに時間がかかる(バックトラック法を用いる)。ただ、うまく変数が選べて割り当てもうまく選べば多項式時間で終了します。

参考文献

- [1] S. A. Cook, The complexity of theorem-proving procedures, Proc. 3rd Ann. ACM Symp. on Theory of computing, pp. 151–158(1971).
- [2] Martin Davis, Hilary Putnam, A Computing Procedure for Quantification Theory, Journal of the ACM, vol. 7, pp.201–215(1960)
- [3] Martin Davis, George Longemann, Donald Loveland, A Machine Program for Theorem-Proving, Communications of the ACM, vol, 5, pp.394–397(1962).
- [4] M. R. Garey, D. S. Johnson, Computers and intractability, A guide to the theory of NP-completeness, Freeman and Company, 1999.
- [5] R. M. Karp, Reducibility among combinatorial problems, Proceedings of a Symposium on the Complexity of Computer Computations, pp. 85–103(1972).