社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.
SS2003-47 (2004-03)

関数型言語MLによるプレスブルガー文真偽判定ルーチンの開発と検証支援システムへの応用

才村　徹也†　　岡野　浩三†　　谷口　健一†

† 大阪大学大学院情報科学研究科　〒560-8531 豊中市待兼山町 1-3
E-mail: †{saimura,okano,taniguchi}@ist.osaka-u.ac.jp

あらまし　本稿では，関数型プログラミング言語 ML により実装した整数プレスブルガー文真偽判定ルーチンについて述べる．このルーチンは一般のプレスブルガー文を入力とするが，ある限定クラスにおいては高速に動作する．研究グループで行った CPU KUE-CHIP2 の検証で用いたプレスブルガー文に対して評価実験を行い，適当な速度で真偽判定が行えることを確認した．また，作成したルーチンを用いた形式的検証支援システムについても述べる．
キーワード　形式的検証，関数型プログラミング言語，ML，プレスブルガー文

An implementation of a decision procedure for Presburger sentences in ML and its application to verification support system

Tetsuya SAIMURA†, Kozo OKANO†, and Kenichi TANIGUCHI†

† Graduate School of Information Science and Technology, Osaka University
Machikaneyama 1-3, Toyonaka-shi, 560-8531 Japan
E-mail: †{saimura,okano,taniguchi}@ist.osaka-u.ac.jp

Abstract　This paper presents an implementation of a decision procedure for Presburger sentences in functional programming language ML. It uses a fast algorithm for inputs in the from of universal quantifier prenex form; for the other inputs, its uses a general known algorithm. We have applied the procedure to verification of CPU KUE-CHIP2. The results show its effectiveness. We also describe an outline of verification support system for ML, where the decision procedure for Presburger sentences will be used as central engine of verification system.
Key words　formal verification, functional language, ML, Presburger sentences

1. まえがき

信頼性の高いソフトウェアの設計，開発において形式的検証は有用であるが，形式的検証における課題として，モデル検査技術 [12] における “状態爆発問題” や，定理証明系の自動化技術 [13] の欠如 (検証者の知識や技術に頼る部分が多い) 等が挙げられる．これを克服するために大きく複雑なソフトウェアを，基本的なモジュールに分割し検証を行う手法 [8], [9] や，形式手法をハイブリッドに用いる手法 [10], [11] などが考案されている．理論的な研究と共に，人の作業と計算機の作業をうまく協調させ，効率良く検証を行う研究も必要である．

我々の研究グループでは，そのような観点から主に代数的言語を対象にプログラムの代数的手法に基づいた仕様記述方法，段階的設計法，プログラムの正しさの検証支援方法や，ハードウェア設計自動化に関する研究を行ってきた [6]．検証支援においては項書換え系に関する種々の理論的結果の実用的利用 [14] や，整数を持つプログラムの効率的検証を目的としたプレスブルガー文真偽判定ルーチンの利用 [1], [5] などに特色があり，今までに在庫管理プログラムなどの例題に対して，その有用性を調べてきた．

現在，モジュールで構成された実用プログラムに対して，効率的な検証支援を行う枠組みを研究するため，関数型言語を対象にした形式的検証システムを含む統合開発環境の構築に取り組んでいる．プログラムの満たすべき性質を等式論理の形で，プログラム中のコメントとして記述し，それを利用し検証を行う．関数型プログラムは，等式論理の集合であり，性質記述との親和性が高い．項書換え系の性質を利用し，性質記述とソースプログラムの情報を用いて，プログラムの正しさを示す論理式を作成する．得られた論理式は，プレスブルガー文真偽判定アルゴリズムを用いてその真偽を判定する．

本稿では，その検証システムで用いるプレスブルガー文真偽判定アルゴリズムを関数型言語 ML で実装し，有用性評価のた

め，実際に検証で用いられたデータに対して適用したことについて述べる．

作成したルーチンは一般のプレスブルガー文を入力の対象とする．ただし，全ての変数が同一限定子に束縛された冠頭標準形プレスブルガー文の場合には，そのような入力のクラスの真偽判定を高速化するためのアルゴリズムとして知られている森岡らの高速化アルゴリズム [1] を適用する．

検証システムの対象と同じ言語を用いることで，同システムで検証を行え，システムの評価とルーチンの保守発展的な開発に役立つと考えている．また，検証式の真偽判定は検証時間に最も深く関わる部分であるが，ML は関数型言語の中でも比較的歴史が深く，ML による検証支援系の実現 (HOL [17]) も世界的に知られている．処理系ライブラリも充実しており，速度の点で実用に足る，と考える．

ルーチンの実用性を調べるため，速度について評価実験を行った．研究グループでは，以前，教育用 CPU KUE-CHIP2 を段階的に設計し，その正しさの証明を代数的手法を用いて行った．その際に生成したプレスブルガー文に対して，本研究で作成したルーチンを適用した．トークン数 (変数，整数項などの総出現数) が 10〜200，変数が 1〜30 個であるような 300 個ほどの式が 1 秒以内に判定でき，実用的な速度で判定が行えることがわかった．

本稿では 2. と 3. でプレスブルガー文真偽判定アルゴリズムの概要を，4. で作成したルーチンの説明，5. で行った評価実験について述べる．最後に 6. で現在構築中の検証支援システムについて説明する．

2. プレスブルガー算術

加算を持つ整数の理論 (整数の集合 Z 上の変数，定数，$+$，$-$，$=$，$\geqq$，$\leqq$，$\vee$，$\wedge$，$\neg$，$\forall$，$\exists$ からなる理論) はプレスブルガー (Presburger) 算術と呼ばれ，その上の閉論理式をプレスブルガー文あるいは P 文と呼ぶ，P 文の真偽は決定可能である [3], [4].

P 文について簡単に説明する．P 文は閉論理式である．論理式はアトムまたは論理定数 (true，false) であり，論理式に，$\vee$，$\wedge$，$\neg$，$\forall$，$\exists$ を施したものも論理式である．ここでアトムとは，整数定数，変数 (整数定数を係数として持つことができる)，およびそれらの有限個の和 (および差) を項と呼び，二つの項を等号，または不等号で結合したものである．プレスブルガー算術はプログラムの InfeasiblePath 検出 [4]，プログラムや回路の正当性証明などに広く利用されている．P 文を全ての変数が同一の限定記号で束縛された冠頭標準形に制限することによる，より速い真偽判定アルゴリズムも知られている [1].

3. プレスブルガー文真偽判定アルゴリズム

以下では，実装に用いた P 文の真偽判定法の概略を述べる．まず基本的なアルゴリズムとして，一般の P 文に対する Cooper のアルゴリズム [3] について簡単に述べ，次に，すべての変数が存在記号で束縛された冠頭標準形で束縛された冠頭標準形 P 文に対する高速化手法について述べる．

3.1 一般のクラスに対するアルゴリズム

真偽を判定すべき P 文

$$\phi \triangleq \exists x_{\mathrm{m}} \ldots x_1 F(x_{\mathrm{m}} \ldots x_1)$$

があるとする (F は，否定記号を含まない式)．最も内側の束縛変数 x_1 に，その変数を含まない幾つかの式 $X_{(1,1)} \cdots X_{(N(1),1))}$ (それらは変数 x_2 から x_{m} と定数の一次結合．また，$N(1)$ は式中に現れる x_1 の個数や定数に依存したある有限の整数値) を代入した各論理式の論理和

$$\begin{aligned}\phi' = \exists x_m ... x_2 \{\ & F\ (X_{(1,1)}, x_2, ..., x_m) \\ & \wedge\ F(X_{(1,2)}, x_2, ..., x_m) \\ & \wedge\ \cdots \\ & \wedge\ F(X_{(1,N(1))}, x_2, ..., x_m)\}\end{aligned}$$

を構成する．(ϕ は ϕ' と等価．この処理を変数消去と呼ぶ)．以後同様に変数 $x_2 \cdots x_{\mathrm{m}}$ を消去していくことで変数のない等価な式が得られ，真偽が決定できる [2].

全称記号 ($\forall$) を含む P 文の場合，$\forall x$ を $\neg\exists x\neg$ に置き換え，最も内側の式から変数消去を行えば同様に真偽判定が行える．否定記号は，$\neg(P \wedge Q) = \neg P \vee \neg Q$ のように式の展開を行なっていき，最終的に否定のかかる不等式で $\neg(x > n)$ を $(x \leqq n+1)$ のようにして置き換えることにより消去できる．

3.2 限定クラスに対するアルゴリズム

Cooper の提案した高速化のアイディアを，直井氏らがアルゴリズムとして具体化・精密化した Cooper・直井の方法 [4] に以下に述べる工夫を適用することで，EPP 文の真偽判定の高速化を行う．EPP 文とは，$\neg$ と等号を含まず，限定子の形はすべて存在記号である冠頭標準形 P 文のことである．EPP 文の形は以下のように表される．

$$\phi \triangleq \exists x_{\mathrm{m}} \ldots x_1 F(x_{\mathrm{m}} \ldots x_1)$$

基本的なアルゴリズムは，上で述べたように，内側の変数から変数消去を行い，変数のない等価な式に変形していく．まず Cooper・直井の方法における変数消去について述べ，続いて適用する工夫について述べる．

3.2.1 変数消去

たとえば，ある整数変数 x_1 を消去するとする．まず，x_1 に対応する新しい変数 I_1 を導入する．I_1 は，とりうる範囲は $1 \leqq I_1 \leqq R_1$ に定まっている (R_1 は式より定まる，ある正の整数定数)．以下，I_i をレンジ属性付き変数 (レンジ変数) と呼ぶ．次に x_1 のいくつかの値 ($val_1 \sim val_k$) に対し，それぞれ，式

$$\phi_j \triangleq \exists x_{\mathrm{m}} \ldots \exists x_2 \exists I_1 F(val_{\mathrm{j}}, x_{\mathrm{m}}, \ldots, x_2) \qquad (1 \leqq j \leqq k)$$

の真偽を，残りの変数 $x_2 \ldots x_{\mathrm{m}}$ を同様に消去して調べる．その様子は木の形で表される．ここで，val_j は x_1 以外の変数，I_1 および定数の一次結合であり，ϕ から定まる．全ての ϕ_j が偽であれば ϕ は偽，一つでも真になれば ϕ は真である．すべての変数 $x_1, \ldots, x_n$ を消去すると，以下の一般型をした，$I_1, \ldots, I_n$ と定数だけからなる式 τ が $x_1, \ldots, x_n$ への各代入ごとに得ら

れる．どの τ も偽なら ϕ は偽，一つでも τ が真なら ϕ は真である．

$$\begin{aligned}\tau = \exists I_1 ... I_n \{ & F'(I_1, ..., I_n) \\ & \wedge \rho_1 \mid \mu_{11} I_1 + \mu_{12} I_2 + \cdots + \mu_{1n} I_n + \nu_1 \\ & \wedge \cdots \\ & \wedge \rho_n \mid \mu_{n1} I_1 + \mu_{n2} I_2 + \cdots + \mu_{nn} I_n + \nu_n \},\end{aligned}$$

where $1 \leqq I_1 \leqq R_1, ..., 1 \leqq I_n \leqq R_n$

ここで，$F'(...)$(以下 ω とする) はいくつかの (不) 等式の論理結合であり，$C \mid exp$ は合同式「$exp \equiv 0 \pmod{c}$」(式 exp が整数定数 C で割り切れること) を表す．各 ρ, μ, ν は整数定数である．

これに対して高速化の工夫 1 から 4 を適用する．工夫 1，2 の目的は変数消去で行われる代入の回数をなるべく減らすこと，工夫 3，4 の目的は，変数消去で得られたレンジ変数からなる式の真偽判定を高速化することである．

3.2.2 工夫 1：式の簡単化

以下の二つの方法を用いることで，式中の変数や不等式を減らし，簡単化する，ϕ の変数消去で行われる代入の回数が減る．

- 冗長な不等式

たとえば，$[x \leqq 0 \wedge x \leqq 2]$ を $[x \leqq 0]$ に置き換えるなどして，冗長な不等式を削除する．ただし，厳密に行うとコストが非常にかかるため，以下の条件を満たす不等式群についてのみ冗長性をチェックする．

- 変数の一次式がまったく同じ不等式同士であること
- 式を表す構文木上において，兄弟または叔父と甥にあたる不等式同士であること

- レンジ属性付き変数だけからなる不等式の真偽決定

レンジ属性付き変数だけからなる項は，その最大値と最小値を次のようにして求めることができる．変数の係数が正の変数に対してはレンジ属性の最大値，係数が負の変数に対しては最小値を代入する，このときの和が項の最大値であり，変数の代入に対して最大値と最小値を逆に扱ったものが項の最小値である．これより，不等式の真偽が決定できる場合がある．項 1 と項 2 について，項 1 の最大値が項 2 の最小値より小さければ，項 1< 項 2 であり，項 1 の最大値が項 2 の最大値より大きければ項 1> 項 2 である．

3.2.3 工夫 2：変数消去順の動的な制御

工夫 1 と同様に，式中の変数や不等式を減らし，変数消去の代入回数を減らす．式の構造などに着目して変数の消去順を動的に制御する．途中で得られた式に対し，その構文木の根により近い位置に出現する変数を，次に消去する．これにより，構文木の根に近い (不) 等式の真偽が定まり，変数を消去した後の式が非常に簡単になる場合が多い．

3.2.4 工夫 3：シンプレックス法を用いた充足不能性チェック

変数消去の結果，得られる式を τ の母式 ω(論理演算子で結合された論理式) を，実数上の線形計画法 (シンプレックス法) を利用してチェックし充足不能であれば，τ を偽とする．レンジ変数に代入する解は式中の整数変数の係数値が大きくなると爆発的に増加するのできわめて時間がかかるが，シンプレックス法にかかる手間は係数に依存しないので，この工夫によって係数値が相当大きな EPP 文に対しても，偽であることの検証を行える場合がある．

3.2.5 工夫 4：連立合同式の解探索順の制御

レンジ変数への値の代入の際，Cooper・直井の方法では，τ 中の連立合同式を解くことにより，解の候補を決定する．このとき，とりうる値の範囲が狭いレンジ変数から順に解を探索し，バックトラック探索の回数を減らす．

4. 実　　装

実装は，SML/NJ を用いて行った．その実装方針，入力とする P 文のクラスを述べ，主なモジュール構成について簡単に述べる．

最初に，入力クラスによって用いるアルゴリズムを選択する．したがって二つのアルゴリズムは，共通モジュールは用いるが，独立して実装することになる．

4.1 入力クラス

入力は一般の P 文である．将来システムで用いることを考えて以下の拡張を行った．

- ブール変数の導入
- if 文，包含，排他的論理和，論理比較等の演算子の追加

プログラムの性質から検証式を作成し，それを P 文で扱うために，整数の理論で扱えない部分はブール変数を用いて表す．二つ目の拡張は，単に表現力の問題で，ルーチンの方で低レベルな論理に等価変形する．

4.2 モジュール構成

実装したプログラムのモジュール構成を図 1 に示す．その中の主なモジュールについて簡単に説明する．ファイルから読み込んだ式は，抽象構文データ構造に格納し (InputExp)，等価変形を行い冠頭標準形の P 文として変数に格納する (prePB)．P 文の入力クラスを判定し，適用するアルゴリズムを決める (PBinit)．あとは真偽判定部において，アルゴリズムにより手順や用いるモジュールが異なるが，基本的には変数消去 (PBEliminate) と式の簡単化 (PBSimplify) を繰り返し，真偽判定を行う．

5. 評 価 実 験

5.1 CPU KUE-CHIP4 の検証例題

研究グループでは，すでに情報工学教育用に京都大学で開発された 8 ビット CPU KUE-CHIP2 [19] を段階的に設計し，その正しさを形式的に示した [20]．

それを基にした検証では，各段階の詳細化の正しさを示す式を作成し，それを P 文真偽判定手続きを用いて判定する．その際に生成される P 文に対して，今回作成した P 文真偽判定ルーチンを適用した．

実験には，過去のデータから復元した P 文を用いた．KUE-CHIP2 は，五つの段階 (レベル) をもって設計された．今回用いたのはレベル 3 からレベル 4 への詳細化の正しさを示す式であり，レベル 3 では各命令の実行手順の決定，レベル 4 では回路の実行効率を上げるためプリフェッチを行うようにしたもの

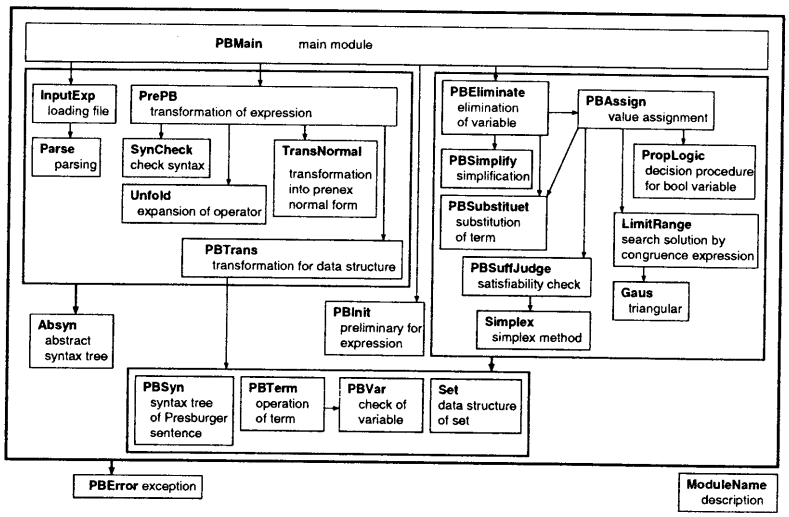

図 1　プレスブルガー文真偽判定アルゴリズムの実装
Fig. 1　Implementation of a decision procedure for Presburger sentences

である．具体的には，各レジスタについて，一命令実行前後における値の変化が，レベル 3 とレベル 4 で変わらないことを示す式を作成した．

トークン数 (変数，論理演算子，整数項などの総出現数) が 10～200，変数は 1～30 個，ブール変数 2～20 個の式が 346 個あった．全ての式は，限定子が全て同一であり，過去の結果より真であることがわかっている．図 2 は，そのプレスブルガー文の一例である．

```
(Token:47. Variable:10, Bool Variable 2)  u: universal quantifier
(( u v1 v2 v3 (((v1 = 12) and ((v2 = 50) or (v3 = 51))) and true)) and
 ( u v4 v5 v6 (((v4 = 12) and ((v5 = 50) or (v6 = 51))) and true))) imply
 (if (v7 == false) then (v8) else (v9) == if (v10 == false) then (v11) else (v12))
```

図 2　プレスブルガー文の例
Fig. 2　An example of proposed procedure for Presburger sentence

5.2　結果と考察

作成したルーチンに適用した結果，全ての式の判定時間の総和は 1 秒以内であった．実験は Pentium4 CPU 2.66GHz，メモリ 1GB のマシンで行った．

実際の検証において用いられる式を，満足のいく速度で判定できたことは，目的である検証支援システムでの有用性，システムに関数型言語を用いる実用性を示す結果になったと考える．

しかし，実験に用いた式が高速化アルゴリズムを利用できる限定された P 文であったこと，一般の P 文での実験も必要であるとも考えられる．そこで，変数の数やトークン数を変えて，ランダムで作成した P 文に対して，Merryland 大学において C で作られた P 文真偽判定ルーチン OmegaCaluculator [15] との比較実験についても行った．変数 10～20 個，トークン数を 500 個程度のブール変数のないプレスブルガー文に対して速度比較実験を行った結果，今回作成したルーチンの方が約 3，4 倍の実行時間が多くかかった．作成したルーチンがほぼ純粋な関数型言語の機能を用いて作成されていることを考慮に入れると，遜色ない結果であると考える．

今後，ルーチンの速度向上についても考える必要もあるが，システムに適用する実用性は十分あると考える．

6.　形式的検証支援システム

作成した P 文真偽判定ルーチンを使用する形式的検証システムを現在構築中である．この節では，まずシステムについて簡単に説明し，次にシステムにおいて重要な部分の設計方針について述べる．最後に，システムの将来的な課題，展望について述べる．

6.1　システムの概要

このシステムでは，適切に設計されたいくつかのモジュールからなる ML のプログラムを対象にしている．検証者がプログラムのソースコードにコメントとして検証用補題を等式論理の形で記述し (性質記述)，それを用いてプログラムの正しさを検証する．具体的には，モジュールのインターフェイスに対して，入力の守るべき前提条件，出力の守るべき後置条件を等式論理で表し，前提条件のもとで後置条件が守られることを証明する．その際，必要であれば補題としてプログラム中で成り立つ性質 (不変表明) を併せてソースコード中に記述する (契約による設計 [18])．

ここで対象としている言語は関数型言語である．関数型言語を対象としたのは関数型言語の次のような特徴をもつためである．

（1）　プログラムそのものを等式の集合として扱える

（2）　(1) の理由から，ソースプログラムの記述が性質記述と親和性が高い

これにより，プログラムにおける関数定義，性質記述は同様に管理することができ，項書換え系の性質を利用することで，論理式への代入，展開が容易に行える．

前提条件のもとで後置条件が成り立つ，という検証補題に対して項書換えを行い，述語 (性質記述として記述)，関数の展開を行い，低レベルの論理で表された検証式を P 文に変換し．それを今回作成した P 文真偽判定ルーチンを用いて真偽判定を行う．

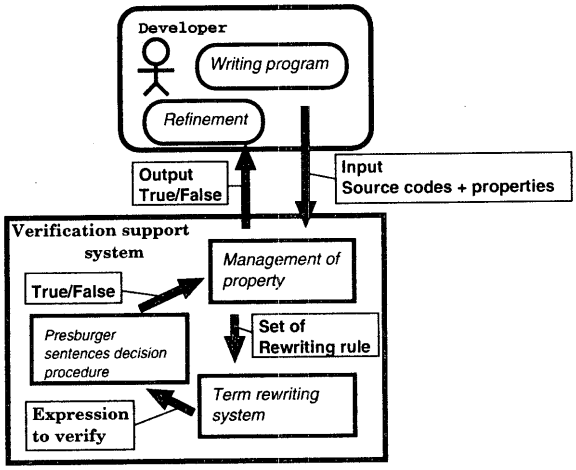

図 3　システムを用いた検証の流れ
Fig. 3　Formal verification flow with the proposed

真であれば，その検証用補題は正しい．それによりモジュール毎の正しさを検証することができ，それらのモジュールからなる大きなモジュールを検証することも可能である．システムを用いた検証の流れは図 3 のようになる．

6.2　設計方針

現在対象として考えている言語は，OCaml [16](StandardML の派生言語) である．プログラムの実行速度のベンチマークでは，高い評価を得ている．OCaml はオブジェクト指向を採り入れた実用的な関数型言語であるが，6.1 で述べたシステムでは，純粋な関数型言語としての性質を利用しているため，システムの実装には注意が必要である．OCaml 言語は純粋な関数型言語ではない機能も有しているが，それを使わずにプログラムすることも可能である．

現在は，形式的検証システムのプロトタイプとして，OCaml 言語にいくつか制限をおいたサブクラスに対する検証支援システムを構築中である．以下では，システムを実装する際の方針を述べる．

6.2.1　言語に対する制限

システムの簡単化のため，入力の OCaml 言語に対して次のような制限を課す．

（1）　オブジェクト指向に関する機能を用いない
（2）　参照型は用いない
（3）　場合分けは If 式だけとする
（4）　関数適用では，全てのパラメータが与えられるとする
（5）　モジュール内でのローカルなモジュール定義はないとする

最初のオブジェクト指向に関する機能を省いたのは，用いる手法が関数型言語に対するものであり，言語固有のオブジェクト指向機能に対する拡張は本質的ではないと判断したためである．

検証手法では純粋な関数型言語の性質を利用しているので，副作用を許す参照型変数を用いると検証が難しくなる．しかし実用プログラムにおいては，参照型変数が使われると考えられるので，今後，対応について検討する必要がある．

しかし，これらの制限を置いても，検証支援システムの実用プログラムに対する有用性はそれほど損なわれないと考える．これらの制限は，言語の表現能力を制限をするが，関数型言語としての能力にはさほど影響を与えない．

また，検証支援システムの対象として，大きなソフトウェアプログラム全体ではなく，その中の検証する必要のあるモジュール群を考えている．そのような部分に対する適用では，実用性も十分あると考える．

6.2.2　性質記述

OCaml の仕様では，モジュールのインターフェイスとして，関数の入力と出力の型をファイルに記述する．このファイルにコメントを用いて，入力に対する前提条件，出力に対する後置条件を記述してやれば，関数の入出力定義との対応が解りやすい．図 4 はソースコード中の性質記述の例である．

```
(*
整数とリストを引数に取り，整数番目の要素を取り出したリストを
返す関数pickupの型定義．
*)

val pickup : (int * 'a list ) -> 'a list
(*
pre (x, l) = ( x>0  and x < List.length(l) )
post  (x, l, pickup) =( List.length(l) -1 == List.length(pickup(x,l)) )
*)
```

図 4　性質記述の例
Fig. 4　An example of description of properties

システムは，このコメントの中の性質記述を読み取り，検証に用いる．また，性質記述で用いる述語や，補助的に不変表明を用いる場合も同様にコメントを用いて，モジュールファイルなどに記述する．

6.2.3　項書換え系の利用

プログラムの正しさは，「前提条件 imply 後置条件」という式の真偽を調べればよいことになる．

検証式の作成には，項書換え系の性質を用いる．「前提条件 imply 後置条件」という最初の式を，関数定義や，述語，性質記述の項書換え規則により，書換えを行えば，最終的に，関数等が展開された低レベルの論理を用いた式を得ることができる．整数とブール変数のみを用いた式であれば，P 文に変換し，真偽判定をすることができる．

実際のプログラムに対する検証では，この枠組みの適用が難しい場合もある．以下では，そのような場合に対する設計方針を述べる．

a）　再帰関数の扱い

関数型言語では，一般に再帰を用いてループを実現する．一般に，If 式などで再帰的に関数を呼び出すか，値を返すかという分岐の構造をとる．このとき，単純な項書換えではループ終了条件が明示的に真とされなければ，再帰的に無限に関数呼び出しについて項書換えを行うことになる．

これらの部分においては，項書換え系による自動的な検証式作成だけでなく，検証者による補助を用いることを考えている．

- 検証者が分岐について指定する．

検証者が，場合分けを行い，その場合における正しさを検証できるようにする．

- 帰納法を利用する.

再帰関数に関する性質を証明するために，補助的な不変表明を用いて，帰納法を用いて解く

b) 名前空間

項書換えは，項の名前による書き換えを行う．したがって，名前空間により用いるべき書換え規則集合が異なる．OCamlでは，Let式を用いて局所的に変数を定義し，値や関数を束縛して用いる機構がある．Let式の影響範囲にある式においては，Let式で定義したで束縛変数の束縛が優先されるため，Let式の外側の式とは別の書き換え規則集合を用いる必要がある．そのような場合には，階層的に項書換えを行うことで対処する．

関数を書き換えたあとの式に対してもその関数の定義された環境によって名前空間は異なる．ソースコード上では同じ名前でも，モジュール名が省略されている場合は，束縛されている値が異なることがある．したがって，文脈から名前への値の束縛のあったモジュールを補う必要がある．

c) 型付け

関数型言語において，全ての式は型を持ち，OCaml言語においても静的に型推論を行い，型チェックを行う．検証システムの項書換えにおいても，型の概念を付加しプログラムとして正しい項書換えを行うべきである．現在は，OCaml自体の型推論による型付け機構を利用して，読み込んだ式の型付けをすることを考えている．

6.3 将来展望

6.2の方針を基にした検証支援システムのプロトタイプを作成し，例題に適用することで有用性を評価することを考えている．例題としては，図書管理システムなどを考えているが，その評価方法を含め検討中である．

システムの発展形として，真偽判定の結果だけでなく検証者にとって有用な何らかのフィードバックを与えることを考えている．現在のシステムでは，P文による真偽判定の結果だけでは，ソースコードのバグか，性質記述のバグかわからない．フィードバックする情報によりプログラムの修正支援を行えるようにする．また，入力のOCaml言語を一般のクラスまで拡張し，そのための検証手法についても考案する．このシステムは，形式的検証を含んだ統合開発環境のメインのツールとして用いることを考えている．他の開発，検証支援としてプログラムの性質の記述支援，性質記述を含むプログラムの情報のドキュメント生成等の機能についても検討している．

7. あとがき

本稿では，関数型プログラミング言語MLの統合開発環境の形式的検証システムに用いるプレスブルガー文真偽判定アルゴリズムを，MLで実装したことについて述べた．入力クラスは一般のプレスブルガー文であるが，限定クラスにおいては，高速に動作する．速度について，過去研究グループで行った検証で用いたP文に対して適用した結果，実用的な速度で判定出来た．今後は，作成したルーチンを用いて，構築中の形式的検証システムを完成させ，例題に適用し，システムの有用性評価を行う．また主幹システムであるプレスブルガー文真偽判定ルーチンの高速化等についても今後の課題として挙げられる．

文献

[1] 森岡澄夫，柴田直樹，東野輝夫，谷口健一，“全ての変数が存在記号で束縛された冠頭標準形プレスブルガー文の真偽判定の高速化手法”，情報処理学会論文誌，Vol.38, No.12, pp.2419–2426, 1997.

[2] 東野輝夫，北道淳司，谷口健一，“整数上の線形制約の処理と応用”，コンピュータソフトウェア，vol.9, No.6, pp.31–39, 1992.

[3] D.C.Cooper, “Theorem Proving in Arithmetic without Multiplication,” Machine Intelligence, No.7, pp.91–99, 1972.

[4] 直井邦彰，高橋直久，“プレスブルガー算術を用いたInfeasible Path検出”，電子情報通信学会論文誌，Vol.J80-D-I, No.3, pp.269–281, 1993.

[5] 柴田直樹，岡野浩三，谷口健一，“凸多面体併合を用いた有理数プレスブルガー文真偽判定アルゴリズムの実装と形式的設計検証への適用”，電子情報通信学会論文誌，Vol.J84-D-I, No.7, pp.999–1008, 2001.

[6] K.Okano, Y.Kitahama, A.Kitajima, T.Higashino, K.Taniguchi, “Formal Verification of CPU in Laboratory Work,” Proceedings of the 2001 International Conference on Microelectronic Systems Education(MSE 2001), pp.32–36, 2001.

[7] 柴田直樹，岡野浩三，東野輝夫，谷口健一，“冠頭標準形有理数プレスブルガー文の真偽判定アルゴリズムの提案”，電子情報通信学会，Vol.J82-DI, No.6, pp.691–700, 1999.

[8] S.Chaki, E.Clarke, A.Groce, S.Jha, H.Veith, “Modular verification of software components in C,” Proceedings of the 25th international conference on Software engineering,pp.385–395 2003.

[9] F.Xie, J.C.Browne, “Verified Systems by Composition from Verified Components,” Proceedings of the 9th European software engineering conference, pp.277–286, 2003.

[10] R.D.Jeffords, C.L.Heitmeyer, “A Strategy for Efficiently Verifying Requirements Specifications Using Composition and Invariants,” Proceedings of the 9th European Software engineering conference, pp.28–37, 2003.

[11] S.Hazelhurst, O.Weissberg, G.Kamhi, L.Fix, “A hybrid verification approach: getting deep into the design,” Proceedings of the 39th conference on Design automation, pp.111–116, 2002.

[12] E.M.Clarke, O.Grumberg, D.A.Peled, “Model Checking,” MIT press, 1999.

[13] J.Harrison, M.Aagaard, “Fast Tactic-based Theorem Proving,” Theorem Proving in Higher Order Logics:13th International Conference, LNCS1869, pp.252–266, 2000.

[14] 松石航也，服部哲，岡野浩三，東野輝夫，谷口健一，“共有メモリ型並列計算機上での正則な項書換え系の一実装法”，電子情報通信学会論文誌，Vol.J81-DI, No.1, pp.28–37, 1998.

[15] “The Omega Project, Frameworks and Algorithms for the Analysis and Transformation of Scienctific Programs,”http://www.cs.umd.edu/project/omega/

[16] “Objective Caml,”http://pauillac.inria.fr/ocaml/.

[17] M.J.C.Gordon, T.F.Melham, “Introduction to HOL: a theorem proving environment for higher order logic,” Cambridge University Press, New York, NY, 1993.

[18] B. Meyer, “Object-Oriented Software Construction,” Prentice-Hall, 1988.

[19] 谷口健一，北道淳司，“LSI設計教育と計算機ハードウェア教育のためのマイクロプロセッサ：KUE-CHIP 2”，電子情報通信学会技術報告，VLD95-92, pp.45–52, 1995.

[20] 北嶋暁，森岡澄夫，島谷肇，東野輝夫，谷口健一，“代数的手法を用いたCPU KUE-CHIP2の段階的設計の正しさの自動証明”，電子情報通信学会論文誌，Vol.J79-D-I, No.12, pp.1017–1029, 1996.