ICOT Technical Memorandum: TM-0051

TM-0051

状況意味論における計算モデルとその実現

向井国昭　　安川秀樹
三吉秀夫　　平川秀樹

April, 1984

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

状況意味論における計算モデルとその実現

目次 ページ

1 はじめに

本メモは状況意味論（situation semantics）の本格的な実現を目指すためのいくつかの実験の報告である。我々はいくつかの実験で状況意味論の考え方に慣れようと思う。それから自然言語理解の新しい実現方式をさぐりたいと思う。

本版で実現した機能は次の通り：

(1) コンストレイント
(2) アンカリング
(3) ロール
(4) 下降型と上昇型混合の簡単な探索戦略

まだ実現していない機能は次の通り：

(1) コンストレイント型
(2) same（等号）
(3) no（否定文）
(4) その他

最初はわかりやすい実現をと心がけたので効率は二の次になっている。

ロールの処理とアンカリングの処理にPrologのunificationが使えるようにデータ構造を工夫した点が本メモの主内容である。それはPrologの項とアンカーの対（closureと呼んでいる）で表現されるものである。不定要素を単にPrologの論理変数に置き換えるだけではロールがうまく扱えなかったために導入したものである。

インタプリタの全プログラム・リストと例題を付録とする。はじめに例題を見ればインタプリタのイメージが容易に把めると思われる。また状況意味論の紹介もかねて主な用語の説明をまとめておいた。

2 状況意味論

2.1 概要

米国の Barwise & Perry が 1981 年頃から発表しはじめた自然言語のモデル論的意味論である [Barwise 81a, 81b, 81c, 81d, 83

Barwise は admissible 集合の仕事 [Barwise 85] で良く知られた数理論理学者であり Perry は哲学者である。両者とも昨年(83)発足した研究所 CSLI のメンバーである。特に Barwise はその所長である。状況意味論が CSLI の中でどのような位置付けにあるか興味深い。Barwise 自身は ACL 1983 Vol 9, NO1. の CSLI の紹介の中で D. Scott の領域理論 [Scott 82] と状況意味論の比較の中から自然言語の意味論とプログラミング言語の意味論との統一的な理論を作るという方向を一例としてあげている。

状況意味論は世界を状況の入れものと考えている。状況はいくつかの命題が成っており それらはいくつかの公理を満たしている。命題は関係, 個体, ロケーション, 不定要素 から構成されているものである。不定要素をのぞくそれらは プリミティブ である。特別なタイプの命題としてコンストレイントがある。コンストレイントは内容的には状況間の2項関係であり 世界に存在を許される状況を制約している。コンストレイントは entailment の形をしており, ある意味で状況の計算規則を与えている。

状況意味論はモンタギュ意味論と同じくモデル論的意味論であるがモンテギュと次の点が少なくとも異なる。

(1) 可能世界意味論ではない。
(2) 文の意味は状況間の2項関係である。
(3) 論理的同値な文成分の置き換え法則に関連する理論的困難がない (believe, know などの attitude verb が扱える)。
(4) (2)とも関連して 意味の計算量が大幅に改良されている [Barwise 81a].

状況意味論のメタ理論は通常の ZF 集合論ではなく KPU [Barwise 75] であることが計算可能な意味論ならしめている点も重要な特徴である。([Mukai 77] も参照のこと).

2.2 モンタギュ意味論の論理的同値の困難

MG（モンタギュ意味論）は可能世界意味論とも言われている。すなわちいくつかの可能世界から成る集合Sを考え文の意味は集合S上の特性函数であると定義する。言い換えると文が成り立つ可能世界をすべて指定することが文の意味である。ある文の補文をそれと等価な他の文で置き換えて得られる文の意味はもとの文の意味と変わらないとするのがいわゆるフレーゲの原理である。MGはフレーゲの原理の上に立っている。

しかしながら believe, doubt などの動詞を考えるとこの原理は成り立たないことが知られている。

(1) $\mathcal{G}$ believe that ϕ_1.
(2) $\mathcal{G}$ believe that not ϕ_2.

今，ϕ_1とϕ_2が等価な文であれば(1)と(2)からは矛盾が出る。したがってϕ_1とϕ_2が等価でないことを知った上でないと(1)と(2)を同一の人物が発言することは許されないことになる。しかしこれは現実の自然言語の使用とは合わない。

MGの文の意味の計算量にも問題がある。ある可能世界における文の真偽値は一般には，到達可能な他の可能世界の真偽値に依存して決まる。文の意味は可能世界の間を走りまわらなければ定まらない。これは計算機処理に向いておらず，また現実の言語使用の効率の良さを説明できない。

以上の2点がMGの分かりやすい問題点であると思われる。状況意味論では believe know などの attitude verb は個体と状況の間の2項関係を表わすと考える。

2.3 状況意味論の素材

状況意味論の素材は次の3つの集合 A, R, L により与えられる：

(1) A : 個体の集合
(2) R : 関係の集合
(3) L : ロケーションの集合．ここでロケーションとは時空領域を指す．L には時間的関係や空間的関係を表わす2項関係が与えられているが本メモの範囲では必要がない．最大のロケーションを仮定しそれを l_u とする．

2.4 状況の構成

定義．次の形の3組をそれぞれ（本メモでは）命題と呼ぶ。

$$(l, \langle r, a_1, a_2, \cdots, a_n \rangle, yes) \quad \cdots (1)$$

$$(l, \langle r, a_1, a_2, \cdots, a_n \rangle, no) \quad \cdots (2)$$

内容的には時空領域 l においてオブジェクト $a_1, \cdots, a_n$ が r 項関係 r にある（yes の場合）ことを表わしている。no の場合は同様に l において $a_1, a_2, \cdots, a_n$ が r の関係にないことを表わしている。

定義．命題から成る集合を状況（situation）と呼ぶ．

定義．(1) と (2) のように yes と no だけが異なるような命題の対を含まないとき その状況はコヒーレント（coherent）であるという．ふたつの状況の和集合がコヒーレントであるとき，それらの状況は両立する という。

2.5 不定要素（indeterminate）の導入

個体，関係，あるいはロケーション それぞれについて不定要素を導入する．命題は不定要素を含んで良いとする．不定要素を含んだ状況を特にイベント型（event type）と呼ぶ。

不定要素を含まない状況のことをcoe（course of event）と呼ぶ。

不定要素に個体を対応させる部分函数をアンカー（anchor）と呼ぶ。イベント型Eをアンカーfにしたがい不定要素を「代入」して得られるイベント型をE[f]で表わす。E[f]がcoeのとき fをEのトータル・アンカーと呼ぶ。

上で導入した不定要素をとくに基本不定要素という。一般の不定要素は次のように定義する。

定義．
(1) 基本不定要素は不定要素である。
(2) 不定要素xとイベント型Eの対 <x, E> も不定要素である。

<x, E> なる形の不定要素のことをロール（role）と呼ぶ。イベント型はロールを使って良い。

イベント型は不定要素を用いて定義され，ロールはイベント型を用いて定義される。すなわちイベント型とロールは互いに他を用いて定義されていることに注意する。

eとEをそれぞれcoeとイベント型とする．あるアンカーfが存在して E[f]⊂e となるとき eはイベント型Eに属する，あるいは eは型Eであるという。

2.6 コンストレイント

次の形の命題 C を コンストレイント と呼ぶ。

$$C := (l_u, \langle involve, E, S \rangle, yes)$$

ここで E はイベント型，S は イベント型の集合 である。

$$S = \{E_1, E_2, \cdots, E_m\}$$

定義． C は上のコンストレイントとする。

(1) coe e_0 が 型 E に属すとき e_0 は C に関して meaningful であるという。そのような e_0 の全体を MF_C で示す。

(2) $E[f] \subset e_0$ となるような任意のアンカー f について，e_1 が どれかの $E_i[f]$ に属するとき e_1 を e_0 の C に関する meaningful option であるという。

(3) e_0 と e_1 を coe とする。次の (a) または (b) を満たすとき e_0 は e_1 を precludeするという。

(a) e_1 が e_0 と両立しない．

(b) e_0 が 型 $E[f]$ に属すようなあるアンカー f が存在して，f の任意の拡張 g に対して e_1 が 型 $-S[g]$ に属す。

ここで スキーマ $T = \{F_1, F_2, \cdots, F_m\}$ に対して $-T$ を次のように定義する。

$$-T = \{\{-P_1, -P_2, \cdots, -P_m\} \mid P_i \in F_i, 1 \leq i \leq m\}$$

ここで，$-(l, \alpha, yes) = (l, \alpha, no)$ ，$-(l, \alpha, no) = (l, \alpha, yes)$ と規約する。

2.7 状況の構造 (structure of situations)

状況のコレクション(collection) M と M_0 の対 $\mathfrak{m}=(M, M_0)$ が次の条件を満たすとき $\mathfrak{m}$ を状況の構造という。

(1) $M_0 \subset M$

(2) M_0 の各状況はコヒーレントである。

(3) M_0 の各状況の部分状況(部分集合のこと)は M に属す。

(4) $X \subset M$ が集合ならばある $e \in M_0$ が存在し, X の各元は e に含まれる:

$$\bigcup X \subset e$$

(5) M は M の任意のコンストレイント C に対して C を満足(respect)する。

ここで任意の $e_0 \in M_0$ の MF_C に対してある $e \in M$ が存在して e が e_0 の C に関する meaningful option であるとき M は C を満た(respect)しているという。

M の元を事実的(factual)状況, M_0 の元を現実的(actual)状況と呼ぶ。

状況の構造 $\mathfrak{m}=(M, M_0)$ が与えられたとき実在的(real)状況はそこで成りたっているすべての命題からなる状況によってclassifyされていると考える。これが現実的状況の意味である。

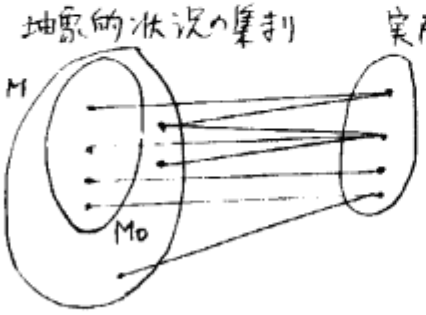

図1 地象的状況と実在的状況の関係(classify)

3 状況の計算アルゴリズムの実現

今回は状況意味論の実験を目的として一部を実現したのでその概要を説明する。

3.1 データ構造

(a) 基本不定要素は ?(名前) の形である。

(b) ロールrは次の形で定義する。

def-role(r, α, ε).

これは

$r := \langle \alpha, \varepsilon \rangle$

を表わしている。 ε はイベント型名である。ロールは $\mathfrak{R}$(名前)の形である。

(c) イベント型 $\varepsilon := \delta$ は次の形で assert しておく。

def-event(ε, δ).

(d) 状況のデータ構造は命題のリストで表現する。

(e) 命題 $(l, \langle r, a_1, a_2, \cdots, a_n \rangle, t)$ は

$$(l', (r', a_1', a_2', \cdots, a_n'), t')$$

で表現する。但し, l', r', a_i', t' は l, r, a_i, t の表現であるとする。

(f) イベント型を表わす不定要素も都合により導入している。現時点における状況意味論では許されていない。

3.2 入力と出力

アルゴリズムの入力は (1)〜(4) である。

(1) イベント型の名前の定義　　$\varepsilon_1 := \delta_1$, $\varepsilon_2 = \delta_2$, ---

(2) ロールの名前の定義

$r_1 := \langle \varkappa_1, E_1 \rangle$, $r_2 := \langle \varkappa_2, E_2 \rangle$, ---

(3) coe X　（但し X はコンストレイントを含む）

(4) イベント型 q

アルゴリズムの出力は 次の条件を満たす すべてのアンカー f である。但し そのようなアンカーが存在しなければ fail する。

(1) $G[f] \subset hull(X)$. 但し $hull(X)$ は次のように帰納的に定義される coe である。

定義. $hull(X) = X_1 \cup X_2 \cup \cdots \cup X_n \cup \cdots$

(a) $X_1 \stackrel{def}{=} X$

(b) $X_{i+1} \stackrel{def}{=} X_i \cup MO_i$

(c) $MO_i \stackrel{def}{=} \cup \{ mo \mid$ mo は X_i のあるコンストレイント C に関する X_i の meaningful-option である $\}$。

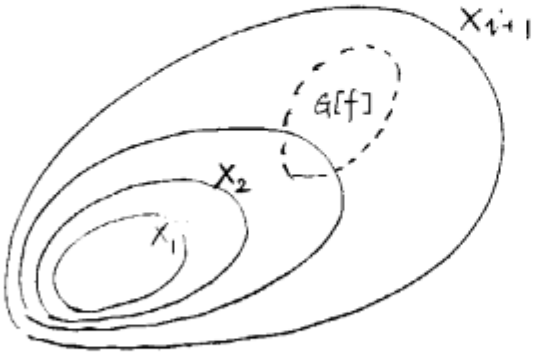

図2 状況計算のイメージ

3.3 closure

状況の不定要素は Prologの論理変数で表現すれば都合が良いように思われるが，実際はうまく行かないようだ。それは次の理由によるものと思われる：

(1) 不定要素は名前（手づら）が意味を持つ。
(2) 同じイベント型から複数の異なるロールが定義できる：

$r := \langle x, E \rangle$
$s := \langle y, E \rangle$

そこでとりあえず，Prologの項と環境 e の対で状況を内部表現した。より正確には次のような対応をとっている。すなわち，

$E(x_1, x_2, \cdots, x_n)$ ：イベント型
f ：アンカー
$f(x_i) = a_i$ または 未定義 $(1 \leq i \leq n)$

が与えられたとき $E[f']$ と f' の対で $E[f]$ を表現する。但し

$$f'(x_i) = \begin{cases} a_i & : f(x_i) \text{が定義されている。} \\ X_i & (\underline{\text{Prologの未使用の自由変数}}) : \text{その他の場合} \end{cases}$$

とする。対 $(E[f'], f')$ のことを closure と仮に呼ぶ。closure を用いることにより アンカー処理を unification 処理に還元している。これは主にアンカー処理の記述の手間を省くためのものである。しかし処理の設計の見通しを良くする効果もあった。次頁に closure の例をあげる。

例　π と f をそれぞれ次の命題とアンカーとする.

$$\pi := at(l, <?r, ?a>, yes)$$

$$f(?r) = p$$
$$f(?a) = b$$

但し，$?r$ と $?a$ のみが不定要素とする.

$$\pi[f] = at(l, <p, b>, yes)$$

π の closure は (π', env) となる。

$$\pi' := at(l, <R, A>, yes)$$
$$env := \{ ?r = R, ?a = A \}$$

R と A は Prolog の未使用な自由変数である。　$\pi[f]$ に相当する closure は　(π'', env') となる.

$$\pi'' := at(l, <p, b>, yes)$$
$$env' := \{ ?r = p, ?a = b \}$$

3.4 状況の探索

最初は 次の形の 各コンストレイント C について，

$C := at(l_u, (involve, E, S), yes)$
S は シングルトン のスキーマをとる不定要素とする.

与えられた coe X の C に関する meaningful-option を X に追加して得られる状況を改めて X とおく。これを可能なだけ繰り返す。(前向き (forward) 探索)

次のステップは 与えられた ゴールのイベント型 G を上で得られた X に対して 逆向き (backward) に探索して アンカー を求める. この探索は Prolog の下降型探索と ほぼ同じである。

3.5 ロールのアンカリング

必要なデータは 環境 env と 関連するロールの定義である:

$$env = \{ r_1 = v_1,\ r_2 = v_2,\ \cdots,\ r_n = v_n \}$$

$$r_i = \langle x_i, E_i \rangle \quad (1 \leq i \leq n)$$

但し，便宜上 r_i はすべて ロールとする。v_i は Prolog の項である。v_i が Prolog の変数であることは アンカーとしての env が r_i に対して未定義であることを意味している。

各 E_i の closure を (E_i', env_i) とおき，'つなぎ'の環境を

$$env' = \{ x_1 = v_1,\ x_2 = v_2,\ \cdots,\ x_n = v_n \}$$

おく。 すべての環境 の和をとり それを env'' とおく。

$$env'' = env' \cup env_1 \cup \cdots \cup env_n$$

$x = u$, $x = w$ なる形の env'' の元の各組み合わせについて u w を unify する。与えられた coe X に各 E_i をアンカリングして得られる結果の env, env'' をそれぞれ同じ記号で書く。env と env'' を合わせて得られる環境をアンカー f とする。f は次の条件を満たしていることが保証されたわけである。

$$f(v_i) = f(x_i) \qquad (1 \leq i \leq n)$$

3.6 環境のマージ

ふたつの環境 env_1 と env_2 をマージして新らしい環境 env_3 を作り出す操作を定義しておく。これは上でも既に使用されている。

$$env_1 := \{ x_1 = v_1, x_2 = v_2, \cdots, x_n = v_n \} \quad (i \neq j \Rightarrow x_i \neq x_j)$$

$$env_2 := \{ y_1 = u_1, y_2 = u_2, \cdots, y_m = u_m \} \quad (i \neq j \Rightarrow y_i \neq y_j)$$

$$env_3 := \{ z_1 = w_1, z_2 = w_2, \cdots, z_p = w_m \} \quad (i \neq j \Rightarrow z_i \neq z_j)$$

但し $\{z_1, z_2, \cdots, z_p\} = \{x_1, x_2, \cdots, x_n\} \cup \{y_1, y_2, \cdots, y_m\}$ であり w_i は次のようにして決められるものとする。

(1) $z_i = x_j = y_k$ なる j と k が存在するときは w_i は v_j と u_k を unify して得られる結果の項とする。
(2) $z_i = x_j$ のときは $w_i := v_j$ と定義する。
(3) $z_i = y_k$ のときは $w_i := u_k$ と定義する。

4 主要述語の機能概要

fcoe(x, y, a) : 与えられたイベント型 x, coe y に対して x[a] が y の(間接的)meaningful option を付け加えて得られる coe に含まれるような アンカー a を返す.

solve(x, y) : ゴール x を coe y に対して下降型に解く。

extend-situation(x, y) : coe x の特別な形のコンストレイントに関する meaningful-option を x に加えることを繰り返して拡張された coe y を得る。

meaningful-option(e, s, y, mo) : mo は コンストレイント c に関する y の meaningful-option である.
c := at(lu, (involve, e, s), yes)

anchor-roles(a, e) : $a = \{r_1 = v_1, \cdots, r_n = v_n\}$ の各ロール r_i を e にアンカリングする. e は coe である。

connect-roles(a, d) : a を環境 $\{r_1 = v_1, \cdots, r_n = v_n\}$ とし ロール r_i が $\langle x_i, d_i \rangle$ と定義されているとする. ロール r_i と 不定要素 x_i の値が ばらず等しくなるように それらの値を unify する. $d = \{d_i$ の closure のボディ$\}$ を返す。

closure(x, y, a) : イベント型 x の closure を(y, a)して, y, a を返す。

merge(x, y, z) : 環境 $x = \{r_1 = v_1, \cdots r_n = v_n\}$ と 環境 $y = \{s_1 = u_1, \cdots, s_m = u_m\}$ をマージして 主環境 $z = \{t_1 = w_1, \cdots, t_p = w_p\}$ を得る。

5 例題

実行例を説明する。実行例は全部で6ケある。そのうちの3ケは談話状況を与えてそれに関する質問を解くものである。その談話状況の発話には I, you の代名詞が含まれている。I, you の意味はロールとしてそれぞれ $I, $you として定義されている。因みに $I は談話状況における発話者として定義されている。談話状況には

at(l, (mo, α, β), yes)

なる命題が2ケ含まれている。その意味は発話αの表わすイベント型がβであることである。関係 mo は文と意味を対応付けるという意味でパーザの機能を表わしている。このことを保証するコンストレイントは次の形で同じく談話状況のなかにおかれている。

at(lu, (involve, [at(?l, (mo, $utter, ?S), yes), ?S), yes)]

他の3つの実行例は、Prologのプログラムとの類似を例示するものである。その中には次の自然数生成プログラムを含んでいる。

```
number(1).
number(s(X)) :- number(X).
```

談話状況についての各質問に要した実行時間はコンパイル版で300ミリ秒程度であった。

おわりに

頭初，我々は，自然言語処理のための意味モデルとしてモンタギュ意味論を使うことを考えて，それを勉強した。内包論理の可能世界意味モデルに基礎をおくモンタギュ意味論は，美しい体系を持っているが believe や know などの attitude verb と呼ばれる動詞の意味がうまく扱えないことや計算機処理上の計算量に問題点があることが分かった。状況意味論は可能世界を捨てて状況を基本的なオブジェクトに採用することによりそれらを克服していると見られる。主にこれらの理由により，我々は状況意味論を意味分析の方法として採用することにして，実験を始めたところである。

但し，可能世界意味論と状況意味論は互いに他を補なう理論であると思われるふしがあるので両者を統合した意味論も存在する可能性は否定できない。

状況意味論は現在，多くの人の興味をひきつつあるようである。国内でも論講会や論文発表[安川84]，[鈴木84]を通じて急速に広まりつつあると感じられる。筆者自身これらにより状況意味論を学ぶことができた。関係者に感謝する。

状況意味論はそのメタセオリである KPU 集合論等の技術的な面も含めると大理論である。筆者の状況意味論の理解も十分とは思っていない。とくに状況意味論における計算モデルについてはまだ満すべきモデルが得られてない。本メモは，かなり制限した範囲での計算モデルの試みであるが，内容的にも形式的にも今後の検討課題は多い。しかしながら，本メモの付録の簡単な例題によっても，同意味論の持つ大きな可能性を示すことができたのではないかと思う。もしそうならば幸わいである。

REFERENCE

[Barwise 75] Jon Barwise: Admissible Sets and Structures, Springer Verlag, 1975.

[Barwise 81 a] Jon Barwise: Some Computational Aspects of Situation Semantics, in Proceedings of the 19th Annual Meeting, Association for Compuational Linguistics, 1981.

[Barwise 81 b] Jon Barwise and John Perry: Scenes and Other Situations, Journal of Philosophy,78,No.7, 1981, pp.369-397.

[Barwise 81 c] Jon Barwise and John Perry: Situations and Attitudes, Journal of Philosophy,78,No.11, 1981, pp.668-691.

[Barwise 81 d] Jon Barwise and John Perry: Semantic Innocence and Uncompromising Situations, in French, Peter A., Uehling, Theodore E., Jr., and Wettstein, Howard K., eds. Midwest Studies in Philosophy, 6. Minneapolis: 1981, pp.387-404.

[Barwise 83] Jon Barwise and John Perry: Situations and Attitudes, MIT Press, 1983.

[Makkai 77] M. Makkai: Admissible Sets and Infinitary Logic, in Handbook of Mathematical Logic, pp.233-281, 1977.

[Scott 82] Danna S. Scott: Domains for Denotational Semantics, in the Proceedings of ICALP '82, Aarhus, Denmark, July 1982.

[鈴木 84] 鈴木浩之：日本語文の意味の状況意味論的な記述，ロジックプログラミングコンファレンス予稿集，1984.

[安川 84] 安川秀樹，何井国昭，平川秀樹，三吉秀夫，鈴木浩之：文章理解システムの構想，情報処理学会第28回全国大会予稿集 PP.943-944, 1984.

付録 1
全プログラムリスト

```
:- public fcoe/3,member/2,indeterminate/1,
        meaningful_option/4.

fcoe(X,Y,A):-
        closure(X,Z,A),
        extend_situation(Y,U),
        solve(Z,U).

solve([],_).
solve([H|T],E):-
        member(H,E),
        solve(T,E).
solve([H|T],E):-
        member(at(X,(involve,MF,MO),yes),E),
        closure((MF,MO),(MF1,MO1),Env),
        anchor_roles(Env,E),
        in_schema(Z,MO1),
        member(H,Z),
        append(MF1,T,U),
        solve(U,E).

extend_situation(X,X):-
        \+ ( member(at(_,(involve,_,S),yes),X),
             indeterminate(S)),!.
extend_situation(X,Y):-
        extend_situation1(X,X,Z),
        extend_situation(Z,Y).

extend_situation1([],X,[]).
extend_situation1([at(lu,(involve,E,S),yes)|X],Y,M):-
        indeterminate(S),!,
        bagof(MO,meaningful_option(E,S,Y,MO),MOs),
        extend_situation1(X,Y,Z),
        multiset_add(MOs,Z,M).
extend_situation1([X|Y],Z,[X|V]):-
        extend_situation1(Y,Z,V).

meaningful_option(E,S,Y,MO):-
        closure((E,S),(E1,S1),Env),
        subset(E1,Y),
        anchor_roles(Env,Y),
        closure(S1,MO,Env1),
        anchor_roles(Env1,Y).

anchor_roles(Env,E):-
        connect_roles(Env,Defs),
        anchor_list(Defs,E).

anchor_list([],_).
anchor_list([H|T],E):-
        subset(H,E),
        anchor_list(T,E).

connect_roles(Env,Defs):-
        new_env(Env,New,EL),
        set_union(EL,[],EL1),
        collect_defs(EL1,Env1,Defs),
        merge_list(Env1,New,_).

new_env([],[],[]).
new_env(['$'(X)=V|R],[Y=V|S],[Name|T]):-!,
        def_role('$'(X),Y,Name),
        new_env(R,S,T).
new_env([_|R],S,T):-
```

```
        new_env(R,S,T).

collect_defs([],[],[]).
collect_defs([X|R],[Env|S],[Def|T]):-
        def_event(X,Def1),
        closure(Def1,Def,Env),
        collect_defs(R,S,T).

indeterminate(X):-var(X),!,fail.
indeterminate('$'(_)).
indeterminate('?'(_)).

closure(X,Y,[]):-var(X),!,Y=X.
closure(X,Y,[X=Y]):-indeterminate(X),!.
closure(X,X,[]):-atomic(X),!.
closure([X|Y],[Z|U],V):-!,
        closure(X,Z,A),
        closure(Y,U,B),
        merge(A,B,V).
closure(X,Y,Z):-
        X=..[F|U],
        closure(U,V,Z),
        Y=..[F|V].

merge([],X,X).
merge([X=W|Y],Z,U):-
        assocq(X,Z,V),!,
        V=W,
        merge(Y,Z,U).
merge([X|Y],Z,[X|U]):-
        merge(Y,Z,U).

merge_list([],Env,Env).
merge_list([X|Y],Env,Env1):-
        merge(X,Env,Env2),
        merge_list(Y,Env2,Env1).

assocq(X,[Y=V|_],V):-X==Y,!.
assocq(X,[_|Y],V):-
        assocq(X,Y,V).

member(X,[X|_]).
member(X,[_|Y]):-
        member(X,Y).

append([],X,X).
append([X|Y],Z,[X|U]):-
        append(Y,Z,U).

union1([],X,X,[]).
union1([X|Y],Z,U,V):-
        member(X,Z),!,
        union1(Y,Z,U,V).
union1([X|Y],Z,U,[X|V]):-
        union1(Y,[X|Z],U,V).

set_union([],X,X).
set_union([X|Y],Z,U):-
        member(X,Z),!,
        set_union(Y,Z,U).
set_union([X|Y],Z,U):-
        set_union(Y,[X|Z],U).

multiset_add([],X,X).
```

```
multiset_add([X|Y],Z,U):-
        multiset_add(Y,Z,V),
        set_union(X,V,U).

in_schema(X,{(X,Y)}).
in_schema(X,{(_,R)}):-!,
        in_schema(X,{R}).
in_schema(X,{X}):-!.
in_schema(X,X).

subset([],_).
subset([X|Y],Z):-
        member(X,Z),
        subset(Y,Z).
```

```
:- op(400,fx,?). % for basic indeterminate
:- op(400,fx,$). % for complex indeterminate (role)
```

付録2 実行例

```
%%%%%%%% sample discourse situation.

sample_coe([
        at(l,(speaking,taro),yes),
        at(l,(addressing,hanako),yes),
        at(l,(saying,[you,are,a,girl]),yes),
        at(l,(saying,[i,am,a,boy]),yes),
        at(l,(saying,[you,are,on,the,right]),yes),
        at(l,(mo,[you,are,a,girl],[at($here,(is_girl, $you),yes)]),yes),
        at(l,(mo,[i,am,a,boy],[at($here,(is_boy, $i),yes)]),yes),
        at(l,(mo,[you,are,on,the,right],[at($here,(be_right,$i,$you),yes)]),
                yes),
        at(l,(mo,[you,are,on,the,left],[at($here,(be_left,$i,$you),yes)]),
                yes),
        at(lu,(involve,[at(?l,(mo,$utter,?s),yes)],?s),yes),
        at(lu,(involve,[at(?l,(be_right,?x,?y),yes)],
                [at(?l,(be_left,?y,?x),yes)]),yes)
        ]).


%%%%%%% event type definition

def_event(ds,[at(?l,(speaking,?a),yes),
              at(?l,(addressing,?b),yes),
              at(?l,(saying,?alpha),yes)]).


%%%%%%% role definition.

def_role($i,?a,ds).
def_role($you,?b,ds).
def_role($here,?l,ds).
def_role($utter,?alpha,ds).


%%%%%%% test cases

test1(X):-
        sample_coe(COE),
        fcoe([
                at(?m,(is_girl,?g),yes),
                at(?l,(is_boy,?b),yes)
               ],COE,X).
test11(X):-
        sample_coe(COE),
        fcoe([
                at(?m,(be_right,taro,?g),yes)
               ],COE,X).

test12(X):-
        sample_coe(COE),
        fcoe([
                at(?m,(be_right,hanako,?b),yes)
               ],COE,X).

test2(M):-fcoe([a,a],[at(lu,(involve,[b],[a]),yes),b,c],M).

test3(X):-fcoe(
              [at(lu,(number,?num),yes)],
```

```
                [at(lu,(number,1),yes),
                 at(lu,(involve,
                        [at(lu,(number,?int),yes)],
                        [at(lu,(number,s(?int)),yes)]),
                       yes)],
                 X).

test4(X):-fcoe([at(lu,(p,?int),yes)],
                [at(lu,(p,1),yes),
                 at(lu,(p,2),yes),
                 at(lu,(p,3),yes)],X).


%%%%%%%% execution.

| ?- test1(X).

X = [?m=1,?g=hanako,?l=1,?b=taro]

yes
| ?- test2(X).

X = []

yes
| ?- test3(X).

X = [?num=1] ;

X = [?num=s(1)] ;

X = [?num=s(s(1))] ;

X = [?num=s(s(s(1)))] ;

X = [?num=s(s(s(s(1))))]

yes
| ?- test4(X).

X = [?int=1] ;

X = [?int=2] ;

X = [?int=3] ;

no
| ?- test11(M).

M = [?m=1,?g=hanako] ;

no
| ?- test12(M).

no
```