付録 E

ラムダ記法と意味処理

本章は、自然言語の文章の意味表現としての談話表示理論 (DRT, Discourse Representation Theory) と、その計算機への実装を詳しく述べることを目的とした講義「自然言語理解システム」の補助資料として用意された。文から談話表示構造 (DRS) への翻訳の準備として、ラムダ記法を導入し、また文の統語構造と語彙の意味表記から「自動的に」文の意味表現が得られることを示す。

E.1 有限個の語から無限通りの表現を生成する「言語」システム

- 日本人の大人が持つ日本語の語彙 (vocabulary) は平均的に 5 万程度
- 我々が見聞きしたり、発話したりする文の数は無限
- 語から文を生成する仕組みにより「無限個の文が生成される」
- 一つは、接続詞などによって（簡単な）語句や文をつないで長い文を作り出すもの
- もう一つは、「再帰的な」構造
- 再帰的な構造の例：関係節—言語学の用語 (日本語文法では、連体修飾節）
  「(日本語の場合）文 S を名詞句 NP の前に置き、S によってその名詞句の指示対象や意味を特定する」用法
  関係節をもつ名詞句を主語や目的語として使える—文の中に文が現れる (このような文の中の文のことを「埋め込み文」[*1]という。

「文の意味表現を求める」というタスク（仕事）を考える。

基本方針 (合成性): それぞれの語に論理式を与え、また文法規則によってどのような語や句がより大きな単位となって文を構成する要素となるかを規定

注意：ここで考える文の要素は、名詞句や動詞句のように我々の言語直感に合致する要素

- それぞれの語を、文法的な特徴からカテゴリに分ける（固有名詞、自動詞、他動詞など）
- それぞれの語に、カテゴリに応じたタイプの論理式を割り当てる
- それぞれのカテゴリの語や句がどのように組み合わされて、文を構成する要素を作

[*1] このような埋め込み文は、文の引用や副詞節などでも現れる。

るかという文法規則を立てる
- その文法規則ごとに、それぞれの語句の意味表現から、どのような意味表現が作り出されるか、という規則をたてる

そのための妥当な方法：ラムダ記法を用いて意味を表現する

E.2　ラムダ記法とラムダ計算

- ラムダ記法は Alonzo Church が 1930 年代に考え出した数学的な道具
- 言語への応用は言語学者の Richard Montague の功績

E.2.1　無名関数を作り出すラムダ

- 数学の問題：「f は実数を定義域とし、$f(x) = x^2 + 3$ であるような関数とする。この時、10 に対する f の値はいくらか？」
- その計算方法：
  10 に対する f の値とは、$f(10)$ を計算すればよい。それには $f(x) = x^2 + 3$ だから、 $f(10) = 10^2 + 3$ となる。これを計算して、答えは 103
- ここで行われたこと： $x^2 + 3$ という式の x に 10 を代入して、$10^2 + 3$ という形にし、算数計算によって、103 を出力する。
- 注意: 関数の定義に用いられる引数を**仮引数**、計算のために与える引数を**実引数**と呼ぶ
- ラムダ式 $\lambda x\ .\ x^2 + 3$ は $x^2 + 3$ を計算する関数を表したもの
  ただしこれは、f のような名前を持たない。だから $f(10)$ を計算せよ、とは書けない。そのかわり、$(\lambda x\ .\ x^2 + 3)(10)$ のように書く
- ラムダ計算では $(\lambda x\ .\ x^2 + 3)(10)$ を $(10)(\lambda x\ .\ x^2 + 3)$ と書いても良い—なぜなら、どちらが (実) 引数でどちらが関数かは普通は明らか
- ここでは $(\lambda x\ .\ x^2 + 3)@(10)$、もしくは $(\lambda x\ .\ x^2 + 3)@10$ と表す—「関数@実引数」 の形に統一
- $(\lambda x\ .\ x^2 + 3)@(10)$ は 103 と等価ではないことに注意。この形から 103 を求めるには、まず $10^2 + 3$ に変形してから算術演算によって計算する
- β 変換：このように $(\lambda x\ .\ x^2 + 3)@10$ を $10^2 + 3$ に変形する操作のこと[*2]。
- 次に二つの引数を持つ関数 $g(x, y) = x^2 + y^2$ をラムダ式で表すことを考える
  これには $\lambda x \lambda y\ .\ x^2 + y^2$ と $\lambda y \lambda x\ .\ x^2 + y^2$ の 2 通りが可能
- ラムダ式では実引数を同時に与えることはできない—1 個の仮引数に対し実引数を 1 個、次の仮引数 1 個に対し実引数を 1 個、というように順番に渡さなければならない。
- まず仮引数 x に最初の実引数が与えられ、次に仮引数 y に次の実引数が与えられる、というように順番が決まれば、ラムダ式の形も決まる。この場合 $\lambda x \lambda y\ .\ x^2 + y^2$ が求めるラムダ式
- これを用いて、$g(3, 2)$ に対応する計算は:

[*2] 最後の算術演算そのものはラムダ計算の範囲外である。

1. 仮引数 x に実引数 3 が渡される: $(\lambda x \lambda y\ .\ x^2 + y^2)@3$
2. $(\lambda x \lambda y\ .\ x^2 + y^2)@3$ に β 変換を施すと、$\lambda y\ .\ 3^2 + y^2$
3. 仮引数 y に実引数 2 が渡される: $(\lambda y\ .\ 3^2 + y^2)@2$。
4. β 変換を施すと、$(3^2 + 2^2)$

E.2.2 ラムダ抽象化

- ラムダ抽象化とは、数式から関数を作り出すような操作のこと
- プログラム作成への例え
  アニメーション作成において、「馬オブジェクトを前方 1m、上 0.5m の位置に同時に移動させ、次に前方 1m 下 0.5m の位置に同時に移動させる」を行うプログラムを書いたとする。
  - ここで、「1m」をラムダ抽象化する
    λx.(馬オブジェクトを前方 x、上 $0.5m$ の位置に同時に移動させ、次に前方 x 下 $0.5m$ の位置に同時に移動させる)
  - これは「馬オブジェクトを前方 xm、上 0.5m の位置に同時に移動させ、次に前方 xm 下 0.5m の位置に同時に移動させる」という「関数」(Ruby で言えばメソッド) を表す
  - さらに続けて「0.5m」、「馬オブジェクト」をラムダ抽象化
    $\lambda o \lambda y \lambda x$.($o$ を前方 x、上 y の位置に同時に移動させ、次に前方 x 下 y の位置に同時に移動させる)
  - 以上のラムダ抽象化から、「任意のオブジェクトをジャンプしながら前方に移動」する関数が得られた

この手法を「文の意味表現」から「個々の語の意味表現を求める」ために用いる

A woman walks

- 一階述語論理で表すと $\exists x(woman(x) \wedge walk(x))$
- この文は $a\ woman$ という名詞句と $walks$ という動詞句から構成されている
- この文の一階述語論理式に対し、$walks$ に対応する $walk$ をラムダ抽象化
  $\lambda P.\exists x(woman(x) \wedge P@x)$
  注: $walk(x)$ を $P@x$ と書き改めたのは、$walk$ は一項述語 (ラムダ記法では関数) で x がその実引数だから— つまり、$walk(x)$ は $walk$「関数」に対する実引数 x の関数適用の結果得られた
- ここまでで $a\ woman$ の意味表現が得られた:$\lambda P.\exists x(woman(x) \wedge P@x)$
- 今度はこれに対して $woman$ をラムダ抽象化する。$woman$ も $walk$ と同様、一項述語なので、同様にして $\lambda Q \lambda P.\exists x(Q@x \wedge P@x)$ が得られる
- これが限定詞 (伝統的な言葉で言えば冠詞)a の意味表現として規定したもの

Every boy laughs

- 一階述語論理で表すと $\forall x(boy(x) \rightarrow laugh(x))$

- この文は *every boy* という名詞句と *laughs* という動詞句から構成されている
- この文の一階述語論理式に対し、*laughs* に対応する意味表現 $laugh$ をラムダ抽象化
  $\lambda P.\forall x(boy(x) \rightarrow P@x)$
  注: $laugh(x)$ が $P@x$ と書き改めた理由は先と同じ
- ここまでで *every boy* の意味表現が得られた:$\lambda P.\forall x(boy(x) \rightarrow P@x)$
- 今度はこれに対して *boy* をラムダ抽象化する。*boy* も $laugh$ と同様、一項述語なので、同様にして、$\lambda Q\lambda P.\forall x(Q@x \rightarrow P@x)$ が得られる。
- これが限定詞 *every* の意味表現として規定したもの

John loves a woman

今度は語彙の意味から文の意味を構成してみる

- E.2.2 節で求めた限定詞 a の意味表現: $\lambda Q\lambda P.\exists x(Q@x \wedge P@x)$
- 名詞 $woman$ の意味表現を $\lambda z.woman(z)$ とすると、$a\ woman$ の意味表現は、これらを関数適用した結果:
  $(\lambda Q\lambda P.\exists x(Q@x \wedge P@x))@(\lambda z.woman(z))$
- これを β 変換すると: $\lambda P.\exists x(\underline{\lambda z.woman(z)@x} \wedge P@x)$
  下線部を β 変換して: $\lambda P.\exists x(woman(x) \wedge P@x)$
- *loves* の意味表現を $\lambda u\lambda v(u@\underline{\lambda z.love(v,z)})$ とすると、*loves a woman* の意味表現は:
  $(\lambda u\lambda v(u@\underline{\lambda z.love(v,z)}))@\ \underbrace{\lambda P.\exists x(woman(x) \wedge P@x)}$
- これを β 変換すると:
  $\lambda v(\underbrace{\lambda P.\exists x(woman(x) \wedge P@x)}\ @\underline{\lambda z.love(v,z)})$
- さらに β 変換: $\lambda v(\exists x(woman(x) \wedge (\underline{\lambda z.love(v,z)})@x))$
- もう一度 β 変換: $\lambda v.\exists x(woman(x) \wedge love(v,x))$
- 最後に *John* の意味表現 $\lambda P.P@john$ に *loves a woman* の意味表現を関数適用:
  $(\lambda P.P@john)@(\lambda v.\exists x(woman(x) \wedge love(v,x)))$
  $(\lambda v.\exists x(woman(x) \wedge love(v,x)))@john$
  $\exists x(woman(x) \wedge love(john,x))$

Every boy loves a woman

語彙の意味から文の意味を構成してみる。途中を省略して以下を仮定

- *every boy* の意味表現: $\lambda P.\forall x(boy(x) \rightarrow P@x)$
- *loves a woman* の意味表現: $\lambda v.\exists x(woman(x) \wedge love(v,x))$

この 2 つの式には変数 x が共に現れているが、これらは無関係で紛らわしいので、(α 変換により) 変数を付け変え、*loves a woman* の意味表現を以下のようにする:
$\lambda v.\exists u(woman(u) \wedge love(v,u))$

- *every boy* の意味表現に対し *loves a woman* の意味表現を実引数として関数適用:
  $(\lambda P.\forall x(boy(x) \rightarrow P@x))@\underline{\lambda v.\exists u(woman(u) \wedge love(v,u))}$
- これを β 変換 $\forall x(boy(x) \rightarrow \underline{(\lambda v.\exists u(woman(u) \wedge love(v,u)))}@x)$
- もう一度 β 変換により、最終形:
  $\forall x(boy(x) \rightarrow \exists u(woman(u) \wedge love(x,u)))$

A pretty woman walks

新たな文法規則と、それに対応する意味規則を考える

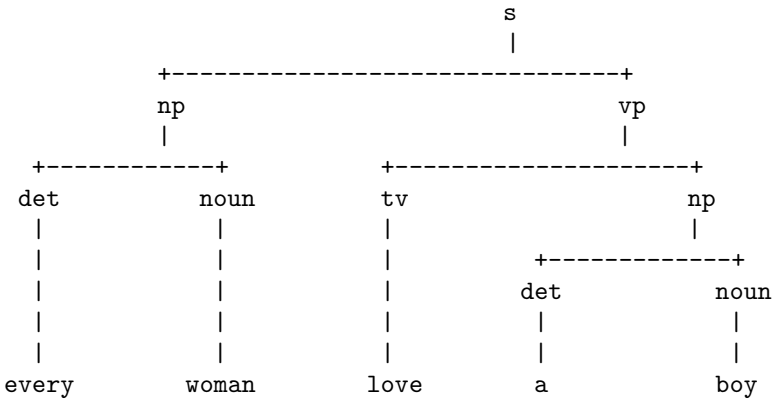
- *A pretty woman walks.* に対して考えられる一階述語論理式:
  $\exists x(pretty(x) \wedge woman(x) \wedge walk(x))$
- 先と同様に *walks* に対する意味表現をラムダ抽象化して *a pretty woman* の意味表現を得る: $\lambda P.\exists x(pretty(x) \wedge woman(x) \wedge P@x)$
- *A pretty woman* が *a* と *pretty woman* から構成されることと、*a* の意味表現が $\lambda Q\lambda P.\exists x(Q@x \wedge P@x)$ であることから、*pretty woman* の意味表現を求める:
  $\lambda z.(pretty(z) \wedge woman(z))$
- *woman* に対する意味表現をラムダ抽象化する: $\lambda R\lambda z.(pretty(z) \wedge R@z)$
  これが形容詞 *pretty* の意味表現
- *pretty* は形容詞 (Adjective)、*woman* は名詞 (Noun) であることから、考えられる文法規則: `n(app(A,N)) --{\rangle} a(A), n(N).`
  参考: 主語や目的語の役割をする英語の名詞句の基本形は、限定詞と名詞の組み合わせ。この例のように「形容詞と名詞の組み合わせ」では名詞句を構成しているとは考えられないが、名詞よりも大きなカテゴリ。そこで NBar (直訳すれば、名詞プラス) というカテゴリ名を与える者もいる。

E.3 意味解析プログラム

タスク 1　自然言語 (ここでは英語) に対する統語論の記述。ここでは **DCG を用いて実現**。

DCG の助けを得て、文の構造を求めることができる

例:



注: DCG の規則は、このような木の部分部分の成り立ちを決めるもの。