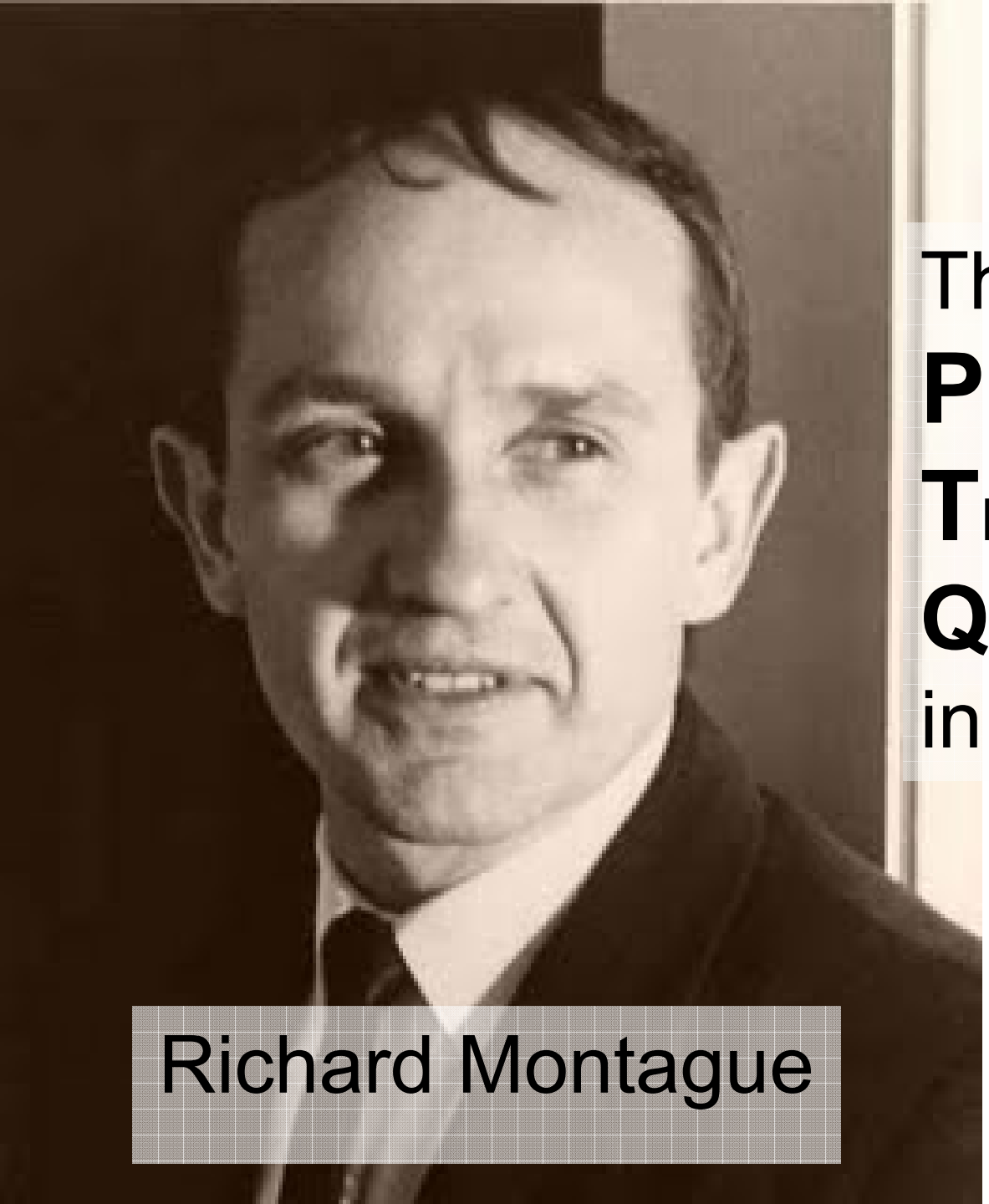
自然言語をラムダ式で解釈する体系PTQ

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

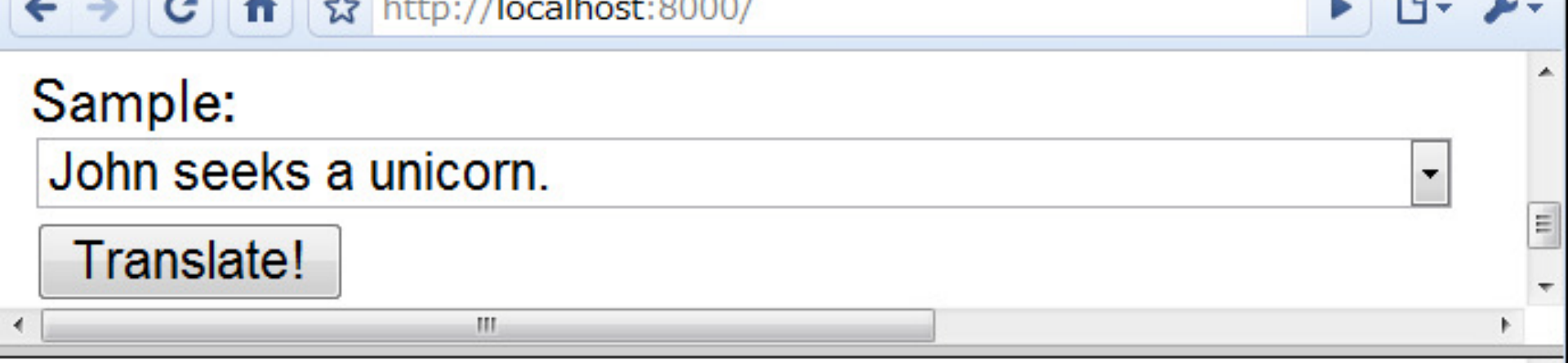
The
Proper
Treatment of
Quantification
in ordinary English

1973

いきなりだけど

Haskellで実装したので

デモ



Parsed

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

Translation

(λ x0 : s→e→t. x0 {john}) ^(seek ^((λ x1, x2 : s→e→t. ∃ x3 : e. x1 {x3} ∧ x2 {x3}) ^unicorn)) : t

Translation (simplified)

seek ^(λ x0 : s→e→t. ∃ x1 : e. unicorn x1 ∧ x0 {x1}) john : t

Translation (MP applied)

try ^(λ x0 : e. ∃ x1 : e. unicorn x1 ∧ find* x1 x0) 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

素朴なアイディア

type 動詞
= 名詞 → 文

type 名詞 = Entity

type 文 = Bool

素朴なアイディア

John :: Entity

walk :: Entity → Bool

John walks.

⇒ walk(John) :: Bool

素朴なアイディア

type 他動詞
= 名詞 → 動詞

find, love :: 他動詞

素朴なアイディアでダメな場合

- **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 → Bool

type 名詞

= 動詞 → 文

= (Entity → Bool) → 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