3SATに対する例題生成系とその評価

2T−9

宮野 英次　岩間 一雄
九州大学工学部情報工学科

1. はじめに

我々は充足可能性問題（SAT）の解法アルゴリズムの平均計算時間を実験的に評価する場合の入力例題の合理的な生成系の開発を目標とするプロジェクトを遂行中である．我々の例題生成系は，答え（*yes* または *no*）の入力に対して，*yes* の例題（充足可能なCNF式）と *no* の例題（充足不能なCNF式）を乱数を用いてそれぞれ独立にランダムに生成して出力する[3]．

例題生成系に対する要求として次の3つを考える．(i) 効率： 出力例題の長さに対して多項式時間内ですべての例題を出力可能である．(ii) 複雑論的安全性： 多様な例題が生成可能で問題本来の難しさを維持している．(iii) 統計的安全性： 生成可能な個々の例題の生成確率も考慮に入れたとき，都合の悪い偏りが生じない．

(i), (ii) については，文献[4]で例題生成系とその複雑論的安全性を示した．本稿では (iii) の安全性について議論する．仮に，例題生成系が問題の本来の難しさを維持しているとする．しかし，例えば，充足可能な例題に関しては，充足不能な例題に対して項に含まれるリテラルの数が比較的多いものばかりがほとんどいつも生成されるのであれば問題がある．しかし，個々の例題の生成確率を厳密に調べることは非常に難しく，正確な分布を解析することも事実上不可能であろう．そこで，生成される例題集合がいくつかの特徴的性質を有するように制御できることを目標とする．本稿では，指定されたリテラルの出現頻度の分布パラメータを満足する充足不能な3リテラル式（各項に正確に3個のリテラルを含む）の例題生成系を与える．本例題生成系が広範囲の例題を生成可能であることを解析および実験により示す．

2. SATの例題生成アルゴリズム

例題の生成系として次のような決定性チューリング機械 G を考える: G は入力列 $r \in \{0,1\}^*$ に従いCNF式 $f(r)$ を出力する．G によって生成される言語を $L(G)$ とすると，$L(G) = \{f(r) | r \in \{0,1\}^*\}$ と定義される．出力の長さを n とするとき，もしも G が $p(n)$ ステップで停止するならば，G は $p(n)$ 時間で動作すると言う．入力列 r は $L(G)$ の中の1つを選ぶために使われ，実際には G の内部でランダムに生成されてもかまわない．すべての充足可能な例題集合を I_{sat}，すべての充足不能な例題集合を $I_{\overline{sat}}$ で表す．

Test-Case Generators for 3SAT and Their Evaluation
Eiji MIYANO and Kazuo IWAMA
Dept. of Comp. Sci. and Comm. Eng., Kyushu Univ.

ある変数 x またはその否定 $\overline{x}$ をリテラル，リテラルの論理和を和項と呼ぶ．すべての変数に対する真偽のある割当をセルと呼ぶ．セル C によって表される変数の真偽の割当が和項 A を偽にするとき，A は C を包含すると言う．和項 A によって和項 B が包含するすべてのセルが包含されるとき，A は B を包含すると言う．

充足可能例題の生成系SAT-GENは，ランダムに1つのセル C_{ans} を考え，C_{ans} を包含しないような和項の集合を出力例題の和項とする．充足不能例題の生成系 $\overline{\text{SAT}}$-GENは，まず，1つのランダム変数 x により $f_{now} = x\overline{x}$ とする．次に，充足不能性を満足する次のような3つの操作を f_{now} に施す： (i) 和項 A をランダムな変数 x により $(A+x)(A+\overline{x})$ と分解する．(ii) $(A+x')$ から x' を削除する．(iii) 和項 A が和項 B を包含するとき，B を削除する．

定理1[3]． (i) SAT-GENは I_{sat} を生成する．(ii) $\overline{\text{SAT}}$-GENは $I_{\overline{sat}}$ を生成する．

SAT-GENは多項式時間内で動作するのに対して，$\overline{\text{SAT}}$-GENは上記 (iii) の操作が生成例題の長さを任意に短くするため，多項式時間内では動作しない．しかし，(iii) の実行回数を多項式回以下にすると，多項式時間内の動作が可能になる．$d(n)$ を n の多項式とするとき，(iii) の実行回数を $d(n)$ 回以下にすることで，$\overline{\text{SAT}}$-GEN$(d(n))$ を構成する．

3. 3SATの例題生成アルゴリズム

リテラル分布を $(N(x_1), N(\overline{x_1}), \cdots, N(x_m), N(\overline{x_m}))$ で表す．ここで，$N(v)$ (v は x_i または $\overline{x_i}$) はリテラル v の出現数を表す．項分布を $(T(1), T(2), \cdots, T(t))$ で表す．ここで，$T(i)$ は正確に i リテラルからなる項の数を表す．本稿では，項分布が $(0,0,t,0,\cdots)$，つまり，いわゆる3-SATに対して，リテラル分布 N を満足する例題生成系を考える．充足可能な3リテラル式の例題生成系3SAT-GENの基本的な操作はSAT-GENとまったく同様であり，次の定理が示される[5]．

定理2. 3SAT-GENはリテラル分布 N を満足するすべての充足可能な3リテラル式を多項式時間内に生成する．

充足不能な3リテラル式の生成についても，基本的な操作に関しては $\overline{\text{SAT}}$-GEN$(d(n))$ とまったく同様である．$\overline{\text{SAT}}$-GEN$(d(n))$ が生成可能なリテラル分布 N に従ったすべての3リテラル式を $\overline{L}_{3,d(n)}$ で表す．

Generator $3\overline{\text{SAT}}$-GEN($d(n)$):

入力： $N=(N(x_1),N(\overline{x_1}),\cdots,N(x_m),N(\overline{x_m}))$.

出力： リテラル分布 N に従い，すべての和項が3リテラルである CNF 式.

ステップ1. ランダムな変数 x で $f_{now}=x\overline{x}$ とする.

ステップ2. (2-1) ～ (2-3) を任意に1つ選び実行する.

(*2-1*) f_{now} の中から和項 A とリテラル v をそれぞれ1つランダムに選ぶ．A を $(A+v)(A+\overline{v})$ と分解して f_{now} を変更する．仮に，新しい f_{now} の中のリテラル u の出現数が $N(u)$ を越えた（つまり，$N(u)+1$ 回）とする．次の (i)，(ii) のいずれかを実行する：(i) u を含み，4リテラル以上の和項 B を選び，B から u を削除する．(ii) $(A+v)$ または $(A+\overline{v})$ を包含するような和項 C を選び，$(A+v)$ または $(A+\overline{v})$ を削除する．u のようなリテラルすべてについて同様の操作を繰り返す.

(*2-2*) f_{now} の中から4リテラル以上の和項 A を1つランダムに選ぶ．A からランダムに1つのリテラル v を削除して f_{now} を変更する.

(*2-3*) 和項 A が和項 B を包含するような f_{now} の2つの和項 A および B をランダムに選ぶ．B を f_{now} から削除して f_{now} を変更する.

(*2-4*) f_{now} のすべての和項が少なくとも3リテラルを持つならば，ステップ4へ.

ステップ3. 入力列 r を使い果たすまでステップ2を繰り返す．ただし，(a) f_{now} のすべての和項が少なくとも3リテラルを含むまでは，(2-1) の (i) の和項 B としてどのような和項を選んでも良い（3リテラル以下でも良い）．(b) (2-1)-(ii) と (2-3) の操作の使用回数の総和を $d(n)$ 回以下とする.

ステップ4. リテラルの削除により f_{now} のすべての和項を3リテラルにする．f_{now} の中の出現数が $N(v)$ 回より少ないリテラル v により3リテラルの和項をランダムに作り，f_{now} の1つの和項とする．N を満足するまでステップ4を繰り返す. □

$3\overline{\text{SAT}}$-GEN($d(n)$) は例題の生成過程で，リテラルの出現回数はせいぜい $N+1$ に限られている．それにもかかわらず，$\overline{\text{SAT}}$-GEN($d(n)$) が生成可能な3リテラル式を生成することができることを示すことができる.

定理3. $3\overline{\text{SAT}}$-GEN($d(n)$) は $\overline{L}_{3,d(n)}$ を多項式時間内に生成する.

証明. $\overline{\text{SAT}}$-GEN($d(n)$) の例題生成過程を和項の生成木とみなす．つまり，例題生成過程のある時点での木の葉で表される和項がそのときの f_{now} の1つの和項を表しているとする．直観的には，$3\overline{\text{SAT}}$-GEN($d(n)$) はこの木を深さ優先順で模倣することができる：(i) 仮に，$\overline{\text{SAT}}$-GEN($d(n)$) の操作の中で，和項 A が和項 B により包含され，A が削除されたとする．このとき，A を生成する前に，B を必ず生成するように模倣する．(ii) 仮に，$\overline{\text{SAT}}$-GEN($d(n)$) の中で，リテラル u が和項 A から削除されたとする．このとき，もしも，A の先祖の和項 A' を根とする部分木に u が含まれなければ，A' から u を削除することで模倣する．厳密な証明は別稿に譲る. □

$3\overline{\text{SAT}}$-GEN($d(n)$) は充足不能な例題の非常に制限された部分集合しか生成することができない．仮に，その例題集合がクラス P であれば，問題の本来の難しさを維持しているとは言えない．$\overline{L}_{3,d(n)}(N_c)$ は例題に含まれるすべてのリテラルの出現数が正確に c であるようなリテラル分布に従う3リテラル式の集合を表すとする．$d(n)=0$ のとき，次の定理が示される：

定理4. $\overline{L}_{3,0}(N_c)$ は $c\geq 5$ について NP 完全である.

4. 3SAT 例題生成系の実働化

$3\overline{\text{SAT}}$-GEN($d(n)$) は乱数を用いるため，指定されたリテラル分布に従った例題を生成することができない可能性がある．そこで，$3\overline{\text{SAT}}$-GEN($d(n)$) の実働化を行い，失敗の可能性を調べた（50変数，リテラル出現頻度一様，10回試行）[1]．完全ランダムに3リテラル式を生成した場合，(項の数)/(変数の数) の比がおよそ4.2では，50% の例題が充足可能であり，例えば，その比が3.5では，ほとんど100% の例題が充足可能であることが知られている[2]．直観的には，その比が小さいほど充足不能例題の生成は難しくなる．これに対して，本例題生成系は広範囲で充足不能な例題を生成することができ，その失敗確率も極めて低いことがわかる.

リテラル出現頻度	3	4	5	6
項 / 変数 - 比	2.00	2.67	3.33	4.00
成功確率	100%	100%	100%	100%

参考文献

[1] 朝廣，岩間．3SAT 例題生成アルゴリズムの実働化．電気関係学会九州支部連合大会，1993年.

[2] James M. Crawford and Larry D. Auton. Experimental results on the cross-over point in satisfiability problems. In *Proc. AAAI Spring Symposium Series*, pp.22-28, 1993.

[3] K.Iwama, H.Abeta and E.Miyano. Random generation of satisfiable and unsatisfiable CNF predicates. In *Proc. 12th IFIP World Computer Congress*, pp.322-328, 1992.

[4] K. Iwama and E. Miyano. Security of test-case generation with known answers. In *Proc. AAAI Spring Symposium Series*, pp.85-91, 1993.

[5] 宮野，岩間．失敗確率の低い3SAT 例題生成系について．夏の LA シンポジウム予稿，1993年.