2J1-E-5-04

A think-programming method based on free texts “almost” in English

Keisuke NAKAMURA[*1]

[*1] Knowrel System, Inc.

This document describes a method for programming for simulating human-thinking based on free English texts and describes three PROLOG-like examples each consisting of a) such type “program” that is “almost” in English and that has natural language templates with variables, b) related query, and c) corresponding simulated “human-thought” result. In order to speed up executing English “programs” and queries that are separated with words and white spaces, we improved original algorithm used from 2012 for Japanese “programs” that are not so separated. We conclude that our think-programming method is useful for automatically simulating some types of real human thoughts that are able to be programmed “almost” in English.

1. Introduction

To use computer to automatically process knowledge and/or information written in natural language in the same manner as Prolog and to achieve comprehensive deduction and solution searching at the predicate logic level, in [Nakamura 2016] etc. we proposed that the knowledge be expressed in a format wherein variables are embedded in natural language.

This document describes 2) feature of the method, 3) examples of program, related query, and corresponding result, 4) our recent improvement for English white spaces, 5) conclusion.

2. Feature of our method

We respect Prolog. But our method differs in some points below. Detailed algorithms are shown in [Nakamura 2016].

・a person uses character types, delimiters, or escape characters to distinguish the constant portions and the variable portions of content that is equivalent to "literal" in Prolog and inputs the content into a computer.

・computer performs automatic unification and/or automatic derivation on text included in the input while treating variables as material that could span the boundaries of the subject, predicate, object etc. of the text.

3. Examples of program, query, and result

3.1 Example 1

```
[Progam]  C:\Users\Keisuke\Documents\humanote\1.jpl
   1: ice cream is cold food
   2: cold food is cooling
   3: $A is $B :- $A is $C ; $C is $B ;
[Query]
   ?ice cream is $X
[Result]
   :$X  = cold food
   :$X  = cooling
   time = 875msec
```

Line 3 teaches computer “transitivity rule” of human-thinking.

Contact: Keisuke NAKAMURA, KnowrelSystem,Inc., phone: +81-76-213-5211, fax: 5239, keisukebecome@icloud.com

Semicolon(;) is used for delimiter of literals because comma(,) in Prolog is too casually used in natural English.

Using lower case of alphabet, distinct case also is available.

It was implemented on 3.6-4.2GHz CPU, 8G Memory, Windows 10 and Visual C++2013. Demonstrations are enabled by downloading open beta version of our software HUMANOTE (see https://github.com/keisukebecome/humanote for public/)

3.2 Example 2

```
[Progam]  C:\Users\Keisuke\Documents\humanote\2.jpl
   1: ///@1.jpl
   2: my favorite food is ice cream
   3: what i eat is $X :- my favorite food is $X ;
[Query]
   ?what i eat is $X
[Result]
   …
   3: ///◆INC_START 2019/02/13_21:52:29  1.jpl
   4: ///LOAD … C:\Users\Keisuke\Documents\humanote\1.jpl
   5: ice cream is cold food
   6: cold food is cooling
   7: $A is $B :- $A is $C ; $C is $B ;
   8: ///◆INC_END  1.jpl
   9: my favorite food is ice cream
   10: what i eat is $X :- my favorite food is $X ;

   :$X  = cooling
   :$X  = cold food
   :$X  = ice cream
   time=3252msec
```

Line 1 at original Program (2.jpl) was expanded to Line 3-8 at resulted memory and computer do unification and derivation on the expanded Program in the memory.

Our system also enables recursive expansion and enables citation/reuse of programs placed in the internet, group LAN, and PC (the above example is PC case).

We call this method “LinkedOpenRule” (named after “LinkeOpenData”) which cites and uses not only new data (fact) in the internet etc. but also new rules both being updated daily, weekly and monthly by other users, groups and companies.

3.3 Example 3 (before improvement as to time)

```
[Progam]  C:\Users\Keisuke\Documents\humanote\3.jpl
   1: we can get to Noto from Kanazawa by car
   2: we can get to Kanazawa from Osaka by JR
   3: we can get to Sendai from Osaka by Shinkansen
   4: we can get to gyuutan shops from Sendai by foot
   5: we can get to Osaka from Toyonaka by Hankyu-line
   6: we can get to Hegura-island from Noto by boat
   7: we can get to $Z from $X by $A and $B :-
                                   we can get to $Y from $X by $A ;
                                   we can get to $Z from $Y by $B ;
[Query]
   ?we can get to $M from Toyonaka by $N
[Result]
   :$M  = Osaka
   :$N  = Hankyu-line

   :$M  = Kanazawa
   :$N  = Hankyu-line and JR

   :$M  = Sendai
   :$N  = Hankyu-line and Shinkansen

   :$M  = Noto
   :$N  = Hankyu-line and JR and car

   :$M  = Hegura-island
   :$N  = Hankyu-line and JR and car and boat

   :$M  = gyuutan shops
   :$N  = Hankyu-line and Shinkansen and foot

   time=124757msec
```

By this experiment, we found that 124757msec. for English program is over 15 times larger than 7818msec. for Japanese program with same logical structure.

In order to improve time, we reviewed original source codes and speeded up the basic algorithm of unification for "almost" English programs and related queries.

4. Speed up for English "program"

For speeding up, we focused on white spaces in English natural language texts that don't exist so much in Japanese texts.

Two (one before and one after) characters neighboring each variable in the template was searched in targeted constant texts in original algorithm [Nakamura 2016].

New algorithm also checks another more before character if the original-variable-neighboring-before-character is white space like ' '.

The improvement result is that original 124757msec. was shorten to 9034msec. with other resulting content being same as the above-mentioned content.

It is only 7.24% of original time cost.

We think that this new algorithm is useful not only for other word-separated (i.e. inflected) natural language like French, German, Spanish and so on, but also agglutinative languages and analytic/isolated languages.

5. Conclusion

We conclude that our think-programming method is useful for automatically simulating some types of real human thoughts that are able to programmed "almost" in English.

6. Next

We will do as below,

・speed up more as to Japanese and English

・experiment as to analytic language such as Chinese language

・create a shared "LinkedOpenRule" web sites in each language with worldwide partners.

References

[Nakamura 2016] Keisuke Nakamura, "Method for Processing Knowledge or Information, Device, and Computer Program", International Publication WO/2016/071942, the World Intellectual Property Organization (WIPO), 2016.