System Development Paper

環境部を必要としないMizar字句・構文解析器の開発

Development of Mizar Lexer and Parser Working without Environment Part (Work in Progress)

中正 和久 [1,*]
Kazuhisa Nakasho[1,*]

1 山口大学大学院創成科学研究科, 山口県宇部市常盤台 2-16-1
1 Graduated School of Science and Technology for Innovation, Yamaguchi University, 2-16-1 Tokiwadai, Ube, Japan

* nakasho@yamaguchi-u.ac.jp

Received: November 9, 2018. Revised: May 17, 2019. Accepted: June 14, 2019.

Abstract

As preparation for development of the theorem search engine, the author is developing Mizar lexer and parser that work without inputting the environment part of Mizar language. In this paper, we outline the requirements and design of this program, and report the current implementation status.

1 緒言

筆者は Mizar 数学ライブラリ (MML) 上で動作する新たな定理検索システムの開発準備として，環境部を入力せずとも動作する Mizar 言語の字句・構文解析器の開発を進めている．現在，MML の定理検索システムとしては，2001 年に Grzegorz Bancerek 氏によって開発された MML Query [1] が広く利用されている．MML Query は形式化ライブラリ検索システムの先駆けであるとともに，現在においても形式化ライブラリ内の定理を包括的に検索することができる唯一稼働中のシステムである [2]．MML Query は高度なパターンマッチングを実現するために検索対象を記述するための独自言語を利用している．このため，利用者はその文法を習得するために相当量の学習時間を費やすとともに，その言語を使って巧みに照合条件を記述する必要がある．今回新たに開発する定理検索エンジンでは，検索システムの利用を容易化するために，Mizar 言語で記述された検索対象からシンボルや変数の型がどのような木構造で結合しているかを特徴量として抽出し機械学習型の検索エンジンへ渡す，という処理フローを計画している．図 1 は新たな定理検索システムの概略図である．開発中の字句・構文解析器は図 1 の * の箇所に相当する．

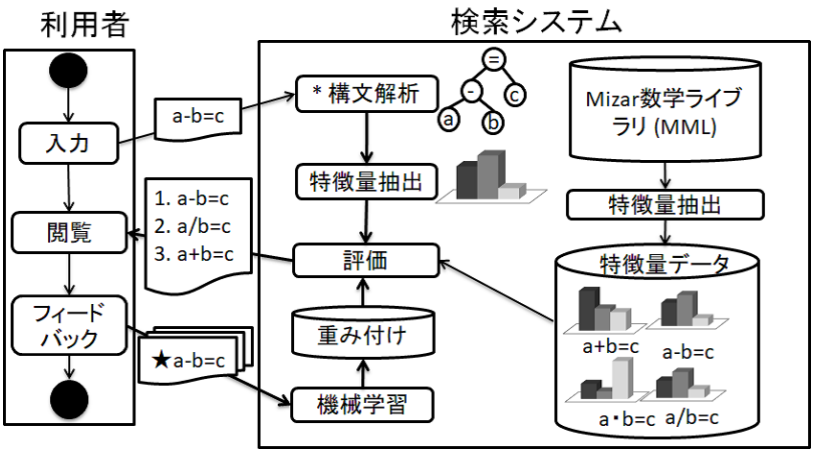

図 1. 新たな定理検索システムの概略図

Mizar 言語は文脈依存文法で構成されており，本文内のシンボル定義が環境部に依存している [3]．しかし，環境部を正確に入力することは Mizar 言語に精通したライブラリ開発者にとっても手間のかかる作業である．このため，検索システムの利用者に環境部の入力を強いることは好ましくない．本プロジェクトでは，環境部を必要とせずに定義・定理等の記述を解析する字句・構文解析器の開発を目標としている．

2 字句・構文解析器の要件

現在，Mizar の実行環境で運用されている字句・構文解析器は，まずファイル中の環境部を解析して文脈自由な WSM 形式に変換する．その後，パーサジェネレータ Bison によって生成されたプログラムによって構文木構造を XML 中間表現へ変換する [4,5]．

今回のプログラムでは環境部の入力を不要とするために，MML に登録されている全てのシンボルが有効であるとして，最長一致による字句解析を行うこととした．このため，本来はシンボルではないトークンがシンボルとして誤認識される可能性がある．しかし，let, reserve などの特定のキーワードの直後に位置するトークンは変数である等の解釈を字句解析器に盛り込むことにより，実用上問題のない程度に誤認識を低減できるものと見込んでいる．

Bison はボトムアップ型の LALR 法を採用しており，一般的にトップダウン型のパーサジェネレータに比べて効率的でサイズも小さい構文解析器を生成する．しかし，ボトムアップ型の構文解析器は，生成されたプログラム中から読み取り文の木構造を類推することが難しい．このため，文法に誤りのある入力が与えられたときに適切なエラーメッセージを出力させることが難しいという欠点がある．Mizar Verifier もまた，初学者にとっては理解が難しい文法エラーを出力する傾向にある．これらの理由および計算機の性能向上を背景として，近年は LL 法やパックラットパーサなどトップダウン型のパーサが利用される事例が増えてきた．今回の開発でも利用者側の利便性を図るために，パーサージェネレータにはトップダウン型（Adaptive LL(*) 型）の ANTLR [6] を採用した．ANTLR は Java, C++, Python, Go, Swift, JavaScript, C# など多くの言語出力に対応しており，

他言語へも移行しやすい．特に，JavaScript にも対応しているため，今後 Web ブラウザや Atom, Visual Studio Code などの各種エディタ上で文法チェックを行うツールの開発に役立つものと期待できる．今回開発では出力言語に Python3 を採用した．

本プロジェクトでは，完全な字句・構文解析器の構築を目指していない．検索システムの利用者にとっては，ストレスなく定理・定義等を入力することができて，構文解析結果に誤りがあれば適切かつ容易に修正することができれば十分である．このため，本プログラムを利用するシステム側においては，入力がどのように構文解析されたかを利用者が確認できることが重要となる．検索システムの開発では，表示に HTML を利用して，構文に誤りがある箇所をハイライトする，シンボルへのハイパーリンクを張る，などの実装を施し，構文解釈結果が利用者側からも閲覧できるようにする予定である．

3 入出力と処理の流れ

ここでは，この字句・構文解析プログラムの入出力および処理の流れについて概説する．図 2 は字句・構文解析器の処理の流れ図である．

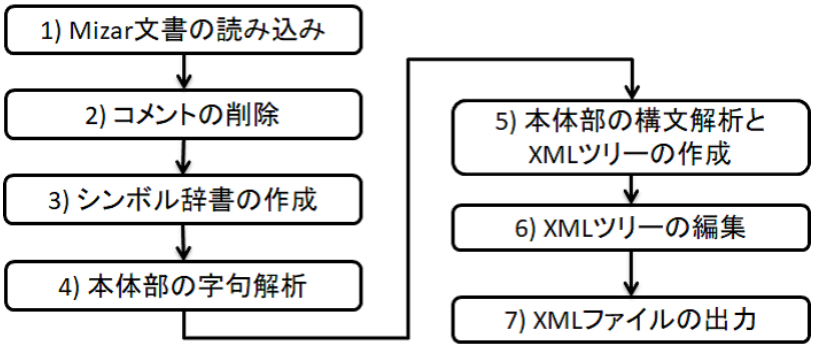

図 2. 字句・構文解析器の処理の流れ

3.1 入出力

入力対象は，Mizar 言語で書かれた theorem, definition, registration, notation, scheme の宣言部を対象とする．定理検索システムでは利用者が証明部を入力する必要はないため，証明部の構文解析は今回プロジェクトのスコープからは除外した．

本プログラムの出力は，既存の Mizar 構文解析器と同様に XML 中間表現を予定している．これは，本プログラムをテストするにあたって既存プログラムとの実行結果の比較を容易にするためである．

3.2 字句解析の流れ

字句解析では MML で定義されている全てのシンボルが有効であるものとみなす．字句解析の前段階では，Mizar 配布バイナリに付属している “mml.vct” ファイルからシン

ボル定義を抽出して辞書を作成している．

字句解析器はまず最初にコメントを削除する．次に前のほうから最長一致原則でシンボル，キーワード，数字，識別子等のトークンを読み取る．トークンが辞書に登録されたシンボルに合致したときは，接頭詞 「“__” + “シンボルの種類” + “シンボルの優先度” + “_”」を付与する．Mizar 言語に含まれるシンボル記号とその種類の対応関係は表 1 を参照されたい．

表 1. シンボル記号とその種類の対応関係

記号	種類
R	Predicate
O	Functor
M	Mode
G	Structure
U	Selector
V	Attribute
K	Left Functor Bracket
L	Right Functor Bracket

字句解析処理の最後に，本プログラムは解析結果をテキストファイルに書き出す．トークンの区切りはスペースまたは改行文字によって表現されている．このため，ANTLR の文法ファイルで字句解析ルールを記述することは容易である．

3.3 構文解析の流れ

Mizar 言語の構文定義は，配布バイナリに含まれている BNF 記述ファイル “doc/syntax.txt” を ANTLR へ渡す文法ファイルに変換している．ANTLR によって生成された構文解析器の雛形に，既存の Mizar 構文解析器が出力する XML 中間表現を出力するコードを追加する予定である．XML 中間表現の出力の前に，字句解析時に付与したシンボルの接頭詞は除去する．

4 実装状況

この節では，本プログラムの実装について，現在の進捗状況を報告する．

4.1 字句解析

let や reserve などのキーワードの直後の変数に対する例外処理以外の作業は完了した．ただし，シンボルの最長一致をとる箇所は簡易実装であるため，今後パフォーマンスを計測した上で必要であればパトリシア木を使うように修正を施したい．

4.2 構文解析

ANTLR へ渡す文法ファイルを作成し，2 ケース (Listing1,2) について正常に構文解析が動作することを確認した．

Listing 1. Theorem1

```
theorem
  for T being Noetherian sup−Semilattice for I being Ideal of T
  holds ex_sup_of I, T & sup I in I;
```

Listing 2. Theorem2

```
theorem
  ((for r,s,t holds (r ∗ s) ∗ t = r ∗ (s ∗ t)) & ex t st for s1 holds s1
  ∗ t = s1 & t ∗ s1 = s1 & ex s2 st s1 ∗ s2 = t & s2 ∗ s1 = t) implies S is Group;
```

通常，LL 法の構文解析器では左再帰除去が必要となるが，ANTLR は自動的に直接左再帰を除去する．このため，構文の修正が必要となるのは間接左再帰のみである．Mizar の BNF 記述ファイルにおいて除去が必要であった間接左再帰は，2 箇所 (Listing 3,4) であった．

Listing 3. BNF1

```
Type−Expression = "(" Radix−Type ")"
  | Adjective−Cluster Type−Expression
  | Radix−Type .
```

Listing 4. BNF2

```
Term−Expression = "(" Term−Expression ")"
  | [ Arguments ] Functor−Symbol [ Arguments ]
  | Left−Functor−Bracket Term−Expression−List Right−Functor−Bracket
  | Functor−Identifier "(" [ Term−Expression−List ] ")"
  | Structure−Symbol "(#" Term−Expression−List "#)"
  | "the" Structure−Symbol "of" Term−Expression
  | Variable−Identifier
  | "{" Term−Expression { Postqualification } ":" Sentence "}"
  | "the" "set" "of" "all" Term−Expression { Postqualification }
  | Numeral | Term−Expression "qua" Type−Expression
  | "the" Selector−Symbol "of" Term−Expression
  | "the" Selector−Symbol
  | "the" Type−Expression
  | Private−Definition−Parameter
  | "it" .
```

5 残課題

現時点で判明している残課題は以下の通りである．

5.1 シンボル判定の例外処理

最長一致によってシンボルに合致するトークンを選ぶだけでは，変数を表す識別子がシンボルと誤って解釈される可能性がある．今後，誤解釈の頻度がどの程度であるかを調査した上で，let や reserve などある種のキーワードの直後にあるトークンを変数識別子とみなすなどの例外処理を字句解析器に追加する．

5.2 演算子の優先順位

Mizar 言語には，開発者が演算子 (Functor) を自由に追加し，それらに優先順位を付与できるという他言語にはない柔軟性がある．現時点では，ANTLR へ渡す文法ファイルを修正することにより演算子の優先順位を構文解析器に反映させる予定である．既に字句解析フェーズにおいて，優先順位はシンボルの接頭詞に付与されている．ANTLR はこれらを読み取ることにより，シンボルの優先順位を反映した木構造を出力することができる．

5.3 XML 中間表現の出力とテスト

本プロジェクトで構築する構文解析器からは，既存の Mizar 構文解析器と同等の XML 中間表現の出力を予定している．既存の Mizar 構文解析器および本プロジェクトの構文解析器によって出力されるの XML 中間表現を比較することによって本プログラムのテストを実施し，一致率の向上を目指す．テストケースには，MML に収録されている theorem, definition, registration, notation, scheme を利用する計画である．

5.4 型検査

Mizar は多重定義および型 (Mode) 継承が可能なため，検索システムの開発時には，Java や C++と同等の型検査を実装する必要がある．

6 結言

本稿では，新たな定理検索システムを構築する準備段階として開発中の環境部を必要としない Mizar 字句・構文解析器について紹介した．現在はプロトタイプ作成が終わって開発の見通しがついた段階で，年度内の完成を目指している．今後は細部を詰めつつ，構文解析精度の向上に努めたい．

Mizar 言語は文法が複雑なため字句・構文解析器の開発が難しく，しばしば周辺ツールの開発におけるボトルネックとなってきた [7]．今回の字句・構文解析器の開発が，このような事情を打開する新たな突破口となることを期待している．

本開発の次に予定している定理検索システムは，来年度前半での完成を目指している．この定理検索システムは，検索エンジンの機械学習用コーパスデータの収集を目的の一つとしており，定理検索結果をデータベースに蓄積する計画である．ここに定理検索システムの利用について，Mizar プロジェクト参加者各位にご協力を賜りたくお願い申し上げる次第である．

Acknowledgments

本研究にあたって，既存の Mizar 字句・構文解析器の仕様解説ならびにソースコード提供を快く引き受けてくださった Adam Naumowicz 先生，Artur Korniłowicz 先生，Radosław Piliszek 氏にこの場をお借りして感謝の意を表する．

参考文献

[1] Bancerek G. Information Retrieval and Rendering with MML Query. In: Borwein JM, Farmer WM, editors. Mathematical Knowledge Management. Berlin, Heidelberg: Springer Berlin Heidelberg; 2006. p. 266–279.

[2] Guidi F, Sacerdoti Coen C. A Survey on Retrieval of Mathematical Knowledge. Mathematics in Computer Science. 2016 Dec;10(4):409–427. Available from: https://doi.org/10.1007/s11786-016-0274-0.

[3] Cairns P, Gow J. Using and parsing the Mizar language. Electronic Notes in Theoretical Computer Science. 2004;93:60–69.

[4] Bylinski C, Alama J. New Developments in Parsing Mizar. In: Jeuring J, Campbell JA, Carette J, Dos Reis G, Sojka P, Wenzel M, et al., editors. Intelligent Computer Mathematics. Berlin, Heidelberg: Springer Berlin Heidelberg; 2012. p. 427–431.

[5] Naumowicz A, Piliszek R. Accessing the Mizar Library with a Weakly Strict Mizar Parser. In: Kohlhase M, Johansson M, Miller B, de Moura L, Tompa F, editors. Intelligent Computer Mathematics. Cham: Springer International Publishing; 2016. p. 77–82.

[6] Parr T, Harwell S, Fisher K; ACM. Adaptive LL (*) parsing: the power of dynamic analysis. ACM SIGPLAN Notices. 2014;49(10):579–598.

[7] Cairns P, Gow J. Integrating Searching and Authoring in Mizar. Journal of Automated Reasoning. 2007 Aug;39(2):141–160. Available from: https://doi.org/10.1007/s10817-007-9073-2.