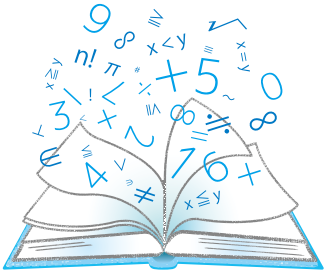

7 SMTソルバーによるプログラム検証

石井大輔（福井大学） 上田和紀（早稲田大学）

SMT：述語論理を扱うSAT技術

高速なSATソルバーは大規模な論理式を解くことを可能にし，多くの応用問題に有用なツールとなったが，万能というわけではなく不得意な問題もある．たとえば，整数・実数の変数を持つ不等式で記述される問題や，配列・リスト・スタックなどのデータ構造について記述した問題を考えよう．こうした問題の求解では無数の場合分けを考える必要があり，そのままでは命題論理式に落とし込むことができない．**SMT** (satisfiability modulo theories) は，こうしたSATが不得手な問題を扱えるようにSATを拡張した技術である．

SATが真偽値をとる命題変数を持つ命題論理を扱うのに対し，SMTでは**述語論理** (predicate logic) を扱う．述語論理式の例として

$$x \geq 1 \vee x > y$$

のような整数あるいは実数の変数 x と y を持った式を考えよう．この式は，命題論理式 $B_1 \vee B_2$ の命題変数 B_1 と B_2 を，不等式 $x \geq 1$ と $x > y$ で詳細化したものと考えられる．また，たとえば $x=1$，$y=1$ のとき充足される（真となる）ので，充足可能である．

このような述語論理式の充足可能性を判定するため，CVC4[☆1]，Z3[☆2]，Alt-Ergo[☆3] といった**SMTソルバー**が開発され，プログラム検証・スケジューリング・プランニング・プログラム合成などへの応用において重要なツールとなっている．

SMT技術は定理証明の自動化技術というもう1つのルーツを持つ．たとえば「整数変数と算術演算子を用いた述語論理式」のように限定した式の評価に特化した**理論ソルバー**群を用い，述語論理に関する推論を行うための技術が，70年代から培われてきた．最新のSMTソルバーではそうした技術をSAT技術と統合し，さまざまな述語を含んだ式を求解することを可能にしている（**図-1**）．

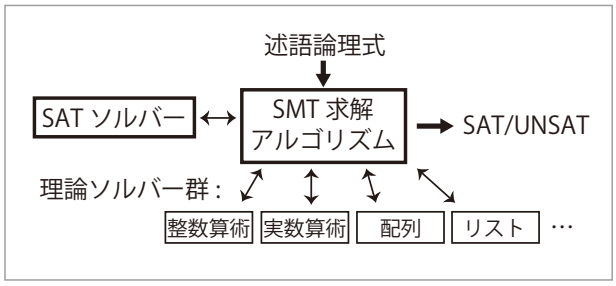

図-1 SMTソルバー

SMTの研究コミュニティでは，問題の標準的な記法とベンチマーク問題集を整備し，SMT-LIB[☆4] として提供している．また，SMT競技会[☆5] が2005年以来毎年開催され，ソルバー開発を牽引している．

SMTを用いたプログラム検証

SMTソルバーの応用事例として，**プログラム検証**を紹介する．プログラムを解析・検証するには多くのアプローチがあるが，たとえばプログラムの記号的実行や有界モデル検査などにSMTソルバーが役立てられている[1]．ここでは，最も伝統的な「プログラム検証」を題材に，SMTがどう役立つのか説明しよう．プログラム検証は，R. W. Floydと

☆1 http://cvc4.cs.nyu.edu/web/
☆2 https://github.com/Z3Prover/z3
☆3 http://alt-ergo.lri.fr/
☆4 http://smtlib.cs.uiowa.edu/index.shtml
☆5 http://www.smtcomp.org/

```
1: //事前条件: x <= 0 /\ y >= 0
2: //事後条件: x <  0 /\ x <= y
3: x = x - 1;
4: y = y - 1;
```

図-2 プログラム P_1

```
1: //事前条件: x>=0 /\ r=0 /\ c=0
2: //事後条件: r = x*y
3: while (c < x) {
4:     r = r + y;
5:     c = c + 1;
6: }
```

図-3 プログラム P_2

C. A. R. Hoare が 60 年代に提案した方法で，契約プログラミング (programming by contract) にも影響を与えた．レビューやソフトウェアテストとは異なり，プログラムが満たすべき仕様を数学的に記述し，その妥当性を示すことによりプログラムの信頼性を担保するものである．SMT ソルバーを利用して検証プロセスを自動化する手法が提案され，いまでも（何度目かの）脚光を浴びている．

★「正しい」プログラム（断片）の例

まず整数型変数 x と y を扱う**図-2** のプログラム P_1 を考えよう．1, 2 行目はコメント，3 行目が「変数 x の値を 1 減らした値を x に代入する」という内容の文，4 行目が変数 y に関する同様の文となっている．コメントでは**事前条件** (precondition) と**事後条件** (postcondition) を与えてプログラムの仕様を記述するものとする．P_1 は，「事前条件 x <= y /\ y >= 0 をみたす状態のもと，3 ～ 4 行目のプログラム本体を実行，終了した後，状態が事後条件 x < 1 /\ x <= y をみたす」という内容の**仕様付きプログラム**である．なお，事前条件中の“/\”は論理積∧を表している．

もう少し複雑な例を 2 つ載せよう．**図-3** の仕様付きプログラム P_2 は，while 文を使った例である．事前条件のもと，3 ～ 6 行目のプログラムを実行後，事後条件「r の値は x と y の積」が成り立つことを表している．

図-4 の P_3 は配列を使った仕様付きプログラムである．len(a) は配列 a のサイズ，old(a) はプログラムが中身を書き換える前の配列 a を表し，プログラムを実行後，「書き換え前の配列 a の第 i 要素の内容が，書き換え後の a の第 j 要素の内容と等しい」ことを表している．

```
1: //事前条件: 0 <= i <= j < len(a)
2: //事後条件: old(a)[i] = a[j]
3: tmp  = a[i];
4: a[i] = a[j];
5: a[j] = tmp;
```

図-4 プログラム P_3

★プログラム検証とは？

プログラム検証では，このような仕様付きプログラムが与えられたとき，その正当性を判定する．つまり，事前条件のもとプログラムを実行したとき，事後条件が必ず成り立つか，あるいは成り立つとは限らないかどうかを判定し，前者が確認できれば正当性が証明されたことになる．この正当性を**検証条件** (verification condition) と呼ばれる述語論理式で記述し，その妥当性（変数の値によらずに論理式が必ず真となるかどうか）を SMT ソルバーで求解することにより，プログラム検証の自動化が可能になる．

P_1 の正当性は，整数算術の述語を含む式

$$x \leq 0 \wedge y \geq 0 \Rightarrow x-1<0 \wedge x-1 \leq y-1$$

として記述できる．この式が妥当であれば P_1 は正当ということになる．式中の含意の前件は事前条件，後件は事後条件にプログラム中の代入結果を反映させたものである[☆6]．

P_2 についても同様なプログラム検証が可能だろうか？　繰り返し構文を使ったプログラムの場合，**ループ不変条件** (loop invariant) と呼ばれる「プログラムの実行が繰り返し文の先頭にきたときに常に成り立つ条件」を見つけることができれば，それを仕様として加え，検証が可能になることが知られている．P_2 の場合，

☆6 仕様付きプログラムから述語論理式への（機械的な）変換方法について興味のある読者は，文献 3）をあたるか，「ホーア論理」，「最弱事前条件」などのキーワードで調べてほしい．

```
// ループ不変条件: c <= x /\ r = c*y
```

を追記し，プログラムの実行開始時および while 文の中身を実行した後にこの条件が成り立つことを要請すればよい．ただし，適切なループ不変条件を見つけるのは簡単でないことが多い．ループ不変条件を付与した $\mathbf{P}_2$ の正当性は，下記の述語論理式として記述できる．

$$(x \geq 0 \wedge r=0 \wedge c=0) \Rightarrow \underline{c \leq x \wedge r=cy} \wedge$$
$$(\underline{\underline{c<x}} \wedge \underline{c \leq x \wedge r=cy} \Rightarrow$$
$$\underline{c+1 \leq x \wedge r+y=(c+1)y} \wedge$$
$$(\neg\underline{\underline{(c<x)}} \wedge \underline{c \leq x \wedge r=cy} \Rightarrow r=xy)$$

式中の一重下線部はループ不変条件から，二重下線部は while 文が与えるループ終了条件から生成された部分式になっている．繰り返し処理をするプログラムの場合，事前条件をみたす各状態から実行される繰り返しの回数が異なってくる．上記の式は，**帰納法**の考え方に基づき，そのような実行例すべての正当性を記述する．式の直感的な内容は，(1)「事前条件のもと，ループ不変条件が成り立つ」，(2)「ループ不変条件が成り立つときにループの中身を 1 回実行した後，ループ不変条件が成り立つ」，(3)「ループを抜けた後，事後条件が成り立つ」となっている．

$\mathbf{P}_3$ の正当性は，配列に関する述語を含む式

$$0 \leq i \leq j < \mathrm{len}(a_1) \Rightarrow$$
$$tmp=a_1[i] \Rightarrow a_2=a_1\langle i \lhd a_1[j]\rangle \Rightarrow$$
$$a_3=a_2\langle j \lhd tmp\rangle \Rightarrow a_1[i]=a_3[j]$$

で記述できる．上式で“$a\langle i \lhd v\rangle$”は「配列 a の第 i 要素に値 v を代入した配列」，“$a[j]$”は「配列 a の第 j 要素」と解釈する．

実際に SMT ソルバーを用いて各述語論理式の妥当性を証明することができる．また，仕様付きプログラムから上記のような検証条件を自動生成し，SMT ソルバーを用いた妥当性判定により検証を行う Why3[☆7]，Frama-C[☆8]，Boogie[☆9] などのツールが開発されている．

```
// 事前条件: 0 <= i1 <= i2 < len(a) ->
//     a[i1] <= a[i2]
// 事後条件: 0 <= r -> a[r] = v
// ループ不変条件: 0 <=s /\ t < len(a) /\
//     (0 <= i < len(a) -> a[i] = v ->
//         s <= t -> s <= i <=t ) /\
//     (0 <= r -> a[r] = v)
s = 0; t = len(a)-1; r = -1;
while (s <= t) {
    m = s + (t - s)/2;
    if (a[m] < v)
        s = m + 1;
    else if (a[m] > v)
        t = m - 1;
    else
        { r = m; s = t + 1; }
}
```

図 -5　二分探索の仕様付きプログラム

★「検証済みソフトウェア」の実現へ

以上説明した技法を使って検証可能な，より実用的な例として，二分探索の仕様付きプログラムを**図 -5** に載せる．このように，仕様付きプログラムの形式で実装を行い，SMT ソルバーを使って正当性を検証し，適宜修正を施しながら (correct by construction)，高信頼なソフトウェアを実現する環境が整いつつある．

SMT ソルバーの仕組み

前章の仕様付きプログラム $\mathbf{P}_1$ の検証条件を例にとってその妥当性を判定する（簡易版の）求解処理を追ってみよう．求解処理では，検証条件の否定である述語論理式

$$\mathbf{F}: \quad x \leq 0 \wedge y \geq 0 \wedge (x \geq 1 \vee x > y)$$

を考え，この式が充足不能であることを確かめることで妥当性を証明する．求解処理は 2 ステップからなる．ステップ 1 では SAT ソルバーを利用して述語論理式中の論理演算子を評価する．ステップ 2 では理論ソルバーを利用して，述語論理式中の算術式や配列に関する制約式などを評価する．両ステップの結果を組み合わせることで元の式の充足可能性が判定できる．

☆7 http://why3.lri.fr/
☆8 http://frama-c.com/
☆9 https://github.com/boogie-org/boogie

ステップ 1　F 中の各不等式を命題変数 B_1，B_2，B_3，B_4 で置き換えた命題論理式

$$e(\mathbf{F}):\quad B_1 \wedge B_2 \wedge (B_3 \vee B_4)$$

を考え，その充足可能性を SAT ソルバーで求解する．もしも結果が充足不能だと，$B_1 \sim B_4$ の元となった不等式の真偽にかかわらず，F も充足不能ということになる．その場合，充足不能を出力して処理を終了する．この例では $e(\mathbf{F})$ を成立させるような命題変数への付値が複数存在するが，そのような付値の 1 つ

$$\alpha = \{B_1 \mapsto 真,\ B_2 \mapsto 真,\ B_3 \mapsto 偽,\ B_4 \mapsto 真\}$$

が結果として得られる．

ステップ 2　ステップ 1 の求解結果に基づき，命題変数 $B_1 \sim B_4$ の元となった不等式から式

$$x \leq 0 \wedge y \geq 0 \wedge \neg(x \geq 1) \wedge x > y$$

を作成する．式は，命題変数の付値が偽の場合は不等式に否定演算子を付け，各不等式の論理積をとったものである．上式の充足可能性を理論ソルバーで判定する．本稿では理論ソルバーの処理については省略するが，整数の線形算術理論に依拠する式であれば，Simplex 法や Fourier-Motzkin 法などの方法を用いることができる．理論ソルバーにより充足解が求まった場合，元の式 F が充足可能ということになるので処理を終了する．この例では，**図 -6** のように各不等式が表す領域を考えると，4 領域の共通部分がないため，充足不能となる．その場合はステップ 1 に戻り，別の充足解について同様に処理を続ける．

求解処理は，最終的に 3 通りの論理積について充足不能であることを確認し，F が充足不能であることを出力する．　**（求解処理おわり）**

冒頭で挙げた SMT ソルバーはいずれも，整数・実数の線形算術理論，配列理論，未解釈関数の理論，ビット・ベクトルの理論など，プログラム検証にとって重要な理論ソルバーを含んでおり，本稿で例示した述語論理式を実際に求解することができる．また SMT ソルバーの実装では，理論伝播 (theory propagation) によりステップ 1 とステップ 2 を密に連携させ，求解処理を効率化したり，Nelson-Open 法により複数の理論が提供する記号が混在する述語論理式を評価したりすることができる．

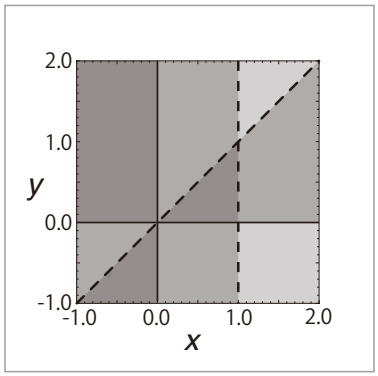

図 -6
述語論理式 F の求解時，ステップ 2 で考える (x, y) 平面の領域（実線は端が領域に含まれること，破線は含まれないことを示す）

参考文献

1) Moura, L. d. and Bjørner, N. : Satisability Modulo Theories : Introduction and Applications, Comm, ACM, 54(9), pp.69-77 (2011).
2) Kroening, D. and Strichman, O. : Decision Procedures, Springer (2008).
3) Bradley, A. R. and Manna, Z. : The Calculus of Computation, Springer (2007).
4) 岩沼宏治，鍋島英知：SMT：個別理論を取り扱う SAT 技術，人工知能学会誌，25(1), pp.86-95 (2010).
5) 梅村晃広：SMT ソルバ・SMT ソルバの技術と応用，コンピュータソフトウェア，27(3), pp.24-35 (2010).

(2016 年 4 月 29 日受付)

石井大輔（正会員）　dsksh@acm.org

2015 年より福井大学大学院工学研究科講師．博士（工学）（早稲田大学）．数値制約プログラミング，ハイブリッドシステムの検証などの研究に従事．

上田和紀（正会員）　ueda@ueda.info.waseda.ac.jp

早稲田大学理工学術院情報理工学科教授，エジプト日本科学技術大学客員教授．プログラミング言語，並行・並列計算，高性能検証などの研究に従事．第 7 回日本 IBM 科学賞，2011 年度日本ソフトウェア科学会基礎研究賞など受賞．2014 年度本会フェロー．