データベースシステム 91－1
（1992. 11. 5）

オブジェクト指向データベースプログラミング言語 DOTPLの動的OIDとメソッド継承

劉 渤江† 塚本 昌彦‡ 西尾 章治郎§
宮原 秀夫†

†大阪大学基礎工学部情報工学科
‡シャープ株式会社技術本部情報技術開発センター
§大阪大学工学部情報システム工学科

ドット記法とIS-A関係を用いて知識を表現する拡張項表現DOTは，簡潔な属性値継承機構を代数的に記述し，オートマトンを用いた推論機構を実現している．本稿では，DOTをオブジェクト指向プログラミング言語として拡張したデータベースプログラミング言語DOTPLを提案する．DOTPLは，IS-A関係の変更に応じて動的にオブジェクト識別子を付与する機構をもち，知識の変更にともなって二つ以上のオブジェクトの同一性が動的に変わりうるモデルを実現する．さらに，実行中のメソッドを動的に切り替える動的メソッド継承の機構をもち，知識の変更にともなって，オブジェクトの特性を動的に変更する機構を提供する．これらのDOTPLの機構によって，知識の更新に動的に対応可能な柔軟なデータベースモデルの実現が可能となる．

Dynamic Object Identity and Method Inheritance in Object-Oriented Database Programming Language DOTPL

Bo-jiang LIU† Masahiko TSUKAMOTO‡ Shojiro NISHIO§
Hideo MIYAHARA†

†Department of Information and Computer Sciences
Faculty of Engineering Science, Osaka University
‡Information System Research and Development Center, Sharp Corporation
§Department of Information Systems Engineering
Faculty of Engineering, Osaka University

Extended Term Representation DOT, which expresses knowledge using *dot notation* as well as *IS-A relation*, specifies algebraically simple inheritance mechanisms of attribute values and realizes inheritance mechanisms based on the structure of automata. In this paper, we propose database programming language *DOTPL* which extends the capability of DOT as an database programming language. DOTPL has the mechanism which dynamically provides *object identifiers* according to the update of IS-A relation. By this mechanism, the equivalence relation of objects are dynamically updated following the change of knowledge. Furthermore, this language DOTPL has dynamic *method inheritance* mechanism which dynamically modifies the methods under execution according to the update of knowledge and realizes dynamic update of the property of objects. These features of DOTPL provide a real world modeling which flexibly handles dynamic update of knowledge. It becomes possible to construct database models which are very flexible to dynamic update of knowledge.

1 はじめに

近年, データベースの研究分野では, 演繹データベース (deductive database)[18]とオブジェクト指向データベース (object-oriented database)[4]に関する研究が盛んに行なわれている. 演繹データベースは一階述語論理に基づいた演繹機構を備え, 強力な数学的基盤が確立されているが, 実世界のデータのモデリング能力の限界が指摘されている. 一方, オブジェクト指向データモデルは, 対象となるデータの複合的構成や属性値の継承などの, 現実世界の対象のもつ性質を有効にとらえモデル化することに優れ, 新しいデータモデルとして注目されてはいるが, 理論的基盤は確立されていない. これら二つの長所を統合したデータベースが, 次世代知識データベースを構築するための強力な基礎モデルの一つと考えられている. このような融合されたモデルを演繹オブジェクト指向データベース (Deductive and Object-Oriented Database: DOOD) とよび[8][15][17], さまざまな研究が盛んに行なわれている (例えば, ψ項[1][2][3], O 項[10], F-logic[6], Quixote[16]など). そこでは, 演繹データベースを出発点として, それぞれの提案するオブジェクト指向の概念を取り入れた項表現を用いて, 演繹データベースのパラダイムで培われた論理の展開法, 問合せやルール表現を基盤として演繹データベースとオブジェクト指向データベースの融合を目指す方法論が採られている.

DOT(Deductive and Object-oriented Term representation) は, このような演繹データベースとオブジェクト指向データベースの発展的融合を実現するための拡張項の一つとして提案された[11][12]. 項表現 DOT は次のような特徴を持つ.

1. 型と実体の区別をしない.
2. DOT 式間の IS-A 関係を用いて知識を表現する.
3. 問合せに対する答えが正規表現で与えられる[12].
4. ある知識が他の知識に依存しているかを示す論理式 (知識の従属性) を求めることができる[14].

型と実体の区別は, 本質的には視点によって変わってくるものである. 例えば, 'A 君' を実体, '人' を型とする場合もあれば, 'A 君' を, 「'名前' が " 田中" で, '親' が '太郎' と '月子' で, ...」といったいくつかの属性値制約をもつ型と考え, その型に属する実体として「4 月 10 日の A 君」あるいは「6 月 10 日の午後 5 時の A 君」などを考えたい場合もある. 従来の応用では, このような様々な視点で同一のものを見ることはなかった. しかし, 知識ベースなどの新しい分野にデータベース技術を導入するにあたっては, このような様々な視点でものを見ることができるようなモデルを考える必要がある.

DOT では, 「人間の親は人間である」という知識を次のように表現する.

人間. 親 IS-A 人間

DOT は, 型と実体を区別せず, ドット記法による仮想的なオブジェクト表現を導入することにより, オブジェクト間の IS-A 関係を簡単に記述する機構を備えている. 項表現 DOT に基づいて定義されるデータモデルを DOT データモデルといい, その代数構造として DOT 代数が提案された[12][13]. DOT 代数においては, 任意の元の上界または下界が正規集合となることが知られており, この性質が有限表現された問合せの答えを導出するうえで重要な役割を担っている. 問合せに対する答えは, 有限オートマトンを構成することによって得られる[11][12].

本稿では, 上記のような特徴を持つ DOT データモデルに対する操作を記述するためのプログラミング言語モデルを提案する. 本稿で提案する DOT プログラミング言語 **DOTPL**(DOT Programming Language) は, 特に, DOT データベースの推論能力を拡張することを目標とするデータベースプログラミング言語 (DBPL) であると同時に, オブジェクト識別性, 継承, メッセージ・センディングなどの機能を備え, オブジェクト指向プログラミング言語 (OOPL) の一つとみなすことができる. オブジェクト指向データベースプログラミング言語 (OODBPL) として本稿で提案する DOTPL は, 次のような特徴をもつ.

1. オブジェクトの識別を動的に行う.
2. メソッドの継承を動的に管理する.

DOTPL では, 双方向の IS-A 関係をもつオブジェクトを同じオブジェクトと見なす. つまり, オブジェクト間の IS-A 関係に従ってオブジェクトを識別する. メソッドはドット記法で表現されるオブジェクトに対して定義され, 実行時に成立する IS-A 関係に従って動的に継承される.

以下 2 章では, DOT データモデルとその機能について簡単に紹介する. 3 章では, DOTPL のシステムモデルを与える. 4 章で, IS-A 関係の動的な機構に基づくオブジェクト識別子の定義, 管理機構を与える. 5 章で, メソッド継承の管理機能を述べる. 6 章では, 関連研究との比較を行う. 最後に 7 章では, 結論として本稿のまとめと今後の研究課題を述べる.

なお, 知識ベースシステムとして有効な機能を果たし得る DOT データベースを高度利用するための高

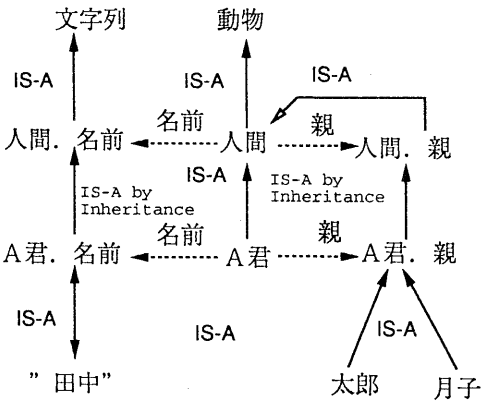

図 1: 'A 君' と '人間' に関する DOT データベース

級核言語として, 本稿で提案する DOTPL が有効に機能することを検証するために, 現在, 全システムを UNIX[1] をベースとするワークステーションで実装中である.

2 項表現 DOT の概要

この章では, 例を用いて項表現 DOT の構文と意味の直感的な説明, および, IS-A 関係に関する推論機能の説明を行う. 詳しくは, 文献[11][12]を参照されたい.

2.1 項表現 DOT の特徴

演繹データベースにオブジェクト指向の概念を取り入れることを目指した拡張項表現に関する研究では, 演繹データベースを出発点として, それぞれの提案するオブジェクト指向の概念を取り入れた項表現を用いて, 演繹データベースのパラダイムで培われた論理の展開法, 問合せやルール表現を基盤として演繹データベースとオブジェクト指向データベースの融合を目指している. DOT には, このような研究と同様, 演繹データベースの論理展開をベースに, オブジェクト指向の考え方を融合したデータベースモデルである.

次の二つの項は, 'A 君', '人間' に関する DOT による記述である.

人間 : 動物 [名前 → 文字列,
　　親 → 人間]　　(1)

A 君 : 人間 [親 ← 太郎,
　　名前 ↔ "田中",
　　親 ← 月子]　　(2)

[1]UNIX は米国 AT&T の登録商標である.

(1) の項は, '人間' に関する性質を表すもので, 「人間は動物であり, 名前は文字列である. 人間の親は人間である. 」ということを表している. (2) の項は 'A 君' に関するもので, 「A 君は人間であり, 名前は"田中"である. 太郎と月子は彼の親である. 」ということを表している.

図 1 は, 項により表現される意味を示したものである. 特に, 「人間の親」は '人間. 親' というような**仮想オブジェクト**で表現される. 仮想オブジェクトは, 通常 DOT 項の中ではオブジェクトとして明示的に現れていないが, これを一般的なオブジェクトとして扱い, **DOT 式**と呼ぶ. ここで, '名前' と '親' のようなオブジェクトの属性を表すシンボルを**ラベル**とよぶ.

DOT 項は, **IS-A 関係**に関する知識を表現しているものと考える. '→', '←' は矢印の向きの IS-A 関係を表す. '↔' は双方向の IS-A 関係を表す. 図 1 において, IS-A 関係を表す実線のアークは **IS-A リンク**とよび, 属性と対応する点線のアークは**属性リンク**とよぶ.

DOT では, 次のような**継承ルール**を導入する.

DOT 継承 : DOT 式 a, b 及び DOT ラベル l に対して, 「a IS-A b」が成立すれば, 「$a.l$ IS-A $b.l$」も成立する. □

前述の項表現を用いて表された知識と上記の継承ルールを用いて, 例えば,

A 君. 親　IS-A　人間. 親
太郎　IS-A　人間. 親
太郎　IS-A　人間

などが演繹できる.

DOT の意味論は, **意味集合**という DOT 式間の 2 項関係により特徴づける. 例えば, (1), (2) の項の意味集合は以下のようになる.

{ (人間, 動物), (人間. 名前, 文字列),
(人間. 親, 人間), (A 君, 人間),
(A 君. 名前, " 田中"), (" 田中", A 君. 名前),
(太郎, A 君. 親), (月子, A 君. 親) }

項表現の意味は, 意味集合を含む最小の IS-A 関係として与えられる.

2.2 DOT の推論機能

DOT では, IS-A リンク (あるいは IS-A 関係) を辿ることにより, 新たな IS-A 関係を導くことができる. 図 1 では, 例えば, '太郎' から 'A 君. 親', 'A 君. 親' から '人間. 親', '人間. 親' から '人間' という IS-A リンクを辿ることにより,

太郎 IS-A 人間

という IS-A 関係を導くことができる.

DOT 問合せは, 次のように表される.

? A 君 [親 → X] (3)

? A 君 [親 ← X] (4)

ここで, Xは DOT 変数である. (3) の問合せは,

A 君. 親 IS-A X

を満たす Xを問合せるものである.

A 君. 親 IS-A A 君. 親
A 君. 親 IS-A 人間. 親
A 君. 親 IS-A 人間
A 君. 親 IS-A 動物

が成立するため, (3) に対する答えは, 以下のように表される.

X : A 君. 親 + 人間. 親 + 人間 + 動物

ここで, '+' は「あるいは」という意味をもつ. また, (4) の問合せは,

X IS-A A 君. 親

を満たす Xを問合せるものである.

A 君. 親 IS-A A 君. 親
太郎 IS-A A 君. 親
月子 IS-A A 君. 親

が成立するため, 答えは,

X : A 君. 親 + 太郎 + 月子

と表される.

問合せの答えは, 無限になる可能性がある. 一般に, DOT 問合せの答えの集合はオートマトンの正規表現を用いて表現できることが示されている. 例えば, DOT 項

A 君 [先祖 ← A 君. 親,
先祖 ← A 君. 先祖. 親]

という知識があるとき, 問合せ ?-A 君 [先祖 ← X] に対する答えは,

X : A 君.(親 + 先祖). 親*

となる. 文献 [11] では, このような正規表現を用いた答を得るための方法を示している.

3 DOTPL のモデル

3.1 基本動機

DOT データベースモデルは, DOT 記法による仮想オブジェクトと IS-A 関係を用いる DOOD モデルとして提案され, その特殊な推論能力が示されてきた. この章では, OODBPL の一つとして本稿で提案する DOTPL モデルの概要について述べる.

DOTPL の設計の目的は, まず第一に, DOT データベースの推論能力を拡張することである. 従来の DOT には, **反射律**, **推移律**, **継承律**という三つの公理に基づく推論能力しかないため, 公理の追加などさらに高度な推論のための準備が必要と思われる. したがって, DOT データモデル自身よりも高い推論能力をもつ言語を導入することを考える. 第二に, DOT データモデルでは, IS-A 関係と属性値継承にのみ着目していたため, オブジェクト識別性やメソッドなどオブジェクト指向の多く特徴の導入は不十分であった.

そこで, DOTPL では, DOT データモデルの特徴を生かしたうえで, オブジェクト指向の概念にとって本質的なオブジェクト**識別性**, メッセージ・センディングなどの特性を新たに導入する.

- **オブジェクト識別性.**
  一般に, オブジェクト識別性の概念は, あるオブジェクトをほかのオブジェクトから区別するための重要な性質である. 通常, オブジェクト識別性を実現するために, **オブジェクト識別子 (oid)** を導入する. DOTPL では, 実行時における IS-A 関係に基づいて, あるオブジェクトを他のオブジェクトから区別する. DOTPL では, 必要に応じてすべてのオブジェクトに対して識別子を発行するが, 双方向に IS-A 関係が成立するオブジェクトに対して同じ識別子を発行する. IS-A 関係が成立するかどうかの判定は, DOT 推論カーネルに基づいて行う.

- **メッセージ・センディング.**
  DOTPL では, 多くの OOPL と同じように, すべての操作は, オブジェクト間のメッセージ・センディングによって起動される. 各オブジェクトは, 自身の手続きを用いて, このメッセージに応じる. この手続きを**メソッド**と呼ぶ. オブジェクト間のメソッドの継承は, IS-A 関係に従って動的に行われる.

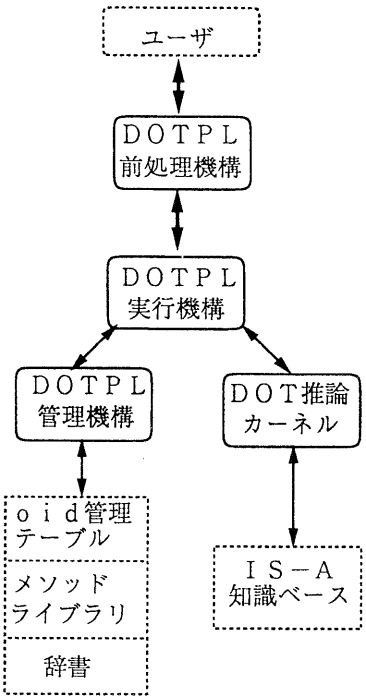

図 2: DOTPL モデル

3.2 DOTPL モデルの構成

DOTPL モデルは図 2 で示すように, **DOT 推論カーネル**, **DOTPL 管理機構**, **DOTPL 実行機構**, **DOTPL 前処理機構**の四つのサブシステムから構成される.

- **DOT 推論カーネル.**
  DOT 推論カーネルは, DOT 式間の IS-A 関係に関する知識を格納する **IS-A 知識ベース**を管理する. IS-A 知識ベースに格納されている知識に従って IS-A 関係に関する推論を行う. IS-A 知識ベースに格納されている知識を Rとする. 他のモジュールからの要求に応じて, 以下の処理を行う.
  - **初期化.** 関係 Rに初期値をセットする. R の初期値は, 空集合である.
  - **追加.** 追加する組 (x, y) に対して, $(x, y) \notin R$ならば, (x, y) を Rに追加する.
  - **削除.** Rから組 (x, y) を削除する.
  - **問合せ.** IS-A 関係に対する問合せは, DOT 項表現を用いて表現される. これらによって, 次の三つの基本機能とこれらを組合せた複合的な問合せが可能となる.
    1. $(d_1, d_2) \in R$が成立するかどうかを調べる.
    2. 「d IS-A X」を満たすオブジェクト X の集合を求める. ここで, Xは変数である. このような集合を d の**上界集合**と呼び, $U(d)$ と記す.
    3. 「X IS-A d」を満たすオブジェクト X の集合を求める. ここで, Xは変数である. このような集合を d の**下界集合**と呼び, $L(d)$ と記す.

    問合せに対する答は, 正規表現を用いて表現する.
  - **極小集合.** DOT 式の有限集合 M の**極小集合 (minimal subset)** を求める. ここで, Mの極小集合 $MS(M)$ を $\{m \in M \mid m'$ IS-A m かつ $m' \in M$ ならば $m \sim m'\}$ と定義する. ただし, 「$a \sim b$」は, 「a IS-A b」かつ「b IS-A a」が成立することを意味する. ここで, 関係 '$\sim$' は同値関係である.

- **DOTPL 管理機構.**
  DOTPL 管理機構は, オブジェクト指向の概念を導入するための管理機構である. DOTPL 管理機構は, DOT 推論カーネルに基づいて以下のものを管理する.
  1. DOT 式とオブジェクト識別子の対からなるテーブル (**oid 管理テーブル**).
  2. DOT 式に対するメソッド定義の集合 (**メソッドライブラリ**).
  3. DOT 式に用いる名前の辞書.

- **DOTPL 実行機構.**
  DOTPL 実行機構は, DOTPL のプログラムの実行を管理するモデルである. DOTPL 言語の基本演算は, 以下の二種類である.
  1. **IS-A 演算子.** 例えば, 「A 君 IS-A 飯田. 家庭教師」.
  2. **非 IS-A 演算子.** 例えば, 「A 君 NOT IS-A 先生」は, 「A 君 IS-A 先生」が成立しないことを表す.

  簡単にいえば, DOTPL プログラムの基本単位は DOT の IS-A 関係である. DOTPL の実行機

構は, DOT 推論カーネルを用いて, DOTPL のプログラムを評価することによって実行される.

- **DOTPL 前処理機構.**
  DOTPL 前処理機構は, DOTPL のユーザインタフェースである. DOTPL 前処理機構は, 項表現インタフェース, DOTPL 言語インタフェース, ユーザインタフェースの部分からなる. 項表現インタフェースは, DOT 項の文法に従って DOT 項をパーズする. DOTPL 言語インタフェースは, DOTPL の応用プログラムを定義する. ユーザインタフェースは, ユーザから発せられた DOTPL の操作に対して, その結果をある方法で表示することにより, DOTPL を直接に利用するツールである.

4 動的な IS-A 関係に基づくオブジェクト識別性

オブジェクト識別性は, オブジェクト指向パラダイムにおいて特に重要な概念である. オブジェクト識別性とは, 一般に, 実世界の中のオブジェクト (実体) を, 計算機上で区別するための手段と考えることができる. 通常, 各オブジェクトにはこれを識別するためのオブジェクト識別子 (oid) が与えられる. 通常のオブジェクト指向システムでは, システムが組み入れるオブジェクト識別子を用いる. このようなシステム中のオブジェクト識別子の重要な役割は, オブジェクト識別子によるオブジェクトへの参照である. つまり, オブジェクト間の意味構造は, オブジェクト識別子によって実現される.

項表現 DOT はオブジェクト識別性の概念をもっていないが, DOTPL では, 項表現 DOT の推論と表現能力を拡張するためにオブジェクト識別性の概念を導入する. DOTPL では, オブジェクト識別性の基礎を DOT 式名と IS-A 関係の集合に置き, オブジェクト識別子は DOT 推論カーネルを用いて実現する. DOTPL では, IS-A 関係の動的な変化に応じて, オブジェクトは動的に区別される.

DOTPL では, オブジェクトは DOT 式によって表され, DOT 式にはオブジェクト識別子が割り振られる. DOT 式 a, b のオブジェクト識別子を $\#(a), \#(b)$ と表すとき, $a \sim b$ であるための必要十分条件が $\#(a) = \#(b)$ となるようにオブジェクト識別子が割り振られる. つまり, 双方向 IS-A 関係を満たす DOT 式オブジェクトに対して同じオブジェクト識別子を与える. 二つのオブジェクトの同値性は, このようなオブジェクト識別子の同一性により定義される. 例えば,

A 君. 親　IS-A　太郎
太郎　IS-A　A 君. 親

のとき,

$$\#(\text{A 君. 親}) = \#(\text{太郎})$$

が成立する. これら二つの DOT 式はシステム内では同一のものとみなされる.

オブジェクト識別子管理部は, DOT 式とオブジェクト識別子の対応関係を表形式で管理し, 以下の手順で DOT 式 d に対してオブジェクト識別子を割り振る.

1. d の同値類を, 例えば,

   $$? \ d : X, X : d$$

   などの問合せを用いて, DOT 推論カーネルに問合せる.

2. oid 管理テーブルのエントリに同値類のメンバ d' があれば, そのオブジェクト識別子 oid を調べる. もしなければ, 新たなオブジェクト識別子 oid を生成する.

3. (d, oid) を登録する.

DOT 推論カーネルに基づいて, 動的に DOT 式オブジェクト間の IS-A 関係を調べることによって, DOT 式オブジェクトの同値類を調べる. 例えば, 図 1 の例で, 'A 君. 名前' にオブジェクト識別子を与えるとき, 以下のことを DOT 推論カーネルで導出する.

A 君. 名前 ～ "田中".

ここで, "田中" に識別子 $oid8473$ が割り当てられている場合, ⟨ A 君. 名前, $oid8473$⟩ を登録する. " 田中" が登録されていない場合は, 使用されていないオブジェクト識別子を生成し (例えば, これを $oid9542$ とする), ⟨ A 君. 名前, $oid9542$⟩ を登録する.

もし, 図 1 の IS-A 関係データベースに対して, 新しい IS-A 関係

A 君. 親　IS-A　花子
花子　IS-A　A 君. 親

(つまり, 'A 君. 親' は '花子' ただ一人であること) を追加すると, DOT 推論カーネルは以下のことを導出する.

$$\#(\text{A 君. 親}) = \#(\text{花子}).$$

そして, 'A 君. 親' と '花子' に対して同じ *oid* (例えば, *oid*8483) を付与する.

ここで, 知識の変更にともない *oid* の変更を行う必要があることを考慮しなければならない. 例えば, 知識の追加によって, それまで異なる *oid* が振られていた DOT 式が同値になり, 同じ *oid* を振り直さなければならなくなる場合がある. このように異なる二つ以上のオブジェクトが知識の変更により一つのオブジェクトになることを, **オブジェクトの融合**とよぶ. また, 知識の削除によって, それまで同じ *oid* が振られていた DOT 式の同値性がくずれ, 異なる *oid* を振り直さなければならなくなる場合がある. このように一つのオブジェクトが知識の変更により二つ以上のオブジェクトになることを**オブジェクトの分裂**とよぶ.

オブジェクトの融合, 分裂により, 従来の OOPL と比べて柔軟なプログラミングが可能になる. ここで, 次章でより詳しく定義するメソッドに関する一例として, 次のプログラムを考える.

A 君 : 人, B 氏 : 人,
A 君. 歩いている (),
B 氏. 持っている (傘)

この例は, 人に対して, '歩いている ()', '持っている (傘)' というメソッドを定義し, 各々, 歩いている人の姿, 傘を持っている人の姿を画面に表示するメソッドであるとする. 最初は, A 君と B 氏は別々の人と認識され, 歩いている人と傘を持っている人が表示される. ここで,

A 君 ~ B 氏

という知識を追加すると, A 君と B 氏の *oid* が一致し, 歩いている人と傘を持っている人が実は同じ人だったことを認識し, オブジェクトの融合が起こる. さらに,

A 君 ~ B 氏

という知識を削除すると, さきほどの A 君と B 氏に再び異なる *oid* が振り直され, 歩いている人と傘を持っている人がやはり違う人だったと認識し, オブジェクトの分裂が起こる.

5 メソッドとその継承管理機構

動的に知識を処理することは, DOT データベースにおいて非常に重要な考え方の一つである. DOTPL における動的な知識処理は二つの面から捉えることができる. 一つの面は, DOT データベースの IS-A 関係を動的に更新する知識処理の機能をもつ点である. この機能は, DOT 推論カーネルにより実現される. もう一方は, DOT データベースの IS-A 関係にもとづいて, この上にオブジェクト自身あるいはオブジェクト (DOT 式あるいは IS-A 関係) 間の知識を抽象的に表現することである. このような機能によってオブジェクトの知識の動的な変更に応じて抽象的な意味を反映する. つまり, このような動的なオブジェクトを導入することにより, 抽象型オブジェクトを実現する. この動的なオブジェクトは, DOTPL のメソッドという.

メソッドは,

$$A :\!- B$$

という形式で表される. A をメソッドのヘッド, Bをメソッドの**本体**と呼ぶ. ヘッドは,

$$d.m(param)$$

という形式で表される. d は DOT 式, m はメソッド名, $param$ はパラメータのリストである. パラメータは, DOT 式あるいは以下で述べるメッセージ・センディングのいずれかである. 本体部はルールの集合からなる. ルールは,

$$C \Leftarrow D;$$

という形式で表される. ここで, Cは変数を含む DOT 項の非空集合あるいは DOTPL のメソッド, Dは変数を含む DOT 項の集合あるいはメソッドである. また, Bは Cに現われる変数のみを含むものとする.

メソッドは, 複数の DOT 式に対して多重定義することが可能である. ここで, m_dは DOT 式 d に対して定義されているメソッド m を表すものとし, 一般にメソッド**定義**とは, メソッドの有限集合を表すものとする.

以下では, オブジェクト '人間' に対して定義される '生活 ()' というメソッドの例を示す.

人間.生活 (P, T) :-
P. トレーニング () $\Leftarrow T$: 朝;
P. 仕事 () $\Leftarrow T$: 昼;
P. 遊び () $\Leftarrow T$: 晩;
P. 寝る () $\Leftarrow T$: 夜;

ここで,

鈴木 : 人, 正午 : 昼

のとき, メッセージ

鈴木. 生活 (鈴木, 正午)

に対して, メソッドのボディに記述されたルールを評価し, 条件を満たすルール

鈴木. 仕事 ()

を評価し, 実行を完了する.

OOPL パラダイムにおいては, すべてのプログラムの実行は, オブジェクト間のメッセージ・センディングによって起動される. メッセージ・センディングは, DOTPL のメソッドの実行を行なうために重要な役割を果たす. DOTPL では, メッセージ・センディングは以下のいずれかの形式で記述する.

$$a.m(param) : b \qquad (5)$$
$$a!m(param) : b \qquad (6)$$

ここで, a と b はオブジェクトであり, $m()$ はメソッドである. $param$ は, DOT 項あるいはメッセージ・センディングのリストである. a と b が同一のオブジェクトであるとき, ': b' は省略してもよい. (5) 式は, レシーバ a に対して, オブジェクト b で有効なすべてのメソッドを起動するということを意味する. (6) 式は, レシーバ a に対して, オブジェクト b で有効なメソッドの一つだけを起動することを意味する. ここで, オブジェクト a で有効なメソッドとは, a の上界集合 $U(a)$ の中で m が定義されているものからなる集合を M とすると, M の極小集合 $MS(M)$ に対して定義されているメソッドのことをいう. (5) 式の場合, メソッドの起動の順序は言語仕様外であり, 実装依存あるいは非決定的である. (6) 式の場合, どのメソッドが起動されるかは言語仕様外であり, 実装依存あるいは非決定的である.

DOT ルールの条件部分は, レシーバとパラメータにある種の制限を加える. 例えば, 次のプログラムを考える.

A 社社員. 定期昇給 (X) :-
X. 昇給 (X, 8%) ⇐ X. 役職 : 平社員;
X. 昇給 (X, 10%) ⇐ X. 役職 : 準管理職;
X. 昇給 (X, 12%) ⇐ X. 役職 : 管理職;

この DOTPL で記述されたプログラムは, A 社社員に対して有効なメソッドで,

X. 役職 IS-A 平社員ならば '昇給 (X, 8%)',
X. 役職 IS-A 準管理職ならば '昇給 (X, 10%)',
X. 役職 IS-A 管理職ならば '昇給 (X, 12%)'

というメソッドを表している.

メソッド実行機構は, メソッドの起動, 実行, 終了を司る. オブジェクト d に対するメソッド m の起動は, 次の手順で行われる.

1. DOT 推論カーネルに対して, d の上界集合 $U(d)$ を問合せる.

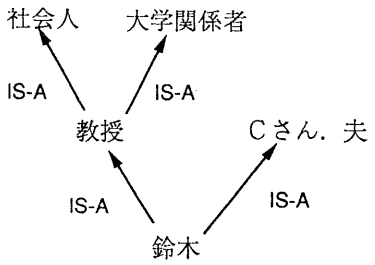

図 3: '鈴木' に関する DOT データベース

2. メソッド管理機構に, $U(d)$ および m を渡し, $U(d)$ の元の中で m が定義されているものからなる集合 M を問合せる.

3. DOT 推論カーネルに, M の極小集合 $MS(M)$ を問合せる.

4. $MS(M)$ の各オブジェクト o に対して定義されているメソッド m_o を起動する.

例えば, 次のような DOT データベースとメソッドが存在するものとする.

R = { (鈴木, 教授), (鈴木, C さん. 夫),
(教授, 大学関係者),(教授, 社会人) },
M = { 仕事 $()_{\text{C さん. 夫}}$, 仕事 $()_{\text{大学関係者}}$,
仕事 $()_{\text{社会人}}$ }.

図 3 はこの例における IS-A 階層を図示したものである. メッセージ・センディング鈴木. 仕事 () に対して, DOTPL 実行管理機構は, まず, DOT 推論カーネルを用いて,

$$U(\text{鈴木}) = [\text{鈴木} + \text{教授} + \text{C さん. 夫} + \text{大学関係者} + \text{社会人}]$$

を求める. DOT 推論カーネルは, 上界問合せに対して正規表現で答える. U(鈴木) の中でメソッド '仕事 ()' が定義されているものは,

$$M = \{ \text{大学関係者, 社会人, C さん. 夫} \}$$

である. メソッド定義は有限集合であるため, この集合は必ず有限集合となる. 集合 M に対する極小集合 $MS(M)$ は次のようになる.

$$MS(M) = \{ \text{大学関係者, 社会人, C さん. 夫} \}.$$

以下, いくつかのメッセージ・センディング仕様に対する DOTPL の実行管理機構の動作の例を示す.

- '鈴木.仕事()' に対しては, '仕事()大学関係者' と '仕事()社会人' と '仕事()C さん.夫' の三つのメソッドが起動される.
- '鈴木.仕事() : 教授' に対しては, '仕事()大学関係者' と '仕事()社会人' の二つのメソッドが起動される.
- '鈴木!仕事()' に対しては, '仕事()大学関係者', '仕事()社会人', '仕事()C さん.夫' のうちのいずれか一つのメソッドが起動される.
- '鈴木!仕事() : 教授' に対しては, '仕事()大学関係者', '仕事()社会人' のうちのいずれか一つのメソッドが起動される.

ここで,

'仕事()大学関係者', '仕事()社会人', '仕事()C さん.夫'

の各々はオブジェクト '大学関係者', '社会人', 'C さん.夫' から別々に継承されるメソッド '仕事()' を意味している.

6 関連研究との比較

この章では, DOTPL と他のオブジェクト指向データベースプログラミング言語との比較を行なう.

F 論理[6]では, オブジェクト識別子がオブジェクトに対する 'ハンドル' と考える. この 'ハンドル' により, オブジェクトに対するすべての情報をアクセスすることができる. 例えば, F 論理における項表現

$$\begin{aligned}&\mathrm{bob}[\mathrm{name} \rightarrow \text{"Bob"};\ \mathrm{age} \rightarrow 40]\\&\mathrm{cs}[\mathrm{dname} \rightarrow \text{"CS"};\ \mathrm{mngr} \rightarrow \mathrm{bob}]\end{aligned}$$

では, 'bob' がオブジェクト識別子として定義される.

F 論理では, メソッドの継承をクラス階層により定義し, 多重継承も許す. オブジェクトの複数の親クラスが同じ名前の異なるメソッドを定義する場合には, メソッド名の競合問題をメソッドのパラメータ化によって解決する. 例えば, 次のようにメソッド '$mthd$' に対して, '$param(mthd, Class)$' のようなメソッド集合を定義する.

$$\begin{aligned}&Class[param(mthd, Class)@X, Y \rightarrow Z]\\&\quad \Leftarrow Class[mthd@X, Y \rightarrow Z].\end{aligned}$$

つまり, F 論理には本質的には同じ名前のメソッドが存在しない. DOTPL では, IS-A 関係を推論することによりメソッドの継承を管理し, メッセージ・センディングを発行する時点でオブジェクトに対して適用するメソッドを決定している. このような動的なメソッド継承管理メカニズムにより, メソッドの多重継承に関する名前競合問題を解決できる.

LLO[9]は, メソッド記述に特徴をもつ. 型チエックによりメソッド継承を実現する. 例えば, メソッド '$mthd(Type)()$' に関するメッセージ・センディング '$mthd(SubType)()$' に対して, '$SubType$' が '$Type$' のサブタイプであれば, '$SubType$' に対してもメソッド '$mthd(SubType)()$' を起動する. しかし, 型とクラスを同一視しているために, 多重継承に関するメソッドの名前競合問題は未解決のまま残っている. DOTPL では, 動的な IS-A 関係の推論機構によりメソッド継承を管理する点で LLO と異なる.

F 論理では, オブジェクト識別子をオブジェクトの外延と考え, オブジェクトのアクセスあるいはオブジェクトへの参照のように使われる. LLO では, オブジェクト識別性が名付け値 (named value) という概念により扱う. 名付け値は, オブジェクト識別子とそれに割り当てられた値の二層概念をもつ. さらに, XSQL[7]では, オブジェクト識別性を論理レベルと物理レベルに分け, 項自体を論理レベルの識別子, 実現時の識別子を物理レベルの識別子とし, 論理レベルでは異なるオブジェクトが物理レベルで同一になる場合がありうるとしている. これらにおけるオブジェクト識別子の考え方は基本的にはほとんど同一のものと考えられる. 本稿のオブジェクト識別子の考え方は, これらの考え方を DOT 式で表されるオブジェクトに拡張し, 上記のようないくつかの問題を解決するために発展させたものである. なお, XSQL のパス式 (path expression) は構文上は本稿のドット式と非常に類似しているが, XSQL では属性値をたどる問合せに用いるのに対し, 本稿では DOT 式自体をオブジェクトとみなし, oid を付与する点が異なる.

7 おわりに

本稿では, DOT データモデルにもとづく演繹オブジェクト指向システムの設計と実現を目標として, OODBPL の側面を備えた DOTPL を提案した. DOTPL は, DOT 推論カーネルにおいて IS-A 関係を推論することにより, 動的なオブジェクト識別と動的なメソッド継承を実現した. DOTPL は, 知識ベースシステムとして有効な機能を果たし得る DOT データベースを高度利用するための核言語として重要である.

現在までに, 図 2 に示した DOTPL モデルの核部分にあたる DOT 推論カーネルの試作システムの実装は完了して評価を開始しており, それと並行してシステム全体を実装中である. 今後は, システム全体の実

装によってDOTPLの有効性を評価する必要があるとともに, 知識更新時のオブジェクト識別子および実行中のメソッドの管理手法について検討する必要がある.

謝辞

著者の一人塚本昌彦は, 日頃ご指導頂くシャープ(株)技術本部河田亨副本部長, 情報技術開発センター千葉徹第4開発部長に深謝の意を表す. なお, 本研究の一部は, 文部省科学研究費および(財)新世代コンピュータ技術開発機構からの研究補助費によるものである.

参考文献

[1] Aït-Kaci,H. : A Lattice Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures, *Ph.D. Thesis, Univ. of Pennsylvania* (1984).

[2] Aït-Kaci,H. :An Algebraic Semantics Approach to the Effective Resolution of Type Equations, *Theoretical Computer Science, Vol.45,* pp.193-351 (1986).

[3] Aït-Kaci,H. and Nasr,R. : LOGIN : A Logic Programming Language with Built-in Inheritance, *Journal of Logic Programming, Vol.3,* pp.185-215 (1989).

[4] Atkinson,M., Bancilhon,F., DeWitt,D., Dittrich,K. ,Maier,D., and Zdonik,S. : The Object-Oriented Database System Manifesto (Invited Paper), *in* [8], pp.223-240 (1991).

[5] Chen,W., Kifer,M., and Warren,D.S. : Hilog as a Platform for Database Languages (or why predicate calculus is not enough), *Proc. of the 2nd Int'l Workshop on Database Programming Language*, Gleneden Beach, Oregon, pp.121-135 (1989).

[6] Kifer,M., Lausen,G., and Wu,J. : Logical Foundations for Object-Oriented and Frame-Based Language, *Technical Report 90/14 (revised), Dept. of Computer Science, Univ. of New York at Stony Brook* (1990).

[7] Kifer,M, Kim,W., and Sagiv,Y. : Querying Object-Oriented Databases, *Proc. of the ACM-SIGMOD Int'l Conf. on Management of Data*, pp.393-402 (1992).

[8] Kim,W., Nicolas,J.-M., and Nishio,S. (Eds.): Deductive and Object-Oriented Database, *North-Holland* (1991).

[9] Lou,Y. and Ozsoyoglu,Z.M. : LLO: An Object-Oriented Deductive Language with Methods and Method Interitance, *Proc. of the ACM-SIGMOD Int'l Conf. on Management of Data*, pp.198-207 (1991).

[10] Maier,D. : A Logic for Objects, *Proc. of Workshop on Foundation of Deductive Database and Logic Programming,* Washington D.C., pp.6-26 (1986).

[11] Tsukamoto, M., Nishio, S., and Fujio, M. : DOT : A Term Representation using DOT Algebra for Knowledge-bases, *Proc. of the 2nd Int'l Conf. on Deductive and Object-Oriented Databases,* pp.391-410 (1991).

[12] Tsukamoto, M., Nishio, S., Fujio, M., and Miyamoto, M. : Query Processing for a Knowledge-base using DOT Algebra, *Proc. of the IEEE 1st Int'l Workshop on Interoperability in Multidatabase Systems,* pp.46-53 (1991).

[13] 塚本昌彦, 西尾章治郎, 長谷川利治 : DOT記法とIS-A関係にもとづく項表現を用いた演繹・オブジェクト指向データベースモデル, コンピュータソフトウェア, *Vol.9, No.1,* pp.10-26 (1992).

[14] 塚本昌彦, 西尾章治郎 : 知識ベースシステムDOTにおける知識の従属性と更新処理, 第9回ソフトウェア科学会大会論文集, pp.197-200 (1992).

[15] Yokota,K. and Nishio,S. : Towards Integration of Deductive Databases and Object-Oriented Databases - A Limited Survey, *Proc. of Advanced Database System Symbosium,* pp.253-261 (1989).

[16] Yokota,K. and Yasukawa,H. : Towards an Integrated Knowledge-Base Management System, *Proc. of the Int'l Conf. on Fifth Generation Computer Systems,* pp.89-112, (1992).

[17] 横田一正 : 演繹オブジェクト指向データベースについて, コンピュータソフトウェア, *Vol.9, No.4,* pp.3-18 (1992).

[18] Ullman,J.D. : Principles of Database and Knowledge-Base System, *Vol.1, Computer Science Press* (1989).