記述論理によるUMLの表現

塩谷, 勇 / 三浦, 孝夫 / SHIOYA, Isamu / MIURA, Takao / 中西, 啓之 / NAKANISHI, Hiroyuki

(出版者 / Publisher)
一般社団法人情報処理学会
(雑誌名 / Journal or Publication Title)
情報処理学会研究報告. データベースシステム (DBS)
(号 / Number)
71
(開始ページ / Start Page)
81
(終了ページ / End Page)
87
(発行年 / Year)
2003-07-16

社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2003－DBS－131 (11)
2003／7／16

記述論理によるUML の表現

中西啓之† 三浦孝夫† 塩谷 勇††

UML (Unified Modeling Language) は, 情報システムの設計開発に関する言語であり, 事実上の業界標準的な手法として知られる. しかし, 定義の曖昧さが残るため, モデル変換, 等価性判定, 冗長性の検証, 無矛盾性の検査などの設計開発過程で要する知的な操作を行うことが極めて難しい. 記述論理による UML 記述は, クラス図のモデル化と推論に試みられており, 従来直観的に扱われていた表現手法に形式的な枠組みを適用できる. 本研究では, 協調図上のオブジェクトの振舞いに対し記述論理による表現を提案する. 記述論理式で表現されたクラス図スキーマとの整合性や冗長性を推論することにより, システム仕様機能を検証できる.

Reasoning in Collaboration Diagrams by Description Logics

HIROYUKI NAKANISHI,† TAKAO MIURA† and ISAMU SHIOYA††

UML (Unified Modeling Language) is a *de fact* standard language for information system design and development. However, because of the ambiguity, we can't utilize intelligent operation like model transformation, examination of equivalence and redundancy as well as consistency. By using Description Logics, we can formalize UML especially for validating model consistency and reasoning that have been made by human-being.

In this investigation, we put our focus on behavior over collaboration diagrams and propose how to describe and reason them. By this approach we can co-evaluate collaboration diagrams and class diagrams.

1. 前 書 き

UML (Unified Modeling Language) は, 情報システムの設計開発に関する言語であり, 事実上の業界標準的な手法として知られる. UML では, 情報システム全容の目的・機能を記述するユースケース, 必要となるオブジェクト情報の構造を表現するクラス図, オブジェクトの振舞いに関する役割を表現する対話図, オブジェクトの状態変化・振舞いを表す状態図やアクティビティ図などから構成される [10),16)]. このうち, 対話図 (協調図および順序図) は, 機能の振舞いを表現しており, ユースケースとともにシステム仕様の機能を示す. またクラス・オブジェクトの役割や実現の詳細を記述しており, 分析過程を通してソースコードやシステムテストの方法を検討できる. UML は記述の方法を定義する言語である. その意味は各種図, OCL (Object Constraint Language) および細部を規定する自然言語で表現されるため, 図の相互の関連性については曖昧さが残る [11)]. この結果, モデル変換, 等価性判定, 冗長性の検証, 無矛盾性の検査などの設計開発過程で要する知的な操作を行うことが極めて難しい.

記述論理 (Description Logics) は, 健全かつ完全でしかも決定可能な推論機構を備えているため, 知識表現・データベース分野で近年注目を浴びている. 記述論理は構造化された情報を扱い, 変数や関数のない, 次数に上限を設けた第1階述語論理の部分クラスである. 論理プログラムと異なり, 選言 (∨) や否定 (¬) を自然に扱うことができる. データベースの立場からは, 設計, 管理, 情報検索, 情報統合などで推論機構を導入し, 同値性, 無矛盾性, 冗長性, 充足可能性の検査・検証ができる. 高度なモデル化機能により, 実体関連データモデルやオブジェクト指向データモデル, あるいは UML クラス図では基本概念が対応しており, スキーマや一貫性制約の記述などに応用できる [5),8),9)]. また様相論理機構を導入できるため, 時制モデルへの応用が提案されている [2)]. 特に, データベーススキーマは $\mathcal{ALCQI}$ と呼ぶ記述論理の部分クラスで表すこ

† 法政大学 工学研究科 電気工学専攻 〒184-8584 東京都小金井市梶野町 3-7-2 (Dept.of Elect.& Elect. Engr., HOSEI University 3-7-2, KajinoCho, Koganei, Tokyo, 184-8584 Japan)
†† 産能大学 経営情報学部 〒259-1197 神奈川県伊勢原市上粕屋 1573 (Department of Management and Information Science, SANNO University 1573, Kamikasuya, Isehara city, Kanagawa 259-1197 Japan)

とができ, スキーマ・クラスの充足可能性・等価性・冗長性などの設計上重要な問題に対して, 決定可能な推論機構を利用できる. また, 濃度制約の充足性判定問題は, 同等の記述論理式を整数線形計画法 (線形プログラミング) に帰着させ, 多項式時間で決定可能である.

記述論理による UML 記述は, クラス図のモデル化と推論に試みられており, 従来直観的に扱われていた表現手法に形式的な枠組みを与える [3),4),6)]. 反面, 論理系はスキーマやオブジェクト表現には効果的であるが, 状態の変化を捉えることが容易ではない. このため, 状態図の形式化やオブジェクト協調などの振舞いに対して適用することは容易でない [14)].

本稿では, 協調図上のオブジェクトの振舞いに対し記述論理による表現を提案する. 協調動作 (operation) はいくつかのメッセージ指令から構成され, 各指令を前提条件や完了条件 P_1, P_2 および発火条件 F で表す. 記述論理式で表現されたスキーマと指令条件の整合性や冗長性を推論することにより, システム仕様機能をデータベースとの関連で検証できる.

次章では UML 協調図の果たす役割を述べ、3 章では記述論理の特徴を要約する. 4,5 章では $\mathcal{ALCQI}$ によるクラス図の記述手法を示し, これを用いて協調動作を記述論理で表現する手法を示す.

2. UML と協調図

UML はソフトウエアシステムの仕様の策定, 構成, 視覚化, 文書化などを支援するための言語体系であり, 多種類の図 (diagram) から構成される. とくに協調図 (Collaboration Diagram) はクラス・オブジェクトの果たす役割の相互関連性を表現している.

UML が想定している協調 (Collaboration) では, (OMG 1.3 によれば,) オブジェクトを想定したスロット, オブジェクト間の関連を表すリンク, および協調を記述するスロット (role という) がある [10)]. クラス図はクラス・オブジェクトの記述を目的としており静的である. 協調図における振舞いは, クラス・オブジェクトが果たすべき役割 (role) を表し動的な特性を示す. 文献 [11)] では, ClassifierRole によりオブジェクトが果たす役割を, AssociationRole により関連が他の役割と共同して果たす役割を示すとしている. リンク (Link) は関連 (Association) のインスタンスであり, クラス・オブジェクト同士がどのように結びつくかを表現する. 対話 (Interaction) とは振舞いの記述であり通信の半順序的列を言う. メッセージ (message) は 通信内容であり, 通常 sender/receiver オブジェクト, 指令 (変換, 質問, パラメタなど) が含れる. メソッドとはこれらの実現である.

協調図には個別の振舞いを示すインスタンスレベルでの記述と, クラス図との整合性検証を目的とする仕様レベルでの記述がある. 特に前者は, オブジェクトとそれらの関連が作業を完了するためにどのように連絡しあうかを表す. オブジェクトとその詳細,(オブジェクトに対する) 予め決められた操作, 他のクラスから呼び出された操作, 動作特性 (メッセージ順序) などが代表的である. 複数のオブジェクトの対話はクラスとそれらの関連 (汎化, 集約を含む) 上で, オブジェクトインスタンスが結びつきあうことで遂行され, クラス図上にリンクがない場合は, 協調動作は生じない.

協調図の分析は, 従来から研究されているデータフロー手法のそれと類似しているため, ソフトウエア各部の動作を検証するための精密な静的・動的条件を得ることができる. 検証のため次の 12 種類の協調動作リンクが考察されている [1)].

A1. 協調ペア (クラス図で ClassifierRole や AssociationRole が定義されている)
A2. 変数定義 (局所変数に値を設定する)
A3. 変数参照 (局所変数を参照する)
A4. オブジェクト定義 (対応するオブジェクトの特定) A5. オブジェクト生成
A6. オブジェクト参照 (メソッド呼び出し)
A7. オブジェクト破壊
B1. 変数定義 (局所変数を定義・使用)
B2. オブジェクト定義 (オブジェクトを定義・使用)
B3. オブジェクト生成使用 (オブジェクトを生成・使用する)
B4. オブジェクト生成・破壊 (オブジェクトを生成・破壊する)
C1. メッセージ順序 (順序を表現)

A1,C1 は協調図リンクと構成が正しい定義であることを, A2,A3 はメソッドの局所変数に対する動作内容を,A4 は他のクラスのオブジェクト参照を示す. A5,A7 はデータベースの状態変化を生ぜしめる内容であり,A6 はメソッド呼び出しである. また B1,B2,B3,B4 は一時オブジェクト・局所変数に対するものであり, 新規オブジェクトやメソッドの解析に用いる.

[例題 1] ここで扱うのは自動販売機の例である。アクタである利用者はパネルで商品の在庫を確認したうえで (クラス ControlPanel が, クラス Goods の属性値を参照する), パネルを操作して商品の選択やお金の投入を行う。また, レジスタでつり銭の準備, 取り出し機で商品を提供する.

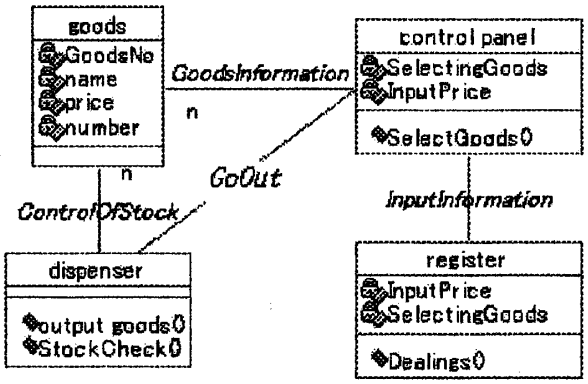

図 1 「自動販売機」クラス図

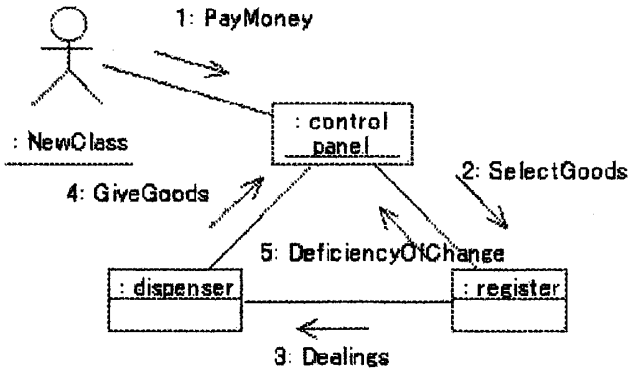

図 3 協調図「つり銭を考慮する自動販売機」

クラス図において，ControlPanel と Goods の間には，商品情報に関する関連が成り立つ．Goods と Dispenser の間では，在庫に関する情報をやり取りする．ControlPanel と Register の間には、パネルで入力された値をレジスタで参照するという関連が定義されている．

このクラスに対して，次の 2 種類の協調図を考えることができる．

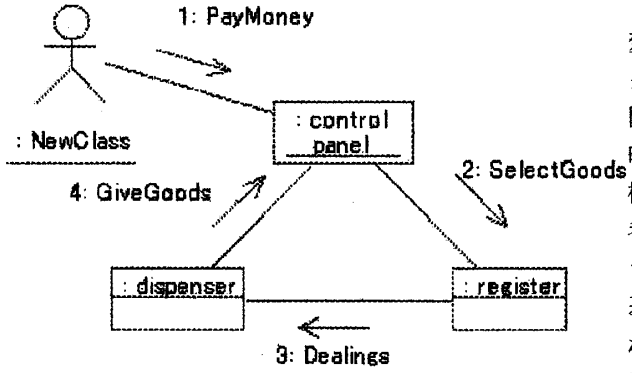

図 2 協調図「自動販売機の使用」

図 2 では利用者が自動販売機にお金を投入し，最終的に商品を手に入れるまでのシステムの振舞いを表している．なお，*Dealings* は Register, ControlPanel, Dipenser 上で定義されたビュー上のメソッドであるとする．

これに対して，図 3 では Change(つり銭) を考慮したときのシステムの振舞いを表す．この場合，Register クラスに属性「つり銭」を追加し，システムの動作中に変更される可能性を考慮せねばならない．

□

本稿では,(クラス図を前提にして) クラス・オブジェクト, 関連とこれらの属性, クラスメソッドとその局所変数に対する分析を行う．動的な検証により，オブジェクト動作, 対話や実行結果の信頼性を高め誤りの原因を突き止めることが容易になる．動作が生じる条件や、各リンク上で前提・完了条件を分析し, メッセージ順序を監視することにより, これらの条件を記述することができる．

3. 記述論理 (Description Logics)

記述論理は構造化された情報を扱う論理系であり，変数や関数のない次数に上限を設けた述語論理の部分クラスである．第 1 階述語論理と違って, 充足性判定問題が決定可能であり, 主要な部分クラスでは多項式時間で処理できるという特徴を有する．記述論理では概念 (Concepts) と役割 (Role) から構成される．前者はオブジェクトクラスを意味しており，後者はオブジェクトインスタンスの属性 (2 項関連) を意味する．基本概念 (primitive concept) によって記号が与えられ，$\sqcap, \sqcup$ などの構成子を用いて式 (expression) が定義される．限量作用子 $\forall, \exists$ は役割を介して定義される．基本概念 C, C' 上の役割 R に対して，$\forall R.C$ とは C のオブジェクト x の任意の R 属性値 y に対して $y \in C'$ となることをあらわす．例えば，*Person*(人間), *Doctor*(医者) 概念と，$CHILD$(子供) 役割に対して，$Person \sqcap \forall CHILD.Doctor$ により，その子供すべてが医者である人物を表現している．$\exists R$ により何かの R 属性が存在することを表す．$C \preceq D$ により包摂関係を表す，すなわちすべての C オブジェクトは D オブジェクトでもある．例えば *Parent*(親) 概念とは *Person* で $CHILD$ 属性を有するオブジェクトであり，$Parent \preceq Person \sqcap \exists CHILD$ と表すことができる．

$\mathcal{L}$ スキーマとは，$\mathcal{C} \preceq \mathcal{E}$ または $\mathcal{C} \doteq \mathcal{E}$ の形の

式の集合を言う．前者は基本概念の仕様とよばれ，$\mathcal{E}$ で指定された式を定義域とすることを意味する．後者は定義概念と呼ばれ，$\mathcal{C}$ 概念のオブジェクトであるための必要十分条件を表している．すべてのスキーマ内の式を充足する解釈をスキーマのモデルという．式 $\mathcal{E}$ がスキーマと無矛盾とは，スキーマと $\mathcal{E}$ が共に空でないモデルを有するときを言う． このような記述論理クラス $\mathcal{ALC}$ では包摂関係を高速に推論することができる．実際多項式時間で決定可能 (decidable) であることが知られる [9]．

記述論理には構成子の種類に応じて多数のクラスが存在する．下図にこれを示す．

構成子名		構文	解釈
概念名		C	$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
トップ		$\top$	$\Delta^{\mathcal{I}}$
原始否定		$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
連言		$E_1 \sqcap E_2$	$E_1^{\mathcal{I}} \cap E_2^{\mathcal{I}}$
全称限量		$\forall R.E$	$\{o \mid \forall o' : (o,o') \in R^{\mathcal{I}} \rightarrow o' \in E^{\mathcal{I}}\}$
非限定存在限量		$\exists R$	$\{o \mid \exists o' : (o,o') \in R^{\mathcal{I}}\}$
存在限量	$\mathcal{E}$	$\exists R.E$	$\{o \mid \exists o' : (o,o') \in R^{\mathcal{I}} \wedge o' \in E^{\mathcal{I}}\}$
disjunction	$\mathcal{U}$	$E_1 \sqcup E_2$	$E_1^{\mathcal{I}} \cup E_2^{\mathcal{I}}$
一般否定	$\mathcal{C}$	$\neg E$	$\Delta^{\mathcal{I}} \setminus E^{\mathcal{I}}$
数値制約	$\mathcal{N}$	$\exists^{\geq m} R$	$\{o \mid \sharp\{o' \mid (o,o') \in R^{\mathcal{I}}\} \geq m\}$
		$\exists^{\leq n} R$	$\{o \mid \sharp\{o' \mid (o,o') \in R^{\mathcal{I}}\} \leq n\}$
数値限量制約	$\mathcal{Q}$	$\exists^{\geq m} R$	$\{o \mid \sharp\{o' \mid (o,o') \in R^{\mathcal{I}} \wedge o' \in E^{\mathcal{I}}\} \geq m\}$
		$\exists^{\leq n} R$	$\{o \mid \sharp\{o' \mid (o,o') \in R^{\mathcal{I}} \wedge o' \in E^{\mathcal{I}}\} \leq n\}$
有限整合	$\mathcal{W}$	$wf(R)$	$\{o_0 \mid \forall o_0, o_1, \ldots (\text{無限列}) \exists i (o_i, o_{i+1}) \notin R^{\mathcal{I}}\}$
ルール結準像	$\mathcal{V}$	$(R_1 \subseteq R_2)$	$\{o \mid \{o' \mid (o,o') \in R_1^{\mathcal{I}}\} \subseteq \{o' \mid (o,o') \in R_2^{\mathcal{I}}\}\}$
ルール名		P	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
逆	$\mathcal{I}$	R^-	$\{(o,o') \mid (o',o) \in R^{\mathcal{I}}\}$
選言	$\mathcal{R}$	$R_1 \cup R_2$	$R_1^{\mathcal{I}} \cup R_2^{\mathcal{I}}$
結合	$\mathcal{R}$	$R_1 \circ R_2$	$R_1^{\mathcal{I}} \circ R_2^{\mathcal{I}}$
反射推移閉包	$\mathcal{R}$	R^*	$(R^{\mathcal{I}})^*$
同一	$\mathcal{R}$	$id(E)$	$\{(o,o) \mid o \in E^{\mathcal{I}}\}$
差	$\mathcal{D}$	$R_1 \setminus R_2$	$R_1^{\mathcal{I}} \setminus R_2^{\mathcal{I}}$

このうち特にデータモデリングでは $\mathcal{ALCQI}$ が適合する [8]．$\mathcal{ALCQI}$ では，概念は，基本記号 A または $\neg C$, $C \sqcap C'$, $C \sqcup C'$, $\forall R.C$, $\exists R.C$, $\exists^{\geq n} R.C$, $\exists^{\leq n} R.C$ の形式であり，役割は P, P^- のいずれかである (A, P は基本概念・基本役割，C, C', R は概念表現および役割表現)．また $C \Rightarrow C'$ を $\neg C \sqcap C'$ と表す．

例えば，多項述語を許したモデル (実体関連モデルや UML クラス図) $\mathcal{DLR}$ は $\mathcal{ALCQI}$ の部分クラスであり，オブジェクト指向モデルは $\mathcal{ALCQ}$ で表せる．これらに応じて決定性判定問題の計算量が知られている．

クラス	充足性判定
$\mathcal{AL}$	P
$\mathcal{ALE}, \mathcal{ALR}, \mathcal{ALER}, \mathcal{ALU}, \mathcal{ALUN}$	NP
$\mathcal{ALC}$	$PSPACE$
$\mathcal{ALCQI}$	$EXPTIME$

データベースで扱うデータは有限である．これに対し論理系ではこの制限は通常ない．充足可能ならば必ず有限モデルが存在するとき，その論理系は"有限モデル推論"であるという．すなわち，充足性は有限モデルに限ってよく，有限モデルがなければ本来充足不能であると断定できる．しかし $\mathcal{ALCQI}$ は有限モデル推論性を満たさない [7]．このため，以下では (1) スキーマでは基本概念だけが高々 1 度しか左辺に表れず，(2) 概念を再帰的に定義せず，しかも (3) 有限性条件 (well-founded) $\mathcal{W}$ を仮定した $\mathcal{ALCQIW}$ に限定して議論を行う．

[例題 2] 商品番号が 1 であるような Goods が存在するかどうかは，

$$Goods \sqcap \exists GoodsNo.\{1\}$$

によって表される．また，数量存在が扱えるので，在庫のある Goods を知るためには

$$Goods \sqcap \exists^{\geq 1} number$$

で，また在庫のない商品は

$$Goods \sqcap \forall number.\{0\}$$

によって表される．□

4. クラス図の記述論理による表現

記述論理による UML クラス図の形式化は，メタモデルやデータフロー等と異なり [13),15)]，決定可能な推論機構を利用することでクラス図の充足可能性・等価性・冗長性などの設計問題に対応することができる．この章では記述論理に基づく知識ベースによって UML が表現され，クラス図の設計・開発に有用であることを示す [3),4),8)]．

クラス C は概念に,C の属性 A (領域 C') については役割 R で $C \preceq \forall R.C', C' \preceq \forall R^-.C$ と解釈する．以下ではこれを $R \preceq C \times C'$ と表す．C が属性 $A_1, .., A_n$ を持ち，各属性 A_i に対応する役割 R_i は $R_i \preceq C \times C_i$ を満足するとき $C \preceq \forall R_1.C_1 \sqcap ... \sqcap \forall R_n.C_n$ なるスキーマ条件が対応する．クラス図では，実体関連モデルと同様に多項関連を扱う (n 項関連を導入し $\mathcal{DLR}$ クラスが提案されている) ため，2 項関連のみを扱う記述論理ではモデルの変換が必要となる．クラス $C_1, .., C_n$ 上の関連 R は R を概念に対応させ，R と C_i とのリンク $\mathbf{r}_i$ を役割と見て $R \preceq \forall \mathbf{r}_1.C_1 \sqcap ... \sqcap \forall \mathbf{r}_n.C_n$ および $C_i \preceq \forall \mathbf{r}_i^-.R$ と n 個の 2 項関連に分解する．濃度制約は (C と R 上の) $\mathbf{r}$ 上の数値限量と考え $(m..n)$ 対応を $C \preceq \exists^{\geq m} \mathbf{r}^-$ および $C \preceq \exists^{\leq n} \mathbf{r}^-$ と解釈する．$m = 0$ あるいは $n = \infty$ のときはこの制約を除外する．

クラス階層 (または汎化 generalization) C **ISA** C' あるいは包含制約 $C \subseteq C'$ は $C \preceq C'$ で表す．これは C, C' がブール表現されていても同様である。また排他制約 $C \| C'$ ($C \sqcap C' = \phi$ を意味する) は $C \preceq \neg C'$ あるいは $C \sqcap C' \preceq \perp$ と表される．

UML クラス図では集約 (Aggregation) によって，クラス C のオブジェクトが (他のクラス C' の) 複数の

オブジェクトから構成されることを表現できる．記述論理では 集約を表わす役割 Ag によって $Ag \preceq C \times C'$ と表す．C と C' の濃度制約 $(m..n)$ (どの $e \in C$ も m 個以上 n 個以下の C' オブジェクトで構成される) も先と同様に $C \preceq (\exists^{\geq m} A.C') \sqcap (\exists^{\leq n} A.C')$ と表せる．C' と C の濃度制約 (どの $e \in C'$ も m 個以上 n 個以下の C オブジェクトにしか参加しない) も逆役割 A^- を用いて同様に表わせる．

[例題 3] 例題 1 のクラス図は，記述論理スキーマでは次のように表現される．

$ControlPanel \preceq \forall GoodsInformation.Goods \sqcap$
$\forall InputInformation.Register$
$Goods \preceq \forall GoodsInformation.Dispenser$
$Dispencer \preceq \forall GoOut.ControlPanel$
$Register \preceq SelectingGoods.Goods \sqcap$
$InputPrice.INTEGER$
$Goods \preceq GoodsNo.INTEGER \sqcap name.STRING \sqcap$
$price.INTEGER \sqcap number.INTEGER$
$ControlPanel \preceq SelectingGoods.Goods \sqcap$
$InputPrice.INTEGER$

一方，このスキーマに課せられた制約条件のうちどのレジスタ動作もひとつの入力でのみ動作するため，濃度制約は次で表現される．

$Register \preceq \exists^{\leq 1} InputInformation$

□

5. 協調図の記述論理による表現

記述論理を用いた振舞いに関し，状態図を用いた研究[14]では $\mathcal{DLR}$ を用いて状態 (state) を概念に，事象 (event) と動作 (action) を概念と関連に対応させている．事象が実行できる条件 (guard) を記述論理式で与え，その充足性と状態の変化動作を推論することができる．しかし，この方法ではデータベース (クラス図) との関係をモデル化できず，協調図の表現には適用できない．本章では，クラス・オブジェクトの振舞いとメソッド内の局所変数あるいは一時オブジェクトの振舞いを直接扱う．

動作が実行できる前提条件 P_1，動作を発火させる発火条件 F，動作完了後に成り立つ完了条件 P_2 を考え，これを記述論理で表現する．この動作の妥当性は次のように表せる：

$$D \models \{(P_1 \sqcap F) \Rightarrow P_2\} \equiv \bot$$

P_1, F は，この動作が生じるための条件であり，例えばデータベース状態の変更，利用者の要求や動作確認，外部事象の発生などがある．常に成立するか考慮の必要がない条件は **TRUE** ($\top$) とする．ここでは P_1, F は D の内容に変更を加えない (解釈を変えない) とする．

データベースのスキーマ D が与えられたとき，P_1, F に対応して，記述論理の推論機構により判断すべき状況は次のものである．

(1) P_1 は D の下で充足するか：$D \not\models P_1 \equiv \bot$
記述論理は決定可能であるから $D \models P_1 \equiv \bot$ でもよい．
(2) 同様に，F は D の下で充足するか：$D \not\models F \equiv \bot$
(3) D, P_1 が充足するとき，F は充足可能か：$D, P_1 \not\models F \equiv \bot$
(4) D, P_1, F は同時に充足可能か：$D, P_1, F \not\models \bot$

言い換えると P_1, F が協調図上で何らかの誤りとみなされるべき状態は次のものである．

(1) $D \models P_1$ ： 前提条件が常に成り立つ
(2) $D \models P_1 \equiv \bot$ ： 前提条件は決して成り立たない
(3) $D \models F \equiv \bot$ ： D からは決して発火しない
(4) $D, P_1 \models F$ ： 前提条件が成り立てば必ず発火する
(5) $D, P_1 \models F \equiv \bot$ ： 前提条件が成り立っても決して発火しない
(6) $D, F \models P_1 \equiv \bot$ ： 発火したが前提が決して成り立たない
(7) $D \models P_1 \sqcap F$ ： 常に発火する
(8) $D \models P_1 \sqcap F \equiv \bot$ ： 前提条件と発火条件は決して同時に成り立たない
(9) $D, P_1, F \models \bot$ ： 同上

完了条件 P_2 に関して，記述論理の推論機構により判断すべき状況は次のものである．

(1) $D \models (P_1 \sqcap F) \Rightarrow P_2$ ： 前提条件，発火条件が成立すれば完了条件が常に成り立つ
(2) $D \models ((P_1 \sqcap F) \Rightarrow P_2) \equiv \bot$ ： 前提条件，発火条件が成立しても決して完了条件が成り立たない
(3) $D \not\models ((P_1 \sqcap F) \Rightarrow P_2) \equiv \bot$ ： 発火後に P_2 は充足可能

このうち (1),(2) は協調図上で何らかの誤りとみなされる．後述するように，(3) の状況はこの対話で実際に使用するオブジェクトが存在しうること (データベースの完了条件の充足性) を保障するだけであり，個々のオブジェクトがどのように変化するかを示すものではない．

2 章で示したように，協調図では 12 種類の (C から C' への) リンクが存在しうる．クラス・オブジェクトのメソッド $m : C \times C'$ を属性の動的計算と考え，m を属性値，その戻り値 C' を領域とみなす．記述論理ではこれらに対応して完了条件 P_2 を次のように表現することができる．

A1 協調ペア (P_2 として $\top$)
クラスから属性への関連は役割 $R \preceq C \times C'$ で，クラス図関連 A へのリンク r をたどる場合は $r \times A \times C$ で定義され，定義の妥当性 (例えば，$D \models R \preceq C \times C'$ などの) 検証を対応させる．
A2 変数定義 ($C \sqcap \exists m.V$)
メソッド属性 $m :\preceq C \times V$ とする．
A3 変数参照 ($C \sqcap \exists m.C'$)

$m : C \rightarrow C'$ とする.

A4 オブジェクト定義 $(C \sqcap \exists m.C')$
戻り値がオブジェクトとなるメソッド $m : C \rightarrow C'$ とみなす.

A5 オブジェクト生成 $(C' \sqcap \exists g^{-}.C)$
クラス C' のオブジェクトを生成するメソッド $g : C \rightarrow C'$ と考える.

A6 オブジェクト参照 $(C \sqcap \exists m.C' \sqcap \Gamma)$
メソッド $m \quad : C \rightarrow C'$ による変化式 Γ に対して.

A7 オブジェクト破壊 $(\neg(C' \sqcap \exists d^{-}.C))$
オブジェクトの破壊メソッド $d : C \rightarrow C'$ と考えられる.

これらの完了条件 P_2 はデータベーススキーマ記述, P_1, F とともに推論することにより, 冗長性あるいは充足不能性を検査できる.

[例題 4] 図 2,3 の協調図について完了条件を抽出する.

協調図 2 の自動販売機では選ばれた品物の金額が投入された金額に一致すると考え, 利用者がパネル情報を生成すると考えることができる. これに対応して, メッセージ 1 ではオブジェクトを生成し, 投入金額と選択された品物を設定する.

P_1: TRUE
F: パラメタ投入金 > 0
P_2: $ControlPanel \sqcap \exists^{\geq 1} InputPrice \sqcap existsSelectingGoods$

メッセージ 2 では, この情報をもとに $SelectGoods*$ メソッドがレジスタオブジェクトを生成し, 支払い情報を指示する. ここでは投入金額と表品価格が一致すること, および在庫確認をメソッド内で検査済みとする.

P_1: 該当商品が存在 $Goods \sqcap \exists SelectingGoods*^{-}.ControlPanel$
F : TRUE
P_2 : $Register \sqcap \exists SelectGoods*$

メッセージ 3, 4 も同様に受け取り機への指示 (GoOut), パネルへの完了メッセージ表示 (GiveGoods) を実行する.

(メッセージ 3)
P_1 : $ControlPanel \sqcap \exists^{\geq 1} InputPrice \sqcap \exists SelectingGoods \sqcap \exists SelectGoods*.(Register \sqcap \exists SelectGoods*)$
F : TRUE
P_2 : $Dispenser \sqcap \exists Dealings$
(メッセージ 4)
P_1, F : TRUE
P_2: $Dispencer \sqcap \exists GoOut^{-}.ControlPanel$

協調図 3 のうち、つり銭処理だけが異なるため, メッセージ 5 では投入金額と表品価格が一致しない処理を扱う. これは, ControlPanel での投入金額 InputPrice と支払いでの投入金額 InputPrice が一致しないことに対応し, この差を DeficiencyOfChange メソッドで扱う.

P_1: $Register \sqcap \exists InputPrice.\{x\} \sqcap \exists SelectGoods*^{-}.(ControlPanel \sqcap \exists InputPrice.\{y\} \sqcap (x \neq y))$
F : TRUE
P_2 : $ControlPanel \sqcap \exists DeficiencyOfChange^{-}.Register$

□

この結果は実行の結果 (振舞いの結果) が整合していることを保障するものでなく, 充足可能性の検査にすぎないことに注意したい. スキーマ D の解釈であるデータベースが, 具体的にどのような状態に遷移するかについて協調図では何も主張しない. 記述論理の充足性は, 何かのモデルの存在を保証するだけであり, そのモデルの計算方法を示すものではない.

実際, A5,A6,A7 はメソッド内で他のオブジェクトの状態を変更するため, 実行の結果は D の解釈を変更するという意味で動的である. 上記の範囲で局所的に充足していても, 新たな解釈は後続する動作で D のモデルではなくなる可能性があり, この動作単独で判断することは合理性がない. これまでデータベース分野では, 解釈に不整合が生じた時点でこれを解消すべく他の部分への変更動作を伝播させるか, あるいはトランザクション理論に従って整合性検査をある時点まで延期する手法がとられている. 後者の場合, メッセージ順序を表わすリンクはトランザクション制御にかかわる情報を提供する.

局所変数の操作あるいは一時オブジェクトの操作はプログラム的には興味深いがデータベースの整合性の観点を有さないため, 本研究では考察しない..

これらの考察は, 複数の動作に対して適応できる. 実際, P_1, P_2, F と P_1', P_2', F' から $P_1 \sqcap P_1', P_2 \sqcap P_2', F \sqcap F'$ を構成し, 同様の考察を加えればよい.

6. 結　　論

UML 記述のうちクラス図と協調図に記述論理を用いて形式的な枠組みを与えた. 本研究による表現を用いれば, データベース (クラス図スキーマ) との整合性や冗長性を推論することにより, 協調図の整合性を検証することができる. オブジェクト状態の変更を伴う場合であっても, 協調図とクラス図から充足可能性の検証ができる. しかし, 協調図だけでは, 充足性 (あるいは充足不能性) を判定できても, 実際にどのようなモデルに変更すればよいかという分析はできない. このためには, 状態図やアクティビティ図などの, 個々のオブジェクトの動作を追跡する推論機構が必要となる.

参 考 文 献

1) Abdrurazik, A. and Offutt, J.: Using UML Collaboration Digrams for Static Checking and

Test Generation, IEEE UML, 2000, pp.383-395
2) Artale, A., Franconi, E. et al: Description Logics for Modelling Dynamic Information, in *Logics for Merging Application of Databases*, Springer, 2003
3) Bernardi, D., Calvanese, D. et al.: Reasoning on UML Class Diagrams using Description Logic Based Systems, *KI2001* Intn'l Workshop on Application of Description Loics, 2001
4) Bernardi, D., Calvanese, D. etal: Reasoning on UML Class Diagrams, Technical Report 11-03, Dipartimento di Informatica e Sistemistica, Universita di Roma "La Sapienza", 2003
5) Borgida,A : Description Logics in Data Management, IEEE TKDE 7-5, 1995, pp.671-682
6) Cali, A., Calvanese, D. et al: A Formal Framework for Reasoning on UML Class Diagrams, Intn'l Symp. on Methodologies for Intelligent Systems (ISMIS02), 2002, pp.503-513
7) Calvanese, D.: Finite Model Reasoning in Description Logics, *KR96*, 1996
8) Calvanese, D., Lenzerini, M. et al: Description Logics for Conceptual Modelling, in *Logics for Databases and Information Systems*, Kluwer, 1998
9) Donini, F.: Reasoning in Description Logics, in *Principle of Knowledge Representation*, CSLI Publication, 1996
10) Object Management Group: OMG UML Specification 1.3, 1999, `www.omg.org.uml/`.
11) Overgaard, G.: A Formal approach to Collaborations in the Unified Modeling Language, IEEE UML, 1999, pp.99-115
12) Roger , M. Simonet, A. et al.: Bringing Together Description Logics and Databases in an Object Oriented Model, *DEXA*, 2002
13) Shiroiwa, M., Miura, T. et al: Meta Model Approach for Mediation, IEEE proc.*COMPSAC*, 2003, to appear
14) Van Der Straeten. R.: Using Description Logic in Object-Oriented Software Development, *DL02*, 2002
15) Wieringa, R.: A Survey of Structured and Object Oriented Software Specification Methods and Techniques, ACM Comp.Surv. 30-4 (1998), pp.459-527
16) ファウラー, M., スコット, K.: UML モデリングのエッセンス (第 2 版), 翔泳社, 2000