特　集

SMT: 個別理論を取り扱うSAT技術

SMT: Satisfiability Modulo Theories

岩沼 宏治　山梨大学大学院医学工学総合研究部
Koji Iwanuma　Interdisciplinary Graduate School of Medical and Engineering, University of Yamanashi
iwanuma@yamanashi.ac.jp

鍋島 英知　（同　上）
Nabeshima Hidetomo　nabesima@yamanashi.ac.jp

keywords: quantifier-free first-order logic, Nelson-Oppen method, background theory, decision procedure

1. はじめに

近年の SAT 技術の進展に伴い，SAT を用いたソフトウェアやハードウエアの形式的検証や人工知能におけるプラニングなどが盛んに研究されている．一般にこれら検証・推論には，等号や算術，配列やリスト，ビットベクターなどに関する背景知識が必要になり，SAT の枠組みでは，これらの背景知識も命題論理の枠組みで記述される．しかしながら実用問題では，背景知識はかなり複雑で量も多いことから，より表現能力の高い一階論理などの論理体系で記述できれば，記述が大変コンパクトで簡潔になり，都合のよいことが多い．**背景理論付き SAT**(Satisfiability Modulo Theories: **SMT**) [Barrett 09b, Ranise 06] とは，このような命題論理よりも表現能力の高い論理体系で記述さた背景理論を，SAT 技法で効果的に取り扱うことを目的とした技術である．既に多くの技術やソルバーが開発されており，2005 年からは SMT ソルバー競技会 SMT-COMP が毎年開催され，ソルバーの発展を後押している．

SMT 技術は**事前処理型 SMT 技術** (eager SMT techniques) と**遅延処理型 SMT 技術** (lazy SMT techniques) の 2 つに大別される．事前処理型 SMT 技術 [Bryand 01, Lahiri 04, Barrett 09b, Navarro-Pérez 07] とは，背景知識や質問を事前に命題論理式へコンパイル・符号化し，既存の高速 SAT ソルバーで最終的に解かせるものである．変換方式として Small-Domain 符号化，直接符号化 (direct encoding)，またはその混合型 [Barrett 09b] など多種多様な方式が考案されている．常に最新の高性能 SAT ソルバーを利用できるメリットがある反面，多くの背景理論に対して符号化方式を個別に作成する必要がある．また符号化出力した命題式は巨大なものになることも多く，メモリオーバーなどの問題に直面することも多い．これに対して遅延処理型 SMT [Armando 00, Flangan 03, Nieuwenhuis 06, Barrett 09b] は，個別の理論に特化した既存ソルバーと最新 SAT ソルバー技術を効果的に組み合わせる目的で発展してきた技術である．この手法では，充足可能性を判定する論理式のアトム（原子論理式）は，まず命題式として取り扱われ，SAT ソルバーで処理される．もし SAT ソルバーで（命題論理式として）充足不可能と判定すれば，処理はそこで終了する．充足可能と判定された場合，SAT ソルバーから命題論理モデルが出力されるので，背景理論の専用ソルバーが起動し，背景理論との無矛盾性を（通常は一階論理の枠組みで）チェックする．無矛盾と判定されれば，そこで処理が終了する．矛盾していれば，SAT ソルバーが別の命題論理モデルの探索を再開し，再度同じことを繰り返す．柔軟性に富む枠組みであり，全体として高速と言われている．SMT-COMP 競技会に参加するソルバーも，その多くが遅延処理型となっている．

本解説では紙面の都合上，遅延処理型 SMT 技術だけを解説する[*1]．まず初めに，背景理論を取り扱うための DPLL ソルバーの改良について解説する．対象とする背景理論やその専用ソルバーには多種多様なものがあり，DPLL との組合せ方にも無数のバリエーションがある．そのため，DPLL の擬似実行コード[*2]を基にした議論は，大変複雑で不明瞭なものになりがちである．これを解決するために，[Nieuwenhuis 06][*3]では，まず DPLL システムの抽象化を試み，状態遷移システムとして定式化している．[Nieuwenhuis 06] で対象とした DPLL は古典的なものではなく，矛盾からの節学習やバックジャンプ法（非時間順バックトラック法）およびリスタート (restart) 法を持つ最新の高速 DPLL 法である．抽象化は，DPLL の論理的な側面に着目して行われており，停止性，完全性と健全性，および種々の計算戦略や手続きの修正・拡張の正当性などに関する議論の多くが，この枠組みの中

∗1　事前処理型 SMT 技術については [Barrett 09b] に包括的な解説がある．また本特集においても，[田村 10] に制約最適化問題における事前処理型技術の包括的な解説があるので，併せて参照して頂きたい．

∗2　高速 DPLL 法を実現する擬似実行コードやデータ構造，ヒューリスティック計算戦略などについては，本特集論文 [鍋島 10] の 3 章に解説されているので，そちらを参照して頂きたい．

∗3　筆頭著者の Robert Nieuwenhuis は，これ以前には一階の等号推論関係で顕著な業績（例えば [Nieuwenhuis 01] を参照のこと）を上げている研究者である．

で明瞭に行える．本解説ではこの抽象化された DPLL システムを基礎にして，SMT への拡張を議論する．

次に本解説では，背景理論の組合せ問題に対処するための Nelson-Oppen の理論 [Nelson 84, Nelson 79, Nelson 80, Oppen 80a] を紹介する．実用問題で必要となる大規模で複雑な背景理論は複数の理論が組み合わされて構成されている．そのような複合型理論に対して新たに専用ソルバーを開発するのは大変な苦労が伴う．個々の背景理論に対して開発された既存の専用ソルバーをそのまま組み合わせて利用できれば，大変都合が良い．Nelson-Oppen 法はそのための理論・技術である．30 年近く前に提案されたものであるが，これを本質的に超えるものはまだ存在しない．現在使われている多くの手法が Nelson-Oppen の方法を基にしており，改良研究が継続して行われている．本解説では，まず分かりやすいインクリメンタル型の Nelson-Oppen 型手続きを説明したのち，[Nelson 79] で導入され Convex 理論において非常に効果的に働く推論手続きを紹介する．

また最後に SMT ソルバー競技会 SMT-COMP と参加ソルバーについて簡単な紹介を行う．既に多くの SMT ソルバーが開発されており，その現状について概観する．

2. 準　　備

本解説では，等号を持つ一階論理の枠内で議論を行う．項，アトム，リテラル，節，論理式，モデル，充足可能性，無矛盾性などを通常のように定義する．引数を持たない述語記号は**命題**と呼ばれる．本解説では等号を持つ一階論理を考えているので，モデルは**等号モデル** (equational model)，即ち等号記号は恒等関係 (identity relation) と解釈するモデルのみを考える．命題論理および一階論理に関する詳細は，文献 (例えば [有川 88, 萩谷 94, 岩沼 09] など）を適宜，参照して頂きたい．

以下では，アトムもしくは命題を $A_1, A_2, \ldots, B_1, B_2, \ldots$ で表記する．またリテラルの列 $l_1\ l_2\ \ldots\ l_n$ をリテラルの連言 $l_1 \wedge l_2 \wedge \ldots \wedge l_n$ と適宜同一視する．リテラルの列（連言）を $\mu, \nu, \ldots$ で表記し，リテラルの選言，即ち節は C または下添数を付記した $C_1, C_2, \ldots$ で表記する．節の列 $C_1, \ldots, C_n$ を節の連言 $C_1 \wedge \ldots \wedge C_n$ と同一視し，また適宜，節集合 $\{C_1, \ldots, C_n\}$ とも同一視する．節集合は F または G で表記する．True で空のリテラルの連言，および空の節集合を表し，論理的には真を表すものとする．

理論 (theory) T とは一階閉論理式 (first-order closed formula) の集合である．論理式 F が T-**充足可能** (T-satisfiable) あるいは T-**無矛盾** (T-consistent) であるとは，$F \wedge T$ が充足可能である場合を言う．$F \wedge T$ を真とするモデルを F の T-**モデル**と呼ぶ．F が T-充足可能でない場合は，T-**充足不能**あるいは T-**矛盾** と言う．G が F の**理論** T **上の論理的帰結**であるとは，$F \wedge \neg G$ が T- 充足不能である場合を言い，$F \models_T G$ と表記する．T-**理論補題** (theory lemma) とは，True $\models_T G$ なる節 G を言う．

基礎アトム (ground atom) とは変数が出現しないアトムである．**基礎リテラル**，**基礎節**，**基礎論理式**なども同様である．

【定義 1】（SMT 問題） 理論 T に対する SMT 問題とは，与えられた基礎論理式 F が T-充足可能か否かを決定する問題である．

SMT 問題では通常，F は基礎節集合に限定される．F 中には T に出現しない定数の出現を許し，**自由定数** (free constant) と呼ぶ．自由定数は，論理的には存在限量子 ($\exists$) で束縛された変数と同じ働きをするものである．

2·1 背景理論の例

次によく用いられる背景理論とその決定手続きについて概説する．

§1 等号理論

未定関数記号を持つ等号理論 T_E (Equality with Uninterpreted Functions: **EUF**) を構成する一階閉論理式の集合は空集合である．即ちモデルに課せられる条件は，等号を恒等関係と解釈する以外には全く無い理論である．T_E は空であるので，関数記号は全て解釈が未定の記号となる．引数が無い関数記号は定数であるので，自由定数である．未定関数記号は対象システムの抽象化のためによく用いられる．例えば，単位節の集合 $\{a*(b+c) = 0,\ b*(a+c) \neq 0,\ a = b\}$ の充足可能性を考える場合，一見すると等号理論に加えて算術理論を導入しなければならないように思えるが，関数記号 $*$ と $+$ を未定関数記号で抽象化すれば，$\{f(a, g(b,c)) = 0,\ f(b, g(a,c)) \neq 0,\ a = b\}$ となり，等号理論だけで充足不能と判断できる．

一般の論理式の T_E-充足可能性は決定不能であるが，基礎式の T_E-充足可能性は決定可能である．効率的な決定手続きは合同関係閉包 (congruence closure) に基づくものが多数研究 [Downey 80, Nieuwenhuis 05] されている．

§2 算術理論と差分論理

算術理論 (arithmetic theory) も基本的で重要な理論である．整数上の算術理論 T_Z は，整数 n それぞれを表す定数記号 c_n と，加算を表す関数記号 $+$，負数を表す単項関数記号 $-$，および不等号を表す記号 $\leq$ をアルファベットとして構成された一階閉論理式のうち，通常の整数集合上の解釈で真となる全ての式を集めた理論である．Presburger 算術としても知られて，論理式の T_Z- 充足可能性は決定可能 [Oppen 78] である．基礎式の T_Z-充足可能性は NP 完全 [Papadimitriou 81] である．乗算記号 $\times$ を付加した理論 $T_Z^{\times}$ は，基礎式の $T_Z^{\times}$-充足可能性も決定不能になる．一方で，実数上の算術理論 T_R は決定可能である．基礎式の T_R-充足可能性は多項式時間決定可能 [Karmakar 84] であるが，多くの場合には，シンプレックス法などの指数時間アルゴリズムの方が早いと言われ，実際によく使用されている．

差分論理 (difference logic) は，アトムが $a - b = t$ も

しくは $a-b \leq t$ の形をした部分論理である．但し a と b は未定関数記号，t は整数を表す定数記号である．高速な決定アルゴリズムが [Nieuwenhuis 05b] で与えられている．乗算を追加すると，基礎式に関しても決定不能になる．実数上の差分論理は決定可能であるが，2 重指数時間がかかる．

§3 配列とリスト

McCarthy の**配列理論**の *read* と *write* 公理 T_A は以下のものである．以下では a は配列，i と j は配列添数，e はデータを表している．

$$\forall a\ \forall i\ \forall e[\mathit{read}(\mathit{write}(a,i,e),i) = e]$$
$$\forall a\ \forall i\ \forall j\ \forall e[i \neq j \rightarrow \mathit{read}(\mathit{write}(a,i,e),j) = \mathit{read}(a,j)]$$
$$\forall a\ \forall b[(\forall i\ (\mathit{read}(a,i) = \mathit{read}(b,i)) \rightarrow a = b]$$

一般の論理式の T_A-充足可能性は決定不能 [Suzuki 80] であるが，基礎式に限定すれば決定可能 [Downey 78, Stump 01] である．

リストの理論 T_L の詳細は省略する（興味のある方は [Manna 03] を参照のこと）が，任意の論理式の T_L-充足可能性は決定可能であるが，非初等的 (non-elementary recursive)[Oppen 80b] である．即ち最大時間計算量は $2^{2^{\cdot^{\cdot^{2^n}}}}$ の形では押さえこめない．一方で基礎式の T_L-無矛盾性は線形時間で決定可能 [Oppen 80b] である．

2·2 DPLL 法の抽象化

本節では DPLL を SMT 問題へ拡張する準備として，まず，手続き型 DPLL 法を状態遷移システムへ抽象化し，**抽象 DPLL システム** (abstract DPLL system) を導入する．システムの**状態**としては，失敗を示す *Fail* と，基礎リテラルの列 μ と基礎節集合（あるいは基礎節の列）F の 2 項組 $\langle \mu, F \rangle$（以下では $\mu \parallel F$ と表記する）を考える．μ は**真偽値の部分割り当て**を表しており，以下では μ を適宜そのように呼ぶ．また μ 中の幾つかのリテラルは DPLL 法の**決定リテラル** *4を表し，l^d のように上付き文字 d を付記して，通常のリテラルと区別する．また節，即ちリテラルの選言 C に新たにリテラル l を追加した節を $C \vee l$ と表記する．

基礎リテラルの列 μ と基礎節集合 F 中に出現する基礎アトムを命題とみなして，命題論理の意味で F が μ の論理的帰結になることを $\mu \models_p F$ と表記する．μ に於いてリテラル l が**未定義である**とは，μ 中に l と $\neg l$ どちらも出現してないことを言う．

【定義 2】（抽象 DPLL システム [Nieuwenhuis 06]) 抽象 DPLL システムとは以下の 7 つの遷移規則からなる状態遷移システムである．

UnitPropagate: $\mu \parallel F,\ C \vee l \implies \mu\, l \parallel F,\ C \vee l$
但し (i) $\mu \models_p \neg C$ かつ (ii) l は μ 中で未定義である．

Decide: $\mu \parallel F \implies \mu\, l^d \parallel F$
但し (i) l または $\neg l$ が F に出現し，かつ (ii) l は μ 中で未定義である．遷移後に導入される l^d を**決定リテラル** (decision literal) と呼ぶ．

Fail: $\mu \parallel F, C \implies \mathit{Fail}$
但し (i) $\mu \models_p \neg C$ かつ (ii) μ は決定リテラルを含まない．

Backjump: $\mu\, l^d\, \nu \parallel F, C \implies \mu\, l' \parallel F, C$
但し (i) $\mu\, l^d\, \nu \models_p \neg C$ かつ (ii) 次の 4 条件を満たす節 $C' \vee l'$ が存在する．
(1) $F, C \models_p C' \vee l'$
(2) $\mu \models_p \neg C'$
(3) l' は μ 中で未定義
(4) l' または $\neg l'$ が，F または $\mu\, l^d\, \nu$ に出現する
上記の C を**矛盾節** (conflicting clause)，$C' \vee l'$ を**バックジャンプ節** (backjump clause) と呼ぶ．

Learn: $\mu \parallel F \implies \mu \parallel F,\ C$
但し (i) C の各アトムは F または μ に出現しており，かつ (ii) $F \models_p C$ である．上記の C を**学習節** (learned clause) と呼ぶ．

Forget: $\mu \parallel F, C \implies \mu \parallel F$
但し (i) $F \models_p C$ である．

Restart: $\mu \parallel F \implies \mathsf{True} \parallel F$

充足可能性を判定する基礎節集合 F が与えられたとき，抽象 DPLL システムでは，初期状態を $\mathsf{True} \parallel F$ とし，遷移規則に従い充足可能性判定計算を進めていく．状態遷移列を $\mathsf{True} \parallel F \implies S_1 \implies S_2 \implies \ldots$ のように表記する．この遷移列は非決定性を持ち，特に遷移規則 Decide の決定リテラルと Backjump に於ける矛盾節およびバックジャンプ節の選択は，システムの性能に極めて大きな影響を及ぼす．幾つかの効果的なヒューリスティック戦略が考案されており，実際の高速 DPLL システムに実装されているが，本解説の範囲外なので省略する．詳細については [鍋島 10] などを参照して頂きたい．

以下ではまず，状態遷移列の有限性（システムの停止性）を示す．状態遷移列 $\mathcal{S}$ が Restart に関して**周期増加性** (increasing periodicity) を持つとは，$\mathcal{S}$ 中における Restart 規則の適用の間隔が順次（真に）大きくなっている場合をいう．

【定理 1】（停止性 [Nieuwenhuis 06]） 以下の 2 つの条件を満たす状態遷移列 $\mathcal{S}$ は必ず有限列である．
(1) $\mathcal{S}$ 中には，Learn と Forget の遷移だけなる無限部分遷移列が存在しない．
(2) $\mathcal{S}$ は Restart に関して周期増加性を持つ．

状態 S が**飽和状態** (saturated state)*5であるとは，S

*4 [鍋島 10] では決定変数と呼ばれている．本解説では DPLL を SMT 問題に拡張して基礎リテラル（一階式）を扱う必要があるために，決定リテラルという用語を用いている．

*5 本解説での飽和 (saturated) という概念は，文献 [Nieuwenhuis

が *Fail* であるか，上記の UnitPropagate, Decide, Fail, Backjump のいずれの遷移規則も適用できない場合[*6]をいう．

［定理 2］（完全性と健全性 [Nieuwenhuis 06]） 任意の状態遷移列 $\mathsf{True} \parallel F \Longrightarrow S_1 \Longrightarrow \ldots \Longrightarrow S_n$ において，S_n が飽和状態であるならば，以下が成り立つ．

(1) S_n が *Fail* であることと，F が充足不能であることは同値である．

(2) S_n が $\mu \parallel F'$ なる形の飽和状態ならば，μ は F の (命題論理の意味での) モデルである．

なお，良く知られているように，Learn, Forget, Restart の 3 つの規則は完全性を保証するためには必要無く，通常はシステムの効率化のために導入されている．

以上の議論では計算戦略，即ち遷移規則の適用順序については殆んど制約を課していない点に注意して頂きたい．通常使用される計算戦略とはかなり異なるものも許容されており，Learn では通常の矛盾解析に基づく学習以外の節学習も許されている．定理 1 と 2 は，様々な計算戦略の停止性，健全性と完全性をかなり一般的に証明することに成功している．

3. 遅延評価型 SMT 技術：DPLL システムの拡張

以下では [Nieuwenhuis 06] に従い，抽象 DPLL システムを SMT 問題へ拡張し，**抽象 DPLL-MT システム** (abstract DPLL Modulo Theories system) を導入する．対象とする理論 T は，基礎リテラルの連言の T-無矛盾性が決定可能であるものに限定し，その決定手続きを **T-ソルバー** (T-solver) と呼ぶ．

この抽象 DPLL システムを基礎とした SMT ソルバーの枠組み（アーキテクチャ）として DPLL(X) がある．X はパラメータであり，対象とした背景理論 T の専用のソルバーが指定される．制約論理プログラミングの枠組み CLP(X) [Jaffer 94] と同様な仕組みである．この DPLL(X) に基づく SMT ソルバーとして Barcelogic があり，第 1 回 SMT ソルバー競技会 SMT-COMP'05 [Barret 09a] において，参加 4 部門（全 7 部門）全てにおいて優勝しており，その後の SMT ソルバーに大きな影響を与えたことを最初に紹介しておく．

【定義 3］（抽象 DPLL-MT システム [Nieuwenhuis 06]） 理論 T が与えられたとき，抽象 DPLL-MT システムとは, UnitPropagate, Decide, Fail, Restart の各規則，および以下の 4 つの遷移規則からなる状態遷移システムである．

T-Propagate: $\quad \mu \parallel F \implies \mu\, l^d \parallel F$
　但し (i) $\mu \models_T l$ かつ (ii) l または $\neg l$ が F に出現し，かつ (iii) l が μ 中で未定義である．

T-Learn: $\quad \mu \parallel F \implies \mu \parallel F, C$
　但し (i) C の各アトムは F または μ に出現し，かつ (ii) $F \models_T C$ である．

T-Forget: $\quad \mu \parallel F, C \implies \mu \parallel F$
　但し (i) $F \models_T C$ である．

T-Backjump: $\quad \mu\, l^d\, \nu \parallel F, C \implies \mu\, l' \parallel F, C$
　但し (i) $\mu\, l^d\, \nu \models_p \neg C$ かつ (ii) 次の 4 条件を満たす節 $C' \vee l'$ が存在する．
　(1) $F, C \models_T C' \vee l'$
　(2) $\mu \models_p \neg C'$
　(3) l' は μ 中で未定義
　(4) l' または $\neg l'$ が，F または $\mu\, l^d\, \nu$ に出現する．

基礎リテラルの連言 μ が T-無矛盾で，かつ基礎節集合 F の命題モデル（即ち，$\mu \models_p F$）になっている場合，μ は F の T-モデルになることに注意する．

抽象 DPLL-MT システムにおける決定リテラル，矛盾節，バックジャンプ節，学習節は定義 2 と同様に定める．上記の T-Propagate は UnitPropagate の拡張であり，非常に強力な推論規則である．計算の文脈に依存した補題の学習を行っており，一階定理証明法の結合型タブロー法 (connection tableaux) における folding-down 操作 [Letz 94] と極めて類似した操作である．T-Propagate は学習節抽出の際に必要な矛盾解析にも影響を及ぼすので，注意が必要である．以下に簡単な実行例を示す．

〔例 1〕 ([Barrett 09b] の例 26.4.1, 26.4.2 および 26.4.4 を一部改変) 整数算術理論 T_Z に対して，F として以下の基礎節の集合を考える．

$$
\begin{aligned}
C_1 &: \ \{\neg(2a_2 - a_3 > 2) \vee A_1)\} \\
C_2 &: \ \{\neg A_2 \vee (a_1 - a_5 \leq 0)\} \\
C_3 &: \ \{(3a_1 - 2a_2 \leq 3) \vee A_2\} \\
C_4 &: \ \{\neg(2a_3 + a_4 \geq 5) \vee \neg(3a_1 - a_3 \leq 6) \vee \neg A_1\} \\
C_5 &: \ \{A_1 \vee (3a_1 - 2a_2 \leq 3)\} \\
C_6 &: \ \{(a_2 - a_4 \leq 6) \vee (a_5 = 5 - 3a_4) \vee \neg A_1\} \\
C_7 &: \ \{A_1 \vee (a_3 = 3a_5 + 4) \vee A_2\}
\end{aligned}
$$

上記の F には 5 個の自由定数 $a_1, \ldots, a_5$ が出現し，8 個の基礎アトムと 2 個の命題変数 A_1, A_2 が混在している．DPLL-MT はまず基礎アトムを命題変数として取り扱って真偽値計算を進め，必要に応じて T_Z-ソルバーを呼び出して 理論 T 上の一階論理式としての真偽値計算を行う．F^p を，F 中の 8 個の基礎アトムを擬似的に命題変数 $B_1, \ldots, B_8$ として表記した節集合とすると，F^p は以下のようになる．

$$
\begin{aligned}
C_1^p &: \ \{\neg B_1 \vee A_1)\} \\
C_2^p &: \ \{\neg A_2 \vee B_2\} \\
C_3^p &: \ \{B_3 \vee A_2\}
\end{aligned}
$$

06] では *final* と呼ばれ，文献 [Barrett 09b] では *exhausted* と呼ばれている．いずれの用語にも日本語化その他に若干問題が感じられるので，本解説では一階定理証明で用いられる" 飽和" を使用している．

*6 Learn と Forget 規則は交互に何回でも適用が可能であるので，その適用可能性を飽和性の定義に含めることは適切ではないことは明らかである．

C_4^p : $\{\neg B_4 \vee \neg B_5 \vee \neg A_1\}$
C_5^p : $\{A_1 \vee B_3\}$
C_6^p : $\{B_6 \vee B_7 \vee \neg A_1\}$
C_7^p : $\{A_1 \vee B_8 \vee A_2\}$

このとき，次のような導出（状態遷移）が可能である．以下では $\Rightarrow$ の推移閉包を $\Rightarrow^+$ を表わす．

$$
\begin{aligned}
& \mathsf{True} \parallel F \\
\Longrightarrow^+ \quad & \neg B_5^d\ B_8^d\ B_6^d\ \neg B_1^d \parallel F \\
& \qquad (4 \text{ 回の } \mathsf{Decide}) \\
\Longrightarrow \quad & \neg B_5^d\ B_8^d\ B_6^d\ \neg B_1^d\ \neg B_3 \parallel F \\
& \qquad (T\text{-Propagate}) \\
\Longrightarrow^+ \quad & \neg B_5^d\ B_8^d\ B_6^d\ \neg B_1^d\ \neg B_3\ A_1\ A_2\ B_2 \parallel F \\
& \qquad (3 \text{ 回の } \mathsf{UnitPropagate}) \\
\Longrightarrow \quad & \neg B_5^d\ B_8^d\ B_6^d\ \neg B_1^d\ \neg B_3\ A_1\ A_2\ B_2 \parallel F, C_8 \\
& \qquad (T\text{-Learn}) \\
\Longrightarrow \quad & \neg B_5^d\ B_8^d\ \neg B_2 \parallel F, C_8 \\
& \qquad (T\text{-Backjump}) \\
\Longrightarrow^+ \quad & \neg B_5^d\ B_8^d\ \neg B_2\ \neg A_2\ B_3 \parallel F, C_8 \\
& \qquad (2 \text{ 回の } \mathsf{UnitPropagate})
\end{aligned}
$$

上記の状態遷移列の詳細は以下の通りである．まず初期状態 $\mathsf{True} \parallel F$ から 4 回の Decide 規則で導出された状態

$$\neg B_5^d\ B_8^d\ B_6^d\ \neg B_1^d \parallel F$$

では UnitPropagate は全く適用できない．ここで $\neg B_3 = \neg(3a_1 - 2a_2 \leq 3)$ が，$\neg B_5 = \neg(3a_1 - a_3 \leq 6)$ と $\neg B_1 = \neg(2a_2 - a_3 > 2)$ の T_Z-論理的帰結となっている，即ち

$$\neg B_5, \neg B_1 \models_{T_Z} \neg B_3$$

であることを利用する．$\neg B_3$ は割り当て $\neg B_5^d\ B_8^d\ B_6^d\ \neg B_1^d$ の中で未定義であるので，$\neg B_3$ に対する T_Z-Propagate 規則が適用できる．続いて UnitPropagate を可能な限り適用すると，A_1, A_2, B_2 が節 C_5, C_3, C_2 から順次生成・付加されて，状態は

$$\neg B_5^d\ B_8^d\ B_6^d\ \neg B_1^d\ \neg B_3\ A_1\ A_2\ B_2 \parallel F$$

となる．ここで T_Z-Learn の適用を考える．まず，真偽値割り当て中の 3 つの基礎アトム $\neg B_5 = \neg(3a_1 - a_3 \leq 6)$, $B_8 = (a_3 = 3a_5 + 4)$, $B_2 = (a_1 - a_5 \leq 0)$ の連言は，実は T_Z-矛盾していることに注意する．よって節 C_8 を

$$C_8 : \quad B_5 \vee \neg B_8 \vee \neg B_2$$

とすれば，$\mathsf{True} \models_{T_Z} C_8$ となるので，明らかに $F \models_{T_Z} C_8$ であり，T_Z-学習節となる．また明らかに

$$\neg B_5^d\ B_8^d\ B_6^d\ \neg B_1^d\ \neg B_3\ A_1\ A_2\ B_2 \models \neg C_8$$

なので，C_8 は現在の真偽値割り当てに関して矛盾節となっており，同時にバックジャンプ節にもなっている．即ち以下が成り立っている．

(1) $F, C_8 \models_{T_Z} B_5 \vee \neg B_8 \vee \neg B_2$
(2) $\neg B_5^d\ B_8^d \models \neg(B_5 \vee \neg B_8)$
(3) B_2 は $\neg B_5^d\ B_8^d$ で未定義
(4) B_2 が F に出現

よって C_8 を利用した T_Z-Backjump 規則により，状態

$$\neg B_5^d\ B_8^d\ \neg B_2 \parallel F, C_8$$

にまでバックジャンプしている．

上の例の C_8 以外にも学習節は多数有る．例えば

$$C_9 : \quad B_5 \vee \neg B_8 \vee B_1$$

も $F \models_{T_Z} C_9$ を満たすので，学習節に成り得る[*7]．この C_9 は C_8 とは違って，T_Z-理論補題ではない (即ち $\mathsf{True} \not\models_{T_Z} C_9$) 点[*8]に注意して頂きたい．$C_9$ も矛盾節であるので，抽象 DPLL-MT システムの遷移列としては以下のようなものも可能である．

$$
\begin{aligned}
& \mathsf{True} \parallel F \\
\Longrightarrow^+ \quad & \neg B_5^d\ B_8^d\ B_6^d\ \neg B_1^d\ \neg B_3\ A_1\ A_2\ B_2 \parallel F \\
& \qquad (3 \text{ 回の } \mathsf{UnitPropagate}) \\
\Longrightarrow \quad & \neg B_5^d\ B_8^d\ B_6^d\ \neg B_1^d\ \neg B_3\ A_1\ A_2\ B_2 \parallel F, C_9 \\
& \qquad (T\text{-Learn}) \\
\Longrightarrow \quad & \neg B_5^d\ B_8^d\ B_1 \parallel F, C_9 \\
& \qquad (T\text{-Backjump}) \\
\Longrightarrow \quad & \neg B_5^d\ B_8^d\ B_1\ A_1 \parallel F, C_9 \\
& \qquad (\mathsf{UnitPropagate})
\end{aligned}
$$

以上が抽象 DPLL-MT システムの動作例であるが，停止性その他について以下の性質が成り立つ．

［定理 3］（停止性 [Nieuwenhuis 06]） 抽象 DPLL-MT システムの任意の状態遷移列 $\mathcal{S}$ は，下記の 2 つの条件を満たすならば，必ず有限列となる．

(1) $\mathcal{S}$ 中には，Learn と Forget の遷移だけなる無限部分遷移列が存在しない．
(2) $\mathcal{S}$ は Restart に関して周期増加性を持つか，もしくは $\mathcal{S}$ で学習された節に対しては決して Forget は適用されていない．

抽象 DPLL-MT システムの状態 S が**飽和状態**にあるとは，S が *Fail* であるか，上記の UnitPropagate, Decide, Fail, T-Propagate, T-Backjump のいずれの規則も適用できない場合をいう．このとき以下が成り立つ．

［定理 4］（完全性と健全性 [Nieuwenhuis 06]） 抽象 DPLL-MT システムの任意の状態遷移列 $\mathsf{True} \parallel F \Longrightarrow S_1 \Longrightarrow \ldots \Longrightarrow S_n$ において，S_n が飽和状態であるならば，以下が成り立つ．

(1) S_n が *Fail* であることと，F が充足不能であることは同値である．
(2) S_n が $\mu \parallel F'$ なる形の飽和状態であり，μ が T-無矛盾ならば，μ は F のモデルである．

上記の定理の (2) のモデル生成では，最終的に，真偽値割り当て μ の T-無矛盾性を保証する必要がある．T-

*7 C_9 のような節は色々な戦略で導きだすことができるが，どの場合でも高速 DPLL 法で用いられる矛盾解析 [鍋島 10] を拡張したアルゴリズムが重要な役割を果たす．T-論理的帰結関係を考慮するように拡張が必要になるが，詳細については [Nieuwenhuis 06] を参照していただきたい．

*8 $F \models_{T_Z} C_9$ であることは，前述の T_Z-Propagate で示した $\neg B_5, \neg B_1 \models_{T_Z} \neg B_3$ を利用すれば示せる．

ソルバーを利用して T- 無矛盾性のチェックを行うが，この方式・戦略に関しては抽象 DPLL-MT システムの中では何も言及しておらず，様々な方式が考えられる．

μ が拡張されるたびに T-無矛盾性チェックを行えば，遷移列探索に対する高い枝刈効果が期待できるが，チェックのオーバーヘッドが問題となる．一般に，T-無矛盾性のチェックは命題論理の場合より計算量的に難しい．チェックを最も少なくする方式は，飽和状態に到達した場合にのみ行う方式である．この改良方式としては，UnitPropagate と Fail が適用できない場合に適用される Decide の直前にチェックを行う方式や，周期的にチェックを行って回数を減らす方式も考えられる．

また飽和状態でないならば，無矛盾性のチェックは完全には行わずに簡便に済ませる方式も考えれる．最近では，T-ソルバーそのものに現在の μ に対する無矛盾性計算の履歴を保存させておき，新しい決定リテラルが追加されたときの計算の高速性（漸増的性能）を要求することが普通となっている．このような仕組みを持つソルバーは**インクリメンタル T-ソルバー**と呼ばれる．

μ 中に T-矛盾を検出した後の動作にも，複数の選択の余地がある．古典的な SAT ソルバーを利用する場合には，矛盾節を抽出し学習したあとに Restart を適用するしかない．矛盾を検出したとき（その矛盾節を学習した後に），矛盾を解消するポイントまで探索履歴を即座に戻す機能，所謂バックジャンプ機能を持つ最新の SAT ソルバー（**オンライン型 SAT ソルバー**と呼ばれる) を用いれば，Restart して今までの計算履歴を全て捨てる必要が無くなり，高い性能を出すことができる．インクリメンタル T ソルバーとオンライン型 SAT ソルバーの組合せが現在の主流の方式となっている．

4. 背景理論の結合法：Nelson-Oppen 法

本章では背景理論の組合せ問題を取り扱う．複数の専用ソルバーの組合せに関する Nelson-Oppen の理論 [Nelson 84, Nelson 79, Nelson 80, Oppen 80a] について紹介する．議論を簡単にするために，ここでは 2 個の背景理論の組合せに限定し，主に [Manna 03, Tinelli 04] に従って解説を行う

本章では，背景理論で使用される記号を区別するために，**シグニチャ**(signature) を導入する．シグニチャ Σ とは関数記号と述語記号の集合である．Σ-**項**とは，Σ 中の関数記号と変数記号から構成される項を言う．Σ-**アトム**，Σ-**リテラル**，Σ-**節**，Σ-**理論** なども全く同様に定める．

互いに素 (disjoint) な 2 つのシグニチャ Σ_1, Σ_2 に対して，それぞれ Σ_1-理論 T_1 と Σ_2-理論 T_2 を考え，T_1 と T_2 はそれぞれ専用ソルバー T_1-ソルバー *Solver*$_1$ と T_2-ソルバー *Solver*$_2$ を持つと仮定する．このとき合併シグニチャ$(\Sigma_1 \cup \Sigma_2)$ と合併理論 $T_1 \wedge T_2$ を考え，$(\Sigma_1 \cup \Sigma_2)$-基礎式 F が任意に与えられたときに，F が $(T_1 \wedge T_2)$-充足可能か否かを決定したい．このとき，Nelson-Oppen の手法は，新たに $(T_1 \wedge T_2)$-ソルバーを作らずに，既存のソルバー *Solver*$_1$ と *Solver*$_2$ だけを使って F の充足可能性の判定を可能にするものである．以下，議論を簡単化するために，対象とする基礎式は,（一般性を失わずに）基礎リテラルの連言 μ に限定する．

Nelson-Oppen の手法は次の 2 つのステップからなる手続きである．

純化ステップ 与えられた μ には，一般に Σ_1 と Σ_2 の記号が混在して出現しているので，μ を書換えて，以下の性質を満たす基礎リテラルの連言 μ_1 と μ_2 に分離する．このステップを**純化** (purification) と呼ぶ．

(1) Γ を自由定数の集合とするとき，各 $i = 1,2$ に対して，μ_i のリテラルは $(\Sigma_i \cup \Gamma)$-リテラルである．μ_1 と μ_2 の間で共有される記号は，自由定数のみである．

(2) $\mu_1 \wedge \mu_2$ の $(T_1 \wedge T_2)$-充足可能と，μ の $(T_1 \wedge T_2)$-充足可能は同値である．

判定ステップ 分離・純化した各 μ_i の T_i-充足可能性を *Solver*$_i$ を用いて別個に判定する．このとき μ_1 と μ_2 の共有自由定数の解釈が同一になるように注意する．最終的にどちらかの μ_i が充足不能であれば，もとの μ も充足不能と判定する．どちらも充足可能であれば，もとの μ も充足可能と判定する．

以下まず純化について説明する．まず，部分式の主関数記号が親と異なるシグニチャに属するならば，それらを抽象化（フラット化とも呼ばれる）して分離する．以下では，項 $f(t_1,\ldots t_n)$ の**最外記号** を f とする．同様にアトム $p(t_1,\ldots t_n)$ の最外記号を p とする．

【定義 4】（抽象化規則） (1) 項 $f(\ldots,t,\ldots)$ において，f が属するシグニチャと t の最外記号の属するシグニチャが異なるならば，$f(\ldots,t,\ldots)$ を $f(\ldots,w,\ldots)$ へ書きかえる．また全く新しい自由定数 w を持つ等号リテラル $w = t$ を μ に追加する．

(2) アトム $p(\ldots,t,\ldots)$ においても，上と全く同様に書き換える．

(3) 正の等号リテラル $s = t$ において，s と t それぞれの最外記号が属するシグニチャが異なるのであれば，$s = t$ を $w = s \wedge w = t$ へ書き換える．但し w は全く新しい自由定数である．

(4) 負の等号リテラル $s \neq t$ において，s と t それぞれの最外記号の属するシグニチャが異なるのであれば，$s = t$ を $w_1 \neq w_2 \wedge w_1 = s \wedge w_2 = t$ へ書き換える．但し w_1 と w_2 は全く新しい自由定数である．

上記の書換えは必ず停止しする．結果として μ は先の条件を満たす 2 つのリテラルの連言 μ_1 と μ_2 に自然に分離できる．μ_1 と μ_2 に共通に出現する自由定数の集合を $shared(\mu_1,\mu_2)$ と表記する．

〔例 2〕（[Manna 03]） 例として T_Z と T_E の合併理論を考える．このとき合併シグニチャ$(\Sigma_Z \cup \Sigma_E)$ 上の基礎

リテラルの連言 μ

$$1 \leq a \ \wedge \ a \leq 2 \ \wedge \ f(a) \neq f(1) \ \wedge \ f(a) \neq f(2)$$

を考える．μ は $(T_Z \cup T_E)$-充足不能である．まず μ を純化すると次の 2 つの基礎リテラルの連言に分割できる．

$$\mu_Z =_{df} (1 \leq a \ \wedge \ a \leq 2 \ \wedge \ w_1 = 1 \ \wedge \ w_2 = 2)$$
$$\mu_E =_{df} (f(a) \neq f(w_1) \ \wedge \ f(a) \neq f(w_2))$$

このとき $shared(\mu_Z, \mu_E) =_{df} \{a, w_1, w_2\}$ である．

次に純化した 2 つのリテラルの連言 μ_1 と μ_2 の “組合せ充足可能性” の判定手続きを考える．[Barrett 09b, Manna 03] ではその入門版として，共有自由定数の解釈の同一性を保証するための “アレンジメント (arrangement)” と呼ばれる正負の等号リテラルの連言を “推測 (guess)” によって生成し，判定計算を毎回行う試行錯誤的な手続きを紹介している．しかしこれは典型的な “生成&検査” 方式であり，あまりに効率が悪い．本解説では，これを少し改善したインクリメンタル判定手続き [Tinelli 04] をまず紹介する．

【定義 5】（インクリメンタル Nelson-Oppen 手続き） 合併理論 $(T_1 \wedge T_2)$ 上のリテラルの連言 μ_1 と μ_2 に対する **$(T_1 \wedge T_2)$-判定木**とは，根節点 $\langle \mu_1, \mu_2, \mathsf{True} \rangle$ に以下の 2 つの推論規則を用いて構成される木である．推論規則中の E は等号リテラルの連言である．全ての葉が $false$ とである判定木を**閉じた木**と呼ぶ．

Contradiction 規則: ある $i \in \{1,2\}$ に対して，$\mu_i \wedge E$ が *Solver*$_i$ により T_i-充足不能と判定されるならば，

$$\frac{\langle \mu_1, \ \mu_2, \ E \rangle}{false}$$

Case Splitting 規則: ある $a, b \in$ *shared*(μ_1, μ_2) に対して $a = b$ と $a \neq b$ が E に出現しないならば，

$$\frac{\langle \mu_1, \ \mu_2, \ E \rangle}{\langle \mu_1, \ \mu_2, \ E \wedge a = b \rangle \qquad \langle \mu_1, \ \mu_2, \ E \wedge a \neq b \rangle}$$

〈命題 1〉（健全性 [Barrett 09b, Manna 03]） 閉じた判定木が存在するのならば，リテラルの連言 $\mu_1 \wedge \mu_2$ は $(T_1 \wedge T_2)$-充足不能である．

上の case-splitting 規則は don’t care 規則である．即ち，条件を満たすある a と b のペアを選択したら，その他のペアは特に考える必要は無い規則である．共有変数は有限であるので上の判定木の構築は必ず停止する．

〔例 3〕（例 2 の続き） インクリメンタル Nelson-Oppen 手続きを例 2 に適用した結果を図 1 に示す．s_3 は $\mu_E \models_{T_E} a \neq w_1$ から導出できる．s_6 は $\mu_E \models_{T_E} a \neq w_2$ から導出できる．s_7 は $\mu_Z \models_{T_Z} (a = w_1 \vee a = w_2)$ から導出できる．

以上の Nelson-Oppen 法は一般には完全ではなく，条件が必要になる．Σ- 理論 T が**安定的無限性** (stably infinite) であるとは，任意の閉論理式 F に対して，F が T-充足可能ならば，必ず，F はある無限領域 T-モデルでも真になる場合を言う．理論 T_E, T_Z, T_R, T_A は全て安定的無限性をもつ [Manna 03]. このとき以下が成り立つ．

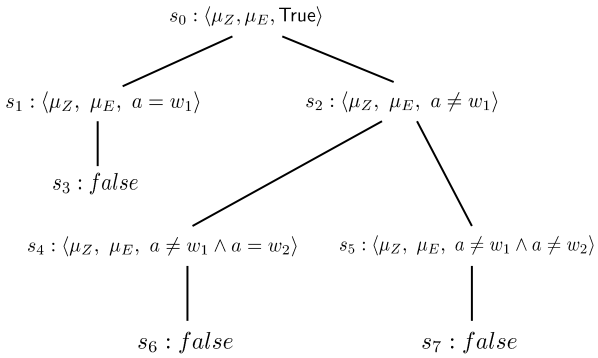

図 1 インクリメンタル Nelson-Oppen 法の例

［定理 5］（完全性 [Barrett 09b, Manna 03]） 理論 T_1 と T_2 はどちらも安定的無限性を持つと仮定する．このとき，リテラルの連言 $\mu_1 \wedge \mu_2$ が $(T_1 \wedge T_2)$-充足不能であるならば，閉じた判定木が存在する．

Nelson-Oppen の判定問題は NP-完全 [Oppen 80a] である．先に示した手続きの最大時間計算量は，理論 T_1 と T_2 のソルバーそれぞれの計算量を $t_1(n)$，$t_2(n)$ とすると，粗く見積もって $O(2^{n^2} \cdot t_1(n) + 2^{n^2} \cdot t_2(n))$ となる（より細かい解析は [Manna 03] を参照のこと）

Nelson-Oppen 法はもともと，等号の連言 E を単に “推測 ”(guess) するのではなく，論理的な推論を併用して導入する形で提案されていた．次にこれを紹介する．

【定義 6】（決定性 Nelson-Oppen 規則 [Nelson 79]）
Contradiction 規則:

$$\frac{\langle \mu_1, \ \mu_2, \ E \rangle}{false}$$

但し, ある $i \in \{1,2\}$ に対して $\mu_i \wedge E$ が T_i-充足不能.

Equality Propagation 規則:

$$\frac{\langle \mu_1, \ \mu_2, \ E \rangle}{\langle \mu_1, \ \mu_2, \ E \wedge a = b \rangle}$$

但し，(i) $a, b \in$ *shared*(μ_1, μ_2)，かつ (ii) $a = b$ は E に出現せず，かつ (iii) ある $i \in \{1,2\}$ に対して，$\mu_i \wedge E \models_{T_i} a = b$

Case Splitting 規則:

$$\frac{\langle \mu_1, \ \mu_2, \ E \rangle}{\langle \mu_1, \ \mu_2, \ E \wedge a_1 = b_1 \rangle \quad \cdots \quad \langle \mu_1, \ \mu_2, \ E \wedge a_n = b_n \rangle}$$

但し，(i) $a_1, \ldots, a_1, b_1, \ldots, b_n \in$ *shared*(μ_1, μ_2)，かつ (ii) $a_1 = b_1, \ldots, a_n = b_n$ は E に出現せず，かつ (iii) ある $i \in \{1,2\}$ に対して，

$$\mu_i \wedge E \models_{T_i} \bigvee_{j=1}^{n} a_j = b_j$$

決定性 Nelson-Oppen 規則を持いた判定木の例を示す.

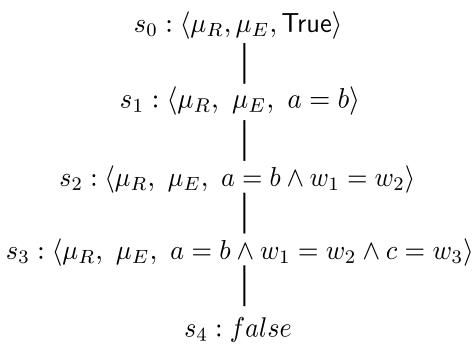

図 2 例 4 に対する決定性 Nelson-Oppen 判別木

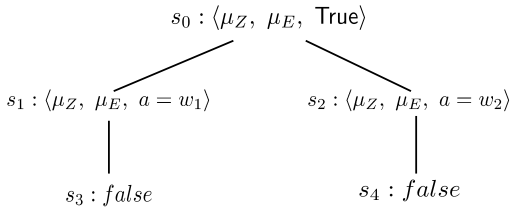

図 3 例 2 に対する決定性 Nelson-Oppen 判別木

〔例 4〕（[Manna 03]） T_R と T_E の合併理論と，合併シグニチャ $(\Sigma_R \cup \Sigma_E)$ 上の基礎リテラルの連言 μ

$$(f(f(a) - f(b)) \neq f(c)) \wedge \\ (a \leq b) \wedge (b + c \leq a) \wedge (0 \leq c)$$

を考える．μ は $(T_R \wedge T_E)$-充足不能である．後半 3 つの単位節から $a = b, c = 0$ が導出できるので，最初の単位節は $f(0) \neq f(0)$ に簡約でき，矛盾している．このとき μ は純化により以下の 2 つのリテラルの連言に分割できる．

$$\mu_R =_{df} (a \leq b) \wedge (b + c \leq a) \wedge (0 \leq c) \wedge (w_3 = w_1 - w_2)$$

$$\mu_E =_{df} (f(w_3) \neq f(c)) \wedge (w_1 = f(a)) \wedge (w_2 = f(b))$$

このとき決定性 Nelson-Oppen システムでは 図 2 に示す推論を行う． s_1 は $\mu_R \models_{T_R} a = b$ より導出される．s_2 は同様に $\mu_E \wedge a = b \models_{T_E} w_1 = w_2$ から，s_3 は $\mu_R \wedge w_1 = w_2 \models_{T_R} c = w_3$ より導出される．このとき $\mu_E \wedge c = w_3$ が T_E- 充足不能であるので s_4 が導出される．

〔例 5〕（例 2 の続き） 例 2 の例に決定性 Nelson-Oppen システムを適用すれば，判定木は図 3 のようになる．全ての葉が false となり，充足不能と判定される．

例 5 では case-splitting 規則が適用されており，逆に適用しないと閉じた木は導出できない．これに対して例 4 では case-splitting は適用されていない．これは有る意味で必然的な現象であり，例 4 は **convex** という性質を満たす理論 T_R と T_E 上の式を対象としているからである．

【定義 7】（Convex theory [Nelson 79]) Σ-理論 T が convex であるとは，任意の Σ-リテラルの連言 μ に対して以下が成り立つ場合をいう．

$a_1, \ldots a_n$ を μ 中に出現する自由定数とするとき，任意の $I' \subseteq \{1, \ldots, n\}$ なる集合 I' に対して以下が同値である

(1) $\mu \models_T \bigvee_{i,j \in I'} a_i = b_j$

(2) ある $k \in I'$ に対して $\mu \models_T a_k = b_k$

明らかに決定性 Nelson-Oppen システムでは，convex な理論 T_1，T_2 に対して case-splitting 規則を適用する必要がなく，非決定的な計算が回避できる．簡単な考察により最大時間計算量は $O(n^4 \cdot t_1(n) + n^4 \cdot t_2(n))$ 以下まで抑えこむことできることが分かる．理論 T_R は convex であり [Nelson 79]，また T_E も convex である [Nelson 80]. 残念ながら T_Z と T_A は convex ではない．例えば T_Z 上の $(a = 1 \wedge b = 2 \wedge 1 \leq c \wedge c \leq 2)$ は，$a = c \vee b = c$ を論理的帰結に持つが，決して $a = c$ や $b = c$ は論理的帰結にならない [Manna 03]．安定的無限性と Convex 理論の間には次の関係が知られている．

［定理 6］（[Barrett 02]） non-trivial モデル（即ち領域が 2 個以上の元からなるモデル）を持つ convex 理論は，安定的無限性を持つ．

Nelson-Oppen の手法は現在も拡張の研究が行われている． 2 つのシグニチャに共通記号が存在する場合，また安定的無限性を持たない場合，あるいは DPLL への組込みを前提とした場合の制限緩和など，多様な研究があり，[Barrett 09b, Manna 03, Tinelli 04, Tinelli 05] に詳しい解説がある．

なお，Nelson-Oppen 法を制限した手法として，Shostak の手法 [Shostak 84] がある．T_E 上の基礎式の真偽判定，即ち合同関係の理論に特化したもので，簡便で高速計算が可能であるとして多くのシステムに実装されてきた．しかし最初に提案された Shostak の手法には幾つかの問題があることが後で判明し，修正と改良 [Ganzinger 02, Shankar 02] が加えられている．結果として最も一般的な方式は，Nelson-Oppen 法とかなり似たものに変容している．

5. SMT ソルバー競技と参加システム

SMT ソルバー開発を促進するために，2005 年より SMT ソルバー競技会 SMT-COMP [Barret 09a] が開催されている．2005 年から 4 回は 国際会議 CAV (Computer Aided Verification) に併設されていたが，2009 年は国際会議 CADE(Conference on Automated Deduction) で開催され，多くのソルバーが参加している．2009 年 SMT-COMP の競技部門とエントリシステム数を表 1 に示した．表 2 には各部門の過去 3 年間の上位 3 位以内のソルバーを記載しており，優勝システムを最上位に示してある．

表 1 を見ると部門ごとのエントリ数がかなり違う．QF_UF, QF_IDL, QF_RDL, QF_BV のような単一理論の部門はエントリが多く，複数の背景理論を組み合わせた部門はエントリ数は少ない．背景理論の組合せには Nelson-Oppen 法

が使用されている．表 2 には 10 ソルバー（延べで 59 ソルバー）が掲っているが，事前処理型ソルバーは Boolector と Beaver の 2 つ（延べ 4 システム）だけであり，残りは全て遅延処理型である．やはり多くの背景理論に対処するためには，遅延処理型のアーキテクチャの方が有利であることが確認される．2009 年の全ての部門にエントリしているのは CVC だけである．CVC は先駆的な SMT ソルバーとして名高い．次に多くの部門にエントリしているのは，Yices と MathSAT であり，性能的にも極めて高いものを示している．Z は 2009 年は参加していないが，それまでに優秀な成績を残している．一階論理の代表的な等号推論体系である superposition [岩沼 09, Nieuwenhuis 01] を用いるなど，特徴あるソルバーである．これらの遅延処理型システムの核となる SAT ソルバは，MathSAT と OpenSAT は MiniSAT をカスタマイズして使用しているが，他は独自に製作した DPLL システムを用いている模様である．各ソルバーの詳細は SMT-COMP のホームページから情報が入手できる．

6. ま と め

SMT 技術は，その応用上の重要性と SAT ソルバの驚異的な発展に支えられて，近年盛んに研究が行われている．論理的にみれば，SMT 問題は一階閉論理式上での基礎論理式の真偽値判定問題であり，SAT 以外にも解集合プログラミング [井上 08] や一階定理証明 [岩沼 09, Nieuwenhuis 01] などによるアプローチも考えられる．また逆に，SMT 技術からの後者 2 つへのアプローチも考えられる．後者 2 つの分野も近年大きく発展してきており，SMT 関連技術の今後の進展が大変興味深い．

Nelson-Oppen の手法も長い時を越えて，その重要性が再認識され研究が続けられている．日本語による Nelson-Oppen 手法に関係した著述は，筆者が Google で検索した範囲においては，一つも発見することはできなかった．本解説が一人でも多くの読者の方々の興味を引き，何らかの SMT 関連の研究の参考あるいは引き金になれば，筆者の望外の喜びとするところである．

◇ 参 考 文 献 ◇

[有川 88] 有川節夫，原口誠：述語論理と論理プログラミング (オーム社,1988)

[Armando 00] A. Armando, C. Castellini and E. Giunchiglia: SAT-based procedures for temporal reasoning. *Proc. of 5th European Conference on Planning, LNCS*, Vol.1809, pp.97–108, 2000.

[Bryand 01] R. Bryand, S. German and M. Velev: Processor Verification Using Efficient Reductions of the Logic of Uninterpreted Functions to Propositional Logic, *ACM Trans. on Computational Logic*, Vol.2, No.1, pp.93–134, 2001.

[Barrett 02] C. W. Barrett, D. L. .Dill and A. Stump: A Generalization of Shostak's Method for Combining Decision Procedures. *Proc. of 4th Inter. WS. on Frontiers of Combining Systems (FroCoS'2002), LNCS*, Vol.2309, pp.132–147, 2002.

[Barret 09a] B. Barrett, M. Deters, A. Oliveras and A. Stump: International Satisfiability Modulo Theories Competition, `http://www.smtcomp.org/`

[Barrett 09b] C. Barrett, R. Sebastiani, S. A. Seshia and C. Tinelli: Satisfiability Modulo Theories. In: *Handbook of Satisfiability*, Chapter 26, pp.825–885 (IOS Press. 2009)

[Downey 78] P. Downey and R. Sethi: Assignment Commands with Array References. *Journal of the ACM*, Vol.25, No.4, pp.652–666, 1978.

[Downey 80] P. Downey, R. Sethi and R. E. Tarjan: Variations on the Common Subexpression Problem. *Journal of the ACM*, Vol.27, No.4, pp.757–771, 1980.

[Flangan 03] J. Flangan, R. Joshi, X. Ou and J. B. Saxe: Theorem Proving Using Lazy Proof Explanation, *Proc. of 15th Inter. Conf. on Computer Aided Verification (CAV), LNCS*, Vol.2725, 2003.

[Ganzinger 02] H. Ganzinger: Shostak Light, *Proc. of 18th Inter. Conf. on Automated Deduction (CADE-18), LNAI*, Vol.2392, pp.332–346, 2002.

[萩谷 94] 萩谷昌己：ソフトウェア科学のための論理学（岩沼書店 1994）

[井上 08] 井上克巳, 坂間千秋：論理プログラミングから解集合プログラミングへ．コンピュータソフトウェア, Vol.25, No.3, pp.20–32, 2008.

[岩沼 09] 岩沼宏治，鍋島英知，井上克巳：一階論理上の等号推論：理論と実際，コンピュータソフトウェア（投稿予定）

[Jaffer 94] J. Jaffar and M. Maher: Constraint Logic Programming: A Survey. *Journal of Logic Programming*, Vol.19/20, pp.503–581, 1994.

[Karmakar 84] N. Karmakar: A New Polynomial-Time Algorithm for Linear Programming, *Combinatorica*, Vol.4, No.4, pp.373–395, 1984.

[Letz 94] R. Letz, C. Goller and K. Mayr: Controlled Integration of the Cut Rule into Connection Tableau Calculi. *Journal of Automated Reasoning*, Vol.13, pp.297-338 (1994).

[Lahiri 04] S. K. Lahiri, S. A. Seshia: The UCLID Decision Procedure, *Proc. of 16th Inter. Conf. on Computer Aided Verification (CAV), LNCS*, Vol.3114, pp.475–478, 2004.

[Manna 03] Z. Manna and C. G. Zarba: Combining Decision Procedures. *Formal Methods at the Crossroads: From Panacea to Foundational Support, LNCS*, Vol.2757, pp.381–422 2003.

[鍋島 10] 鍋島英知，宋剛秀：高速 SAT ソルバーの原理, 人工知能学会誌, Vol.25, No.1, 2010.

[Navarro-Pérez 07] J. A. Navarro-Pérez and A. Voronkov: Encodings of Bounded LTL Model Checking in Effectively Propositional Logic, *Proc. of 21st Inter. Conf. on Automated Deduction (CADE-21), LNAI* Vol.4603, pp.346–361, 2007.

[Nelson 84] G. Nelson: Combining Satisfiability Procedures by Equality Sharing. In: *Automated Theorem Proving: After 25 years*, Vol.29 of *Contemporary Mathematics*, pp.201–211 (American Mathematical Society 1984).

[Nelson 79] G. Nelson and D. C. Oppen: Simplification by Cooperating Decision Procedures. *ACM Transactions on Programming Languages and Systems*, Vol.1, No.2, pp.245-257, 1979.

[Nelson 80] G. Nelson and D. C. Oppen: Fast Decision Procedures Based on Congruence Closure. *Journal of the ACM*, Vol.27, No.2, pp.356–364, 1980.

[Nieuwenhuis 05] R. Nieuwenhuis and A. Oliveras: Proof-Producing Congruence Closure. *Proc. of 16th Inter. Conf. on Term Rewriting and Applications (RTA'05), LNCS*, Vol.3467, pp.453–468, 2005.

[Nieuwenhuis 05b] R. Nieuwenhuis and A. Oliveras: DPLL(T) with Exhaustive Theory Propagation and its Application to Difference Logic. *Proc. of 17th Inter. Conf. on Computer Aided Verification (CAV'05), LNCS*, Vol.3576, pp.321–334, 2005.

[Nieuwenhuis 06] R. Nieuwenhuis, A. Oliveras and C. Tinelli: Solving SAT and SAT Modulo Theories: from an Abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T) *Journal of the ACM*, Vol.53, No.6, pp.937–977, 2006.

[Nieuwenhuis 01] R. Nieuwenhuis and A. Rubio: Paramodulation-Based Theorem Proving, In: *Handbook of Automated Reasoning*, Vol.1, Chapter 7, pp.373–443 (Elsevier Science B.V. 2001).

[Oppen 78] D. C. Oppen: A $2^{2^{2^{pn}}}$ Upper Bound on the Complexity of Presburger Arithmetic. *Journal of Computer and System Sciences*,

表 1 SMT-COMP2009 における部門とエントリシステム数

部門名	エントリ数	部門説明
QF_UF	5	Uninterpreted Functions
QF_IDL	7	Integer Difference Logic
QF_RDL	6	Real Difference Logic
QF_UFIDL	4	Integer Difference Logic with Uninterpreted Functions
QF_LIA	4	Linear Integer Arithmetic
QF_LRA	4	Linear Real Arithmetic
QF_UFLIA	3	Linear Integer Arithmetic with Uninterpreted Functions
QF_UFLRA	3	Linear Real Arithmetic with Uninterpreted Functions
QF_AX	3	Arrays with Extensionality
QF_AUFLIA	3	Linear Integer Arithmetic with Uninterpreted Functions and Arrays
QF_BV	8	Fixed-size Bit-vectors
QF_AUFBV	4	Bit-vectors with Arrays and Uninterpreted Functions
AUFLIA+p	1	Quantified Linear Integer Arithmetic with Uninterpreted Functions and Arrays
AUFLIA-p	1	Quantified Linear Integer Arithmetic with Uninterpreted Functions and Arrays
AUFLIRA	1	Quantified Arrays, Uninterpreted Functions, and Linear Arithmetic

表 2 SMT-COMP における部門別の上位 3 位以内のソルバー

年度	QF_UF	AF_RDL	QF_UFIDL	QF_LRA	QF_UFLRA	QF_AUFLIA	Q_BV
2009	Yices2 proto	Yices2 proto	MathSAT4.3	Yices2 proto	Yices2 proto	Yices2 proto	MathSAT4.3
	MathSAT4.3	Sateen3.5	Yices2proto	Sateen3.5	MathSAT4.3	MathSAT 4.3	Boolector 1.2
	OpenSAT	MathSAT4.3	veriT200907	MathSAT4.3	CVC3 2.0	CVC3 2.0	Beaver-smtcomp
2008	Yices2 proto	Z3.2	Z3.2	Yices2 proto	Z3.2	Z3.2	Boolector
	Z3.2	Yices2 proto	Barcelogic1.3	Z3.2	MathSAT4.2	Barcelogic1.3	Z3.2
	MathSAT4.2	MathSAT4.2	MathSAT4.2	—	Barcelogic1.3	CVC3 1.5	Beaver-1.0
2007	Z3.0.1	Yices1.0.10	Yices1.0.10	Yices1.0.10	—	Yices1.0.10	Spear v1.9
	Yices1.0.10	Z3.0.1	Z3.0.1	Z3.0.1	—	Z3.0.1	Z3.0.1
	Barcelogic1.2	MathSAT 4.0	Barcelogic1.2	Barcelogic1.2	—	CVC3 1.2	Yices1.0.10

Vol.16, No.3, pp.323–332, 1978.

[Oppen 80a] D. C. Oppen: Complexity, Convexity and combinations of theories, *Theoretical Computer Science*, Vol.12, pp.291–302, 1980.

[Oppen 80b] D. C. Oppen: Reasoning about recursively defined data structures. *Journal of the ACM*, Vol.27, No.3, pp.403–411, 1980

[Papadimitriou 81] C. H. Papadimitriou: On the Complexity of Integer Programming, *Journal of the ACM*, Vol.28, No.4, pp.765–768, 1981.

[Ranise 06] S. Ranise and C. Tinelli: Satisfiability Modulo Theories: Trends and Controversies. *IEEE Intelligence Systems Magazine*, Vol.21, No.6, pp71–81, 2006.

[Shankar 02] N. Shankar and H.Rueß: Combining Shostak Theories. *Proc. of Inter. Conf. on Rewriting Techniques and Applications (RTA), LNCS*, Vol.2378, pp.1–18, 2002.

[Stump 01] A. Stump, C. W. Barret, D. L. Dill and J. Levitt: A Decision procedure for an extensional theory of arrays. *Proc. of 17th IEEE Symp. on Logic in Computer Science*, pp.29-37, 2001.

[Suzuki 80] N. Suzuki and D. Jafferson: Verification decidability of Presburger array programs. *Journal of the ACM*, Vol.27, No.1, pp.191–205, 1980.

[Shostak 84] R. E. Shostak: Deciding Combination of Theories. *Journal of the ACM*, Vol.31, No.1, pp.1–12, 1984

[田村 10] 田村直之，丹生智也，番原陸則：制約最適化問題と SAT 符号化人工知能学会誌, Vol.25, No.1, 2010.

[Tinelli 04] C. Tinelli: The Combination Problem in First-Order Logic The Unsorted Case. in *Proc. of Combination of Decision Procedures: Summer School 2004*. http://verify.stanford.edu/summerschool2004/

[Tinelli 05] C. Tinelli and C. G. Zarba: Combining Nonstably Infinite Theories. *Journal of Automated Reasoning*, Vol.34, No.3, pp.209–238 2005.

〔担当委員：××○○〕

19YY 年 MM 月 DD 日 受理

著 者 紹 介

岩沼 宏治（正会員）

1985 年東北大学大学院・電子及通信工学専攻修士課程修了．同年山形大学情報工学科助手．現在，山梨大学大学院・医学工学総合研究部教授．博士 (工学)．人工知能の基礎，特に非単調論理，定理自動証明プログラム，WEB 知的処理，系列データマイニング等の研究と開発に従事．19 87,89, 90, 91 年人工知能学会全国大会優秀論文賞受賞．2004 年 FIT 優秀論文賞受賞

鍋島 英知（正会員）

1998 年豊橋技術科学大学大学院・工学研究科情報工学専攻修士課程修了．2001 年神戸大学大学院・自然科学研究科情報メディア科学専攻博士課程修了．同年山梨大学工学部コンピュータ・メディア工学科助教．2009 年同大学大学院・医学工学総合研究部准教授．神戸大学博士 (工学)．アクション理論，プランニング，オートマトン，定理自動証明プログラム，SAT プランニングなどの研究に従事．2004 年 FIT 優秀論文賞受賞．