3O3-OS-04a-6

センター試験英語の計算文章題の自動解答器構築に向けて

Towards Building An Automatic Solver of Arithmetic Word Problems in NCTUA English Examinations

磯崎 秀樹　小村 祐介
Hideki Isozaki　Yusuke Omura

岡山県立大学
Okayama Prefectural University

「ロボットは東大に入れるか」では、センター試験を計算機に自動で解かせる取り組みを行っており、筆者は英語を担当している。最近、英語の試験に現れる計算問題に着手した。本稿では、問題を解くための Prolog のプログラムを自動生成して実行するというアイデアに基づき、まず、理想的な Prolog のプログラムを手書きで作成し実行してみた。さらに、プログラムを自動生成する方法を考察した。

1. はじめに

センター試験の英語には計算の必要な問題がある。2015 年でいえば問 4B である [*1]。この問題を日本語で要約すると、以下のようになる。

キャンプ場に関する情報を読んで以下の問いに答えよ。

- 問 1: 水遊びが好きな人が興味を持つのはどのキャンプ場か?
- 問 2: 2 人で 9 泊したい。コンピュータを使うので電気が欲しい。これに適したキャンプ場に泊まるには、1 泊あたりいくらかかるか？
- 問 3: 4 人家族が犬と一緒に宿泊したい。予算は 3 泊で 100 ドルである。バーベキューとサイクリングがしたい。この家族はどのキャンプ場を選ぶか？

このような問題を解くには、条件にマッチするものを探し出したり、四則演算を行ったりする必要がある。本稿では、このような問題を自動で解くことを考える。

算数の文章題を自動で解く研究は、昔 [安西 85, Bobrow 68] からあり、近年でも盛んに行われている [Zhou 15, Shi 15, Roy 15]。東ロボ・プロジェクトでも、(日本語の) 数学の問題に取り組んでいる [松崎 15] が、数学の試験は日本語で書かれており、幾何学・整数・数列・微積分・確率などの高校レベルの高度なものが多いのに対して、英語の試験に出てくる計算問題は、英語で書かれている点を除けば、小学生でもわかる、**制約充足と四則演算**の範囲にとどまる。

制約充足が基本的な処理になるので、本稿では以下の 2 段階の方法を採用する。

1. 英語の文章から Prolog のプログラムを自動生成する。
2. その Prolog のプログラムを実行することで解答を得る。

本稿では、第 1 段階を人手で行い、どのようなプログラムを作ればよいか、実装上どのような課題があるかを探る。英語の質問を Prolog のプログラムに変換して質問に答えるシステムとして、CHAT-80 [Warren 82] が有名である。

これまでの算数の文章題を解く研究は、四則演算の範囲にとどまり、制約充足はあまり考えられていない。このため、本稿では Prolog を採用した。Prolog は最近あまり使われていないが、複雑な条件を記述したり、条件を満たす解を見つけ出すのに最適なプログラミング言語であり、IBM の Watson でも利用されている [*2]。

2. 人手で Prolog のプログラムを作る

計算問題は、図表・説明文・問題文などがある。ここではこれらを Prolog のプログラムに変換する。

2.1 表を Prolog にする

この問題には以下のような表がある。

Campground Information

Camp-ground	Site Type (available spaces)	Site Rate/ night	Max. People	Max. Stay	Facilities	Restrictions
Apricot	Tents (15)	$20	4	15 nights	BG	—
Maple	Tents (20)	$24	5	12 nights	BG PG	—
Orange	Deluxe Cabins (5)	$96	7	7 nights	K E HS	No pets
Stonehill	Standard Cabins (10)	$32	6	14 nights	E HS	No fireworks

Site Rate=Rate per site (up to the maximum number of people); Max.=Maximum

K Kitchen, E Electricity, BG Barbecue Grill, HS Hot Shower, PG Playground

まず、この表を Prolog のプログラムに変換する。何泊・何ドルなどのように、単位のついた数字が出現するが、これらは `nights(N)`、`dollars(D)` のような表記を用いることにする。通常のプログラミング言語では、これらの表記は関数呼び出しになるが、Prolog では関数呼び出しは行われず、対応する木構造を表すだけである。

第 3 列の Site Rate/night の内容は以下のように表せる。

```
site_rate_per_night('Apricot', dollars(20)).
site_rate_per_night('Maple', dollars(24)).
site_rate_per_night('Orange', dollars(96)).
site_rate_per_night('Stonehill', dollars(32)).
```

なお、Prolog では大文字で始まる識別子が変数であり、小文字で始まる識別子が定数なので、大文字で始まる識別子を定数として扱いたい場合、シングルクオートで囲む。

同様に第 4 列の Max. People は以下のように表せる。

```
max_people('Apricot', 4).
max_people('Maple', 5).
max_people('Orange', 7).
max_people('Stonehill', 6).
```

*1 http://www.dnc.ac.jp/data/shiken_jouhou/h27/jisshikekka/27honshiken_mondai.html

*2 http://www.cs.nmsu.edu/ALP/2011/03/natural-language-processing-with-prolog-in-the-ibm-watson-system/

同様に第 5 列の Max. Stay は以下のように表せる。

```
max_stay('Apricot',nights(15)).
max_stay('Maple',nights(12)).
max_stay('Orange',nights(7)).
max_stay('Stonehill',nights(14)).
```

第 6 列は以下のように表せる。

```
facilities('Apricot', ['BG']).
facilities('Maple', ['BG', 'PG']).
facilities('Orange', ['K', 'E', 'HS']).
facilities('Stonehill', ['E', 'HS']).
```

第 7 列は以下のように表せる。

```
restrictions('Apricot', []).
restrictions('Maple', []).
restrictions('Orange', ['No pets']).
restrictions('Stonehill', ['No fireworks']).
```

何を述語にして、何を引数にするかについては、いろんな方法が考えられるが、ここでは、列見出しを述語として、行見出しを第一引数とした。

なお、試験問題の画像から表の内容を読み取る手法については [磯崎 15] を参照されたい。

2.2 説明文を Prolog にする

この問題の説明文は以下のようなものである。実際には、あと 2 つのキャンプ場にも同様の説明文が存在する。

The campgrounds in Green National Park are open from April 1 to November 30.

Apricot Campground

Walking trails from this campground lead you to the top of Green Mountain. Enjoy the fantastic view from the top. You can also enjoy cycling on the bike trails in the woods.

Maple Campground

Maple Campground has direct access to Green River. Have fun doing such activities as fishing, boating, and swimming. You can also enjoy a campfire by the river.

この説明文を Prolog のプログラムに変換したい。ここでは以下のように、人手でキーワード、キーフレーズを抽出してリストにした。

```
description('Apricot', ['Green Mountain', 'bicycle']).
description('Maple', ['Green River', 'fishing',
 'boating', 'swimming', 'campfire']).
description('Orange', ['Orange Lake', 'water skiing',
 'fishing', 'swimming', 'bird-watching']).
description('Stonehill', ['pine tree forest', 'bicycle',
 'giant pine trees', 'wild animals', 'hiking']).
```

もっと自動変換しやすい表現については後で議論する。

2.3 問題文を Prolog にする

最後に各問題文を Prolog のプログラムに変換する。

問 1 は以下のような問題である。

問 1 A man who likes water activities is looking at the website. Which are the campgrounds he is most likely to be interested in? **39**

① Apricot and Maple Campgrounds
② Maple and Orange Campgrounds
③ Orange and Stonehill Campgrounds
④ Stonehill and Apricot Campgrounds

ここには、「water activities」という言葉があり、上記の `description` に現れる fishing, boating などが water activities であることが分からなければ解けない。ここで WordNet の利用が考えられるが、WordNet に water activity は登録されておらず、`WordNet-3.0/dict/data.noun` を grep して見つかったのは `Sea_Scout` だけであった。

ここでは、以下のように人手で `isa` という述語を導入することで対応する。

```
q1(Campground) :-
   description(Campground, Description),
   isa(X, 'water activity'),
   member(X, Description).

isa('fishing', 'water activities').
isa('boating', 'water activities').
isa('swimming', 'water activities').
isa('water skiing', 'water activities').
```

問 2 は、以下のような問題である。

問 2 Two people are making plans to stay in Green National Park for nine nights. They want to enjoy nature, but they need a power supply to use their computers. How much will they have to pay per night for the site they are likely to choose? **40**

① $20
② $24
③ $32
④ $96

2 人で 9 泊できて、電気がある、という条件を並べてから 1 泊の値段を調べる。

```
q2(Pay) :-
   max_people(Campground, MaxPeople),
   MaxPeople >= 2,
   max_stay(Campground, nights(N)), N >= 9,
   facilities(Campground, Facilities),
   member('E', Facilities),
   site_rate_per_night(Campground, Pay).
```

このプログラムを自動生成するとき、`power` を `'E'` に対応づけるところが課題となる。

問 3 は、以下のような問題である。

問 3 A family of four is planning a four-day camping trip with their dog. Their budget for a camp site is under 100 dollars for three nights. Their main interests for the trip are barbecuing and bicycle riding in the national park. Which campground is this family most likely to choose? **41**

① Apricot ② Maple ③ Orange ④ Stonehill

ペット不可ではなく、4 人で 3 泊できて、100 ドル以下で、バーベキューグリルと自転車がある、という条件を並べる。

```
q3(Campground) :-
    max_people(Campground, MaxPeople), MaxPeople >= 4,
    restrictions(Campground, Restrictions),
    \+ member('no pets',Restrictions),
    max_stay(Campground, nights(N)), N >= 3,
    site_rate_per_night(Campground, dollars(PerNight)),
    PerNight*3 =< 100,
    facilities(Campground,Facilities),
    member('BG', Facilities),
    description(Campground,Description),
    member('bicycle', Description),
```

キャンプ場の特徴は、今回の実装では、`restrictions`, `description`, `facilities` という 3 つの述語に分けて記録しているが、一つの述語にまとめてもよい。

このプログラムを自動生成する場合には、`with their dog` を `'no pets'` と対応づけるところが課題となる。

2.4 実行結果

以上の Prolog のプログラムを実行して、正解が得られるか確認する。実験は SWI-Prolog 7.2.2 を用いて行った。

以下の通り、実行結果はすべて正解であった。

```
?- setof(C,q1(C),CL).
CL = ['Maple', 'Orange'].

?- q2(Pay).
Pay = dollars(32)

?- q3(C).
C = 'Apricot'
```

3. Prolog プログラムの自動生成

すでに述べたように、上記の Prolog プログラム生成を自動化するには、いくつかの課題がある。たとえば、`description` でキーワードを抽出する方法を実装しなければならない。

3.1 説明文の自動変換

上記の `description` のキーワードは、単語ユニグラムと単語バイグラムである。そこで、以下のように説明文そのものの単語リストを第二引数とする簡便な方法が考えられる。

```
descWords('Orange',
 ['water','skiing','is','popular','on','the','lake']).
```

説明文に特定のユニグラムが含まれているかどうかは `member` で確認できる。特定のバイグラムが含まれているかどうかは、以下の `bigram` を用いて確認することができる。

```
bigram([A,B],[A,B|_]).
bigram(AB,[_|L]) :- bigram(AB,L).
```

実行すると以下のようになり、`['water', 'skiing']` が含まれていることがわかる。

```
?- descWords(CampGround, DescWords),
    bigram(['water','skiing'],DescWords).
CampGround = 'Orange',
DescWords = [water, skiing, is, popular, on, the, lake]
```

なお、この `bigram` は生成にも使える。

```
?- descWords(CampGround, DescWords),
    bagof(Bigram,bigram(Bigram,DescWords), BGbag).
CampGround = 'Orange',
DescWords = [water, skiing, is, popular, on, the, lake],
BGbag = [[water, skiing], [skiing, is], [is, popular],
    [popular, on], [on, the], [the, lake]].
```

ただし、これだけでは、否定されている特徴を扱えないので、否定の処理が課題となるだろう。

3.2 必要なオントロジー的知識の自動追加

また、`fishing` や `boating` が `water activities` であることを定義しておかなければならない。

`water activities` は WordNet には登録されていない。Wikipedia の `water activities` は「水分活性」という化学的な指標であり、別物である。

そこで、Pantel et al. [Pantel 04] にならい、1-billion コーパスを `grep 'water activities such as'` することで得られる。その結果、以下の 2 文が得られた。

```
There are three off-leash zones where dogs can run
free and even enjoy water activities such as
kayaking and surfing .
```

```
Mr Jordan called for full-day training for
participants of white water activities such as
rafting , jet boating and river boarding ;
mandatory up-to-date buoyancy aids ; and for
life jackets fitted with crotch straps and D-rings
to attach safety ropes .
```

この文から以下のプログラムを生成できる。

```
isa('kayaking', 'water activities').
isa('surfing', 'water activities').
isa('rafting', 'water activities').
isa('jet boating', 'water activities').
isa('river boading', 'water activities').
```

残念ながら、今回必要な `fishing`, `swimming` などは得られなかった。この問題は、コーパスを増やしたり、`grep` するパターンを増やすことで改善される可能性がある。

3.3 質問文の自動変換

残る問題は、問題文を Prolog に変換することである。Prolog のプログラムで英語の質問文を解析して答える初期の研究として CHAT-80 [Warren 82] がある。CHAT-80 では構文解析も Prolog で行っているが、ここでは、Python のプログラムで Prolog のプログラムを生成するというアプローチを採用するので、Enju [*3] などの高機能・高精度な構文解析器を利用することもできる。

```
Which campgrounds have water activities?
```

といった制約充足的な問題文であれば、Prolog のプログラムに変換するのは容易であろう。しかし、

```
A man who likes water activities is loking at
the website. Which are the campgrounds he is
most likely to be interested in?
```

のような遠回しな表現を前述の `q1(Campground)` のようなプログラムに変換するのは難しい。そこで `is most likely to` を `will` に書き換えることが考えられる。

問 2 や問 3 はもっと明確な制約充足問題である。これらの問題文を Prolog のコードに変換することが課題となる。問 3 の第 2 文を Enju で解析し、XML 出力する。この XML を `enjutree` 環境 [磯崎 11] で図示すると、図 1 の構造を持つ構文木が得られる。この XML は簡単な正規表現で、以下の Prolog の項に変換できる。

```
sentence(c(c(c(tok(their)),
             c(c(tok(budget)),
               c(c(tok(for)),
                 c(c(tok(a)),
                   c(c(tok(camp)),
                     c(tok(site)))))))),
          c(c(c(tok(is)),
              c(c(tok(under)),
                c(c(c(tok(100)),
                    c(tok(dollars))))))),
            c(c(tok(for)),
              c(c(c(tok(three)),
                  c(tok(nights))))))).
```

この形になってしまえば、構文木上での構文を考慮した各種変形は、Prolog で自由にできる。たとえば、`c(c(tok(100)), c(tok(dollars)))` を `dollars(100)` に変形したければ、以下のような述語を用意すればよい。

```
unitNumb(C, U) :-
    C = c(c(tok(N)),c(tok(Unit))), U =.. [Unit,N].
```

*3 http://www-tsujii.is.s.u-tokyo.ac.jp/enju/

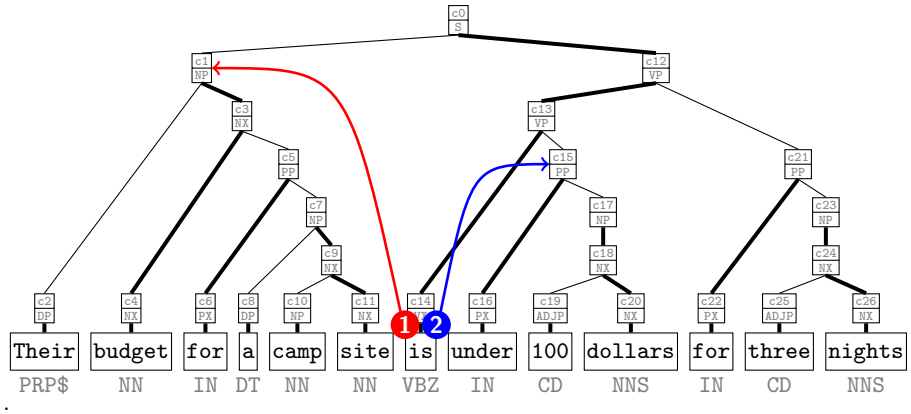

図 1: Enju の出力する構文木

すると、

```
| ?- unitNumb(c(c(tok(100)), c(tok(dollars))),U).
U = dollars(100).
| ?- unitNumb(c(c(tok(three)), c(tok(nights))),U).
U = nights(three).
```

が得られる。

あとは、制約を表す様々なパターンの文に対して、Prolog のコードに変換するためのルールを書き下していくことが必要であるが、ここをルールベースでなはない手法で作ることが重要である。

4. まとめ

本稿では、英語センター試験に現れる計算問題を自動で解くプログラムについて、

1. 説明文・図表・問題文をそれぞれ Prolog のプログラムに変換する。
2. Prolog のプログラムを実行して解を得る。

という二段階の解法を考え、2015 年のセンター試験の問 4B に適用してみた。その結果、Prolog のプログラムを人手で作り、正解が得られることを確認した。次にその Prolog のプログラムを自動生成する方法を検討した。第一段階の自動化については、今回行った小規模な実験により、オントロジー知識の不足など、いくつかの課題が明らかになった。

参考文献

[安西 85] 安西 祐一郎, 上原 譲, 田村 淳：算数の文章題を解くシステムにおける問題の内部表現について, 情報処理学会研究報告　知識工学と人工知能 41-1 (1985)

[Bobrow 68] Bobrow, D.: *Natural Language Input for a Computer Problem-Solving System*, pp. 135–215, MIT Press (1968)

[磯崎 11] 磯崎 秀樹：HPSG に基づく英日翻訳と TEX による XML の可視化, 電子情報通信学会研究報告 NLC2011-25 (2011)

[磯崎 15] 磯崎 秀樹, 伊藤 圭汰, 荒木 良元：論文 QA のための画像処理　表を読む , 言語処理学会年次大会, pp. 139–142 (2015)

[松崎 15] 松崎 拓也, 横野 光, 宮尾 祐介, 川添 愛, 狩野 芳伸, 加納 隼人, 佐藤 理史, 東中 竜一郎, 杉山 弘晃, 磯崎秀樹, 菊井 玄一郎, 堂坂 浩二, 平 博順, 南泰浩：「ロボットは東大に入れるか」プロジェク代ゼミセンター模試タスクにおけるエラーの分析, in *Proc. of the Annual Meeting of the Association for Natural Language Processing* (2015)

[Pantel 04] Pantel, P. and Ravichandran, D.: Automatically Labeling Semantic Classes, in *Proc. of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT/NAACL)*, pp. 321–328 (2004)

[Roy 15] Roy, S. and Roth, D.: Solving General Arithmetic Word Problems, in *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1743–1752 (2015)

[Shi 15] Shi, S., Wang, Y., Lin, C.-Y., Liu, X., and Rui, Y.: Automatically Solving Number Word Problems by Semantic Parsing and Reasoning, in *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1132–1142 (2015)

[Warren 82] Warren, D. H. D. and Pereira, F. C. N.: An Efficient Easily Adaptable System for Interpreting Natural Language Queries, *Computational Linguistics*, Vol. 8, No. 3–4 (1982)

[Zhou 15] Zhou, L., Dai, S., and Chen, L.: Learn to Solve Algebra Word Problems Using Quadratic Programming, in *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 817–822 (2015)