BDD を用いたガード付きフラグメントの充足可能性判定

Checking Satisfiability of Guarded Fragment Using BDD

佐藤 貴洋 † 田辺 良則 †† 萩谷 昌己 †

Takahiro Sato Yoshinori Tanabe Masami Hagiya

† 東京大学情報理工学系研究科
Graduate School of Information Science
and Technology, University of Tokyo
{takahiro,hagiya}@is.s.u-tokyo.ac.jp

††(独) 科学技術振興機構
Japan Science
and Technology Agency
tanabe.yoshinori@aist.go.jp

本発表では，BDD(Binary Decision Diagram) を用いて，一階述語論理のガード付きフラグメント (Guarded Fragment) の充足可能性判定を効率よく行う自動証明器について述べる．この証明器は，与えられたガード付きフラグメントの文を充足するモデルが存在するか否かを判定する．ガード付きフラグメントが木モデルを持つ性質を利用して，タブローを構成する．矛盾したノードの削除を繰り返し，与えられた文を含むノードが削除されていなければ充足可能となる．BDD を用いることにより，ノードを実際に枚挙することなく，効率良くタブローの構成を行うことができる．

1 はじめに

一階述語論理のガード付きフラグメント (guarded fragment) は，限量子 ($\forall$ と $\exists$) の適用が常にガードと呼ばれる論理式に制限された部分体系である．ガード付きフラグメントは，命題様相論理の自然な拡張になっており，命題様相論理の論理式をガード付きフラグメントの論理式に変換することが可能である．命題様相論理と比べて，ガード付きフラグメントは一階述語論理の部分体系であるため，様々な応用が考えられ，近年活発に研究が進められている [1].

ガード付きフラグメントは，命題様相論理と同様に，以下のような重要な性質を有する．

- 充足可能性問題は決定可能である．
- 木モデル性，有限モデル性を有する．

本研究では，BDD を用いてガード付きフラグメントの効率的な充足可能性決定手続きを実装した．SMV[2] をはじめ，モデル検査に BDD が有効であることはよく知られている．充足可能性判定に BDD を用いることはあまり一般的ではないが，最も基本的な様相論理である K の充足可能性判定に BDD が有効である事が報告されている [3]．また，我々は先に，BDD を用いて 2 方向 CTL の充足可能性決定手続きの実装を行った [4]．戸沢は，XML の処理に応用するために，有限二分木に対する制限された 2 方向 μ 計算の充足可能性決定手続きの実装を行っている [5].

本論文の構成は以下のようである．第 2 節では，ガード付きフラグメントの論理式，および正形式について定義する．第 3 節では，充足可能性決定手続きのアルゴリズムについて述べ，そのアルゴリズムの正しさについて簡単に議論する．第 4 節では，この手続きの BDD を用いた実装について述べる．

2 ガード付きフラグメント

本節では，一階述語論理のガード付きフラグメントについて簡単に紹介する．

2.1 論理式

PS を述語記号の集合 (各々引数を取る)，Var を変数記号の集合，Const_0 を定数記号の集合とする．等号と関数記号を含まない一階述語論理の論理式の部分集合として，ガード付きフラグメント (GF) を以下のように帰納的に定義する．

- $P \in \mathrm{PS}$, P が m 引数をとり，$t_1, \ldots, t_m \in \mathrm{Var} \cup \mathrm{Const}_0$ のとき，$P(t_1, \ldots, t_m) \in \mathrm{GF}$
  ($\mathrm{このような}\ P(t_1, \ldots, t_m)$ を，特に原子論理式という．)
- $\varphi_1, \varphi_2 \in \mathrm{GF}$ ならば，
  $\neg\varphi_1, \varphi_1 \wedge \varphi_2, \varphi_1 \vee \varphi_2, \varphi_1 \rightarrow \varphi_2, \varphi_1 \leftrightarrow \varphi_2 \in \mathrm{GF}$
- $\mathbf{x} \subseteq \mathrm{Var}$, p が原子論理式, $\varphi \in \mathrm{GF}$, $\mathrm{free}(\varphi) \subseteq \mathbf{x} \subseteq \mathrm{free}(p)$, $\mathrm{const}(\varphi) \subseteq \mathrm{const}(p)$ のとき，
  $$\exists \mathbf{x}.(p \wedge \varphi), \forall \mathbf{x}.(p \rightarrow \varphi) \in \mathrm{GF}$$

但し，$\varphi \in \mathrm{GF}$ に対して，$\mathrm{free}(\varphi) \subseteq \mathrm{Var}$, $\mathrm{const}(\varphi) \subseteq \mathrm{Const}_0$ を，それぞれ φ 中の自由変数の集合, 定数の集合とする．3 番目の規則において，$\mathbf{x}$ は変数の有限列を表すが，包含関係の中では，有限列を構成する変数の集合とみなす．また，3 番目の規則の原子

論理式 p を特に，限量子 $\exists\mathbf{x}$ または $\forall\mathbf{x}$ のガードと呼ぶ．以後，$\exists\mathbf{x}.(p \wedge \varphi), \forall\mathbf{x}.(p \rightarrow \varphi)$ をそれぞれ，$(\exists\mathbf{x}.p)\varphi, (\forall\mathbf{x}.p)\varphi$ と書く．

2.2 正形式

ガード付きフラグメントの論理式が正形式 (positive form) であるとは，次の 2 条件を満たすことをいう．

- 否定記号が原子論理式の直前にしか現れない．
- 各々の変数について，それが束縛される場所が 1 箇所しかない．(束縛変数の名前はすべて異なる．)

以下，特に断らない限りは正形式の論理式のみを考えることにする．

3 充足可能性判定

ガード付きフラグメントの充足可能性を判定するアルゴリズムについて述べる．

3.1 タブローの構成

φ_0 を，定数記号が現れず，自由変数も持たない正形式の論理式とする．自然数 k を，$k = \mathrm{width}(\varphi_0) = \max\{\ |\mathrm{free}(\varphi)|\ \mid\ \varphi$ はφ_0の部分論理式 $\}$ によって定義する．以下，$\mathrm{Const} \subseteq \mathrm{Const}_0$ および $|\mathrm{Const}| = k$ を満たす集合 Const を一つ取り，固定する．

$\mathrm{cl}(\varphi_0)$ を，自由変数を持たない正形式論理式の集合で，次の条件を満たす最小のものとする．

- $\varphi_0 \in \mathrm{cl}(\varphi_0)$
- $\neg p \in \mathrm{cl}(\varphi_0) \Rightarrow p \in \mathrm{cl}(\varphi_0)$　(p は原子論理式)
- $\varphi_1 \wedge \varphi_2 \in \mathrm{cl}(\varphi_0) \Rightarrow \varphi_1 \in \mathrm{cl}(\varphi_0),\ \varphi_2 \in \mathrm{cl}(\varphi_0)$
- $\varphi_1 \vee \varphi_2 \in \mathrm{cl}(\varphi_0) \Rightarrow \varphi_1 \in \mathrm{cl}(\varphi_0),\ \varphi_2 \in \mathrm{cl}(\varphi_0)$
- $(\exists\mathbf{x}.p)\varphi \in \mathrm{cl}(\varphi_0), \mathbf{c} \subseteq \mathrm{Const}$
  $\Rightarrow p(\mathbf{c}/\mathbf{x}) \in \mathrm{cl}(\varphi_0), \varphi(\mathbf{c}/\mathbf{x}) \in \mathrm{cl}(\varphi_0)$
- $(\forall\mathbf{x}.p)\varphi \in \mathrm{cl}(\varphi_0), \mathbf{c} \subseteq \mathrm{Const}, p(\mathbf{c}/\mathbf{x}) \in \mathrm{cl}(\varphi_0),$
  $\Rightarrow \varphi(\mathbf{c}/\mathbf{x}) \in \mathrm{cl}(\varphi_0)$

但し，$\mathbf{c} \subseteq \mathrm{Const}$ とは，$\mathbf{c}$ が Const の要素の有限列であることを示している．$\varphi(\mathbf{c}/\mathbf{x})$ は，φ の中に現れる $\mathbf{x}$ の変数を，対応する $\mathbf{c}$ の要素で置き換えたものを示している．なお，$\mathbf{x}$ と $\mathbf{c}$ の長さは同じである．

また，6 番目の条件のように，$(\forall\mathbf{x}.p)\varphi$ 中の原子論理式 p に対して，それに対応する $p(\mathbf{c}/\mathbf{x}) \in \mathrm{cl}(\varphi_0)$ が存在するとき，p は $p(\mathbf{c}/\mathbf{x})$ に適合する，という．

次を満たす $\Gamma \subseteq \mathrm{cl}(\varphi_0)$ を，φ_0 型 (φ_0-type) と言う．

- $\varphi \in \Gamma$ は，原子論理式であるか，$(\exists\mathbf{x}.p)\varphi'$ か $(\forall\mathbf{x}.p)\varphi'$ の形のいずれか
- $(\forall\mathbf{x}.p)\varphi \in \Gamma,\ \mathbf{c} \subseteq \mathrm{Const},\ p(\mathbf{c}/\mathbf{x}) \in \Gamma$
  $\Longrightarrow\ \varphi(\mathbf{c}/\mathbf{x}) \in \Gamma$

φ_0 型 Γ で $\varphi \in \mathrm{cl}(\varphi_0)$ が「成立している」ことを示す関係 $\Gamma \Vdash \varphi$ を次のように定義する．

- φ は，原子論理式であるか，$(\exists\mathrm{x}.p)\varphi'$ か $(\forall\mathrm{x}.p)\varphi'$ のいずれかのとき
  $\Gamma \Vdash \varphi \iff \varphi \in \Gamma$
- $\Gamma \Vdash \neg\varphi \iff \varphi \notin \Gamma$
- $\Gamma \Vdash \varphi_1 \vee \varphi_2 \iff \Gamma \Vdash \varphi_1$ または $\Gamma \Vdash \varphi_2$
- $\Gamma \Vdash \varphi_1 \wedge \varphi_2 \iff \Gamma \Vdash \varphi_1$ かつ $\Gamma \Vdash \varphi_2$
- $\Gamma \Vdash \varphi_1 \rightarrow \varphi_2$
  $\iff \Gamma \Vdash \mathrm{PNF}(\neg\varphi_1)$ または $\Gamma \Vdash \varphi_2$
  (但し，$\mathrm{PNF}(\neg\varphi_1)$ は $\neg\varphi_1$ を正形式に変換したもの)
- $\Gamma \Vdash \varphi_1 \leftrightarrow \varphi_2$
  $\iff \Gamma \Vdash \varphi_1 \rightarrow \varphi_2$かつ$\Gamma \Vdash \varphi_2 \rightarrow \varphi_1$

すべての φ_0 型の集合を T_0 で表し，$T \subseteq T_0$ を満たす T を，タブロー (tableaux) と言う．

φ_0 型 Γ と $T \subseteq T_0$ が次の条件を満たすとき，Γ は T で矛盾しない，という:　任意の $\Gamma \Vdash (\exists\mathbf{x}.p)\varphi$ に対して，$\Gamma' \in T$ と $\mathbf{c} \in \mathrm{Const}$ が存在して，

- $\Gamma' \Vdash p(\mathbf{c}/\mathbf{x})$ かつ $\Gamma' \Vdash \varphi(\mathbf{c}/\mathbf{x})$
- $\mathrm{const}(\theta) \subseteq \mathrm{const}((\exists\mathbf{x}.p)\varphi)$ を満たす任意の $\theta \in \mathrm{cl}(\varphi_0)$ に対して，$\Gamma \Vdash \theta \iff \Gamma' \Vdash \theta$

この時，$\Gamma, \Gamma' \in T$ 間の関係を $\Gamma \sqsupset \Gamma'$ と表す．

3.2 アルゴリズム

定数記号が現れず，自由変数も持たない (ガード付きフラグメントの) 論理式 φ_0 が与えられたときに，その充足可能性を判定するアルゴリズムは以下のようになる．

自然数 n に対して，

$$T_n := \{\Gamma \in T_{n-1}\ \mid\ \Gamma \text{ は } T_{n-1} \text{ で矛盾しない.}\}$$

とする．T_0 は既に定義したように，すべての φ_0 型の集合とする．$(T_n \mid n \in \omega)$ は，有限集合の減少列なので，いつかは必ず $T_n = T_{n-1}$ となる．この T_n を，$\overline{T}$ で表す．

充足可能性アルゴリズムは，$\Gamma \Vdash \varphi_0$ なる $\Gamma \in \overline{T}$ が存在すれば,「YES」と判定し，存在しなければ「NO」と判定する．

3.3 健全性

3.3.1 準備

$(V, \to_V)$ を高さ ω 以下の木とする．(ω は最小の無限順序数．)

$v_1, v_2 \in V$ に対し，以下を満たす $\pi : n \to V$ $(1 \le n \in \omega)$ を，v_1 から v_2 へ至るパスと呼び，$\pi : v_1 \rightsquigarrow v_2$ と表す:

- $\pi(0) = v_1$, $\pi(n-1) = v_2$
- すべての $i < n-1$ について，$\pi(i) \to_V \pi(i+1)$ または $\pi(i+1) \to_V \pi(i)$.

さらに，π が単射である時，これを v_1 から v_2 へ至る最短パスと呼び，$\pi : v_1 \overset{s}{\rightsquigarrow} v_2$ と表す.

v_1 から v_2 へ至る 2 つのパス $\pi_1 : n_1 \to V$ と $\pi_2 : n_2 \to V$ $(n_1 \le n_2)$ に対し，次のような e が存在するとき，π_1 は π_2 の部分パスである，といい，$\pi_1 \le \pi_2$ と表す.

- $e : n_1 \to n_2$, e は狭義単調増加
- $\pi_2(e(i)) = \pi_1(i)$

補題 3.1 $v_1, v_2 \in V$ とする.
(1) v_1 から v_2 へ至る最短パスがただ一つ存在する.
(2) $\pi_0 : v_1 \overset{s}{\rightsquigarrow} v_2$, $\pi_1 : v_1 \rightsquigarrow v_2 \implies \pi_0 \le \pi_1$

証明 やさしい. ∎

3.3.2 健全性

φ_0 がガード付きフラグメントのときには，アルゴリズムが「Yes」と判定したら，φ_0 を満たすモデルが構成できることを示す.

木 $(V, \to_V)$ を作る．同時に，関数 $\Gamma : V \to \overline{T}$ と関数 $C : \{\ (v, v') \mid v \to_V v' \text{または} v' \to_V v\ \} \to \mathscr{P}(\mathrm{Const})$ も定める．

$V = \varnothing$ としてスタートする．$\varphi_0 \in \Gamma_0 \in \overline{T}$ なる Γ_0 を一つ選ぶ．V に新しい要素 r_0 を加え，これを V の根とする．$\Gamma(r_0) = \Gamma_0$.

未処理のノード $v \in V$ をとる．各 $(\exists \mathbf{x}.p)\varphi \in \Gamma(v)$ に対して，V に新しい要素 v' を加える．$\Gamma(v)$ は $\overline{T}$ で矛盾しないので，次を満たす $\Gamma' \in \overline{T}$ と $\mathbf{c} \subseteq \mathrm{Const}$ がとれる.

- $p(\mathbf{c}/\mathbf{x}) \in \Gamma'$, $\varphi(\mathbf{c}/\mathbf{x}) \in \Gamma'$
- 任意の $\mathrm{const}(\theta) \subseteq \mathrm{const}(\varphi)$ なる $\theta \in \mathrm{cl}(\varphi_0)$ に対して，$\theta \in \Gamma \iff \theta \in \Gamma'$

そこで,

$$\Gamma(v') = \Gamma'$$
$$C(v, v') = C(v', v) = \mathrm{const}(\varphi)$$

と定める．以上で $(V, \to_V)$, Γ, C の構成ができた.

モデル $\mathcal{M} = (M, (P^{\mathcal{M}})_P)$ を定めるのだが，まず，集合としては,

$$M = V \times \mathrm{Const} = \{(v, c) \mid v \in V,\ c \in \mathrm{Const}\}$$

とする．述語記号 P の解釈 $P^{\mathcal{M}}$ は，以下のようにする.

$v_1, v_2 \in V$ とする．$C' \subseteq \mathrm{Const}$ と $\pi : v_1 \overset{s}{\rightsquigarrow} v_2$ に対し，すべての $i < \mathrm{dom}(\pi) - 1$ で $C' \subseteq C(\pi(i), \pi(i+1))$ となるとき，$v_1 \rightsquigarrow v_2 \,/\, C'$ と表す．$C' = \{c\}$ のときには，$v_1 \rightsquigarrow v_2 \,/\, c$ とも書く.

$A \subseteq M$, $v^* \in V$ とする．任意の $(v, c) \in A$ に対し，$v \rightsquigarrow v^* \,/\, c$ となるとき，v^* を A の代表ノードと呼ぶ.

そこで，述語記号 P に対し,

$$P^{\mathcal{M}} = \{\ \mathbf{m} \subseteq M \mid \mathbf{m} \text{ の代表ノード } v^* \text{ が存在して, } P\mathbf{m}_{\mathrm{C}} \in \Gamma(v^*)\ \}$$

と定義する．ただし，集合 $\mathbf{m}$ は M の要素の有限列であるが，M の部分集合とも考える．また，$A \subseteq M$ に対し，A_{V} と A_{C} は，それぞれ V 成分，Const 成分への射影とする:

$$A_{\mathrm{V}} = \{v \in V \mid (v, c) \in A \text{ for some } c\}$$
$$A_{\mathrm{C}} = \{c \in \mathrm{Const} \mid (v, c) \in A \text{ for some } v\}$$

補題 3.2
(1) $\varphi \in \mathrm{cl}(\varphi_0)$, $v_1, v_2 \in V$, $v_1 \rightsquigarrow v_2 \,/\, \mathrm{const}(\varphi)$ とすると，$\varphi \in \Gamma(v_1) \iff \varphi \in \Gamma(v_2)$
(2) $\mathbf{m} \subseteq M$ とし，v_1^* と v_2^* をともに $\mathbf{m}$ の代表ノードとすると，$P\mathbf{m}_{\mathrm{C}} \in \Gamma(v_1^*) \iff P\mathbf{m}_{\mathrm{C}} \in \Gamma(v_2^*)$
(3) $\mathbf{m} \subseteq M$ とし，v^* を $\mathbf{m}$ の代表ノードとすると，$\mathbf{m} \in P^{\mathcal{M}} \iff P\mathbf{m}_{\mathrm{C}} \in \Gamma(v^*)$

証明
(1) は，$\overline{T}$ で矛盾しないことから明らか．(2) は (1) より $v_1^* \rightsquigarrow v_2^* \,/\, \mathbf{m}_{\mathrm{C}}$ が言えればよいが，$\pi : v_1^* \overset{s}{\rightsquigarrow} v_2^*$ は，$\pi_1 : v_1^* \overset{s}{\rightsquigarrow} \mathbf{m}_{\mathrm{V}}(j)$ と $\pi_2^{-1} : \mathbf{m}_{\mathrm{V}}(j) \overset{s}{\rightsquigarrow} v_2^*$ を連結したものの部分パスであることから従う．$\mathbf{m}_{\mathrm{V}}(j)$ は $\mathbf{m}_{\mathrm{V}}$ の j 番目の要素を表す (j は何でもよい)．(3) は (2) から明らか. ∎

$\mathcal{M}$ の拡張 $\mathcal{M}(v)$ を，定数の解釈を $c^{\mathcal{M}(v)} = (v, c)$ としたもの，として定義する.

補題 3.3 $v \in V$ とし，φ を $\mathrm{cl}(\varphi_0)$ に属する論理式の部分論理式とする．$\mathrm{free}(\varphi) = \mathbf{x}' \oplus \mathbf{x}''$, $\mathbf{a} \subseteq M$, $\mathbf{a} = \mathbf{a}' \oplus \mathbf{a}''$, $\mathrm{len}(\mathbf{x}') = \mathrm{len}(\mathbf{a}')$, $\mathrm{len}(\mathbf{x}'') = \mathrm{len}(\mathbf{a}'')$ であり，また，$\mathbf{a}' \cup \{(v, c) \mid c \in \mathrm{const}(\varphi)\}$ が代表ノード v^* を持つとする．このとき,

$$\mathcal{M}(v) \models \varphi[\mathbf{a}] \iff \mathcal{M}(v^*) \models \varphi(\mathbf{a}'_{\mathrm{C}}/\mathbf{x}')[\mathbf{a}'']$$

である．ただし，集合 A, B が $A \cap B = \varnothing$ であるとき，$A \cup B$ を $A \oplus B$ と記している．また，$\varphi[\mathbf{a}]$ は φ の自由変数を $\mathbf{a}$ の要素で置き換えた結果を表す．

証明　φ の構成に関する帰納法による．面倒だがやさしい．■

補題 3.4　φ_0 がガード付きフラグメントの場合には，$\varphi \in \mathrm{cl}(\varphi_0)$, $v \in V$ とすると，次が成り立つ．

$$\varphi \in \Gamma(v) \implies \mathcal{M}(v) \models \varphi$$

証明　φ の構成に関する帰納法．

原子論理式のときと，その否定と，主論理記号が $\wedge, \vee$ のときは，$P^{\mathcal{M}}$ の定義と φ_0 型の定義から従う．

$(\forall)$:　$\varphi = (\forall \mathbf{x}.p)\psi \in \Gamma(v)$ とする．$\mathcal{M}(v) \models p[\mathbf{m}]$ なる $\mathbf{m} \subseteq M$ を任意にとり，$\mathcal{M}(v) \models \psi[\mathbf{m}]$ を示せばよい．p は $P\mathbf{t}$ という形をしている．$\mathbf{a} \subseteq M$ を，$\mathbf{t}(\mathbf{m}/\mathbf{x})$ の中の各定数 c を (v, c) で置き換えた結果として定義する．$\mathcal{M}(v) \models p[\mathbf{m}]$ だから，$\mathbf{a} \in P^{\mathcal{M}}$ である．よって，$\mathbf{a} = \mathbf{m} \cup \{(v, c) \mid c \in \mathrm{const}(p)\}$ の代表ノード v^* が存在して，$P\mathbf{a}_C \in \Gamma(v^*)$．一方，$\mathrm{const}(\varphi) = \mathrm{const}(p)$ だから，$\varphi \in \Gamma(v^*)$．φ_0 型の定義により $\psi(\mathbf{a}_C/\mathbf{x}) \in \Gamma(v^*)$．帰納法の仮定から，$\mathrm{M}(v^*) \models \psi(\mathbf{a}_C/\mathbf{x})$．補題 3.3 を適用して，$\mathcal{M}(v) \models \psi[\mathbf{m}_C]$．したがって，$\mathcal{M}(v) \models \varphi$ である．

$(\exists)$:　$\varphi = (\exists \mathbf{x}.p)\psi \in \Gamma(v)$ とする．$\Gamma(v)$ は $\overline{T}$ で矛盾しないから，$v \to_V v'$ なる $v' \in V$ と $\mathbf{c} \subseteq \mathrm{Const}$ があって，$p(\mathbf{c}/\mathbf{x}) \in \Gamma(v')$ かつ $\psi(\mathbf{c}/\mathbf{x}) \in \Gamma(v')$．帰納法の仮定から $\mathcal{M}(v') \models p(\mathbf{c}/\mathbf{x}) \wedge \psi(\mathbf{c}/\mathbf{x})$, すなわち $\mathcal{M}(v') \models (p \wedge \psi)(\mathbf{c}/\mathbf{x})$．$\mathbf{m} \subseteq M$ を，$\mathbf{m} = \{v'\} \times \mathbf{c}$ で定義すると，v' は，$\mathbf{m} \cup \{(v, c) \mid c \in \mathrm{const}(\varphi)\}$ の代表ノードになる．したがって補題 3.3 が適用できて $\mathcal{M}(v) \models (p \wedge \psi)[\mathbf{m}]$．すなわち，$\mathcal{M}(v) \models \varphi$ である．■

補題 3.4 を $\varphi = \varphi_0, v = r_0$ として適用することで，次が得られる．

定理 3.5　自由変数を持たない論理式 φ_0 に対し，2 節のアルゴリズムを適用して「Yes」と判定されれば，φ_0 は充足可能である．■

3.4　完全性

本論文では紙数の都合上，証明を省略するが，次の定理が成り立つ．

定理 3.6　由変数を持たない論理式 φ_0 が充足可能ならば，2 節のアルゴリズムは「Yes」と判定する．■

4　BDD による実装

4.1　BDD の使用

3.2 節で述べたアルゴリズムを BDD を用いて実装するために，次のように，各概念に対応する BDD の表現を定める．

まず，集合 C を，$C = \{\varphi \in \mathrm{cl}(\varphi_0) \mid \varphi$ は，原子論理式であるか, $(\exists \mathbf{x}.p)\varphi'$ か $(\forall \mathbf{x}.p)\varphi'$ の形のいずれか $\}$ と定める．C の各要素に，BDD 変数を一つずつ割り当てる．φ に割り当てられた BDD 変数を $e(\varphi)$ で表す．このとき，$\Gamma \in 2^C$ に対する BDD を，同じ記号を用いて，

$$e(\Gamma) = \bigwedge\{e(\varphi) \mid \varphi \in \Gamma\} \wedge \bigwedge\{\neg e(\varphi) \mid \varphi \in C \setminus \Gamma\}$$

によって定義する．例えば，p, q を原子論理式として，$e(p) = x_1, e(q) = x_2, e((\exists \mathbf{x}.p)q) = x_3$ であれば，$e(\{p, q\}) = x_1 \wedge x_2 \wedge \neg x_3$ となる．さらに，$T \subseteq 2^C$ に対応する BDD を，

$$e(T) = \bigvee\{e(\Gamma) \mid \Gamma \in T\}$$

によって定義する．

$\psi \in \mathrm{cl}(\varphi_0)$ は，C の要素のブール結合として書くことができる．そのブール結合における C の要素 φ を $e(\varphi)$ で置き換えて得られる BDD を，同じ記号を用いて $e(\psi)$ と表す．例えば，上記の BDD 変数割り当てによれば，$e(p \wedge \neg q \vee (\exists \mathbf{x}.p)q) = x_1 \wedge \neg x_2 \vee x_3$ と書くことができる．

次に，φ_0 型間の関係 $\sqsupset$ を表現するため，各 BDD 変数 x に対して，新たな BDD 変数 x' を導入する．BDD b に対して，b に現れる各々の変数 x を対応する変数 x' で置き換えたものを b' で表す．このようなプライム付きの BDD は，関係 $\Gamma \sqsupset \Gamma'$ において，Γ' に相当する φ_0 型を表すために用いられる．

3.2 節のアルゴリズムの BDD による実装を，擬似コードとして図 1 に示す．`type` は φ_0 型全体の集合を，`tabl[l]` は集合 T_l を，それぞれ表現したものである．また，$(e(\varphi))'$ を $e'(\varphi)$ と表記している．

4.2　実行例と評価

前節の擬似コードで示した手続きを，Java 言語のプログラムとして作成した．BDD ライブラリには，JavaBDD [6] を使用した．

表 1 に示されたガード付きフラグメントの論理式をいくつか用意し，これらを入力として実行時間と

```
BDD type, tabl[];
Boolean satisfiable (PNF φ0) {
    type := ⋀{e(φ) ∧ e(p(c/x)) → e(ψ(c/x)) | φ = (∀x.p)ψ ∈ cl(φ0), c ⊆ Const}
    int l := 0;
    tabl[0] := type;
    repeat {
        tabl[l+1] := tabl[l] ∧ consis(l);
        l := l+1;
    } until (tabl[l] == tabl[l-1])
    if(e(φ0) ∧ tabl[l] != 0) {return true;} else {return false;}
}
BDD consis(int l) {
    return ⋀{e(φ) → ∃x⃗(tabl[l]' ∧ require1(φ) ∧ require2(φ)) | φ = (∃x.p)ψ ∈ cl(φ0)} ;
}
BDD require1(PNF φ) { return ⋁{e'(p(c/x)) ∧ e'(ψ(c/x)) | φ = (∃x.p)ψ, c ⊆ Const} ; }
BDD require2(PNF φ) { return ⋀{e(θ) ↔ e'(θ) | θ ∈ cl(φ0), const(θ) ⊆ const(φ)} ; }
```

図 1: BDD を用いた実装

表 1: 評価に用いたガード付きフラグメントの論理式

No.	論理式
1	$(\exists x.D(x,x))(\forall y.D(y,x))(R(y,x) \wedge R(x,y))$
2	$(\exists x.D(x))\neg(C(x) \wedge \neg C(x)$ $\vee(\forall x.R(x,y))((\forall z.R(y,z))P(z) \rightarrow P(y))$ $\vee(\forall x.R(x,y))((\forall z.R(y,z))P(z) \rightarrow P(y))$ $\vee C(x) \wedge \neg C(x))$
3	$(\exists x.P(x))((\forall y.R(x,y))(A(y) \rightarrow B(y))$ $\wedge(\forall y.R(x,y))A(y)$ $\wedge\neg(\forall y.R(x,y))B(y))$

使用メモリ量 (BDD の大きさ) の評価を行った．表 1 の論理式はそれぞれ，No.1,2 は充足可能，No.3 は充足不可能である．表 2 に，それぞれの論理式に対応して，(A) 実際のタブロー T_l の要素数 (の最大値)，(B) このタブローを表現するのに用いられた BDD $e(T_l)$ のノード数 (の最大値)，及び，(C) プログラムの終了時に使用中の BDD ノード数を示す．BDD を使用せずに実装しようとすると，集合 T_l を直接的に扱わなくてはならないため，少なくとも (A) の数のノードを作成する必要があり，論理式が大きくなると実装不可能になる．しかし，本研究では BDD を使用したために，T_l 自体の表現に必要なメモリ量は (B) に示すノード数だけでよい．プログラム全体でも，使用されるノード数 (GC によって再利用される量も含んでいる) は，(C) の程度である．これらの結果から，

表 2: T_l の要素数と BDD ノード数および実行時間

No.	(A)	(B)	(C)	実行時間 (ミリ秒)
1	1118	28	1296	321
2	20608	25	1574	591
3	62208	181	2952	461

BDD を使用することによって，素朴な実装に比べて必要なメモリ量が大幅に少なくなっていることがわかる．

参考文献

[1] Grädel, E., Thomas, W., and Wilke, T.(eds.): *Automata, Logics, and Infinite Games: A Guide to Current Research*, Lecture Notes in Computer Science, Vol.2500, Springer, 2002.

[2] McMillan, K.: *Symbolic Model Checking*, Kluwer Academic Publ., 1993.

[3] Pan, G., Sattler, U., and Vardi, M. Y.: BDD-Based Decision Procedures for K, *Proceedings of the 18th International Conference on Automated Deduction*, Springer-Verlag, 2002, pp. 16-30.

[4] 田辺良則，高橋孝一，山本光晴，佐藤貴洋，萩谷昌己: BDD を用いた 2 方向 CTL 論理式充足可能性決定手続きの実装, **コンピュータソフトウェア**, 投稿中, 2004.

[5] Tozawa, A.: On Binary Tree Logic for XML and Its Satisfiablity Test, **第 6 回プログラミングおよびプログラミング言語ワークショップ** (*PPL2004*), 2004.

[6] JavaBDD: `http://javabdd.sf.net/`.