2F4-OS-01a-1

依存型意味論による照応・前提計算の実装に向けて

Presupposition projection as type checking

佐藤 未歩 *1　戸次 大介 *1*2*3
Miho Satoh　Daisuke Bekki

*1お茶の水女子大学 Ochanomizu University
*2国立情報学研究所 National Institute of Informatics
*3独立行政法人科学技術振興機構, CREST CREST, Japan Science and Technology Agency

Dependent type semantics (DTS) is a framework of natural language semantics based on dependent type theory extended with underspecified terms. It has been predicted that the projections of presuppositions and anaphora can be calculated by means of type inference/type checking in DTS. In this paper, we formalized and implemented a set of rules for a type inference/type checking algorithm for a decidable fragment of dependent type theory, which is still sufficient for describing semantic representations in DTS. We demonstrate that this implementation correctly calculates presuppositions of sentences in a decidable and compositional manner.

1. はじめに：前提投射問題と DTS

前提 (presupposition)・照応 (anaphora) 現象は、含意 (entailment) と同様、自然言語の意味論の主要な説明対象である。(1a) は含意の例、(2a) は前提の例であるが、(1b)(1c) と (2b)(2c) のように、否定、条件文の後件に埋め込まれた時、両者は推論について異なるパターンを示す。

(1) a. Sweden cherishes its king. ⊢ Sweden cherishes somebody.
b. Sweden does not cherish its king. ⊬ Sweden cherishes somebody. ⊬ Sweden cherishes somebody.
c. If Sweden cherishes somebody, Sweden cherishes its king. ⊬ John cherishes somebody.

(2) a. Sweden cherishes its king. ⊢ Sweden has a king.
b. Sweden does not cherish its king. ⊢ Sweden has a king. ⊢ Sweden has a king.
c. If Sweden has a king, Sweden cherishes its king. ⊬ Sweden has a king.

したがって、「前提を持つ言語表現が他の言語表現と組み合わせられた時、全体としてどのような前提を持つか」という問題は、自然言語の推論の計算において重要な意味を持つ。これは前提投射 (presupposition projection) の問題として知られ、Stalnaker, Kartunnen, Heim らによる研究に遡るが、のちに van der Sandt[5] らによって前提は照応に還元されうることが示され、前提投射もまた照応解決と統一的に定式化されている。

しかし前提投射の計算の定式化には難題も残されている。その一つは (3) に示す局所適応 (local accomodation) と呼ばれる現象である（Heim 1983）。(3) では前提の内容 (=*it has a king*) が代名詞 *it* の指示対象を含んでいるが、この *it* は量化子 *every* のスコープ内で束縛変項照応として解釈されるため、前提投射は量化子 every のスコープを超えない範囲に止めなければならない。DRT に基づく van der Sandt のシステムは、局所適応の場合を考慮して複雑なものとならざるをえない。

(3) [Every nation]$_1$ cherishes its$_1$ king. ⊢ Every nation has a king.

連絡先: 佐藤未歩，お茶の水女子大学大学院人間文化創成科学研究科戸次研究室，東京都文京区大塚 2-1-1, satoh.miho@is.ocha.ac.jp

それに対して、依存型意味論 (Dependent Type Semantics, DTS: Bekki[1]) は、自然言語の証明論的意味論であり、未指定項 (underspecified term) によって前提・照応の意味を表示する。DTS では文脈受け渡し (context-passing mechanism) と呼ばれる機構によって、命題に局所文脈 (local context) と呼ばれる「先行文脈の証明項」が渡されるが、各未指定項は引数としてこの局所文脈を受け取る関数である。DTS における照応解決は、未指定項を具体的な証明項に置き換えることに相当する。そのような証明項は一般に複数存在し、異なる先行詞に対応している。

DTS の特徴は、前提投射の計算は「未指定項に対する型チェック問題」に帰着し、前提束縛・照応解決の計算は「証明探索」に帰着する点にある。van der Sandt[5] の前提投射の計算とは異なり、DTS における型チェックは、型理論におけるもっとも一般的な操作の一つである。したがって DTS は、型理論・証明論と自然言語の意味論を架橋し、俯瞰する視点を提供する。

ただし Bekki[1] においては、これは単なる示唆に留まるものであった。本論文の貢献は、未指定項を含む依存型理論の型チェック・型推論システムを定義し、実装したことである。特に、本論文のシステムでは、(3) のような局所適応のケースもその他の前提・照応と同じアルゴリズムにおいて計算可能である。(3) の DTS における意味表示は図 1 の通りである。

次節では、(3) を例に取り、本研究が提案する「型チェックによる前提投射計算」の機構を詳述する。またこの機構によって、依存型意味論による前提投射・照応解決の計算が、統一的かつ計算可能な枠組みで実現することを示す。

2. DTS における型チェック・型推論規則

本研究の体系は Löh による Agda の型推論の体系 [3] を元に、DTS のための新たな項を追加したものである。この体系では、項は inferable term（型推論可能な項）と checkable term（型推論可能な項）に分かれている（図 2 参照）。ただし、型推論可能ならば型チェック可能である。また、値は neutral term と value の 2 つに分かれている。以下、inferable term を $M_{\uparrow}$、checkable term を $M_{\downarrow}$ とする。

この体系での型推論・型チェック規則は相互再帰的に定義されており、$(CHK)(VAR)(CON)(TYPE)(\Pi F)(\Pi I)(\Pi E)(\to E)(\to F)(\to I)(\Sigma F)(\Sigma I)(\Sigma E)(\wedge I)(\wedge E)(\wedge F)(ANN)(\top F)(\top I)(\bot F)(ASP)$ が存在する。各規則においては、下段に $:_{\downarrow}$ が現れるものが型チェック規則、$:_{\uparrow}$ が現れるものが型推論規則となっている。例として、$(TYPE)$ 規則を以下に示す。

$$\frac{}{[L]\ \Gamma \vdash_{\sigma} \mathsf{type} :_{\uparrow} \mathsf{kind}\ [L]}\ (TYPE)$$

$$\lambda c.\left(u{:}\begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{nation}(x) \end{bmatrix}\right) \to \mathbf{cherish}(\pi_1(u), \pi_1(@_1(c,u)) : \begin{bmatrix} y{:}\mathbf{entity} \\ \mathbf{kingOf}(y, \pi_1(@_2(c,u)) : \begin{bmatrix} z{:}\mathbf{entity} \\ \neg\mathbf{human}(z) \end{bmatrix})) \end{bmatrix})$$

図 1: (3) Every nation cherishes its king. の意味表示

$$\begin{array}{rcl} v & ::= & n \mid \mathsf{type} \mid \mathsf{kind} \mid \top \mid \bot \mid () \mid (x{:}v) \to v' \mid \begin{bmatrix} x{:}v \\ v' \end{bmatrix} \mid \lambda x.v \mid (v, v') \mid v \to v' \mid \begin{bmatrix} v \\ v' \end{bmatrix} \\ n & ::= & x \mid c \mid @_i \mid nv \mid \pi_i n \\ M_\uparrow & ::= & x \mid c \mid \mathsf{type} \mid (x{:}M_\downarrow) \to M_\downarrow \mid M_\uparrow M_\downarrow \mid \begin{bmatrix} x{:}M_\downarrow \\ M_\downarrow \end{bmatrix} \mid (M_\uparrow, M_\uparrow) \mid \pi_i M_\uparrow \\ & \mid & M_\downarrow : M_\downarrow \mid M_\downarrow \to M_\downarrow \mid \begin{bmatrix} M_\downarrow \\ M_\downarrow \end{bmatrix} \mid () \mid \top \mid \bot \\ M_\downarrow & ::= & M_\uparrow \mid \lambda x.M_\downarrow \mid M_\downarrow M_\uparrow \mid (M_\downarrow, M_\downarrow) \mid @_i \end{array}$$

図 2: Neutral term, Value, inferable term, checkable term

これらの規則の詳しい説明は、次節において文 (3) 意味表示 (図 1) の型チェックを通じて詳しく見ていく。ただし、$(TYPE)$ $(\to I)(\wedge E)(\wedge F)(\top F)(\top I)$ の 6 個の規則は型チェックの過程において呼び出されないため、説明は省略する。

各規則の $\vdash$ の左側に現れている Γ は、型環境（変数の型を保持する環境）である。また、$\vdash$ の下つき文字として現れている σ はシグネチャ（定数の型を保持する環境）であり、型チェックを実行する際にはあらかじめ各定数とその型をシグネチャに定義しておく必要がある。$[L]$ はアスペランド環境（各アスペランドに対する型チェックのジャッジメントを保持する環境）であり、型推論・型チェック規則の実行に伴い更新されていく。

3. 型チェック・型推論の過程

図 1 の意味表示に対する型チェックの過程を例に、本研究の型チェックシステムを解説する。具体的には、以下の型チェックを行うことになる。

$$d : \mathsf{type} \vdash_\sigma \lambda c.(\cdots) :_\downarrow d \to \mathsf{type} \tag{4}$$

(1) は $\lambda x.M$ の形式に対する型チェックなので、以下の (ΠI) 規則が適用される。

$$\frac{[L]\ \Gamma,\ x : v \vdash_\sigma M :_\downarrow v'\ [L']}{[L]\ \Gamma \vdash_\sigma \lambda x.M :_\downarrow (x{:}v) \to v'\ [L']}\ (\Pi I) \tag{5}$$

したがって、環境に $c : d$ が追加され、

$$d : \mathsf{type}, c : d \vdash_\sigma \left(u{:}\begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{nation}(x) \end{bmatrix}\right) \to \mathbf{cherish}(\cdots) :_\downarrow \mathsf{type} \tag{6}$$

が呼び出される。(6) は Π-type に対する型チェックである。ここで、Π-type は inferable term であり、inferable term は checkable term でもある。これを保証するのが以下の (CHK) 規則である。

$$\frac{[L]\ \Gamma \vdash_\sigma M :_\uparrow v\ [L']}{[L]\ \Gamma \vdash_\sigma M :_\downarrow v\ [L']}\ (CHK) \tag{7}$$

(CHK) 規則を経由して、以下の (ΠF) 規則が適用される。

$$\frac{[L]\ \Gamma \vdash_\sigma A :_\downarrow \mathsf{s}_1\ [L'] \quad A \twoheadrightarrow_\beta v \quad [L']\ \Gamma,\ x : v \vdash_\sigma B :_\downarrow \mathsf{s}_2\ [L'']}{[L]\ \Gamma \vdash_\sigma (x{:}A) \to B :_\uparrow \mathsf{s}_2\ [L'']}\ (\Pi F) \tag{8}$$
$$(\mathsf{s}_1, \mathsf{s}_2 \in \{\mathsf{type}, \mathsf{kind}\})$$

ここで前述の $[L]$、すなわちアスペランド環境は、規則の上段に複数の操作がある場合には、下段の $[L]$ を上段の最初の操作に引き渡し、その結果として更新された $[L']$ を次の操作に引き渡している。そして、上段の操作が全て終了したときのアスペランド環境 $[L'']$ が全体の結果となる。

(ΠF) 規則の上段ではまず、

$$d : \mathsf{type}, c : d \vdash_\sigma \begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{nation}(x) \end{bmatrix} :_\downarrow \mathsf{type} \tag{9}$$

が呼び出される。$\begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{nation}(x) \end{bmatrix}$ は Σ-type であり、Σ-type は inferable term であるので、(CHK) 規則を経由して以下の (ΣF) 規則が適用される。

$$\frac{[L]\ \Gamma \vdash_\sigma A :_\downarrow \mathsf{s}_1\ [L'] \quad A \twoheadrightarrow_\beta v \quad [L']\ \Gamma,\ x : v \vdash_\sigma B :_\downarrow \mathsf{s}_2\ [L'']}{[L]\ \Gamma \vdash_\sigma \begin{bmatrix} x{:}A \\ B \end{bmatrix} :_\uparrow \mathsf{s}_2\ [L'']}\ (\Sigma F) \tag{10}$$
$$(\mathsf{s}_1, \mathsf{s}_2 \in \{\mathsf{type}, \mathsf{kind}\})$$

(ΣF) 規則の上段ではまず、

$$d : \mathsf{type}, c : d \vdash_\sigma \mathbf{entity} :_\downarrow \mathsf{type} \tag{11}$$

を行う。**entity** は定数であり、定数は inferable term であるので、(CHK) 規則を経由して以下の (CON) 規則が適用される。

$$\frac{(c, v) \in \sigma}{[L]\ \Gamma \vdash_\sigma c :_\uparrow v\ [L]}\ (CON) \tag{12}$$

(CON) 規則では、シグネチャ σ から定数 **entity** に対応する型を見つけ、それを返す。いま、$\mathbf{entity} : \mathsf{type} \in \sigma$ であるので、**entity** に対する型推論の結果として type が返り、これにより (11) の型チェックが成功する。(ΣF) 規則の上段では次に、**entity** に対して β-reduction を行う。その結果、環境に $x : \mathbf{entity}$ が追加され、(ΣF) 規則の上段の最後の操作である

$$d : \mathsf{type}, c : d, x : \mathbf{entity} \vdash_\sigma \mathbf{nation}(x) :_\downarrow \mathsf{type} \tag{13}$$

が呼び出される。ここで、$\mathbf{nation}(x)$ は Application であるが、Application には $M_\uparrow M_\downarrow$、すなわち関数部分が inferable であり引数部分が checkable であるような Application と、$M_\downarrow M_\uparrow$、すなわち関数部分が checkable であり引数部分が inferable であるような Application の 2 種類が存在する。このうち、前者は inferable term であり、後者は checkable term である。Application に対して型チェック規則が呼び出された際には、$\mathbf{nation}(x)$ のような inferable である Application には以下の (ΠE) 規則、checkable である Application には後述の $(\to E)$ 規則が適用される。

$$\frac{[L]\ \Gamma \vdash_\sigma M :_\uparrow (x{:}v) \to v'\ [L'] \quad [L']\ \Gamma \vdash_\sigma N :_\downarrow v\ [L''] \quad v'[N/x] \twoheadrightarrow_\beta v''}{[L]\ \Gamma \vdash_\sigma MN :_\uparrow v''\ [L'']}\ (\Pi E) \tag{14}$$

(ΠE) 規則の上段ではまず、Application の関数部分である **nation** の型を推論する。**nation** は定数であるので前述の (CON) 規則によって型が $\mathbf{entity} \to \mathsf{type}$ であると分かる。(ΠE) 規則では次に、

$$d : \mathsf{type}, c : d, x : \mathbf{entity} \vdash_\sigma x :_\downarrow \mathbf{entity} \tag{15}$$

が呼び出される。x は変数であり、変数は inferable term であるので、(CHK) 規則を経由して以下の (VAR) 規則が適用される。

$$\frac{(x, v) \in \Gamma}{[L]\ \Gamma \vdash_\sigma x :_\uparrow v\ [L]}\ (VAR) \tag{16}$$

(VAR) 規則では、環境 Γ から変数 x に対応する型を見つけ、それを返す。いま、$x : \mathbf{entity} \in \Gamma$ であるので、x に対する型推論の結果として **entity** が返り、これにより (15) の型チェックが成功する。(ΠE) 規則の上段では最後に、Application の返り値の型である type に対して β-reduction を行う。type に β-reduction を行った結果は type であるので、(14) の (ΠE) 規則の返り値は type となる。これにより、(13) の型チェックが成功するので、(8) の (ΠF) の上段では次に、$\begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{nation}(x) \end{bmatrix}$

に β-reduction を行う。その結果、$u:\begin{bmatrix}x{:}\mathbf{entity}\\ \mathbf{nation}(x)\end{bmatrix}$ を環境に追加し、以下の (17) の型チェックを行う。ただし、以下、$\Gamma' \stackrel{def}{\equiv} d:\mathsf{type}, c:d, x:\mathbf{entity}, u:\begin{bmatrix}x{:}\mathbf{entity}\\ \mathbf{nation}(x)\end{bmatrix}$ とする。

$$\Gamma' \vdash_\sigma \mathbf{cherish}(\pi_1(u), \pi_1(@_1(c,u):[\,\dots\,])) :\downarrow \mathsf{type} \tag{17}$$

$\mathbf{cherish}(\cdots)$ は inferable な Application であるので、前述のように (ΠE) 規則が適用され、関数部分 **cherish** の型を (CON) 規則によって推論する。**cherish** の型は $\begin{bmatrix}\mathbf{entity}\\ \mathbf{entity}\end{bmatrix} \to \mathsf{type}$ であるので、次に

$$\Gamma' \vdash_\sigma (\pi_1(u), \pi_1(@_1(c,u):[\,\dots\,])) :\downarrow \begin{bmatrix}\mathbf{entity}\\ \mathbf{entity}\end{bmatrix} \tag{18}$$

を呼び出す。$(\pi_1(u), \pi_1(@_1(c,u):[\,\dots\,]))$ は Pair であり、2 項以上の述語を扱う際には必ずこの項が現れる。Pair に対する型チェック規則として、以下の (ΣI) 規則が存在する。

$$\frac{[L]\ \Gamma \vdash_\sigma M :\downarrow v\ [L'] \quad v'[M/x] \twoheadrightarrow_\beta v'' \quad [L']\ \Gamma \vdash_\sigma N :\downarrow v''\ [L'']}{[L]\ \Gamma \vdash_\sigma (M,N) :\downarrow \begin{bmatrix}x{:}v\\ v'\end{bmatrix}\ [L'']}\ (\Sigma I) \tag{19}$$

しかし、(ΣI) 規則では Pair の型は Σ-type となっている。そのため、(18) での型チェックに (ΣI) 規則を適用することはできず、(CHK) 規則を通じて

$$\Gamma' \vdash_\sigma (\pi_1(u), \pi_1(@_1(c,u):[\,\dots\,])) :\uparrow \tag{20}$$

を呼び出すことになり、Pair に対する型推論規則として以下の (∧I) 規則が適用される。

$$\frac{[L]\ \Gamma \vdash_\sigma M :\uparrow v\ [L'] \quad [L']\ \Gamma \vdash_\sigma N :\uparrow v'\ [L'']}{[L]\ \Gamma \vdash_\sigma (M,N) :\uparrow \begin{bmatrix}v\\ v'\end{bmatrix}\ [L'']}\ (\wedge I) \tag{21}$$

(∧I) 規則の上段では、まず

$$\Gamma' \vdash_\sigma \pi_1(u) :\uparrow \tag{22}$$

が呼び出される。$\pi_1(u)$ は Projection であるので、以下の (ΣE) 規則が適用される。

$$\frac{[L]\ \Gamma \vdash_\sigma M :\uparrow \begin{bmatrix}x{:}v\\ v'\end{bmatrix}\ [L']}{[L]\ \Gamma \vdash_\sigma \pi_1 M :\uparrow v\ [L']}\ (\Sigma E) \tag{23}$$

$$\frac{[L]\ \Gamma \vdash_\sigma M :\uparrow \begin{bmatrix}x{:}v\\ v'\end{bmatrix}\ [L'] \quad v'[\pi_1 M/x] \twoheadrightarrow_\beta v''}{[L]\ \Gamma \vdash_\sigma \pi_2 M :\uparrow v''\ [L']}\ (\Sigma E) \tag{24}$$

ここで、(ΣE) 規則には (23), (24) の 2 種類が存在する。Projection によって Pair の第 1 要素を取るのが (23) の (ΣE) 規則であり、第 2 要素を取るのが (24) の (ΣE) 規則である。(22) の型推論では、(23) の (ΣE) 規則が適用される。(ΣE) 規則は、Projection の引数である u の型を推論し、その型が Σ-type であったならば、その第 1 要素に対応する部分の型を返す。(22) の型推論では、u の型は前述の (VAR) 規則によって $u:\begin{bmatrix}x{:}\mathbf{entity}\\ \mathbf{nation}(x)\end{bmatrix}$ であることが分かる。これは Σ-type であるので、その第 1 要素に対応する部分の型である **entity** を返す。(21) の $(\wedge I)$ 規則では次に、

$$\Gamma' \vdash_\sigma \pi_1(@_1(c,u):\begin{bmatrix}y{:}\mathbf{entity}\\ \mathbf{kingOf}(\cdots)\end{bmatrix}) :\uparrow \tag{25}$$

が呼び出される。$\pi_1(@_1(c,u):[\,\dots\,])$ は Projection であるので、前述の (ΣE) 規則が適用され、

$$\Gamma' \vdash_\sigma @_1(c,u):\begin{bmatrix}y{:}\mathbf{entity}\\ \mathbf{kingOf}(\cdots)\end{bmatrix} :\uparrow \tag{26}$$

が呼び出される。$@_1(c,u):\begin{bmatrix}y{:}\mathbf{entity}\\ \mathbf{kingOf}(\cdots)\end{bmatrix}$ はアノテーションであるので、以下の (ANN) 規則が適用される。

$$\frac{[L]\ \Gamma \vdash_\sigma A :\downarrow \mathsf{s}\ [L'] \quad A \twoheadrightarrow_\beta v \quad [L']\ \Gamma \vdash_\sigma M :\downarrow v\ [L'']}{[L]\ \Gamma \vdash_\sigma M : A :\uparrow v\ [L']}\ (ANN) \tag{27}$$
$$(\mathsf{s} \in \{\mathsf{type}, \mathsf{kind}\})$$

(ANN) 規則の上段ではまず、

$$\Gamma' \vdash_\sigma \begin{bmatrix}y{:}\mathbf{entity}\\ \mathbf{kingOf}(\cdots)\end{bmatrix} :\downarrow \mathsf{type} \tag{28}$$

が呼び出される。$\begin{bmatrix}y{:}\mathbf{entity}\\ \mathbf{kingOf}(\cdots)\end{bmatrix}$ は Σ-type なので、前述のように (CHK) 規則を経て (ΣF) 規則が適用される。まず、**entity** $:\downarrow$type の型チェックは (11) と同様に成功するので、$y:\mathbf{entity}$ を環境 Γ' に追加し、

$$\Gamma', y:\mathbf{entity} \vdash_\sigma \mathbf{kingOf}(y, \pi_1(@_2(c,u):[\,\dots\,])) :\downarrow \begin{bmatrix}\mathbf{entity}\\ \mathbf{entity}\end{bmatrix} \tag{29}$$

が呼び出される。$\mathbf{kingOf}(\cdots)$ は inferable な Application であるので、前述のように (ΠE) 規則が適用され、関数部分 **kingOf** の型を (CON) 規則によって推論する。**kingOf** の型は $\begin{bmatrix}\mathbf{entity}\\ \mathbf{entity}\end{bmatrix} \to \mathsf{type}$ であるので、次に

$$\Gamma', y:\mathbf{entity} \vdash_\sigma (y, \pi_1(@_2(c,u):\begin{bmatrix}z{:}\mathbf{entity}\\ \neg\mathbf{human}(z)\end{bmatrix})) :\downarrow \mathsf{type} \tag{30}$$

を行う。$(y, \pi_1(@_2(c,u):\begin{bmatrix}z{:}\mathbf{entity}\\ \neg\mathbf{human}(z)\end{bmatrix}))$ は Pair であるので、前述のように $(\wedge I)$ 規則が適用され、第 1 要素 y の型を (VAR) 規則によって推論する。y の型は **entity** であるので、次に第 2 要素である $\pi_1(@_2(c,u:\begin{bmatrix}z{:}\mathbf{entity}\\ \neg\mathbf{human}(z)\end{bmatrix}))$ の型を (ΣE) 規則によって推論する。$@_2(c,u):\begin{bmatrix}z{:}\mathbf{entity}\\ \neg\mathbf{human}(z)\end{bmatrix}$ はアノテーションであるので、前述のように (ANN) 規則が適用され、アノテーション部分である $\begin{bmatrix}z{:}\mathbf{entity}\\ \neg\mathbf{human}(z)\end{bmatrix}$ が型 type を持つかどうか、(CHK) 規則を経て (ΣF) 規則によって確かめる。**entity** の型は (CON) 規則により type であると分かる。$\neg\mathbf{human}(z)$ については、$\neg\mathbf{human}(z) \stackrel{def}{\equiv} \mathbf{human}(z) \to \bot$ と定義されているので、

$$\Gamma', y:\mathbf{entity} \vdash_\sigma \mathbf{human}(z) \to \bot :\downarrow \mathsf{type} \tag{31}$$

が呼び出され、以下の $(\to F)$ 規則が適用される。

$$\frac{[L]\ \Gamma \vdash_\sigma A :\downarrow \mathsf{s}_1\ [L'] \quad [L']\ \Gamma \vdash_\sigma B :\downarrow \mathsf{s}_2\ [L'']}{[L]\ \Gamma \vdash_\sigma A \to B :\uparrow \mathsf{s}_2\ [L'']}\ (\to F) \tag{32}$$
$$(\mathsf{s}_1, \mathsf{s}_2 \in \{\mathsf{type}, \mathsf{kind}\})$$

$(\to F)$ 規則は前述の (ΠF) 規則とほぼ同じ働きをするので、まず $\mathbf{human}(z)$ が型 type を持つのかどうか確かめ、それから $\bot$ が型 type を持つのかどうか確かめる。$\mathbf{human}(z)$ については、前述の (13) などと同様に (ΠE) 規則と (VAR) 規則によって型 type を持つことが確かめられる。$\bot$ については、$\bot$ は inferable term であるので、(CHK) 規則を通じて以下の $(\bot F)$ 規則を適用することにより、型 type を持つことが確かめられる。

$$\frac{}{[L]\ \Gamma \vdash_\sigma \bot :\uparrow \mathsf{type}\ [L]}\ (\bot F) \tag{33}$$

これにより、$\begin{bmatrix}z{:}\mathbf{entity}\\ \neg\mathbf{human}(z)\end{bmatrix}$ が型 type を持つことが分かったので、次に、(ANN) 規則の残りの操作を行う。すなわち、

$$\Gamma', y:\mathbf{entity} \vdash_\sigma @_2(c,u) :\downarrow \begin{bmatrix}z{:}\mathbf{entity}\\ \neg\mathbf{human}(z)\end{bmatrix} \tag{34}$$

を呼び出す。$@_2(c,u)$ は関数部分 $@_2$ が checkable、引数部分 (c,u) が inferable であるので checkable な Application であり、以下の $(\to E)$ 規則が適用される。

$$\lambda c.\left(u{:}\begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{kingOf}(x,\mathbf{sweden}) \end{bmatrix}\right) \to \mathbf{cherish}(\mathbf{sweden}, \pi_1(@_1(c,u)) : \begin{bmatrix} y{:}\mathbf{entity} \\ \mathbf{kingOf}(y, \pi_1(@_1(c,u)) : \begin{bmatrix} z{:}\mathbf{entity} \\ \neg\mathbf{human}(z) \end{bmatrix})) \end{bmatrix}) \tag{39}$$

$$d : \mathsf{type}, c : d, u : \begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{kingOf}(x,\mathbf{sweden}) \end{bmatrix} \vdash @_1 : \begin{bmatrix} d \\ \begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{kingOf}(x,\mathbf{sweden}) \end{bmatrix} \end{bmatrix} \to \begin{bmatrix} y{:}\mathbf{entity} \\ \mathbf{kingOf}(y, \pi_1(@_2(c,u)) : \begin{bmatrix} z{:}\mathbf{entity} \\ \neg\mathbf{human}(z) \end{bmatrix})) \end{bmatrix} \tag{40}$$

$$d : \mathsf{type}, c : d, u : \begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{kingOf}(x,\mathbf{sweden}) \end{bmatrix}, y : \mathbf{entity} \vdash @_2 : \begin{bmatrix} d \\ \begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{kingOf}(x,\mathbf{sweden}) \end{bmatrix} \end{bmatrix} \to \begin{bmatrix} z{:}\mathbf{entity} \\ \neg\mathbf{human}(z) \end{bmatrix} \tag{41}$$

図 3: (2c) If sweden has a king, sweden cherishes its king. の意味表示

$$\frac{[L]\ \Gamma \vdash_\sigma N :_\uparrow v\ [L'] \quad [L']\ \Gamma \vdash_\sigma M :_\downarrow v \to v'\ [L'']}{[L]\ \Gamma \vdash_\sigma MN :_\downarrow v'\ [L'']}\ (\to E) \tag{35}$$

$(\to E)$ 規則ではまず、引数部分 (c,u) の型を推論する。(c,u) は Pair であるので $(\wedge I)$ 規則が適用され、(VAR) 規則によって $c : d, u : \begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{human}(x) \end{bmatrix}$ であると分かる。したがって (c,u) の型は $\begin{bmatrix} d \\ \begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{human}(x) \end{bmatrix} \end{bmatrix}$ であると推論できる。$(\to E)$ 規則では次に、

$$\Gamma', y : \mathbf{entity} \vdash_\sigma @_2 :_\downarrow \begin{bmatrix} d \\ \begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{human}(x) \end{bmatrix} \end{bmatrix} \to \begin{bmatrix} z{:}\mathbf{entity} \\ \neg\mathbf{human}(z) \end{bmatrix} \tag{36}$$

を呼び出す。$@_2$ は underspecified term であるので、以下の (ASP) 規則が適用される。

$$\frac{}{[L]\ \Gamma \vdash_\sigma @_i :_\downarrow v\ [L, (\Gamma \vdash @_i : v)]}\ (ASP) \tag{37}$$

(ASP) 規則では、型チェックを行う $@_i$ に対応するジャッジメントをアスペランド環境に追加する。(36) では、

$$d : type, c : d, u : [\,\dots\,], y : \mathbf{entity} \vdash @_2 : \begin{bmatrix} d \\ [\,\dots\,] \end{bmatrix} \to \begin{bmatrix} z{:}\mathbf{entity} \\ \neg\mathbf{human}(z) \end{bmatrix}$$

がアスペランド環境に追加される。この規則により、未知であった $@_2$ の型を定めることができる。

ところで、先ほどの推論で $@_2$ の型を定めることができたのは、$@_2$ を含む項、すなわち $@_2(c,u)$ にアノテーションがついていたからである。もしアノテーションがついていなかったと仮定すると、先ほどの推論の途中で $@_2(c,u) :_\uparrow$ という操作を行わなくてはならない。しかし、$@_2$ は inferable term ではため、$@_2(c,u) :_\uparrow$ という操作で $@_2$ の型を推論することはできない。$@_i$ の型を決定できるのは、$@_2(c,u)$ のような $@_i$ を含む項が型チェック規則で呼び出されたときのみなのである。そして、$@_i$ を含む項が型チェック規則で呼び出されるようにするためには、アノテーションが不可欠なのである。

これまでの推論で、(36) の型チェックが成功するので、(34) の型チェックにも成功する。また、これにより $\pi_1(@_2(c,u : \begin{bmatrix} z{:}\mathbf{entity} \\ \neg\mathbf{human}(z) \end{bmatrix}))$ の型が **entity** であることが分かり、(30) の型チェックに成功する。

以下、これまでの推論と同様にして、$(\to E)$ 規則、$(\wedge I)$ 規則、(VAR) 規則、(ASP) 規則などを用いることによって残る $@_1$ の型も判明し、以下のような結果が得られる。

$$d : \mathsf{type}, c : d, u : \begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{nation}(x) \end{bmatrix} \vdash @_1 : \begin{bmatrix} d \\ \begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{nation}(x) \end{bmatrix} \end{bmatrix} \to \begin{bmatrix} y{:}\mathbf{entity} \\ \mathbf{kingOf}(y, \pi_1(@_2(c,u)) : \begin{bmatrix} z{:}\mathbf{entity} \\ \neg\mathbf{human}(z) \end{bmatrix})) \end{bmatrix} \tag{38}$$
$$d : \mathsf{type}, c : d, u : \begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{nation}(x) \end{bmatrix}, y : \mathbf{entity} \vdash @_2 : \begin{bmatrix} d \\ \begin{bmatrix} x{:}\mathbf{entity} \\ \mathbf{nation}(x) \end{bmatrix} \end{bmatrix} \to \begin{bmatrix} z{:}\mathbf{entity} \\ \neg\mathbf{human}(z) \end{bmatrix}$$

4. 実装：前提投射の計算

第 3 節で述べたアルゴリズムをプログラミング言語 Haskell を用いて実装した。

このアルゴリズムでは、文 (3) の意味表示（図 1）を入力すると、(38) のアスペランド環境を出力するが、この表示は (3) の前提を正しく与えている。まず、$@_2$（*it* に対応する）であるが、仮に「国家は人ではない」という知識を表す証明項を nh とすると、$\lambda c.(\pi_1\pi_2(c), nh(\pi_2\pi_2(c)))$ という証明項が指定された型を持つので、$@_1$ の型に現れる $@_2$ はこの証明項でおきかえることができる（その結果、**kingOf** の第 2 引数は $\pi_1 u$ となる）。このとき $@_1$ の型には自由変更 u が現れているため、(ΠI) 規則によって global context の中の u を discharge する。この操作によって得られる型が意味するのは「全ての国に、そこの国王が存在する」という命題である。これを global context に追加することが、いわゆる local accommodation に相当する。

また、このアルゴリズムは (3) に限らず、様々な文の前提・照応投射を正しく計算することができる。たとえば (2c) の DTS における意味表示は (39) であるが、これを入力とすると、(40)(41) を出力する。このうち (41) は、*it* が *Sweden* を指すとすれば、(40) の $\pi_1(@_2(c,u))$ は **sweden** に置き換わる。すると、$@_1$ は単に $\lambda c.\pi_2(c)$（すなわち、条件文の前提をそのまま返す関数）で置き換えれば良いことがわかる。したがって、(39) は全体としては前提を持たないことが正しく予測される。紙面の都合により (2a)(2b) における前提投射の計算の説明は省略する。

5. おわりに

本研究では、依存型意味論の型チェック・型推論アルゴリズムを定式化し、実装した。これにより、局所適用のような複雑な場合を含む前提投射の問題が統一的に計算可能になったと言える。

参考文献

[1] Bekki, Daisuke. 2014. Representing Anaphora with Dependent Types. In Logical Aspects of Computational Linguistics (8th international conference, LACL2014, Toulouse, France, June 2014 Proceedings), N.Asher and S.Soloviev (Eds), LNCS 8535, pp.14-29, Springer, Heiderburg.

[2] Bekki, Dai suke and Eric McCready. 2014. CI via DTS. In Proceedings of LENLS11, pp.110-123.

[3] Löh, A., C.McBride, and W.Swierstra. 2010. A Tutorial Implementation of a Dependently Typed Lambda Calculus. Fundamenta Informaticae - Dependently Typed Programming, Vol. 102, No. 2, pp. 177-207.

[4] Martin-Löf, Per. 1984. Intuitionistic Type Theory. vol. 17. Naples: Italy: Bibliopolis.

[5] Van der Sandt. 1992. Presupposition projection as anaphora resolution. Journal of Semantics, Vol.9, pp. 333-377.