

循環的な概念定義を含むフレーム表現の意味論

伊 藤 昭[†] 熊 本 忠 彦[†]

フレーム表現は、我々の持つ素朴な世界観に適合した実用性の高い知識表現形式である。しかしながら、KL-ONE, UNITS など定式化されている形では、基本的には一階述語論理の記述能力を越えることができず、人の対話などに現れる柔軟な意味表現を行うには不十分なことが多かった。フレーム表現の記述能力を拡張するために、循環的な概念の定義を可能にしようとする試みがある。確かに、人は一見循環的とも見える表現を自由に操ることで、広い範囲の概念を直観的に定義している。しかしながら、これまでの循環的定義導入の試みでは、定義がクラス概念のみを用いてなされていたため、有用な一部の概念を定義できないという欠点があった。そこで我々は、クラスとインスタンスとを同時に表現できるフレーム表現の特徴を生かし、インスタンス変数を用いて、より広い範囲の循環的定義を可能とする概念定義の枠組の提案を行う。また本枠組のもとで、「もの」概念だけではなく状況一般を概念として理解することで、自然言語の発話の意味としての「状況タイプ」の表現を行う。

A Semantics for a Frame Representation with Circular Concept Definition

AKIRA ITO[†] and TADAHIKO KUMAMOTO[†]

A frame representation (FR) is a practical method for representing our naive concepts about the world. However, the expressive power of the FR is within that of the first order predicate calculus, and cannot fully express the meanings of our daily conversation. To extend the expressive power of FR, some authors have incorporated the mechanism of the circular explanation/definition of concepts. Surely, people often use the circular explanation of concepts, and somehow manage to communicate the intended meaning. However, the extension so far is restricted to the circular definition with the class concepts. We propose the new framework which allows the wider range of circular definition utilizing FR's capability to represent the instance concepts. In this framework, the situation is understood as a kind of concepts, and the differences of information contained in various situations can be reduced to the subsumption of these concepts.

1. はじめに

フレーム表現は、世界に存在する様々な「もの」や「もの」と「もの」との関係性を直観的に表現するのに適しており、人の持っている素朴な実在論に適合した実用性の高い知識表現形式である^{1)~4)}。また、論理による表現に比べると記述（推論）能力が弱い反面、推論方法などを利用者が定義することで効率のよい知識表現を構成することができる。しかしながら、これまで KL-ONE⁵⁾, UNITS⁶⁾ など定式化されている形では、フレーム表現は基本的に一階述語論理の記述能力を越えることができず、人の対話などに現れる柔軟な知識表現を行うには不十分なことが多かった。

フレーム表現の記述能力を拡張するために、循環的

な概念の定義を扱えるようにしようとする Baader⁵⁾, Nebel⁷⁾ らの試みがある。例えば、「生粋の江戸っ子といえるのは、生粋の江戸っ子の両親に育てられた奴だけだ」というような、一見問題のない表現もよく考えると循環的であり、ある人が「生粋の江戸っ子」であることをどのように判断したらいいのか悩んでしまう。それでも、人はこのような循環的とも見える表現を自由に操ることで、広い範囲の概念を直観的に定義したり説明したりしている。実際、この表現で述べられていることは概念間の関係/制約であり、これが完全なクラス定義になっているかどうかは明らかではない。しかしながら、我々の日常会話ではこのような表現は例外ではなくむしろ一般的であり、自然言語の知識表現を考えるものとしては避けて通ることのない問題である。

我々はある概念を定義するとき、そこに属するイン

[†] 郵政省通信総合研究所
Communications Research Laboratory

スタンスを取り上げ、そのインスタンスが満たさなければならない制約/関係を述べることで、暗 (Implicit) にその概念を定義/説明する。しかしながらこれまでの循環的定義導入の試みでは、定義がクラス概念のみを用いてなされていたため、我々の日常会話表現からの変換が必要であったり、また有用な一部の概念の定義ができないという欠点があった。そこで我々は、フレーム表現の特徴であるクラスとインスタンスとを同時に表現できる点を生かし、インスタンス変数を用いた、より広い範囲の循環的定義を可能とする概念定義の枠組の提案を行う。また本枠組のもとで、「もの」概念だけではなく、状況一般を概念として理解することで、自然言語の発話の意味としての「状況タイプ」⁸⁾を表現することを考える。このとき、状況に含まれている情報量の差は概念間の包摂 (subsumption) 関係として捉えられる。

以下、2章ではフレーム表現の定式化を行い、3章では再帰的なクラス定義を導入する。4章ではインスタンスを表現する概念としてインスタンス変数を導入し、クラス定義の拡張を行う。5章ではこのインスタンス変数を用いた状況の記述を行い、知識の検索が状況概念 (状況タイプ) の包摂を用いて記述されることを示す。6章では基本的なユーザインターフェース関数を紹介する。付録では、制限された範囲ではあるが、こうしたクラス (概念) 間の包摂関係を決定するアルゴリズムを与える。

2. フレーム表現の定式化

2.1 クラスとインスタンス

オブジェクトインスタンス

世界を記述するために、まず「もの」概念に対応するものとして、オブジェクトを導入する。ここでは、実際に存在する「もの」を表現するオブジェクトをインスタンスオブジェクト ω と呼び、有限個または無限個の属性 a_k 、属性値 v_k の組からなる属性記述子のリストを用いて表現する。

$$\omega = ((a_1 \ v_1)(a_2 \ v_2) \dots)$$

なお、インスタンス ω の属性 a_k のとり属性値を $\omega.a_k$ と表記する。

全く同じ属性値リストを持っていたり、実世界における異なった実在には、別々のインスタンスを対応させる。このような「もの」を表すインスタンスの全体集合として Ω を定義する。以後、本論文では Ω を実世界と同一視し、これをフレーム表現で記述すること

を試みる。

インスタンスの記述において、属性値を特に指定する必要のない属性については、属性記述子を省略して

$$\omega = ((a_1 \ v_1) \dots (a_k \ v_k))$$

と記述することもある。しかしながら、 Ω の世界のインスタンス ω は、たとえその一部の属性を用いて部分的に記述されていても、それ以外の属性を持っていないという意味ではない。また、二つのインスタンス ω_1, ω_2 が全く同じ属性記述子のリスクを持っていたり、それぞれ別個のインスタンスである。

クラス

クラスは、インスタンスを要素とする集合であり、インスタンスのとり属性値に対する制約として記述される。この制約は、属性 a_k 、属性型 t_k の組からなる属性型記述子の有限個のリスト (クラス記述と呼ぶ) で表現される。クラスは、普通自分自身を特定するためのクラス名を持っている。

$$\text{Class } C = ((a_1 \ t_1)(a_2 \ t_2) \dots (a_k \ t_k))$$

インスタンス ω がクラス C に属する (クラス C のインスタンスである) とは、クラス記述に現れるすべての属性について、属性値が属性型の制約を満たすこととし、 $\omega \in C$ で表す。この時、インスタンスがクラス記述で指定された以外の属性を持っていることは構わない。

インスタンスと違って、同じクラス記述を持つクラスは同じクラスと考え、区別をしない。従ってクラス記述においては、属性、属性型の組で記述される属性型記述子の省略は許されない。この違いは、インスタンスは初めから「存在」するものであって、その属性の一部が属性記述子で与えられるのに対して、クラスの記述では属性型記述子によってクラスが定義されているところからきている。

多くのクラスでは、すでに定義されているクラスの下位概念 (特殊化) としてクラスが定義されるため、次のようなクラス定義の表記法を導入する。

$$C_1 = ((a_1 \ t_1)(a_2 \ t_2) \dots (a_k \ t_k)) \text{ のとき}$$

$$C_2 = ((a_1 \ t_1) \dots (a_k \ t_k)(a_{k+1} \ t_{k+1}) \dots (a_m \ t_m)) \\ = (C_1 \ (a_{k+1} \ t_{k+1}) \dots (a_m \ t_m))$$

ただし、 $a_1, \dots, a_m$ はすべて異なっていなければならない。このとき、 C_1 を C_2 のスーパークラス、 C_2 を C_1 のサブクラスと呼び

$$C_1 \supseteq C_2$$

と記述する。

クラスをそれが属するインスタンスの集合と同一視

すれば, C_2 は C_1 の部分集合となっている. また, C_1 , C_2 を二つのクラスとすると, その積集合 $C_1 \cap C_2$, 和集合, $C_1 \cup C_2$ で表されるインスタンスの集合も一種のクラスと考えることができる. これを属性型記述子のリストにより定義されるクラスと区別するために, 派生 (Derived) クラスと呼ぶことにする.

もう少し形式的な意味づけを行うために, モデル理論による形式化を与えておく.

定義 1 モデル理論 我々が形式化しようとしているフレーム表現システムを $\mathbf{F}$ とする. モデル理論における解釈とは, 解釈空間 Ω (インスタンスの集合), 属性名の集合 $A = \{a_i\}$, $x \in \Omega$ に対して, その属性値 $x.a_i$ を対応させる関数 $\cdot$, クラス C に対して Ω の部分集合 $I(C)$ を対応させる解釈関数 I の四つ組 $\{\Omega, A, \cdot, I\}$ のことである. $\square$

ある解釈が $\mathbf{F}$ のモデルであるとき,

$$I(C_1 \cap C_2) = I(C_1) \cap I(C_2)$$

$$I(C_1 \cup C_2) = I(C_1) \cup I(C_2)$$

$$C_1 \subseteq C_2 \iff I(C_1) \subseteq I(C_2)$$

が成り立つ.

2.2 原始オブジェクト型と属性型

原始オブジェクト型 原始オブジェクト型は本システム外で定義されている基本的なオブジェクトを導入するための基となる型で, 内部構造 (属性記述子) を持たないものである. ここでは, 数, シンボル, リストなど, 標準の LISP で定義されている型を考えておく. ただし誤解のない限り, 整数などの「型」をクラス, 実際の 1, 2, 3 などの実体をインスタンスとみなし, 本システムで定義されるクラス, インスタンスの一部として取り扱う.

属性型 クラス記述に現れる属性型は, クラス, 派生クラス, 列挙型, または集合型のいずれかである. 属性型 := クラス (原始オブジェクト型を含む)

| 派生クラス

| インスタンスの列挙型 $[i_1 i_2 \dots i_k]$

| クラス C の集合型 $(*: C)$

列挙 (enumerate) 型 $[i_1 i_2 \dots i_k]$ インスタンスの列挙型は, 属性値がそのインスタンスのいずれかであることを表す. 例えば, 列挙型 $[1 2 3 4]$ は属性値が整数 1, 2, 3, 4 のいずれかであることを表す. ただし, 要素が一個の場合 $[i]$ を i と記述することがある.

集合型 $(*: C)$ 集合型 $(*: C)$ は, 属性値がクラス C に属するインスタンスの集合であることを表す. 例えば, 人の持つ「兄弟」という属性の属性型は, 人

の集合型 $(*: \text{human})$ で表される. KL-ONE などでは, 一つのオブジェクトが同一の属性名 (Role) に対して複数の属性値をとることができるが, 本システムではこれを認めていないため, それを補う必要から集合型が導入されている.

ここでモデル理論を用いて, 前に属性値が属性型を満たすと述べたことを形式的に定義しておく.

定義 2 (属性型制約) いま, クラス記述の集合を $\{C_i\}_{i=1 \dots k}$ とし, $C_i = (\dots (a_j^i t_j^i) \dots)$ とする. このとき, $\mathbf{F}$ のモデルとなるすべての解釈について, $\omega_i \in I(C_i)$ ($\subseteq \Omega$) とすれば,

(1) t_j^i がクラス $C_i \Rightarrow \omega_i.a_j^i \in I(C_i)$,

(2) t_j^i が列挙型 $[i_1 i_2 \dots i_n] \Rightarrow$

$$\omega_i.a_j^i \in [I^*(i_1) I^*(i_2) \dots I^*(i_n)],$$

(3) t_j^i が集合型 $(*: C_i) \Rightarrow$

$$\omega_i.a_j^i = [\bar{\omega}_1 \bar{\omega}_2 \dots \bar{\omega}_n],$$

$$\text{かつ } \bar{\omega}_s \in I(C_i), s = 1 \dots n,$$

が成り立つ. ただし $[\dots]$ はインスタンスの集合を表す. $\square$

t_j^i を, $\{I(C_i)_{i=1 \dots k}\}$ を引数とし Ω の部分集合 (集合型では, Ω の列の集合) を値とする関数と見れば, 属性型制約を $\omega_i.a_j^i \in t_j^i(\{I(C_i)_{i=1 \dots k}\})$ と書くことができる. $I^*(i)$ については説明が必要であるが, 当面原始オブジェクト (数, シンボルなど) のみがインスタンスとして記述可能であるとし, これについては $I^*(i) = i$ が成り立つものと仮定しておく.

3. 再帰的クラス定義

クラス定義に, 再帰的に同じクラスが出現することも可能であるが, このとき, 形式的にはクラス定義に曖昧性が生じることがある.

例 神戸っ子

Kobekko = (Human (father Kobekko) (mother Kobekko))

(「神戸っ子」とは, 父も母も「神戸っ子」の人のことである)

一見もっともそうなこの記述は, おそらく期待される意味を持たない. 実際, これを「Human の部分集合であって, そのすべての要素の father 属性, mother 属性が再びその集合の要素であるような集合を探せ」と理解すると, このようなものは一般に複数個存在し, 有効なクラスの定義とはなり得ない. しかしながら, このような部分集合の中で最大集合が存在することが知られており⁵⁾, 次のような形で, 再帰的定義で与え

られるクラス概念の意味を定義する。なお、以下では式が複雑化するのを避けるため、クラス C_i と、対応するインスタンスの集合 $I(C_i)$ とを同一視する。

定義 3 (クラスの再帰的定義) 全インスタンスの集合を Ω とする。クラス記述 $C_1, C_2, \dots, C_k$, ($C_i = ((a_1^i \ t_1^i) (a_2^i \ t_2^i) \dots)$) により、クラス $C_1, C_2, \dots, C_k$ が相互再帰的に定義されている (すなわち、属性型記述子 t_j^i はクラス $\{C_i\}_{i=1 \dots k}$ の関数 $t_j^i(\{C_i\}_{i=1 \dots k})$ である) とする。このとき、適当に Ω の部分集合 $U_1, U_2, \dots, U_k$ を決めて、クラス記述の属性制約を満たす集合の組 $\{C_i\}_{i=1 \dots k} = \{U_i\}_{i=1 \dots k}$ を作る。 $U_i = \phi$ (空集合) の場合も含めれば、このような集合は必ず存在する。このような集合の組が複数あるとし、これを $\{U_i^{(j)}\}_{i=1 \dots k, j=1 \dots s}$ とする。このとき、そのような組のなかで最大のものを $\{U_i\}_{i=1 \dots k}$ として、これをクラス記述 $C_1, C_2, \dots, C_k$ で定まるクラスの定義とする。 $\square$

このような集合の組が存在することを以下に示す。

補題 属性型 t_j^i をクラス $\{C_i\}_{i=1 \dots k}$ の関数と考えたとき、 t_j^i は $\{C_i\}_{i=1 \dots k}$ の単調増加関数である。すなわち、 $\forall j, C_j \subseteq C'_j$ ならば、 $t_j^i(\{C_i\}_{i=1 \dots k}) \subseteq t_j^i(\{C'_i\}_{i=1 \dots k})$ である。

証明 属性型を構成するクラス、派生クラス、列挙型、集合型について、この性質が成り立っていることを示せばよい。クラスについてはこの性質は明らかである。派生クラスについては、 $C_1 \cup \cap C_2 \subseteq C'_1 \cup \cap C'_2$ より、この性質は明らかである。列挙型は定数である (クラスを含まない)。集合型については、 $x \in (* : C)$ なら $x = (i_1 \ i_2 \dots)$, $i_1 \in C \subseteq C'$ である。したがって、 $x \in (* : C')$ である。 $\square$

定理 1 クラス記述 $C_1, C_2, \dots, C_k$, ($C_i = ((a_1^i \ t_1^i) (a_2^i \ t_2^i) \dots)$) を満たす Ω の部分集合の組を $\{U_i^{(j)}\}_{i=1 \dots k, j=1 \dots s}$ とする。このとき $U_i = \cup_j U_i^{(j)}$ とすると、 U_i は、クラス記述を満たす最大の集合である。すなわち、 $\forall i, U_i^{(j)} \subseteq U_i$, かつ U_i はクラス記述の制約を満たす。

証明 $U_i^{(j)} \subseteq U_i$ は明らかである。今、 $\omega_i \in U_i$ とする。 $\exists j, \omega_i \in U_i^{(j)}$ であるから、 $\forall a_1^i, \omega_i. a_1^i \in t_1^i(\{U_k^{(j)}\}_{k=1 \dots k})$ となる。ところが、 t_1^i は $\{U_k^{(j)}\}_{k=1 \dots k}$ の単調増加関数であり、 $U_k^{(j)} \subseteq U_k$ であるから、 $\omega_i. a_1^i \in t_1^i(\{U_k^{(j)}\}_{k=1 \dots k}) \subseteq t_1^i(\{U_k\}_{k=1 \dots k})$ となる。すなわち、クラス記述の制約を満たす。 $\square$

このようにして再帰的なクラス定義では、条件を満たす集合が複数存在するときには、その最大集合 (その存在を証明した) をもって、クラス定義で定まる集

合とするのである。上の例でいえば、

$Kobekko = (\text{Human} (\text{father } Kobekko) (\text{mother } Kobekko))$

で定まるクラスは、 $Kobekko = \text{Human}$ である。少々排他的ではあるが Human が lives-in (～に住んでいる) という属性を持っているものとすれば、

$Kobekko = (\text{Human} (\text{father } Kobekko) (\text{mother } Kobekko) (\text{lives-in Kobe}))$

により、「先祖代々神戸に住んでいる人」という期待する答えに合ったものを得る。

クラスから派生クラスを構成するための演算子として、否定 $\neg (\neg A = \Omega - A)$ を含めていないのは、派生クラスで自由に再帰的定義ができることを保証するためである。否定が再帰的定義に使われるとクラスが定まらないのは、

$\text{Man} = \text{Human} \cap \neg \text{Woman}$

$\text{Woman} = \text{Human} \cap \neg \text{Man}$

が有効なクラス定義になっていないことから容易に理解される。実際、 Man , Woman の集合のうち一方を大きくすれば必然的に他方が小さくなり、上の条件を満たす最大集合の組 $\{\text{Man}, \text{Woman}\}$ は存在しない。

4. インスタンス変数

前に述べたように、 Ω のインスタンスは実世界の「もの」と一対一に対応しており、「私のクラスの友達 (の誰か)」というような変数としての機能を残した概念を表現できない。そこで、このような目的に対応するため、インスタンス変数 $x : C$ を用いて、クラス C のあるインスタンスという概念を表現する。また、次の形式により内部に属性型記述を持つインスタンス変数を導入する。

$i = (\dots (a_j \ x : t_j) \dots)$

これは、無名のクラス $C = (\dots (a_j \ t_j) \dots)$ を定義して、 $i = y : C$ と記述するのと等価である。また、次の記述法を導入する。

$\text{Class } C = ((a_1 \ t_1) \dots (a_k \ t_k))$ のとき、

$i = (\text{Class } C(a_1^* \ v_1^*) \dots (a_k^* \ v_k^*))$

$= ((a_1^* \ v_1^*) \dots (a_k^* \ v_k^*) (a_1 \ x_1 : t_1) \dots (a_k \ x_k : t_k))$

(ただし、 $(a_1 \dots a_k)$ で、 $(a_1^* \dots a_k^*)$ と重複するものは省く。)

ここで、後ろに追加された属性記述子は、クラス記述 $\text{Class } C$ に現れ、インスタンス記述には現れていない属性について、インスタンス変数を用いて属性記述子

を追加したものである。

次に、このインスタンス変数を用いることによって、新しい型の再帰的クラス定義の方法を導入する。

例 いま、ナルシシストを自分自身を愛する人と考えると、この定義として

$x : \text{Narcissist} = (\text{Human} (\text{loves } x : \text{Narcissist}))$
のような記述が考えられるが、これをクラスを用いた循環的定義で記述することはできない。例えば、

$\text{Narcissist} = (\text{Human} (\text{loves Narcissist}))$

としてしまえば、ナルシシストはナルシシストの仲間の誰かを愛するとなってしまう、必ずしも自分を愛する必要がなくなってしまう。

$x : \text{Narcissist}$ は、ナルシシストの性質を記述しているが、これをあるインスタンスがナルシシストである(必要十分)条件を表していると見れば、これはナルシシストのクラス定義となる。我々は、意味ネットワークの表記法を借りて⁴⁾、この関係を図1のように表す。この例でわかるように、インスタンス変数を用いた一組のクラス定義が与えられたとき、その中に出てくる同一のインスタンス変数は同一のものを指す。従って、同じクラスに属する異なったインスタンス変数を表現するには、異なった変数名 $x_1 : C$, $x_2 : C$, ... などを用いる必要がある。

クラス名が不要な場合、これを省略することも可能である。むしろこれが本来の用法であり、わざわざクラス名を導入することなしに概念を導入できる。例えば、

$x = (\text{Human} (\text{brought-up-by } y : \text{Human}) (\text{brother nil}) (\text{grandfather } y : \text{Human}))$

により、「祖父に育てられた一人っ子」のような概念を簡単に表すことができる。また、 $x = (\text{Human} (\text{loves } x))$ とすれば、 x により名前を持たないが自分自身を愛する人という概念を表現することができる。

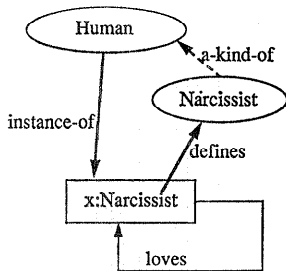


図1 ナルシシストのインスタンス変数による定義
Fig. 1 Definition of Narcissist using instance variables.

さて、前に属性型形式において、インスタンスの列挙型 $[i_1 i_2 \dots i_k]$ を導入したが、このインスタンスは本来 Ω の世界のインスタンスではあり得ない。なぜなら、 Ω はモデル世界であり、クラス記述は概念(クラス)の世界であり本来異なる世界に属している。列挙型 $[i_1 i_2 \dots i_k]$ に現れるインスタンスは、実はここで導入したインスタンス変数のことである。実際、上の Narcissist の定義における loves スロットは、属性値に $x : \text{Narcissist}$ というただ一個のインスタンスを要素とする列挙型となっている。ただし、前にも述べたように、原始オブジェクトについては内部構造(属性記述)を持たないため、この区別の必要はなかった。

前節で出てきたクラスの再帰的定義は、クラス名の所をそのクラスのインスタンス変数で置き換えることで、インスタンス変数のみを用いたクラス定義に変換することができる。ただしこのとき、すべてのインスタンス変数に異なった名前を与えるものとする。また、インスタンス変数の型指定をクラスだけに限るのではなく、属性型に拡張することも容易である。しかしながら、このことによって特にシステムの記述能力が増えるわけではないので、我々はインスタンス変数の右辺での利用を列挙型に限ることとする。

定義4 (インスタンス変数によるクラスの再帰的定義) フレーム表現システム F のクラスが、インスタンス変数を用いて次の式で再帰的に定義されているものとする。

$$x_i^0 : C_i = (\dots (a_i^j t_j^i) \dots), \quad i=1 \dots k$$

ここで t_j^i はクラス $\{C_i\}_{i=1 \dots k}$ 以外に $\{\{x_j^s : C_i\}_{s=1 \dots n_j}\}_{i=1 \dots k}$ を含むことになる。なお、左辺には C_i を定義するために各 $x_i^0 : C_i$ が一回ずつ出てくるだけであるが、右辺には任意のインスタンス変数が任意個存在してよい。

解釈 $\{\Omega, A, \cdot, I\}$ が F のモデルであるとは、すべての $\omega_i \in I(C_i) (\subseteq \Omega)$ について、 ω_i によって決まる $\{\{x_j^s : C_i\}_{s=1 \dots n_j}\}_{j=1 \dots k}$ から Ω への関数 I^* で

$$I^*(x_i^0 : C_i) = \omega_i, \quad \text{かつ}$$

$$I^*(x_j^s : C_i) \in I(C_i)$$

を満足するものが存在して、この I^* により定義2で述べた型制約 $\omega_i, a_i^j \in t_j^i (\{I(C_i)_{i=1 \dots k}\})$ が成り立つことである。□

定義2で未定義のまま残されていた I^* の意味が、ここで初めて明らかにされている。もちろん、前節で行った再帰的クラス定義の最大集合存在定理の証明では、インスタンスがクラスに依存しないことが仮定さ

れていた。従って、証明は若干の修正を要する。定理の証明を見直してみると、証明において本質的なことは θ_i を $\{C_i\}_{i=1\dots k}$ の関数と見たとき、その単調増加という性質であった。このことを確認して、証明の修正を行う。

定理 2 インスタンス変数を含む形で与えられたクラス記述を $C_1, C_2, \dots, C_k$, ($C_i = ((a_1^i \ \theta_1^i)(a_2^i \ \theta_2^i) \dots)$), またこれを満たす Ω の部分集合の組を $\{\{U_i^{(j)}\}_{i=1\dots k}\}_{j=1\dots s}$ とする。このとき $U_i = \bigcup_j U_i^{(j)}$ とすると U_i は、クラス記述を満たす最大の集合である。すなわち、 $\forall i, U_i^{(j)} \subseteq U_i$, かつ U_i はクラス記述の制約を満たす。

証明 $U_i^{(j)} \subseteq U_i$ は明らかである。今、 $\omega_i \in U_i$ とする。 $\exists j, \omega_i \in U_i^{(j)}$ であるから、 ω_i で決まる関数 I^* が存在して、 $\omega_i = I^*(x_i^0 : C_i)$, かつ $I^*(x_i^j : C_i) \in U_i^{(j)}$ である。また、この I^* によって、 $\forall a_i^j, \omega_i, a_i^j \in \theta_i^j(\{U_h^{(j)}\}_{h=1\dots k})$ となる。このとき、 I^* は $I^*(x_i^j : C_i) \in U_i^{(j)} \subseteq U_i$ であるから、同じ I^* を用いた θ_i^j により、 $\omega_i, a_i^j \in \theta_i^j(\{U_h^{(j)}\}_{h=1\dots k}) \subseteq \theta_i^j(\{U_h\}_{h=1\dots k})$ となる。すなわち、 $\{U_i\}_{i=1\dots k}$ はクラス記述の制約を満たす。 $\square$

神戸っ子の例では、

$x : \text{Kobekko} = (\text{Human (father } y : \text{Kobekko)}$
 $(\text{mother } z : \text{Kobekko}) (\text{lives-in Kobe}))$

となる。

なお、制限された範囲ではあるが、付録にインスタンス変数を用いた再帰的クラス定義における、包摂 (subsumption) $\subseteq$ 関係を決定するアルゴリズムを与える。

5. 概念と状況

これまででは、インスタンスの集合であるクラスを概念の定義としてきた。しかしながら、再帰的なクラス (概念) 定義を導入すると、一つの概念 (インスタンスの集合) は他の概念と相互に関係付けられることになる。例えば、次のような概念定義が与えられたとする。

例. 同い年の夫婦

$x : \text{onaidosi-husband} = (\text{Human (gender male)}$
 $(\text{age } z : \text{integer}) (\text{spouse } y : \text{onaidosi-wife}))$
 $y : \text{onaidosi-wife} = (\text{Human (gender female)}$
 $(\text{age } z : \text{integer}) (\text{spouse } x : \text{onaidosi-husband}))$

4章で述べたように、一組の概念定義において同一のインスタンス変数は、同一のものを表している。さて、この概念定義により「自分と同じ年の妻を持つ亭

主」という概念が構成されるが、それと同時に「自分と同じ年の亭主を持つ妻」という概念も自動的に構成されている。そこで、これをもっと自然に考えて、

$S = (x : \text{onaidosi-husband } y : \text{onaidosi-wife})$

により、「同い年の夫婦」という概念を表すものと考ええる。この概念定義は、4章で導入した意味ネットワーク型表記では図2で表される。

このように、我々は「概念」をインスタンス変数のリスト (有限集合) で表されるものとする。いま、あるインスタンス変数 $x : C$ によって、一つのクラスが表されているものとする。このとき、 $\bar{S}[x : C]$ により、 $x : C$ から属性関係により結合されているすべてのインスタンス変数のリストを表すものとする。例えば、

$S = \bar{S}[x : \text{onaidosi-husband}]$
 $= \bar{S}[y : \text{onaidosi-wife}]$
 $= (x : \text{onaidosi-husband } y : \text{onaidosi-wife})$

である。このとき、この S をクラス定義 onaidosi-husband (または onaidosi-wife) が持っている情報と考えるのである。

今、夫婦を

$x_0 : \text{husband}$
 $= (\text{Human (gender male) (spouse } y_0 : \text{wife}))$
 $y : \text{wife} = (\text{Human (gender female)}$
 $(\text{spouse } x_0 : \text{husband}))$

で定義すると、 onaidosi-husband は husband の下位概念 ($\text{onaidosi-husband} \subseteq \text{husband}$) になっている。一般に、二つのインスタンス変数 $i_1 = x_1 : C_1$, $i_2 = x_2 : C_2$ について、その対応するクラスが $C_1 \subseteq C_2$ を満たすとき、 i_1 を i_2 の特殊化 (インスタンス)、 i_2 を i_1 の一般化 (インスタンス) と呼び、 $i_1 \leq i_2$ と記述する。このとき、 i_1 の方が i_2 より多くの情報を持っており、

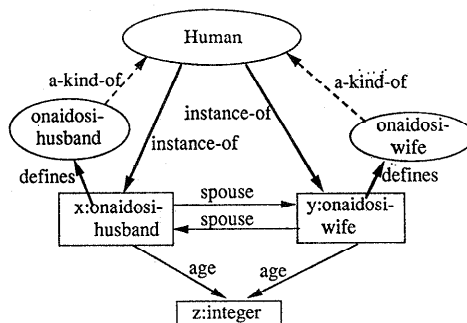


図2 同い年の夫婦の定義

Fig. 2 Definition of married couple of the same age.

詳しい状況の記述である。 $\leq$ は対応するクラスにおける包摂 (subsumption) 関係に相当する。 $i_1 \leq i_2$, かつ $i_1 \geq i_2$ のとき, $i_1 \equiv i_2$ と書く。 $\leq$ は推移律を満たす。また, $\equiv$ は同値関係である。例えば, $x: \text{onaidosi-husband} \leq x_0: \text{husband}$ である。

二つのインスタンス変数 i_1, i_2 が, 一方が他方の特殊化ではないが共通の特殊化インスタンス変数 i を持つ ($i \leq i_1, i \leq i_2$) とき, それらを単一化可能と呼び, $i_1 \sim i_2$ と記述する。これは, 二つのインスタンス変数 i_1, i_2 の持つ情報が, 矛盾なく統合され得ることを意味している。(これは, 必ずしも実世界における同じ「実在」に対応していることを意味しない。そう考えても「矛盾」が生じないということである。) また, $\sim$ は同値関係ではない。なお, このとき比べられているのは, 単に i_1, i_2 ではなく, $\bar{S}[i_1], \bar{S}[i_2]$ であることを注意すべきである。

さて我々は, 状況 S を演算 $\bar{S}[]$ で閉じているインスタンス変数のリスト $S = (i_1 i_2 \dots)$ を用いて表現することとする。すなわち, $\forall i, \bar{S}[i] \subseteq S$ ($\subseteq$ はリストを集合と見なしたときの包含関係) が成り立つ。このとき, 状況の「持っている」情報は, 個々のインスタンス変数の持っている情報の和となる。 S は世界の状況を部分的に記述しており, 状況意味論⁸⁾ という状況タイプに相当する。

インスタンス変数のリスト $S^0 = (i_1^0 i_2^0 \dots)$, $S^1 = (i_1^1 i_2^1 \dots)$ が与えられたとき, すべての i_j^0 に対して i_k^1 が存在して $i_k^1 \leq i_j^0$ が成り立つとき, $S_1 \leq S_0$ と記述し S_1 の方が S_0 よりも詳しい状況の記述であるという。 $S_1 \leq S_2$, かつ $S_1 \geq S_2$ のとき, $S_1 \equiv S_2$ と書く。インスタンスのときと同じようにして, $\leq$ は推移律を満たす。また, $\equiv$ は同値関係である。

定理 3 $i_1 \leq i_2$ ならば, $\bar{S}[i_1] \leq \bar{S}[i_2]$ である。(証明略)

演算 $\bar{S}[]$ は, 一つのインスタンス i から属性を通してたどられるインスタンス全部を結合するため, 場合によっては $\bar{S}[i]$ が「大きくなりすぎる」ことがある。そこで, あるインスタンス変数のリスト $S_0 = (i_1 \dots i_n)$ に対して, 各 i_i の属性値に S_0 に含まれないインスタンス変数が現れた場合, これを切り捨ててしまうことで得られる状況を $\bar{S}[i_1, \dots, i_n]$ と書く。一般に, $\bar{S}[i_1, \dots, i_n] \leq \bar{S}[i_1, \dots, i_n]$ が成り立つ。

6. ユーザインターフェース関数

今, 状況 S がインスタンス変数のリスト $S = (i_1 i_2 \dots$

$i_k)$ で記述されているとする。これを知識ベースと考えると, これを利用する側にとっては, 基本的に二つの操作, すなわち知識の問い合わせ Ask, 知識の追加 Tell が定義されていれば十分である。

知識に対する問い合わせとしては, 次のようなものが考えられる。

search-subconcept(i, S)

$i' \leq i$ を満たす $i' \in S$ を探し, 見つければこれを返す。これは, i で記述されている状況 $S_0 = \bar{S}[i]$ が, S の中に含まれているかどうかを調べるものである。しかしながら, いつも S_0 のすべての情報が S に含まれているわけではない。もっと一般的には, 次を用いる。

search-unifiable-concept(i, S)

これは, $i' \sim i$ を満たす $i' \in S$ を探し, 見つければ i' と共通の特殊化 i^* ($i^* \leq i', i^* \leq i$) とを返す。

一方知識の追加は, もし知識ベース S が追加しようとする知識 i について何も知らない場合は, 単純にそれを追加する

create-concept (i, S)

だけでよい。しかしながら, 知識ベース S が部分的に i についての情報を持っている場合, すなわち **search-unifiable-concept(i, S)** が成功する場合は, その返す値を i', i^* として

update-concept(i', i^*, S)

を用いて知識ベースに知識を追加する。

ここで導入した関数 **search-subconcept**, **search-unifiable-concept**, **create-concept**, **update-concept** はすべて属性を再帰的にたどる形で処理される。すなわちこれらの関数では常に, 前章で定義された $\bar{S}[i]$ 全体が処理の対象となるのであって, 例えば **update-concept** では, i から属性によって結合されているすべてのインスタンスが自動的に **update** されることになる。

例として, 一つの素朴な対話モデルを記述する。ある共通の実世界 W を表現する二つのモデル W_1, W_2 が対話 (情報交換) を行っているとする。ただし, モデルは実世界を部分的に表現しているが, 誤った知識を含んではないものとする。

W_1 からの発話は, W_1 の部分集合である ($i_1 \dots i_n$) から構成される $S = \bar{S}[i_1 \dots i_n]$ の形で相手側 W_2 に伝えられる。当然, $S \leq W_1$ である。 W_2 がこの発話を解釈し, 有意味な情報を抽出しようとするれば, $i_1 \in S$ について, **search-unifiable-concept(i_1, W_2)** を行う。このとき, 成功すれば結果として $i_1' \in W_2$, および共通

の特殊化 i^* が返される。もし、 $i^* \leq i_1$ であれば、 i^* には W_2 の知らない新しい知識が含まれていることになる。このとき、 $\text{update-concept}(i_1, i^*, W_2)$ により、新しい知識を追加することができる。

i_1 が存在しないときは、 W_2 は i_1 について何も知らないことを意味し、このときは i_1 を W_2 にコピーすれば良い。一方、複数の i_1 が得られた場合、 W_1 にはそれを区別する情報がないので、a) W_1 に問い合わせ、 i_1 についてのより詳しい情報を受けとる、b) 実世界を調べて i_1 についての情報を収集する、c) 曖昧性を残したまま処理を進める、のいずれかを選ぶことになる。a) の場合、問い合わせられた W_1 は、初めに構成した $S = S[i_1 \dots i_n]$ の情報量が不足していたのだと考え、より詳しい情報 $S' = S[i_1' \dots i_n']$ 、($n' > n$, $S' \leq S$) を W_2 に教えることになる。

7. ま と め

型定義に再帰的な定義を許しているシステムは少ない。また、オブジェクト（インスタンス）の比較においても、再帰的にその属性（スロット）を比較するものは少ない。実際、CESP⁹⁾、Effel¹⁰⁾ などのオブジェクト指向言語においても、スロットを再帰的にたどる機能は提供されていない。これは、一般的にこのような手続きが、停止性を保証できないためである。例えば、前出 Kobekko の例を Prolog のプログラムとみると、これは一般には停止しないものとなる。

循環的なクラス定義の意味を定めようとする試みは、Baader^{5), 6)}、Nebel^{7), 13), 14)} などによってなされてきた。しかしながら、クラスのみを用いた定義では、一部の有用な概念を定義することができなかった。また、彼らの研究ではクラス概念の存在証明や、subsumption 検査の計算の複雑さが問題となっており、付録にあるような簡単なアルゴリズムは与えられていない。

我々は、自然言語の対話を表現することを目的に、すべての意味（知識）をクラス概念としてとらえることを試みた。しかしながら、クラス概念を記述するに当たっても、インスタンスの同一性（equality）を記述する必要がある、これを表現するためにインスタンス変数を導入した。また、世界（状況）の記述をインスタンス変数の集合として捉えることで、状況の持つ情報量の差を概念の subsumption として統一的に捉えられることを示した。

インスタンス変数（変数を含むインスタンス）の考

えは UNITS³⁾ などでも用いられているが、我々はこれを単に変数を含むインスタンスと考えるのではなく、より積極的に変数が動くことによって定まる集合、すなわちクラス概念を定義するものとする。また、「もの」概念だけではなく状況一般を概念として理解することで、自然言語の発話の意味としての「状況タイプ」⁹⁾ を表現することを行った。このときより特殊（specific）な状況はより一般的な状況に包摂（subsume）されると考えられ、また状況に含まれている情報量の差は概念間の包摂関係に帰着される。

さて、インスタンスに対しては同一性（従って、対応するスロット値が等しい）が、またクラスに対しては包摂（subsumption）関係が、構成要素間の基本関係となる。UNITS では、Indefinite individual フレーム（我々のインスタンス変数に相当）に対しては、インスタンスの同一性（equality）を基本関係として、スロット値の問い合わせや検査が定義されている。これに対して我々の表現では、状況タイプがクラス概念として定義されているため、包摂が構成要素間の基本概念となる。そこで、我々は包摂を用いて知識への問い合わせ、書き込み機能を定義することで、理論の一貫性を保証した。

インスタンス変数による知識の表現は、英語における不定冠詞“a”の用法とよく似ている。例えば、

A man is mortal.

は、クラスとしての人の条件を述べているとも、またある一人の人の条件を述べているとも考えられる。これは、今まで自然言語やフレーム表現の曖昧さと見なされることが多かった^{11), 12)} が、我々はむしろ逆にこのような記述を基本概念と考える立場をとる。

なおこの枠組は、LISP (CLOS) を用いて実装されており、現在これを用いた時間と世界の変化を記述するシステムの開発を行っている。

参 考 文 献

- 1) Minsky, M.: A Framework for Representing Knowledge, in P. H. Winston (ed.) *The Psychology of Computer Vision*, McGraw-Hill, New York (1975).
- 2) Brachman, R. J. and Schmolze, J. G.: An Overview of the KL-ONE Knowledge Representation System, *Cognitive Science*, Vol. 9, pp. 171-216 (1985).
- 3) Stefik, M.: An Examination of a Frame-structured Representation System, *IJCAI '79*, pp. 845-852 (1979).

- 4) Reichgelt, H.: *Knowledge Representation, An AI Perspective*, Ablex Pub., Norwood, NJ. (1991).
- 5) Baader, F.: Terminological Cycles in KL-ONE-based Knowledge Representation Language, *AAAI '90*, pp. 621-626 (1990).
- 6) Baader, F.: Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles, *IJCAI '91*, pp. 446-451 (1991).
- 7) Nebel, B.: Terminological Cycles: Semantics and Computational Properties, in Sowa, J. (ed.) *Principles of Semantic Networks*, Morgan Kaufman, San Mateo (1990).
- 8) Barwise, J. and Perry, J.: *Situations and Attitudes*, The MIT Press, Cambridge (1983).
- 9) CESP 言語, 3版, AI 言語研究所.
- 10) Meyer, B.: オブジェクト指向入門, アスキー出版 (日本語訳) (1990).
- 11) Woods, W.: What Is in a Link: Foundations of Semantic Networks, Bobrow, D. and Collins, A. (eds.) *Representation and Understanding: Studies in Cognitive Science*, Academic Press, New York (1975).
- 12) Brachman, R. J.: What Is-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks, *IEEE Computer*, pp. 30-36 (1983).
- 13) Nebel, B.: Computational Complexity of Terminological Reasoning, *Artif. Intell.* Vol. 34, pp. 371-383 (1988).
- 14) Nebel, B.: Terminological Reasoning Is Inherently Intractable, *Artif. Intell.* Vol. 43 pp. 235-249 (1990).

付 録

一般にクラスの間の包摂 (subsumption) $\subseteq$ 関係を求めることは, クラス定義に再帰的定義を含まなくても, 計算量として手に負えない (intractable) 問題であることが知られている^{13),14)}. ここでは, 少し問題を簡単化して, 属性型記述において選言 (disjunction) を認めないものを考える.

さらに, 次の簡単化を行う. 1) 型制約がクラスであるものは, それをそのクラスのインスタンス変数で置き換える. 2) 集合型は出現しないものとする (この条件の緩和は比較的容易である). したがって, 最終的にはクラス C_i は以下のように定義されていることになる.

$$x_i^j: C_i = (\dots(a_i^j v_i^j)\dots)_{j=1\dots k}$$

ここで v_i^j は, 原始オブジェクト, またはインスタンス変数 $x_i^j: C_i$ のいずれかである.

$x: C = (\dots(a_i v_i)\dots)$ が $x': C' = (\dots(a_i' v_i')\dots)$ のサ

ブクラスである ($C \subseteq C'$) かどうかの検査は, 次のような再帰的な手続きで調べられる.

subsume (C, C') =

s-stack を空にする.

i-stack を空にする.

subsume1($x: C, x': C'$) を呼ぶ.

subsume1($x: C, x': C'$) =

$x: C, x': C'$ の定義式を展開する.

$x: C = (\dots(a_i v_i)\dots)$

$x: C' = (\dots(a_i' v_i')\dots)$

任意の a_i' に対して, $a_i' = a_i$ となる a_i が存在して,

1) v_i, v_i' がともに原始オブジェクトである場合,

原始オブジェクトの関係として $v_i \subseteq v_i'$

2) v_i, v_i' がともにインスタンス変数である場合,

これを $v_i = y: Y, v_i' = y': Y'$ とすると,

もし, $(y': Y' = z)$ かつ $z \neq y: Y$ となるものが i-stack にあれば失敗,

そうでなければ push ($y': Y' = y: Y$) to i-stack

もし, $(Y \subseteq Y')$ が s-stack になれば,

push ($Y \subseteq Y'$) to s-stack

subsume1($y: Y, y': Y'$)

これは, 言葉で説明するとおおよそ次のようになる. 二つのクラスの subsumption は, その定義にある属性型制約の subsumption に帰着される. このとき, 1) 一つのインスタンス変数は複数のインスタンス変数を subsume することができない, 2) 再帰的呼び出しの中で一度調べた subsume は 2 回目以降に出現したときは (成り立っているものとして) 調べなくてもよい.

これが, 必要条件であることは明らかであるが, 十分条件であることは次のようにして確かめられる.

(十分条件) いま, Ω の解釈を I とする. $\omega \in I(C)$ のとき, $\omega \in I(C')$ を示そう. ω によって決まる I^* が存在して, $\omega = I^*(x: C)$ である. $\tilde{I}^*$ を I^* と i-stack とから決まる関数とする. すなわち, $(y': Y' = y: Y)$ が i-stack にあれば, $\tilde{I}^*(y': Y') = I^*(y: Y)$ である. 左辺式 $y': Y'$ に同じものが出現しないことは i-stack の作り方からわかる. さて, $\omega = I^*(x: C) = \tilde{I}^*(x': C')$ である. また, $\omega, a_i^j = \omega, a_i = v_i(I^*) = v_i'(\tilde{I}^*)$ である. ここで $v(I^*)$ は, その中にインスタンス変数が出てきたときには, それに I^* を作用させたものである. $U_j = \{\tilde{I}^*(x_j^j: C_j)\}_{j=1,2,\dots}$ とすると, U_j は最大集合ではないがクラス定義を満たす集合の組となる. また, $\omega = \tilde{I}^*(x': C') \in U'$, かつ C' は最大集合だったから, U'

$\subseteq C'$ である。

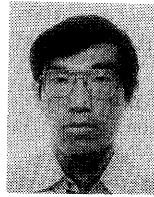
□

また、このアルゴリズムの停止性は次のようにして確認される。

(停止性) 再帰的に呼び出されるクラス間の subsumption 検査は、相互再帰的に定義されたクラス (有限個と仮定されている) の間について行えばよい。このとき、ある検査中に一度出てきた組み合わせは stack に登録され、二度目以後調べなくてもよい。したがって、最悪の場合でも相互再帰的に定義されたすべての組み合わせについて検査を行えば、このアルゴリズムは必ず停止する。

(平成 5 年 2 月 10 日受付)

(平成 5 年 6 月 17 日採録)



伊藤 昭 (正会員)

昭和 47 年京都大学理学部物理学科卒業。昭和 45 年同大学院理学研究科博士課程修了 (理学博士)。同年郵政省電波研究所 (現通信総合研究所) 入所。以後、通信ネットワーク、知識処理、ヒューマンインターフェースの研究に従事。現在、同所知識処理研究室長。電子情報通信学会、人工知能学会各会員。



熊本 忠彦 (正会員)

昭和 63 年筑波大学第三学群情報学類卒業。平成 2 年同大学院修士課程修了。同年、郵政省通信総合研究所入所。現在、同所関西先端研究センター知識処理研究室にて、自然言語処理の研究に従事。特に、日本語話し言葉による人間-計算機間対話に興味を持つ。人工知能学会会員。