解析表現文法を対象とした構文ファイルのチェックツール Tamias の開発

宮地　俊宏 [1)]・片山　徹郎 [2)]

Tamias: a Syntax File Checker for Parsing Expression Grammar

Toshihiro MIYAJI, Tetsuro KATAYAMA

Abstract

Parsing Expression Grammar (PEG) proposed by Bryan Ford has the higher expressive ability than traditional Backus-Naur Form (BNF). PEG is deterministic and has no ambiguity. However, PEG has problems such as “Prefix Capture”. “Prefix Capture” is a problem of hiding the language to be accepted according to the order of choice. In addition, when confirming the behavior of the parser, it is possible to check only the top level non-terminal symbols, and it is necessary to rebuild the parser for each change in the syntax file. To support checking syntax files including such mistakes, this paper proposes Tamias: a syntax file checker to support checking the PEG syntax files. Tamias has PEG interpreter which can check production rules of PEG. By using the PEG interpreter, Tamias can verify the behavior of production rules and measure the reach rate. By measuring the reach rate, Tamias can prove that Prefix Capture has not occurred for any input string.

Keywords: Syntax Analysis, Parser, Parsing Expression Grammar, Packrat Parsing

1.　はじめに

Chomsky 階層における 2 型文法の文脈自由文法を表現する記法として BNF(Backus-Naur Form)がある [1)]。BNF はプログラミング言語の構文定義において伝統的に使用されているが、Dangling-else 問題を代表とするあいまいな文法を記述する可能性がある。あいまいな文法は、1 つの文字列に対する解釈が複数できることからプログラミング言語の仕様では許されない。さらに任意の文脈自由文法に対して、文法があいまいであるかを判定するアルゴリズムは存在しないことが証明されている [2,3)]。

近年、あいまいでない文法を生成できる解析表現文法(Parsing Expression Grammar: PEG)が Bryan Ford によって提唱された [4)]。解析表現文法は、選択に順序を付けた順序選択が特長の 1 つである。順序選択は、高々1 つの式に解析が成功するため、複数の解釈が生まれない。しかしながら、順序選択はあいまい性の排除は可能だが、Prefix Capture という新たな問題が生じる。Prefix Capture とは、選択肢の順序によって受理すべき言語を隠蔽する問題である。さらに、解析表現文法に対して文法ミスのチェックまたは受理すべき言語を受理できるかといった検証は、構文解析器ジェネレータで出力した構文解析器で行なう [5)]。しかし従来の方法では、解析表現文法の生成規則を変更したとき、最初から構文解析器を再度生成する必要があり手間がかかってしまうという問題がある。

そこで本研究では、解析表現文法における構文ファイルのチェックの支援を目的として、構文ファイルのチェックツール Tamias を開発する。

1) 工学専攻 機械・情報系コース 大学院生

2) 情報システム工学科 教授

2.　解析表現文法

2.1 定義

解析表現文法Gは、次の 4-タプルで表せる。

$$G = (V_N, V_T, R, e_S)$$

ここで、V_Nは非終端記号(Non-terminal symbol)の有限集合、V_Tは終端記号(Terminal symbol)の有限集合である$(V_N \cap V_T = \phi)$。Rは生成規則(Production rule)の有限集合で、V_Nに属する要素Aと解析表現eを用いて$A = e$の形で記述する [6)]。解析表現とは、入力文字列に対する命令であり、終端記号または非終端記号を用いて記述する。また、解析表現には演算子を記述できる [6)]。入力文字列sを解析表現eに適用させたとき、eはsの部分文字列と照合して”成功”もしくは”失敗”を示す。sには読み取り位置を指すポインタpがあり、eが成功であれば、sの一致した部分文字列の数だけpを進める。

解析表現eが連接$e_1\ e_2$で表現されている場合、すべての解析表現e_1, e_2が解析成功であればpを進めて成功を示し、1 つでも解析に失敗すればpを解析開始位置まで戻す。

解析表現eが順序選択e_1/e_2で表現されている場合、e_1の解析が成功すればpを進めて成功を示し、e_1が失敗すればe_2の解析を始める。

よって、pはsの先頭から末尾まで移動する。sの解析がすべて成功した場合、pはsの末尾を示す。このときsを受理したという。もし、解析が途中で失敗した場合は、sを拒否したという。また、e_Sとは、Rに属する最上位の生成

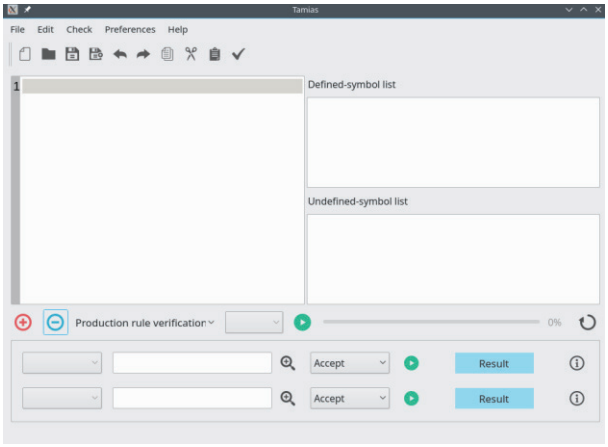

図 1. Tamias のユーザインタフェース.

規則の解析表現であり、開始表現(Start expression)と呼ばれる。

2.2 Prefix Capture

Prefix Capture とは、選択肢の順序によって受理すべき言語を隠蔽する問題である。例えば、生成規則$A = 'a'/'aa'$は、文字列aaを受理しない。これは、文字列aaの部分文字列aが 1 つ目の選択肢に照合するからである。この生成規則における問題は、選択肢aとaaの順序を入れ替えることで簡単に解決する。しかしながら、順序選択の選択肢が複数の非終端記号を導出する場合には、すべての入力文字列に関して Prefix Capture が起こっていないことを証明することは、非常に困難となる。

3. Tamias

本研究で開発した Tamias は、解析表現文法で記述した構文ファイルのチェックツールである。Tamias のユーザインタフェースを、図 1.に示す。Tamias は、生成規則エディタ部、シンボルリスト部、生成規則検証部の 3 つの領域からなる。

- 生成規則エディタ部
  構文ファイルの読み込みおよび記述が可能なエリア。生成規則エディタ部の各行に生成規則を記述する。記述した生成規則は、Tamias 内部のデータ構造へ格納する(4.1 節参照)。
- シンボルリスト部
  生成規則エディタ部で記述している解析表現文法の非終端記号を表示するエリア。ここで、表示する非終端記号を以下の 3 つに分類する。
  - ➢ 定義済シンボル(Defined-symbol)
    非終端記号をA、解析表現をeとしたときの生成規則$A = e$のAを定義済シンボルと呼ぶ。
  - ➢ 未定義シンボル(Undefined-symbol)
    非終端記号をA、解析表現をeとしたときの生成規則$A = e$において、eに含まれる非終端記号の集合から定義済シンボルの集合を引いた集合の要素を、未定義シンボルと呼ぶ。
  - ➢ テスト可能シンボル(Testable-symbol)
    非終端記号をA、解析表現をeとしたときの生成規則$A = e$においてAが終端記号を導出できるとき、Aをテスト可能シンボルと呼ぶ。テスト可能シンボルの集合は、定義済シンボルの集合の部分集合である。
- 生成規則検証部
  生成規則エディタ部で記述している解析表現文法の生成規則に対して、PEG インタプリタを用いて動的に検証するエリアである。生成規則検証部で使用できる機能は以下の 2 つである。
  - ➢ 生成規則の動作検証
    テスト可能シンボルを左辺にもつ生成規則に対して、ユーザが指定した入力文字列を受理するか拒否するかをテストできる。
  - ➢ 生成規則の到達率測定
    テスト可能シンボルを左辺にもつ生成規則の解析表現に含まれる順序選択の到達率(Reach rate)を測定する。到達率の測定は、1 つの生成規則に対して複数の動作検証を実行することで求めることができる。到達率は、次の計算式から求める：

$$\text{到達率} = \frac{\text{解析成功した選択肢の個数}}{\text{順序選択の選択肢の個数}} \times 100.$$

生成規則の検証に必要な要素は、テスト可能シンボル、入力文字列、期待出力の 3 つである。生成規則の検証は、テスト可能シンボルおよび入力文字列による PEG インタプリタの解析結果と期待出力を比較する。期待出力には、受理(Accept)または拒否(Reject)を指定できる。

4. Tamias の実装

4.1 Tamias で用いるデータ構造

Tamias では、生成規則エディタ部の生成規則を内部のデータ構造へ格納している。Tamias 内部のデータ構造は 2 種類であり、以下に示す連想配列である。

- 行番号格納配列
  「行番号」と「非終端記号」の組を複数もつ連想配列である。ここで連想配列の値となる非終端記号は、生成規則エディタ部で記述した生成規則の左辺である。
- 生成規則格納配列
  「非終端記号」と「非終端記号列」の組を複数もつ連想配列である。ここで連想配列のキーとなる非終端記号は、生成規則エディタ部で記述したの左辺である。連想配列の値となる非終端記号列は、生成規則の右辺にある非終端記号を、順序選択による選択肢ごとに分割した記号列である。

4.2 PEG インタプリタ

PEG インタプリタとは、本研究で開発した Tamias 内部で動作する解析表現文法を対象としたインタプリタである。PEG インタプリタの入力は、生成規則検証部でユーザが指定したテスト可能シンボルと入力文字列の 2 つである。出力は、入力で受け取った非終端記号における解析表現の出力である。PEG インタプリタは、常に最上位の非終端記号から解析を始める再帰的下向き構文解析器とは異なり、開始表現をユーザが指定できることが特長の 1 つである。また、構文解析手法に Packrat 構文解析 [7] を採用しているため、解析時間は入力文字列の長さに関して線形に保たれる。Packrat 構文解析手法では、前回の読み取り位置を保存することで、解析の再評価を省略できる。

PEG インタプリタでは、以下の 3 種類のデータを用いる。

- 解析表現文法
  生成規則エディタ部でユーザが記述した生成規則の一部を分割し、「非終端記号」と「解析表現」の組として表現した連想配列である。ここで、連想配列のキーとなる非終端記号は、テスト可能シンボルが導出する非終端記号のすべてである。
- メモ化テーブル
  構文解析で必要となる非終端記号に関する読み取り位置を格納した連想配列である。メモ化テーブルの各行は、解析処理に対応しており、各列は入力文字列の読み取り位置に対応している。メモ化テーブルを参照することによって、すでに結果を格納している読み取り位置では、保存した値を用いて再び解析する処理を省くことができる。
- 入力文字列
  文字列と読み取り位置を表すポインタを要素にもつデータである。文字列は、生成規則検証部において、ユーザが記述した文字列である。

4.3 PEG インタプリタのアルゴリズム

PEG インタプリタのアルゴリズムを、以下に示す。以下のアルゴリズムは、ユーザが指定したテスト可能シンボルを起点として、終端文字を処理するまで再帰的に実行する。

1. 入力として受け取ったテスト可能シンボルが、メモ化テーブルに保存しているかどうかを調べる。
2. 1.で保存していれば、格納している読み取り位置まで入力文字列のポインタを進めて、終了する。
3. 1.で保存していなければ、非終端記号に対応する解析表現を取得する。取得した解析表現の選択肢ごとに 4.の処理を繰り返す。もし取得した選択肢が 1 つでも成功すれば、入力文字列のポインタを進めて終了する。
4. 入力として受け取った連接に含まれる解析表現ごとに 5.を実行する。実行した解析表現に対する 5.の処理が 1 つでも失敗すれば 4.を実行開始したときの読み取り位置までポインタを戻して終了する。
5. 入力として受け取った解析表現に対して演算子ごとに構文解析処理を分ける。もし受け取った解析表現に演算子がなければ、その非終端記号を入力として 1.を実行する。

4.4 生成規則の動作検証アルゴリズム

Tamias の生成規則検証部で生成規則の動作検証アルゴリズムを、以下に示す。

1. ユーザが選択したテスト可能シンボルを取得する。
2. ユーザが指定した入力文字列を取得する。
3. ユーザが選択した期待出力を取得する。
4. 1.で取得したテスト可能シンボルと 2.で取得した入力文字列から PEG インタプリタを実行する。

4.5 到達率測定のアルゴリズム

Tamias の生成規則検証部で到達率を測定するアルゴリズムを、以下に示す。到達率の測定は、1 つのテスト可能シンボルにおける生成規則を複数回、動作検証することで求めることができる。よって、以下の 1.から 4.は各々の動作検証に対して実行する。

1. ユーザが選択したテスト可能シンボルを取得する。
2. ユーザが指定した入力文字列を取得する。
3. ユーザが選択した期待出力を取得する。
4. 1.で取得したテスト可能シンボルと 2.で取得した入力文字列から PEG インタプリタを実行する。
5. 生成規則格納配列において 1.で取得したテスト可能シンボルに対応する解析表現から、順序選択の選択肢の個数を取得する。PEG インタプリタの実行結果から、解析成功となった個数を取得する。取得した 2 つの値から以下の計算式に従って到達率を計算する。

$$\text{到達率} = \frac{\text{解析成功した選択肢の個数}}{\text{順序選択の選択肢の個数}} \times 100\ [\%]$$

5. Tamias の動作実験と評価

本研究で開発した Tamias が正しく動作することを検証するため、以下に示す 2 点について Tamias の生成規則の動作検証機能および到達率の測定機能に対して確認する。

1. 誤りを含む解析表現文法を適用し、正しくエラー報告もしくは動作検証ができることを確認する。
2. 誤りを含む解析表現文法を正しい解析表現文法に修正したのち、エラー報告をしない、または正しく動作検証できることを確認する。

5.1 生成規則の動作検証に関する実験

生成規則の動作検証に用いる、加法および乗法を表し

```
Sum      = Product '+' Product / Product
Product  = Value '*' Value / Value
Value    = [0-9]
```

図 2. 加法および乗法を表した文法.

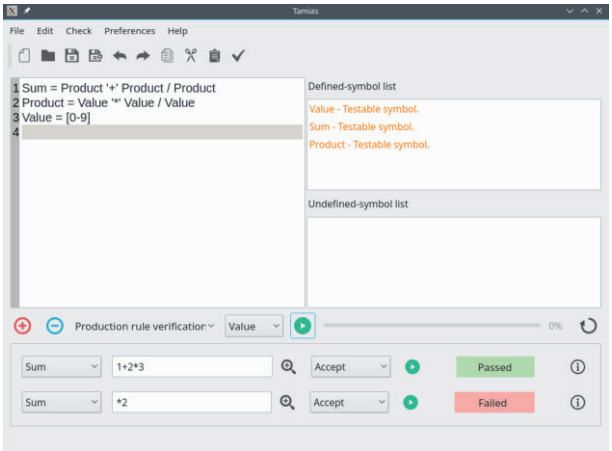

図 3. 生成規則の動作検証に関する実験結果.

た文法を、図 2.に示す。また、生成規則の動作検証結果を、図 3.に示す。

図 3.の下部にある生成規則検証部では、2 つの文字列「1+2*3」、「*2」を対象とした生成規則の検証を実行し、期待出力をともに”受理(Accept)”に設定している。

図 3.に示すように文法に沿った文字列「1+2*3」の処理では生成規則の検証が成功し、文法に沿っていない文字列「*2」では検証が失敗している。

5.2 到達率測定に関する実験

到達率測定に用いる Prefix Capture が発生する文法を、図 4.に示す。また、到達率の測定結果を図 5.に示す。

図 4.の文法に対して、生成規則検証部で到達率を測定し、Prefix Capture が発生していないことの検証を試みる。図 5.の上部から、入力文字列+i および++i において、到達率が 50%となり 100%でないことが分かる。これは、入力文字列が生成規則「Op = '+' / '++'」の 1 つの選択肢'+'に照合したからである。したがって、図 4.の文法に Prefix Capture が含まれることを確認できる。

また、図 4.の一部の生成規則を「Op = '++' / '+'」に変更し、Prefix Capture を含まない文法を適用して到達率を測定した画面を、図 5.の下部に示す。

図 5.下部から、到達率が 100%を示していることが分かる。したがって、到達率の計算式から

$$\text{到達率} = \frac{\text{解析成功した選択肢の個数}}{\text{順序選択の選択肢の個数}} \times 100 = 100$$

順序選択の選択肢の個数 = 解析成功した選択肢の個数となることから、すべての順序選択の選択肢に対して構文解析できることがいえる。そのため、任意の入力文字列に関して Prefix Capture が発生しないことを証明できる。

5.3 Tamias の評価

本節では、Tamias の生成規則の動作検証機能および到達率の測定機能に対する実験結果の考察を以下に述べる。

- 生成規則の動作検証に関する実験
  生成規則エディタ部で生成規則を記述し、動作検証を実行した。PEG インタプリタは、ユーザが選択したテスト可能シンボルを対象とした構文解析を行ない、テストを実行できた。このことから、Tamias は構文解析器を作り直すことなく、ユーザが選択した非終端記号に関して動作検証を実行できるといえる。
- 到達率測定に関する実験
  Prefix Capture が発生する文法に関して、Tamias を用いて到達率を測定した。Prefix Capture が発生する文法では、到達率は 100%とはならなかった。そこで、適用した文法を Prefix Capture が発生しない文法に変換したところ、到達率は 100%となった。到達率が 100%となる結果は、すべての入力文字列に対して順序選択の選択肢が正しく動作することを意味する。よって、Tamias は測定対象の文法に関して、到達率を測定することで Prefix Capture が発生する文法かどうかを検証することができる。

以上から Tamias は、解析表現文法を記述した構文ファイルのチェックができるといえる。

6. 関連研究

本多らは、解析表現文法の開発を支援するために PEG デバッガを提案した [8]。PEG デバッガは、解析表現文法の文法開発を動的に支援するためのデバッグ支援ツールであり、Nez パーサライブラリ [9]の一部として実装している。PEG デバッガは、独自のインタプリタ上で動作し、ブレークポイントの設置、ステップ実行、スタックトレースの表示等の機能をもつ。PEG デバッガは、1 つの入力文字列に関して動作の確認または Prefix Capture が発生しているかどうかの確認を行なうことができる。しかしながら、任意の入力文字列に関して Prefix Capture が発生していないことの証明はできない。これに対して Tamias は、PEG インタプリタを用いて動作検証を実行でき、到達率という指標から、任意の入力文字列に関して Prefix Capture が発生していないことを証明できる。

森らは、構文解析器を差分的に合成できるような構文解析器生成系を提案し、そのプロトタイプを実装した [5]。従来手法では、解析表現文法における構文の一部を変更した場合、対応した構文解析器を最初から再度作り直す必要があった。森らの提案手法では、差分的に合成される解析表現文法と対応して、合成した構文解析器を作成

```
A  = Op Id
Op = '+' / '++'
Id = [0-9]
```

図 4.　Prefix Capture を含む文法.

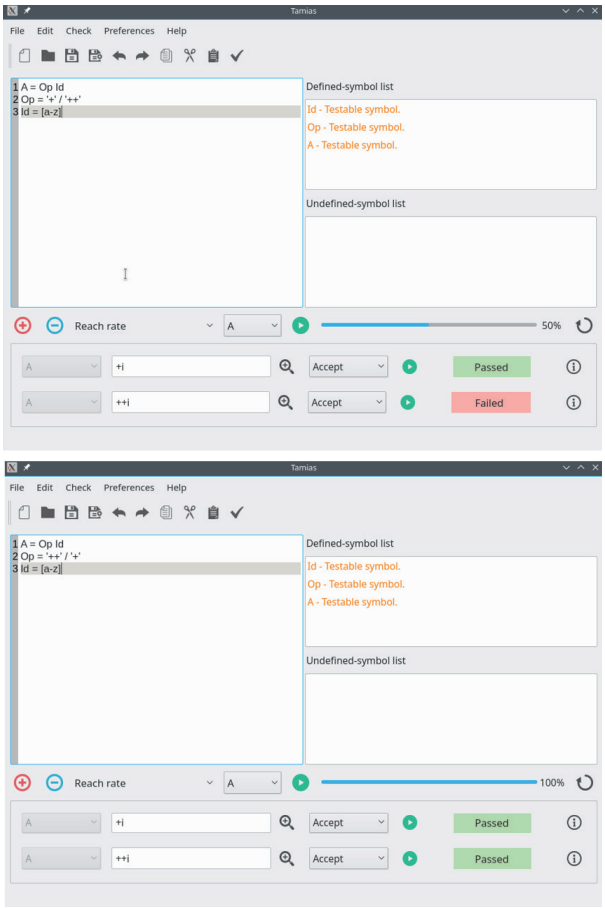

図 5.　到達率測定に関する実験結果.

できる。また、差分的に文法を合成する状況下で、既存の構文解析器よりも高速に構文解析器を生成できることを示した。Tamias は、インタプリタによって、この問題を解決している。すなわち、ユーザが生成規則エディタ部で構文の一部を変更した場合は、PEG インタプリタが解析できる非終端記号を動的に変化させることによって対応できる。

7.　おわりに

本研究では、解析表現文法を対象とした構文ファイルのチェックの支援を目的として、構文ファイルのチェックツール Tamias を開発した。

本研究で開発した Tamias は、ユーザが読み込んだ構文ファイルまたは入力した解析表現文法を静的または動的に解析し、構文ファイルのチェックを可能とするツールである。Tamias は、生成規則エディタ部、シンボルリスト部、生成規則検証部の 3 つのエリアをもつ。生成規則エディタ部では、解析表現文法における構文ファイルの読み込みおよび生成規則の記述ができる。記述した生成規則の非終端記号は、Tamias が独自に定義した 3 種類の非終端記号に分類し、シンボルリスト部に表示する。そして生成規則検証部において、シンボルリスト部で表示した非終端記号をもとに抽出した生成規則に関して動的な検証を実行できる。

Tamias が正しく構文ファイルをチェックできることを確認するために、解析表現文法を生成規則エディタ部へ記述し、実験を行なった。生成規則の動作検証と Prefix Capture が発生していないことの検証は、Tamias 内部の PEG インタプリタを実行することで正しく検証できることを確認した。以上から、Tamias は、解析表現文法を対象とした構文ファイルのチェックが可能であるといえる。

以下に、今後の課題を挙げる。

- 部分解析表現の対応
  解析表現の中に括弧を用いて解析表現を入れ子にするような生成規則は認識しない。今後、括弧を記述できるようにすることで、Tamias で動的検証できるテスト可能シンボルが増えると考えられる。
- ジェネレータ機能の実装
  現在の Tamias では、生成規則エディタ部で記述した解析表現文法から構文解析器を生成できる機能はない。ジェネレータ機能を付けることによって、十分に正しさが検証できた文法に対して構文解析器を作成することができると考えられる。
- 抽象構文木の生成機能の実装
  抽象構文木は、解析表現文法を解析した構造を木構造で表現したものである。それゆえ、抽象構文木の生成は、構文解析した結果から構造をユーザが把握しやすくなると考えられる。
- 生成規則の自動テスト機能の実装
  現在の Tamias では、任意の入力文字列に関して、Prefix Capture が起こらないことを確認する作業は手間がかかる。現状では、すべての選択肢を解析するだけの入力文字列をユーザが作成しなければならない。もし、順序選択の選択肢が複数の非終端記号を導出する場合は、すべての選択肢を解析する入力文字列を作成することは困難である。
  そこで、生成規則エディタ部に記述した終端記号から Tamias が自動的に入力文字列を生成し、生成規則の動作検証を実行するように改良する。それによって、任意の入力文字列に関して、Prefix Capture が起こっていないことを確認する手間が削減できると考えられる。

参考文献

1) John W Backus: The syntax and semantics of the proposed international algebraic language of the Zurich acm-gamm conference. Proceedings of the International Comference on Information Processing, 1959.
2) David G Cantor: On the ambiguity problem of backus systems. Journal of the ACM (JACM), Vol.9, No.4, pp.477-479, 1962.
3) Robert W Floyd: On ambiguity in phrase structure languages. Communications of the ACM, Vol.5, No.10, p.526, 1962.
4) Bryan Ford: Parsing expression grammars: a recognition-based syntactic foundation. In ACM SIGPLAN Notices, Vol.39, pp.111-122, 2004.
5) 森健輔, 脇田建: 差分的に記述された解析表現文法に対する構文解析器の合成, 日本ソフトウェア科学会大会論文集, Vol.30, pp.641-650, 2013.
6) Roman R Redziejowski: Parsing expression grammar as a primitive recursive-descent parser with backtracking. Fundamenta Informaticae, Vol.79, No.3-4, pp.513-524, 2007.
7) Bryan Ford. Packrat parsing:: simple, powerful, lazy, linear time, functional pearl. In ACM SIGPLAN Notices, Vol.37, pp.36-47, 2002.
8) 本多峻, 倉光君郎: 解析表現文法の開発支援のためのデバッガの実装と評価. 日本ソフトウェア科学会大会論文集, Vol. 32, pp. 1-6, 2015.
9) Nez: Open grammar language and tools. http://nez-peg.github.io/, (accessed 2019/02/14)