

自然言語をラムダ式で 解釈する体系PTQ のHaskell実装



酒井 政裕
@masahiro_sakai

今日の話

自然言語は
関数型言語！

元ネタ

The
Proper
Treatment of
Quantification
in ordinary English

1973

A black and white portrait of Richard Montague, a middle-aged man with short dark hair, wearing a dark suit jacket, a white shirt, and a dark tie. He is looking slightly to the left of the camera with a neutral expression.

Richard Montague

いきなりだけど

Haskellで実装したので

デモ

Sample:

John seeks a unicorn.

Translate!

Parsed

F4 john (F5 seek (F2 a unicorn)) : t

Translation

$(\lambda x_0 : s \rightarrow e \rightarrow t. x_0 \{john\}) \wedge (\text{seek } \wedge ((\lambda x_1, x_2 : s \rightarrow e \rightarrow t. \exists x_3 : e. x_1 \{x_3\} \wedge x_2 \{x_3\}) \wedge \text{unicorn})) : t$

Translation (simplified)

$\text{seek } \wedge (\lambda x_0 : s \rightarrow e \rightarrow t. \exists x_1 : e. \text{unicorn } x_1 \wedge x_0 \{x_1\}) \text{ john} : t$

Translation (MP applied)

$\text{try } \wedge (\lambda x_0 : e. \exists x_1 : e. \text{unicorn } x_1 \wedge \text{find* } x_1 x_0) \text{ john} : t$

改めて今日の話

自然言語は
関数型言語！

どういうことか？

- 品詞は型
- 単語の並びは関数適用

品詞 (範疇)

John :: 名詞

walk :: 動詞

John walks. :: 文

素朴なアイデア

type 動詞

= 名詞 → 文

type 名詞 = Entity

type 文 = Bool

素朴なアイデア

John :: Entity

walk :: Entity → Bool

John walks.

⇒ walk(John) :: Bool

素朴なアイデア

type 他動詞

= 名詞 → 動詞

find, love :: 他動詞

素朴なアイデアでダメな場合

- **John or Mary walks.**
- **Every man walks.**

Entity として解釈不能

PTQのHack

type 動詞 = 名詞 → 文

ではなく

type 名詞 = 動詞 → 文

要は高階関数

type 動詞

= Entity \rightarrow Bool

type 名詞

= 動詞 \rightarrow 文

= (Entity \rightarrow Bool) \rightarrow Bool

翻訳

John or Mary

$\Rightarrow \lambda f \rightarrow f \text{ John}' \vee f \text{ Mary}'$

$:: (\text{Entity} \rightarrow \text{Bool}) \rightarrow \text{Bool}$

John or Mary walks

$\Rightarrow (\lambda f \rightarrow f \text{ John}' \vee f \text{ Mary}') \text{ walk}'$

$= \text{walk}'(\text{John}') \vee \text{walk}'(\text{Mary}')$

この先の話

- 冠詞と量化 (これが面白い)
- 内包と外延
- 時制と様相
- 他の文型

ただし、
今回は省略

PTQのHaskell実装

- HackageDB で PTQ
% cabal install PTQ
- Haskellは記号処理が得意
- こういうのはちょー簡単