

定理証明支援系「Mizar」による 記述を補助するエディタ拡張機能 の研究

山口大学大学院 谷口 広途
山口大学 中正 和久

目次

1. 目的
2. 先行研究
3. 実装
4. 評価
5. まとめ
6. 謝辞

目次

1. 目的
2. 先行研究
3. 実装
4. 評価
5. まとめ
6. 謝辞

Mizarを利用しやすくするための
エディタ拡張機能の作成

目次

1. 目的
2. 先行研究
3. 実装
4. 評価
5. まとめ
6. 謝辞

Mizarの開発環境：Emacs for Mizar

The screenshot shows the Emacs editor interface for Mizar. The left pane displays a Mizar proof file named `test.miz`. The right pane shows the output of the Mizar verifier, including version information and a compilation error.

引用定義の参照 (Reference to cited definitions): A callout box points to the definition of `TARSKI:def 2` in the proof file, which states: `func {y, z} -> set means :: TARSKI:def 2 x in it iff x = y or x = z;`

エラー箇所へのリンク (Link to error location): A callout box points to the error message in the output: `c:/Users/w041ff/Desktop/test/test.miz:59:13: *321: Predicate symbol or "is" expected`. The error location is highlighted in red in the original image.

```

emacs@DESKTOP-3141NHJ
File Edit Options Buffers Tools Index Mizar Hide/Show Help
Th1,XBOOLE_1:3;

theorem Th3:
  for f being Function for x,y being object st f = [x,y]
  holds x = y
proof
  let f be Function, x,y be object;
  assume
  A1: f = [x,y];
  then
  A2: {x} in f by TARSKI:def 2;
  A3: {x,y} in f by A1, func {y, z} -> set means
  :: TARSKI:def 2
  x in it iff x = y or x = z;
  A4: {x} = [a,b] by A2;
  A5: {a} = {a,b} by A4,ZFMISC_1:5;
  A6: x = {a} by A4,ZFMISC_1:4;
  consider c,d being object such that
  A7: {x,y} = [c,d] by A3,RELAT_1:def 1;
  A8: x = {c} & y = {c,d} or x = {c,d} & y = {c} by A7,ZFMISC_1:6;
  then c = a by A5,A6,ZFMISC_1:4;
  hence thesis by A2,A3,A4,A5,A7,A8,FUNCT_1:def 1;
end;

theorem Th4:
  (id X)..Y c= Y
proof
  let x be object;
  assume x in (id X)..Y;
  then ex y being object st [y,x] in id X & y in Y by RELAT_1:1;
end;

```

```

Running verifier on test ...
Make Environment, Mizar Ver. 8.1.09 (Win32/FPC)
Copyright (c) 1990-2019 Association of Mizar Users

Verifier based on More Strict Mizar Processor, Mizar Ver. 8.1.09 (Win32/FPC)
Copyright (c) 1990-2019 Association of Mizar Users
Processing: c:/Users/w041ff/Desktop/test/test.miz

Parser [7321 *1] 0:00
MSM [7326 *1] 0:01
Analyzer [7326 *1] 0:06
Checker [7326 *1] 0:33
Time of mizarng: 0:40
c:/Users/w041ff/Desktop/test/test.miz:59:13: *321: Predicate symbol or "is" expected

```

-\--- test.miz 2% L74 (Mizar hs) S\%*- *compilation* All L15 (Compilation)

Mark set

Emacs for Mizarの課題

1. インストールが難しい

- 解説サイトが少ない
- ソースコードのコピペが作業が必要
- Emacsの知識が必要

2. キーバインドが独特

- 「C-c RET」「M-.」などの記法
- 一般的なショートカットキーと異なる

初心者にとってハードルが高い

他の定理証明支援系の開発環境

- Visual Studio Code上の開発環境に関心が高まっている
 - PVS,Lean,Coq等は既にサポートされている

● 実装状況

○：実装済み ×：未実装

	ベースエディタ	入力補完機能	ホバー表示機能	定義元へ飛ぶ機能	ライブ診断機能
VSCoDe-PVS	VSCoDe	○	○	○	○
Lean	VSCoDe	○	○	○	○
Isabelle/jEdit	jEdit	○	○	○	○
CoqIDE	N/A	×	×	×	×
Emacs for Mizar	Emacs	×	○	○	×

目次

1. 目的
2. 先行研究
3. 実装
4. 評価
5. まとめ
6. 謝辞

VSCode上でMizarをサポートする拡張機能を作成

実装機能

- シンタックスハイライト機能
- 自動インデント機能
- エディタ上でMizarコマンドを実行する機能
- ホバー表示機能
- 定義元へ飛ぶ機能（実装中）
- 入力補完機能（実装中）

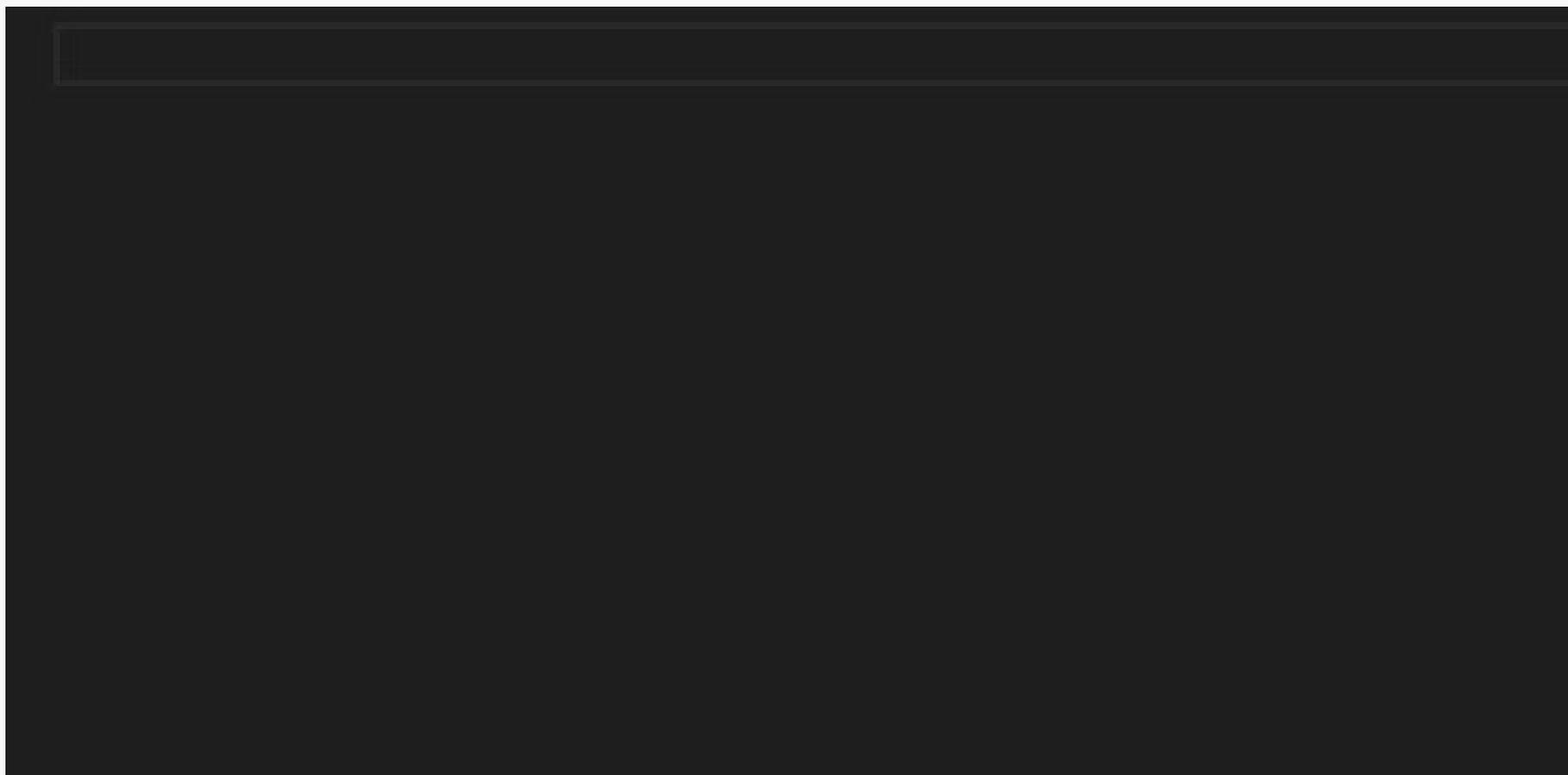
シンタックスハイライト機能



```
48 begin :: Semilattice of type widening
49
50 registration
51   cluster reflexive -> complete for 1-element RelStr;
52   coherence;
53 end;
54
55 definition
56   let T be RelStr;
57   mode type of T is Element of T;
58 end;
59
60 definition
61   let T be RelStr;
62   attr T is Noetherian means
63     :Def1:
64     the InternalRel of T is co-well_founded;
65 end;
```

```
48 begin :: Semilattice of type widening
49
50 registration
51   cluster reflexive -> complete for 1-element RelStr;
52   coherence;
53 end;
54
55 definition
56   let T be RelStr;
57   mode type of T is Element of T;
58 end;
59
60 definition
61   let T be RelStr;
62   attr T is Noetherian means
63     :Def1:
64     the InternalRel of T is co-well_founded;
65 end;
```

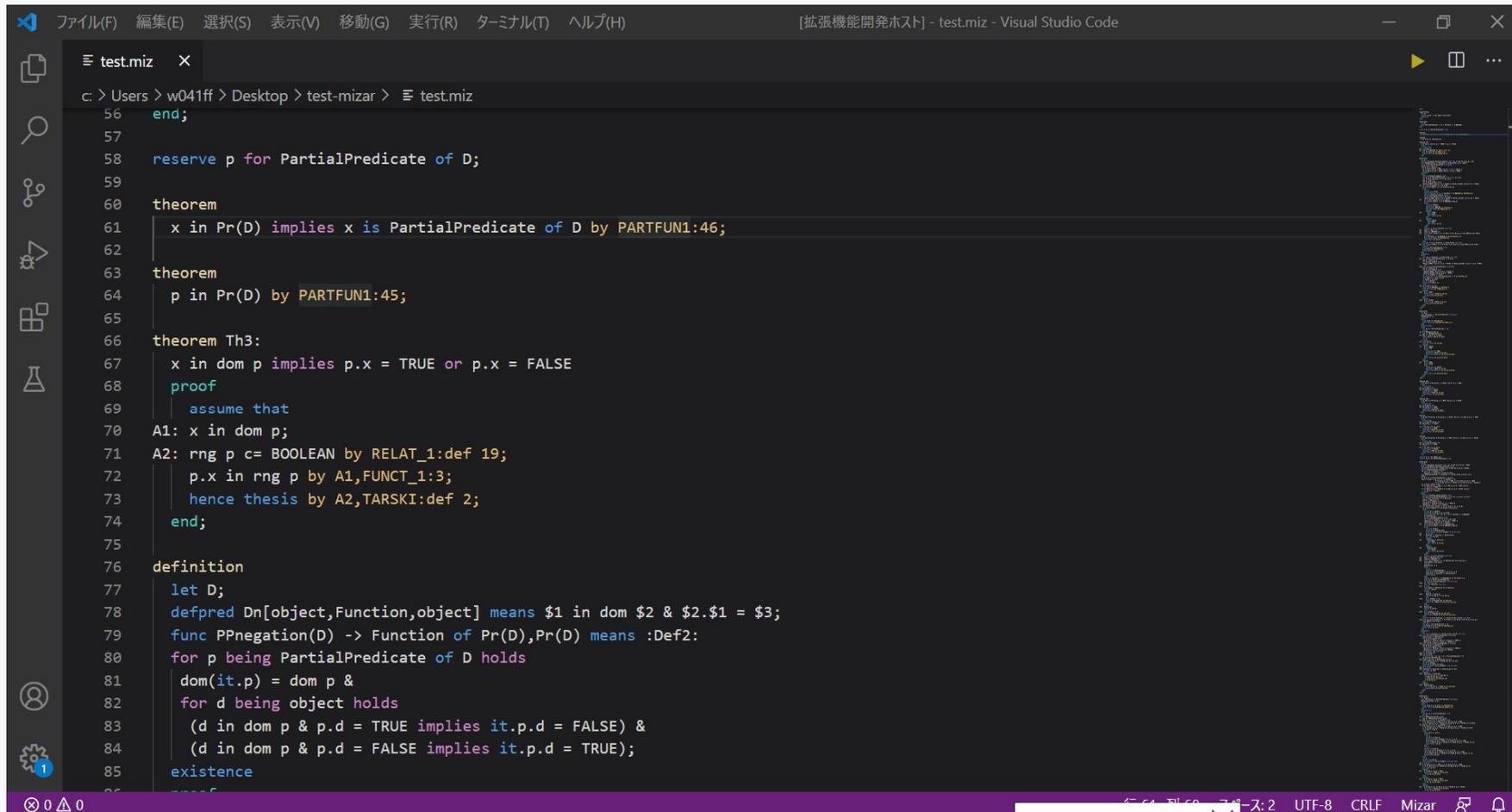
自動インデント機能



ホバー表示機能

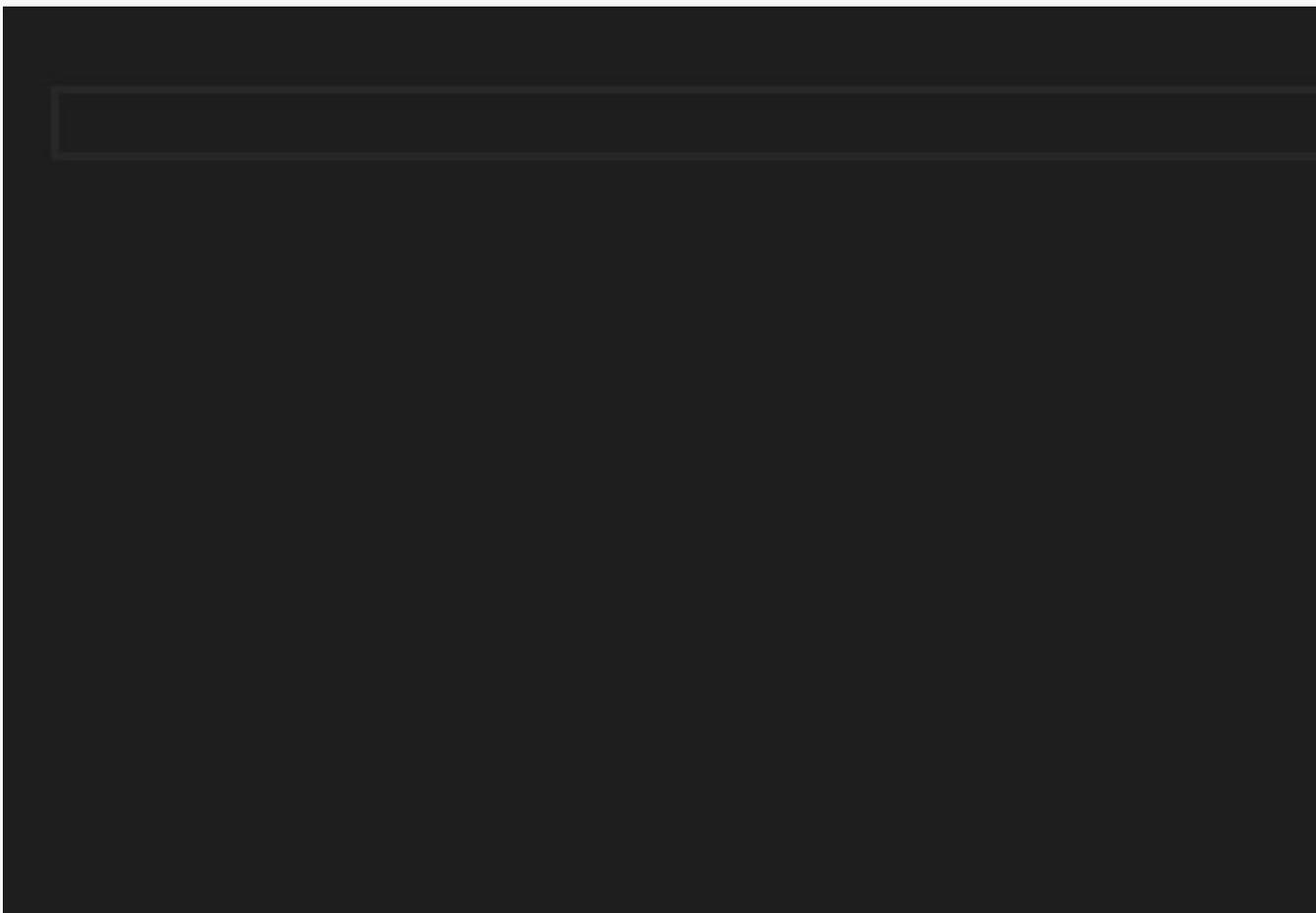
```
and
A10: for p being PartialPredicate of D holds
  dom p = dom(g.p) &
  for d being object holds
    (Dn[d,p,TRUE] implies g.p.d = FALSE) &
    (Dn[d,p,FALSE] implies g.p.d = TRUE);
  let x be Element of Pr(D);
  reconsider p = x as PartialPredicate of D by PARTFUN1:46;
  thus dom(f.x) = dom p by A9
  .= dom(g.x) by A10;
  let a be object;
  assume a in dom(f.x);
  then
A11: a in dom p by A9;
  then p.a in BOOLEAN by PARTFUN1:4;
  then per cases by TARSKI:def 2;
  suppose
A12: p.a = TRUE;
  hence f.x.a = FALSE by A9,A11
  .= g.x.a by A10,A11,A12;
```

定義元へ飛ぶ機能



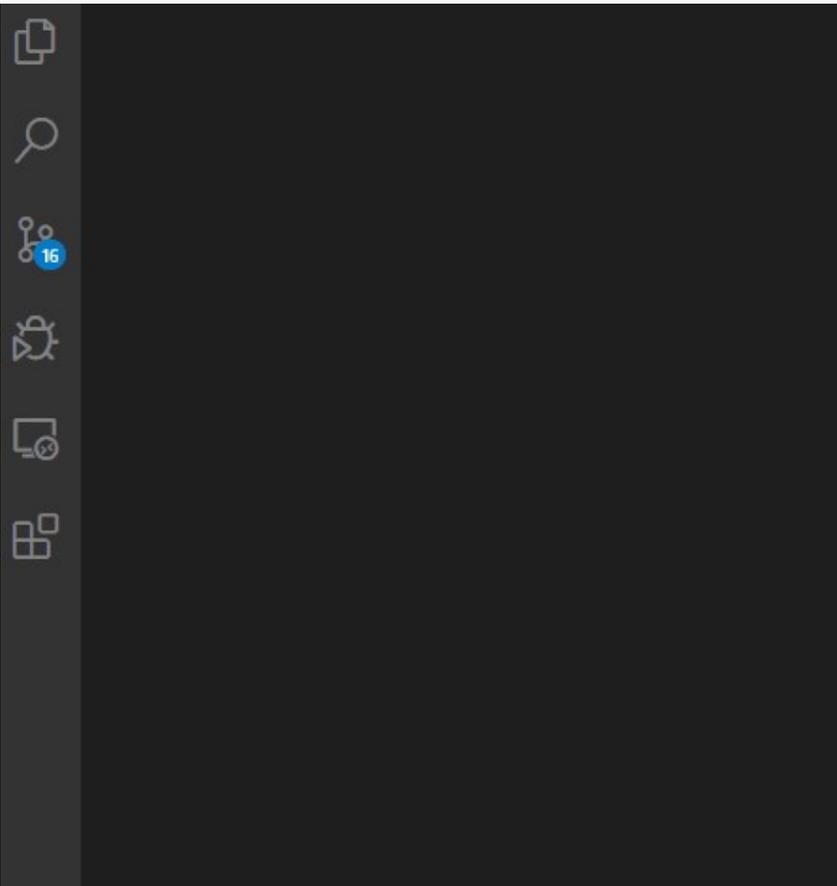
```
test.miz
c: > Users > w041ff > Desktop > test-mizar > test.miz
56 end;
57
58 reserve p for PartialPredicate of D;
59
60 theorem
61   x in Pr(D) implies x is PartialPredicate of D by PARTFUN1:46;
62
63 theorem
64   p in Pr(D) by PARTFUN1:45;
65
66 theorem Th3:
67   x in dom p implies p.x = TRUE or p.x = FALSE
68   proof
69     assume that
70 A1: x in dom p;
71 A2: rng p c= BOOLEAN by RELAT_1:def 19;
72   p.x in rng p by A1,FUNCT_1:3;
73   hence thesis by A2,TARSKI:def 2;
74   end;
75
76 definition
77   let D;
78   defpred Dn[object,Function,object] means $1 in dom $2 & $2.$1 = $3;
79   func PPNegation(D) -> Function of Pr(D),Pr(D) means :Def2:
80   for p being PartialPredicate of D holds
81     dom(it.p) = dom p &
82     for d being object holds
83       (d in dom p & p.d = TRUE implies it.p.d = FALSE) &
84       (d in dom p & p.d = FALSE implies it.p.d = TRUE);
85   existence
```

入力補完機能



インストール方法

- Visual Studio Codeを開く
- Extensionsをクリックし,
“Mizar”と検索
- “Mizar Extension”のinstallを
クリック



N-gramを用いた入力予測

- N-gram
 - N個の単語単位で、単語を切り出す手法

- 2-gram(bi-gram)の例

- 「I am a teacher」 -> 「I am」 「am a」 「a teacher」
- 「I am sleeping」 -> 「I am」 「am sleeping」
- 「I play soccer」 -> 「I play」 「play soccer」



学習（カウント）して入力予測に利用

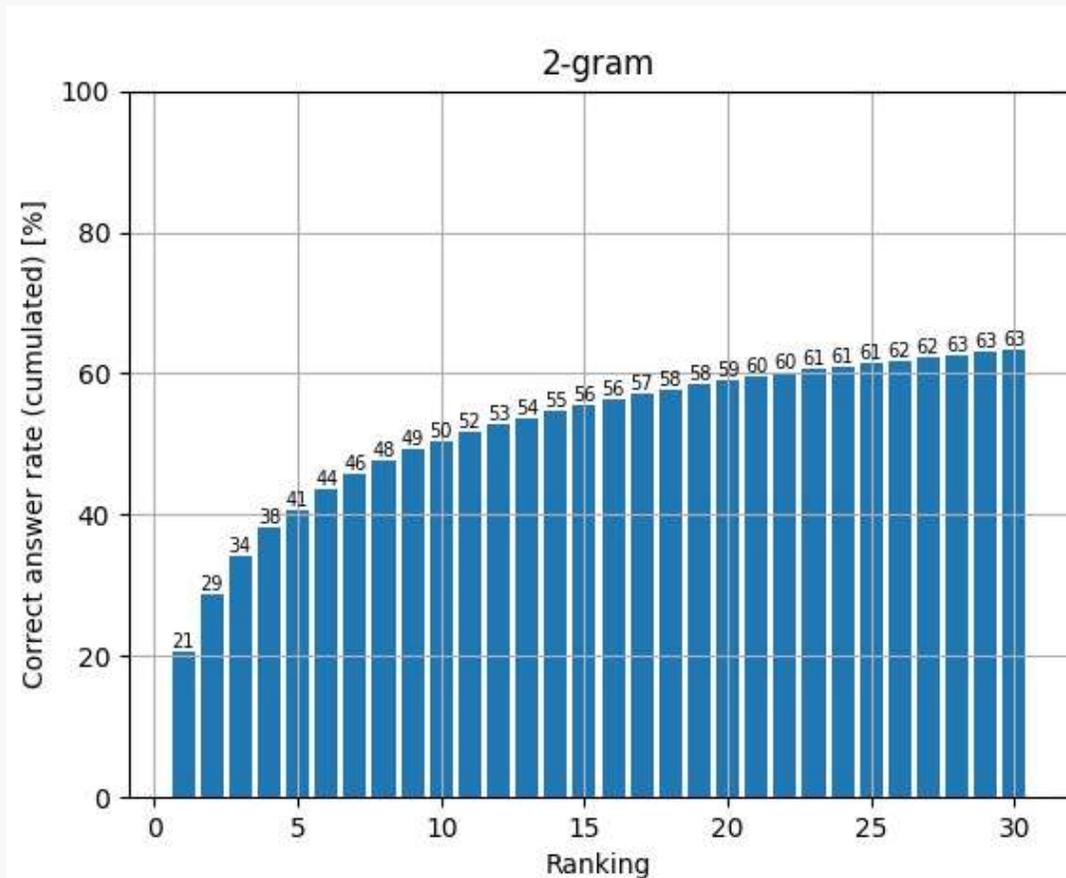
Mizarライブラリの学習と評価方法

- Mizarライブラリ1100ファイルで単語の切り出し，カウント
 - 2-gramによる切り出し
 - 3-gramによる切り出し
 - 4-gramによる切り出し
- 残りの255ファイルでそれぞれの予測精度を評価

目次

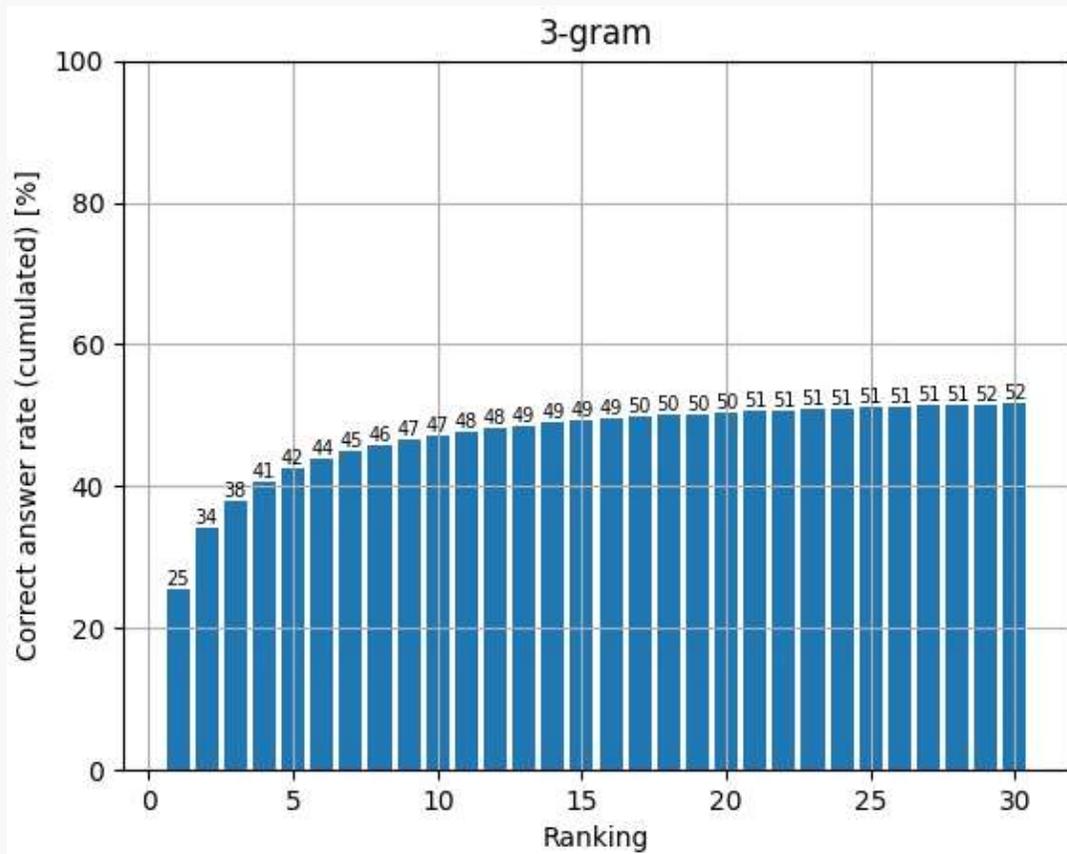
1. 目的
2. 先行研究
3. 実装
4. 評価
5. まとめ
6. 謝辞

2-gramの精度



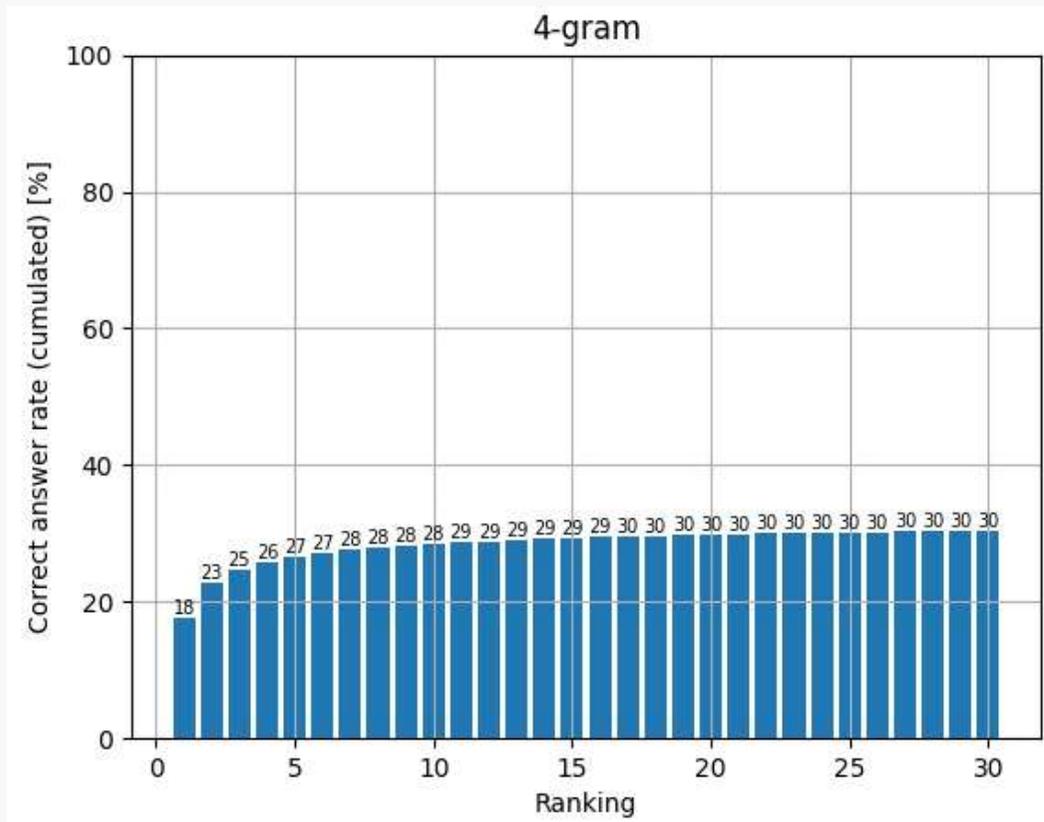
- 第1候補は21%の正解率
- 30候補以内で63%の正解率

3-gramの精度



- 第1候補は25%の正解率
 - 2-gramは21%
- 30候補以内で52%の正解率
 - 2-gramは63%

4-gramの精度



- 第1候補は18%の正解率
- 30候補以内で30%の正解率

実装状況の比較

○：実装済み
△：実装中
×：未実装

	ベースエディタ	入力補完機能	ホバー表示機能	定義元へ飛ぶ機能	ライブ診断機能
VSCoDe-PVS	VSCoDe	○	○	○	○
Lean	VSCoDe	○	○	○	○
Isabelle/jEdit	jEdit	○	○	○	○
CoqIDE	N/A	×	×	×	×
Emacs for Mizar	Emacs	×	○	○	×
本研究	VSCoDe	△	○	△	×

目次

1. 目的
2. 先行研究
3. 実装
4. 評価
5. まとめ
6. 謝辞

成果と課題

● 成果

- 「独特なショートカットキー」「インストールの手間」を解消

● 課題

- 「定義へ飛ぶ」「入力補完」機能のリリース
- 入力補完機能の精度向上
 - 変数名を型名に置き換える
 - Word2vecなど、他の機械学習の手法で挑戦

目次

1. 目的
2. 先行研究
3. 実装
4. 評価
5. まとめ
6. 謝辞

謝辞

- 本研究はJSPS科研費 JP20K19863の助成を受けたものです。

機能の詳細

- 入力補完機能：入力中にキーワードと識別子を提案する機能
- ホバー表示機能：有益なポップアップを表示する機能
- 定義元へ飛ぶ機能：識別子が定義されたファイルを開く機能
- ライブ診断機能：入力中にエラー情報を表示する機能