第5回

フォーマル・ベリフィケーションの基本原理（4）

――論理関数の充足可能性判定手法

藤田昌宏

今回は，論理関数の充足可能性判定手法(SAT手法)について解説する．これは，論理関数全体が‘1’となる入力値を見つけるアルゴリズムである．SAT手法は2分決定グラフとはかなり性質の異なる手法であり，互いに相補的に利用されている．

（編集部）

今回は，2分決定グラフ（BDD：binary decision diagram）とともに，非常によく利用される「論理関数の充足可能性（satisfiability）判定手法」について，その基礎と使いかたを重点的に説明したいと思います．論理関数の充足可能性判定手法は，一般にSAT手法と呼ばれています．

ここでは，設計者などのEDAユーザの立場から見たSAT手法について説明します．

●2分決定グラフを補完する論理式充足可能性判定手法

2分決定グラフでは，いったんグラフのサイズが大きくなってしまうと，その後の処理によって加速度的にサイズが大きくなってしまう（グラフのサイズが爆発してしまう）ということがよく起きます．これは，多くの場合，前回（本誌2003年10月号，pp.149-155）説明した変数順の動的最適化などを行っても回避できません．

このため，2分決定グラフを利用するEDAツールを使っていると，グラフが大きくなりすぎて，よくコンピュータがメイン・メモリ不足に陥ります．つまり，利用している計算機がメモリ・スワップを頻繁に起こして，ディスクがうなっている状態になり，計算処理が先に進まなくなります．フォーマル・ベリフィケーション・ツールの実行中にこのような状態になったときは，多くの場合，内部で利用している2分決定グラフのサイズが爆発しています．

これに対して，これから説明するSAT手法では，基本的に場合分けで処理を進めていくので，最初の問題さえメイン・メモリに収まれば，途中でメイン・メモリ不足に陥ることはほとんどありません．その意味では，計算機は静かに検証の計算を続けます．ただし，場合分けの数が膨大となり，処理に非常に時間がかかることはよくあります．

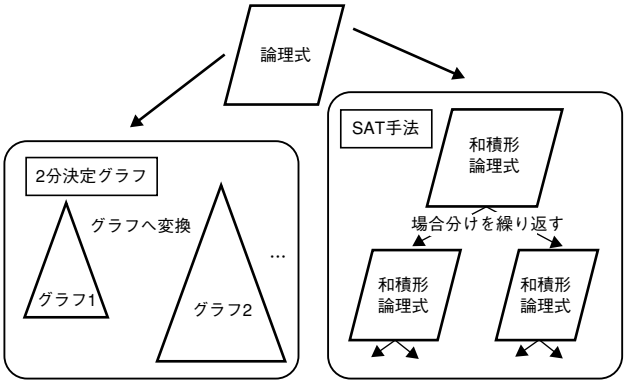
このように，SAT手法は2分決定グラフとはかなり性質の異なる手法であり，互いに相補的に利用されています．

●論理式充足可能性判定手法とは？

論理関数の充足可能性判定問題とは，「与えられた論理関数全体を‘1’とするような変数値の割り当てが存在するか否か」を判定することで，SAT問題とも呼ばれています．一般的には，論理式を任意の形で与えてもかまわないのですが，通常，SAT問題を解くSATアルゴリズム（ここではSAT手法と呼ぶ）は，和積形論理式で問題が与えられたと仮定して作られています．この理由は，半分くらいは研究上の歴史的な経緯であり，あとの半分は数学的な理由です．

2分決定グラフは，与えられた条件（論理式で与えられる）を満たす解をグラフの形ですべて表現しています（つまり，解をすべて保持している）．これに対して，SAT手法は，解を一つ見つけようとします．すべての解を順次見つけていくことも可能ですが，一つずつ見つけていくことになるので，解の数が多い場合には，実用的ではありません．もし，解が存在しないような論理式の場合，2分決定グラフでもSAT手法でも同じ結果になります．この関係を図示したものが**図1**です．

SATアルゴリズムは，与えられた論理式が充足可能な場合，つまり，論理式全体を‘1’とするような変数の値の組み合わせが存在する場合はその値の組み合わせの例を戻り値として返します．また，充足不可能な場合，つまり，ど



〔図1〕論理回路からCNFへの変換

2分決定グラフでは，いったんグラフのサイズが大きくなってしまうと，その後の処理によって加速度的にサイズが大きくなってしまうということがよくある．このため，2分決定グラフを利用するEDAツールを使っていると，グラフが大きくなりすぎて，よくワークステーションやパソコンがメイン・メモリ不足に陥る．これに対してSAT手法では，基本的に場合分けで処理を進めていくので，最初の問題さえメイン・メモリに収まれば，途中でメイン・メモリ不足に陥ることはほとんどない．ただし，場合分けの数が膨大となり，処理に非常に時間がかかることはよくある．このように，SAT手法は2分決定グラフとかなり性質の異なる手法であり，互いに相補的に利用することができる．

のような変数の組み合わせに対しても，与えられた論理式全体を‘1’とはできない場合は，充足不可能であると証明されたことになります．この場合，基本的には，変数の値の組み合わせの全パターンを調べることになります．ただし，さまざまなくふうにより，調べなくてもよいとわかる部分は省くようにして計算を効率化しています．

なお，一般に「検証問題」は，対象とする論理式全体を‘1’とする変数の組み合わせがないことを確認する問題となります．そのような変数の組み合わせがないと，「設計は正しい」という形で定式化される場合が多いのです．

実際には，与えられた論理式が大きすぎる，つまり，非常に多数の変数を含んでいるような場合には論理式全体を‘1’とする変数の組み合わせが見つからず，完全には調べきれないということがよく起こります．その場合，SATアルゴリズムの適用結果は，「調べた範囲では解がなかった」ということになります．これは，一種の論理シミュレーションを行って，できるだけ確認したことに相当します．この場合，論理シミュレーションによって設計検証を行ったときと基本的に同じですが，より網羅的でよりシステマティックにそれぞれの場合を調べている点が異なります．

また，後で述べるように，もともとすべての解を探索しないアルゴリズムもあります．例えば，ある基準でランダム・ウォークを行いながら解を探すというようなアルゴリズムです．この種のアルゴリズムは，解がないことは証明できませんが，解がある場合に高確率で解を発見できるのであれば実用的な価値は高いといえます．実際の検証問題でも有効に利用されています．これは，シミュレーションによって機能検証を行う際に，重み付け乱数によるパターン生成を行うことにかなり近いと言えます．

一般に，二つの設計が等価であることの検証や，ある設計がある性質を満たしているか否かを調べる「モデル・チェッキング（プロパティ・チェッキング）」の問題は，ある論理式を‘1’とするような変数の組み合わせが「存在しない」ことを調べる問題に帰着できる場合がほとんどです．したがって，論理式が充足不可能であることを調べることになります．その意味では，2分決定グラフを利用してもSAT手法を利用しても，同じように処理できます．

ただし，先ほども述べたように，一般的に，2分決定グラフは計算途中でも解のすべてを保持していますから，解があるかないかだけを調べるという観点では，ある意味，むだな処理をしている部分もあります．これに対して，SAT手法は解があればそれを一つだけ見つけるようになっているので，より効率的であるとも言えます．実際，2分決定グラフよりもSAT手法のほうが，検証できる可能性がより高いということも事実です．

●和積形への変換のよしあしによって検証効率が変わる

上でも述べましたが，SAT手法は入力として和積形の論理式を受け付けるようになっています．もちろん，一般の形の論理式を受け付けるものもありますし，そうしてもよいのですが，研究開発上の歴史的な経緯もあり，通常，和積形論理式で問題を与えています．

ここで言う「和積形論理式」とは，変数またはその否定（これらはリテラルと呼ばれる）の和の形をした項の積を取った形です．このため，与えられた検証問題を事前にこの和積形論理式に変換する必要があります．設計検証の立場から見ると，この変換のよしあしによって検証効率が変化するので，これは非常に重要な処理であると言えます．また，本質的に同じ機能の設計を取り扱っている場合でも，SAT問題への変換後の式の大きさは，その設計の書きかたやHDLの記述のしかたに大きく影響を受けます．

●基本はレゾリューション演算

さて，ここでいうSAT問題とは，和積形論理式が与えら