DEWS2004 6-C-05

# 記述論理による協調図の形式化

中西　啓之†　　三浦　孝夫†

† 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
E-mail: †{i03r3229,miurat}@k.hosei.ac.jp

**あらまし**　UML(統一モデリング言語) は, オブジェクト指向分析や設計で広く利用されているが, 整合性を検査するための方法が確立されていない．他方，UML 協調図では属性値や状態の変化ではなく, オブジェクト間の関係と通信を宣言的に表現しているため分析には適している．本研究では協調図を記述論理を用いて形式化し, 記述論理の推論機構を用いて推論を行う.

**キーワード**　UML, 記述論理

# Formalizing UML Collaborations by using Description Logics

Hiroyuki NAKANISHI† and Takao MIURA†

† Dept.of Elect.& Elect. Engr., HOSEI University　3-7-2, KajinoCho, Koganei, Tokyo, 184–8584 Japan
E-mail: †{i03r3229,miurat}@k.hosei.ac.jp

**Abstract**　Although UML(Unified Modeling Language) is widely used by Object-Oriented Analysis (OOA) /Design (OOD), the method for checking consistency is not established yet. On the other hand, in UML Collaboration diagram, relation and communication among objects are expressed declaratively without any state transitions nor attribute value changes. In this research, we describe how to formalize collaboration diagrams in a framework of Description Logics so that we can reason consistency, validity or redundancy among them. Here we propose our formal framework and show hot to establish reasoning by some inference tools.

**Key words**　UML, Description Logics

## 1. 前　書　き

対象問題について深い理解を得るために, その対象をある目的または観点から眺め, 本質的な部分を捉えるためにモデリング言語と呼ばれるものが用いられる．中でも,UML は, オブジェクト指向分析や設計等の分野で広く利用されているが, 整合性を検査するための方法が確立されていない．ただし，UML の中でも協調図においては, オブジェクト間の関係と通信を宣言的に表現しているため, 手続き的な順序を考慮する必要がなく, 論理との相性がよい．そこで, 論理を用いた UML 協調図の形式化を考えることにより, 協調図の整合性を検査する方法の確立を目指す.

関連する研究として, 仕様記述言語 Object-z を用いた協調図の形式化が行われ [1], UML の矛盾やあいまい性を指摘することに成功しているが, その分析には人の手が必要であり, 矛盾やあいまい性を自動的に検出する仕組みはない.

記述論理は一階述語論理のサブクラスであり, 協調図を論理で捉えることができる．また, その推論エンジンは, 計算機により自動的に包摂関係を推論することができる．本研究では, これらを用いることにより, 協調図の形式化の手法, 協調図の整合性の自動的な検査方法を示す.

2 章では UML とそのメタモデルについて述べ,3 章では記述論理とその推論エンジンについて述べる．4 章では UML 協調図の形式化について述べ,5 章で協調図の例を用いて推論を行い, その結果について考察し, 6 章で結びとする.

## 2. UML とメタモデル

### 2.1 UML の形式化

UML メタモデルは，UML を用いて UML 自身を記述するものである．表現するモデルの構造がどうなっているのか, モデルがどのような意味を持っていると解釈すべきかを示している．また,UML メタモデルは抽象的なモデルであり, 宣言的な意味論を中心としていてるので, この抽象的なメタモデルを用いた実装は, その意味論に従わなければならない.

UML メタモデルは, 抽象的なパッケージ (モデル要素のグループ化したもの) 群に体系化されていて, 静的構造を規定する Foundation(基盤), モデルの動的な振る舞いを規定する言語の上位構造である BehavioralElements(振る舞い要素), モデル要素がどのようにパッケージやサブシステムの中で組織されるかを規定する ModelManagement(モデル管理) 等の最上位パッ

ケージに分解される [2]．各パッケージは, 抽象構文, 適格性規則, 意味論といったもので構成されている．抽象構文は構成要素とそれらの関係を定義するメタクラスを示す図で構成され, 関係の多重度用件からなる適格性規則を示している．適格性規則は,UML モデルで満たすべき不変条件を OCL を用いて定義し, メタモデルで定義された属性と関連に関する制約を規則として規定する．意味論は, 原則として自然言語で記述され, 構成要素の意味を定義する．

Foundation パッケージの下位には, 基本メタモデルに必要な基本概念の仕様を記述する Core パッケージがある（図 2）．その更に下位には, 振る舞い要素群に必要な核となる概念を規定する CommonBehavior, 特定タスクを実行するための振る舞いの文脈を規定する Collaborations 等のパッケージが含まれる．

### 2.2 協調図とメタモデル

協調図は, 役割およびそのリンクをまとめた 1 つの相互作用を示し, 異なる役割を演じるオブジェクト間の関係を表す．協調図では, オブジェクト間の関連と共に, メッセージの流れも表現できる．協調図でメッセージを表現するには, 2 つのオブジェクトをつなぐ実線 (関連) のそばに, 先端をメッセージ受信側に向けた矢印を付加すればよい．図 1 に協調図の例を示す．例では, 学生が講義を受けたい教師を指名し (メッセージ 1), その担当講義を 1 つずつ登録していき (メッセージ 1.1), その結果, 教師はどの学生が自分の担当講義を履修するのかを知る (メッセージ 1.i.1)．図 1 には, クラス Person,Course が存在し, クラス Person は,3 つの役割を演じている．また, メッセージ 1.i.1 は, メッセージ 1.1 が 1 回送信されるごとに送信される．

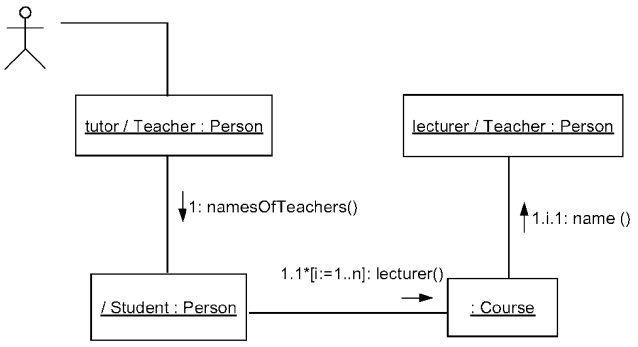

図 1　協調図の例

本研究では, 各 UML 図の基本概念を記述している Core パッケージ, UML 協調図の意味論を記述している Collaborations パッケージに焦点をあてている．以下に Core パッケージと Collaborations パッケージの抽象構文を示す.

Core パッケージは, オブジェクトモデルの開発に必要な基本的な抽象的構成要素および具象メタモデル構成要素を定義する．抽象的構成要素は, 重要な構成要素の具体化のために共通して使われ, インスタンス化できない．一方, 具象メタモデル構成要素は, オブジェクトモデル製作者が使用するモデル化構成要素を反映し, インスタンス化できる．Core パッケージに定義された抽象的構成要素には,ModelElement(モデル要素),GeneralizableElement(汎化可能要素),Classifier(分類子) があり, 具象メタモデル構成要素には,Class(クラス),Attribute(属

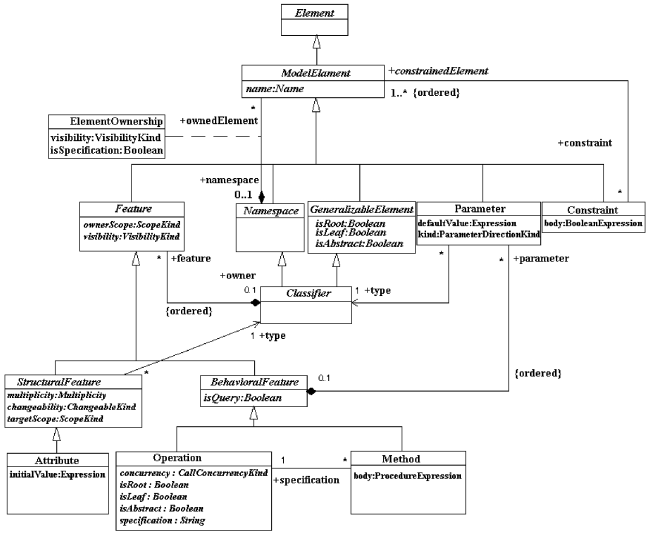

図 2　Core

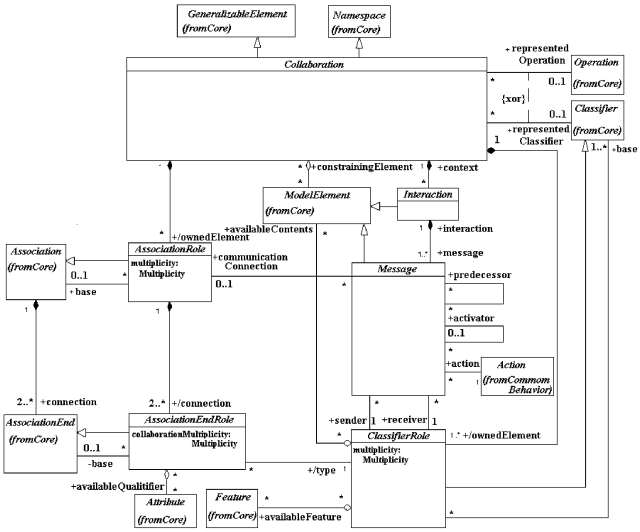

図 3　Collaborations

性),Operation(操作),Association(関連) がある．

Collaborations パッケージは, モデル内のモデル要素の使い方を規定する．UML で,Collaboration は,Operation(操作) や Classifier(分類子) がどのように Classifiers と Associations(関連) の集合によって実現されるのか,Collaboration に関与するオブジェクト間のやりとりはどう定義するのかを記述する．例えば,Collaborations パッケージでは,GeneralizableElement が Collaboration の汎化となっている．Collaboration は ClassifierRole に対して必ず ownedElement という関連をもつ．また,Collaboration は Operation に対して representedOperation,Classifier に対して representedClassifier の関連をもつ．各関連は多くとも一つしか存在できず, 二つのうちどちらか一つしか成り立たないことが示されている．ここで, 操作は振る舞いをもたらすためにオブジェクトから要求されるサービス, 分類子は振る舞いと構造上の特性を記述する要素, 関連は分類子間の意味的関係を定義する．

図 1 で表されたもの以外に, 協調図が満たすべき条件が多数

存在する．これらは適格性規則と呼ばれ, 同じ構成要素間の関係に関する条件や, 他のパッケージの要素との間に成り立つ条件を表す．それらの条件には以下のようなものがある. [1]

1. Collaboration は,Classifier か Operation のどちらかである.
2. 全ての AssociationRole は,Collaboration に含まれる ClassifierRole だけに関連している.
3. Collaboration において, 全ての ClassifierRoles と AssociationRoles は,Namespace(名前空間) にある Classifiers,Associations に関連されている.
4. モデル要素に含まれている要素のみに,Constrain(制約) を設けられる.
5. もし 2 つの ClassifierRoles あるいは AssociationRoles が協調内で名前を持っていないなら, それらは異なった基盤を持っている.
6. 協調の親と子で, 同じ名前を持っている役割 (AssociationRole あるいは ClassifierRole) は, その役割の特化であるに違いない
7. 協調 (Classifier を表すことについてのケースで) 内のすべての Interaction 図は representedClassifier に送られたメッセージから始まる
8. ある協調が他の協調を特化したものである場合, 親協調が持つ全ての ClassifierRoles を含まなくてはならない.
9. ある協調が他の協調を特化したものである場合, 少なくともその Intarection(相互作用) の間, 親に存在しているすべてのメッセージを含んでいなくてはならない.

また, 意味論とは図の構成要素の解釈方法を形式的に定義するものであり, 抽象構文内の Instance(インスタンス),Stimulus(刺激) に基づいて規定されている．インスタンスは操作の集合が適用され, 操作の結果を格納する状態を持つ実体を定義し, 刺激はオブジェクト間の関係を示す．以下に協調図の構成要素のもち得る意味を示す.（a）オブジェクト間に関係が存在するか,（b）他のオブジェクトに関与したか（c）オブジェクトの 1 つがパラメーターの通過によって他のオブジェクトを知っているかが存在する [1].

先の図 1 には,Classifier として Person,Course, ClassifierRole として Teacher,Student,AssociationEndRole として Tutor,lecturer, Message として namesOfTeachers（）,lecturer（）,name（）がある.

## 3. 記述論理 (Description Logics)

### 3.1 記述論理の表現方法

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 [4]〜[6]．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.

［例題 1］ 記述論理式の例を示す.

(1) 構成要素 AssociationRole から構成要素 Association へ, 多くても 1 つの役割 base しか関連をもたない:
$AssociationRole \preceq \exists^{\leqq 1} base.\ Association$

(2) Teacher,Student は, 構成要素 ClassifierRole のインスタンスである: $Teacher \sqcap Student \preceq ClassifierRole$

(3) nameOfTeachers() は, 構成要素 Message のインスタンスである: $nameOfTeachers() \preceq Message$

(4) メッセージ nameOfTeachers() の送信者は Teacher で, 受信者は Student である: $nameOfTeachers() \preceq sender.\ Teacher \sqcap receiver.\ Student$

□

上述のような記述論理を $\mathcal{AL}$ クラスと呼ぶ．$\mathcal{AL}$ 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 [4].

データモデリングでは $\mathcal{ALCQI}$ が適合する [5]．$\mathcal{ALCQI}$ では，概念は，基本記号 $A$ または $\neg C$, $C \sqcap C'$, $C \sqcup C'$, $\forall R.\ C$, $\exists R.\ C$, $\exists^{\geqq n} R.\ C$, $\exists^{\leqq n} R.\ C$ の形式であり，役割は $P, P^-$ のいずれかである．ここで,$A, P$ は基本概念・基本役割, $C, C', R$ は概念表現および役割表現を表す.

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．$\mathcal{ALCQI}$ は有限モデル推論性を満たさない [6]．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.

### 3.2 RACER

RACER システムは, 記述論理 $\mathcal{ALCQHI}_{R+}$のために最適化された計算を実行する推論エンジンであり, 記述論理式をユーザが入力することで, 計算機により包摂関係の推論が自動的にで

きる．また,RACER はフリーソフトとして公開されている [3]．ここで,$\mathcal{ALCQHI}_{R+}$は, 数値制約, 役割階層，逆の役割と他動詞の役割で拡張された基本的な論理 $\mathcal{ALC}$ である．しかし, 本研究では UML モデルが記述対象なので,$\mathcal{ALCQIW}$ の枠組みだけを使用する．RACER では, 記述論理式を表すために, 記号を項と呼ばれる文字に置き換える必要がある．概念 C が C' に包摂される場合,implies を用いて,(implies C C') と表現する．限量作用子 ∀,∃ は all,some で表し,(some R C) と表現する．他に, 一般否定 ¬は,not, 逆役割は inv で表す．また, 概念 C のインスタンス I は,(instance I C) と表現し, インスタンス I と I' が役割 R で関連している場合,(related I I' R) と表現する．

［例題 2］ 先の例を RACER で扱う式を用いて示す．

(1) 構成要素 AssociationRole から構成要素 Association へ, 多くても 1 つの役割 base しか関連をもたない:

(implies AssociationRole (at-most 1 base Association))

(2) Teacher,Student は, 協調図の構成要素 ClassifierRole のインスタンスである:

(instance Teacher ClassifierRole)

(instance Student ClassifierRole)

(3) nameOfTeachers() は, 協調図の構成要素 Message のインスタンスである: (instance nameOfTeachers() Message)

(4) メッセージ nameOfTeachers() の送信者は Teacher で, 受信者は Student である:

(related nameOfTeachers() Teacher sender)

(related nameOfTeachers() Student receiver)

□

RACER には, 記述論理の内容を調査するための質問機能が備わっている．概念 C が充足可能かどうかを知るためには,$(concept - satisfiable?C)$ と表現し, 概念 C が C' に包摂されているかどうかを知るためには,$(concept - subsumes?CC')$ と表現する．

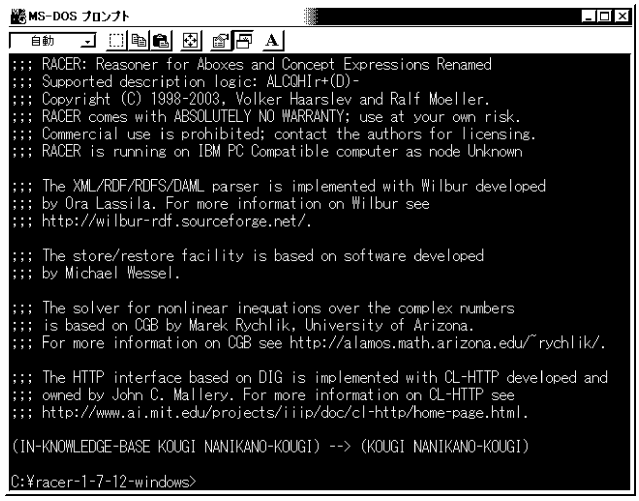


図 4　RACER

## 4. 協調図の形式化

本研究では, 協調図の構文と意味を定義しているメタクラスを記述論理に変換することで, 図の形式化を行う．協調図の定義を直接行っているのは Collaborations パッケージであり, 協調図のメタモデル Collaborations とその上位である Common-Behavior, UML 全体の基本メタモデル Core を記述論理で捉えなおす必要がある．また, パッケージを記述論理を用いて表現するためには, 協調図の満たすべき構文と意味論とを記述論理に変換する必要がある．

### 4.1 協調図の構文の形式化

#### 4.1.1 協調図の構造の形式化

ここでは,Collaborations パッケージの抽象構文の要素間にある関係を記述論理で捉えなおしていく．例えば, 汎化関係は, 子クラス（概念 Collaboration）が親クラス（概念 GeneralizableElement）を包摂するという関係で捉えなおす.

$$GeneralizableElement \preceq Collaboration$$

クラス間の関連は, 例えば, 概念 Collaboration から概念 ClassifierRole へ役割 ownedElement で関連すると捉えなおす．また, 関連 Operation に対して役割 representedOperation, 概念 Classifier に対して役割 representedClassifier で関連すると捉えなおす．要素間の多重度は限量子を用いて,$\exists^{\leqq 1}$ と捉えなおす.

$$Collaboration \preceq ownedElement.ClassifierRole \sqcap (\exists^{\leqq 1} representedOperation.\ Operation \sqcup \exists^{\leqq 1} representedClassifier.\ Classifier)$$

上述の考え方を用いて,Collaboration パッケージの残りの要素を記述論理で捉えなおしたものを以下に示す.

$$Namespace \preceq Collaboration$$
$$ModelElement \preceq Interaction \sqcup Message$$
$$Interaction \preceq context.Collaboration \sqcap message.\ Message$$
$$Message \preceq \exists^{\leqq 1} communicationConnection.AssociationRole \sqcap sender.ClassifierRole \sqcap receiver.ClassifierRole \sqcap action.\ Action \sqcap \exists^{\leqq 1} activator.\ Message$$
$$AssociationRole \preceq multiplicity.Multiplicity \sqcap \exists^{\leqq 1} base.\ Association \sqcap \exists^{\geqq 2} connection.\ AssociationEndRole$$
$$AssociationEndRole \preceq collaborationMultiplicity.\ Multiplicity \sqcap \exists^{\leqq 1} base.\ AssociationEnd \sqcap type.\ ClassifierRole$$
$$ClassifierRole \preceq multiplicity.\ Multiplicity \sqcap base.\ Classifier \sqcap availableContents.ModelElement \sqcap availableFeature.\ Feature$$
$$Operation \preceq Classifier$$

#### 4.1.2 記述論理による Collaboration の制約条件

ここでは, 適格性規則を記述論理に捉えなおす．以下に適格性規則の変換結果を示す.

1. $((representedClassifier.\ Classifier) \sqcap (representedOperation.\ Operation)) \doteq \bot$
2. $\exists type^{-}.\ (base^{-}.\ (connection^{-}.\ AssociationRole))) \preceq \forall base^{-}.\ ClassifierRole$
3. $(name^{-}.\ (base^{-}.\ AssociationRole) \preceq \forall name^{-}.\ Association) \sqcap (name^{-}.\ (base^{-}.\ ClassifierRole) \preceq \forall name^{-} Classifier)$
4. $\forall base^{-}.\ constrainingElements \preceq ModelElement$
5. $((\neg(name^{-}.\ AssociationRole) \sqcup \bot) \sqcap (name^{-}.\ AssociationRole \sqcup \neg\bot)) \sqcup \exists_{\leq 1} base^{-}.\ AssociationRole$
7. $((\neg(representedClassifier^{-}.\ Classifier) \sqcup \bot) \sqcap (\neg(\bot) \sqcup representedClassifier^{-}.\ Classifier)) \sqcup (\neg(base^{-}.\ (receiver^{-}.\ (message0^{-}.\ Interaction)) \sqcup representedClassifier^{-}.\ Collaboration) \sqcap (\neg(representedClassifier^{-}.\ Collaboration) \sqcup base^{-}.\ (receiver^{-}.\ (message0^{-}.\ Interaction)))$
9 $(message^{-}.\ (parent^{-}.\ (interaction^{-}.\ (parentInt^{-}.\ (generalization^{-}.\ GeneralizationElement))))) \preceq message^{-}.\ Interection$

適格性規則 1 は, 前節の記述論理式に存在する Classifier,Operation 双方から役割 representedClassifier, 役割 representedOperation で関連する Collaboration が, 存在しない (⊥) と捉えなおす. 適格性規則 3 の上二行は, 概念 AssociationRole から役割 base で関連する概念が存在し, 更にその概念から役割 name で関連する概念が, 概念 Association から役割 name で関連する概念に包摂されると捉えなおす. 適格性規則 7 は,representedClassifier への始めのメッセージを役割 message0 で捉えなおしている. このように, 基本的には先に述べたパッケージ内の要素間の関係を表わした記述論理式を用いて捉えなおす. 完全には記述論理に置き換えられない部分に関しては, 必要ならば新たに記述論理の役割を定義する. また, 各要素の名前は, 概念 String へ関連する役割 name として捉えなおしている. 適格性規則 6 は, 変数を用いた論理式で表わす必要があるが, 記述論理では変数を扱えず, 適格性規則 8 は RACER では表現できない. しかし, これらの条件が正しく成り立っているかどうかは, 後述する.

### 4.2 協調図の意味論の形式化

構文的な制約により Collaboration における役割を示すが, ここではインスタンスレベルで, 協調図における役割に従っているインスタンスとリンクを考える. オブジェクト間に存在する, 構文的制約がどういう意味に対応づけられるかを記述論理式で捉えなおし, 以下に示す. a,c では, 刺激の送信者（概念 sender）と受信者（概念 receiver）,b では, 役割 isCreateAction,isDestroyAction で関連している概念があると捉えなおす.

a オブジェクト間に関係が存在するか
$(sender^{-}.\ Stimulus \preceq instance^{-}.\ (connection^{-}.\ Link)) \sqcap (receiver^{-}.\ Stimulus \preceq instance^{-}.\ (connection^{-}.\ Link))$

b 他のオブジェクトに関与したか
$(isCreateAction^{-}.\ (action^{-}.\ Stimulus2) \sqcap sender^{-}.\ Stimulus2 = sender^{-}.\ Stimulus \sqcap instantiation^{-}.\ (action^{-}.\ Stimulus2) \preceq classifier^{-}.\ (receiver^{-}.\ Stimulus)) \sqcap (isDestroyAction^{-}.\ (action^{-}.\ Stimulus3) \sqcap receiver^{-}.\ Stimulus = receiver^{-}.\ Stimulus3)$

c オブジェクトの 1 つが, パラメーターの通過によって他のオブジェクトを知っているか
$(receiver^{-}.\ Stimulus2 = sender^{-}.\ Stimulus) \sqcap receiver^{-}.\ Stimulus \preceq argument^{-}.\ Stimulus2$

## 5. 記述論理による推論

### 5.1 推 論 手 順

ここでは, 前章までで記述論理により形式化された協調図の枠組みを用いて, 図 1 の協調図との間に矛盾が存在しないかを RACER で確かめる方法について述べる. 以下の作業が最後まで滞りなく進めば, 記述論理で捉えなおされた協調図の満たすべき抽象構文, 適格性規則と協調図の例との間に矛盾が存在しないことが確認される.

（1） 図の要素間の関係を Tbox として記述する
（2） 要素に対する適格性規則を Tbox として記述する
(記述論理で表しきれないものは,
質問機能によって無矛盾であることを確認する)
（3） 協調図の例を Abox として記述する
（4） RACER を起動
（5） 記述内容に矛盾があれば停止
（6） 2. で表現できなかったものを質問機能で確認
（7） エラーが表示されなければ矛盾がないことがわかる

### 5.2 推論に用いる例

ここでは, 図 1 の協調図を用いて推論を行う. 以下に, 図 1 に存在する概念や役割を記述論理で捉えなおしたものを示す.

$Person \sqcap Course \preceq Classifier$
$Teacher \sqcap Student \preceq ClassifierRole$
$/Teacher : Person \sqcap /Student : Person \sqcap : Course \preceq AssociationEnd$
$Tutor \sqcap Lecturer \preceq AssociationEndRole$
$nameOfTeachers() \sqcap lecturer() \sqcap name() \preceq Message$
$Student \sqcap Teacher \preceq base.\ Person$
$Tutor \sqcap Lecturer \preceq base./teacher : Person$
$nameOfTeachers() \preceq$

$$
\begin{aligned}
&\quad sender.\ Teacher \sqcap receiver.\ Student \\
&lecturer() \preceq \\
&\quad sender.\ Student \sqcap receiver.\ CourseCR \sqcap \\
&\quad predecesser.\ nameOfTeachers() \\
&name() \preceq \\
&\quad sender.\ CourseCR \sqcap receiver.\ Teacher \sqcap \\
&\quad predeseccer.\ (nameOfTeachers() \sqcap lecturer())
\end{aligned}
$$

上述の記述論理式は,3 章で述べた方法により RACER 内で使われる式に変換し, それを RACER に入力する.

### 5.3 協調図の推論

RACER 上で前述の例を記述論理の ABox として記述し, 推論を行わせる. RACER は, 利用者が記述した論理が矛盾ないものならば, 何も矛盾を知らせるメッセージが出ることなく動作を終了する. これをもって, 協調図の無矛盾性を確認できたと考える.

協調図の適格性規則の多くについては, 例を RACER 上で記述し推論させた結果, 矛盾は発見されない. 例えば, 適格性規則 1 については,Classifier と Operation との両方に関連を持つ個体は存在しないので, 例では適格性規則 1 を満たす. 適格性規則 2 についても,AssotiationRole が想定されていないので矛盾なく成立する. しかし, 適格性規則 7 については, 矛盾を知らせるメッセージが表示される. 原因としては, 適格性規則 7 では, 図内での最初のメッセージのあり方に関する条件を述べているが, 最初のメッセージが設置されていないことが挙げられる.

### 5.4 質問機能による協調図の無矛盾性確認

4 章で述べた通り, 記述論理式で表現し切れなかった Collaboration の適格性規則を RACER のもつ質問機能によって確認する手法について述べる.

適格性規則 6 については, 設計者が記述した記述論理内で, 同じ名前を持つ概念が存在するかを検索する. それは,$(concept - satisfiable?\quad \exists_{\geq 2} name^{-}.\ r)$ という質問を RACER にすることで確認できる. そのような概念が存在する場合には, 一方が他方を包摂しているかを質問で確認する. 概念 C1,C2 が存在する時,C1 が C2 を包摂するかを質問するためには,(concept-subsumes? C1 C2) と記述すればよい. その結果, 矛盾を知らせるメッセージが表示されれば, 適格性規則 6 に関する矛盾として判断する.

適格性規則 8 については, 継承関係が存在する協調がある場合に, 親側に存在する分類子役割を検索する. そして, 適格性規則 6 と同様に検索した結果に発見したものが, 子側に存在する分類子役割に包摂されているかを質問する. その結果, 矛盾を知らせるメッセージが表示されれば, 適格性規則 8 に関する矛盾として判断する.

発見された矛盾は, 適格性規則 7 に関するものである. この問題を解決するためには, 役割 representedClassifier で関連される概念を設定し, 協調図の最初のメッセージを設置すればよい ($message0 \preceq Message,\quad message0 \preceq representedClassifier.\ Person$). 以上の結果から, 実際に記述論理を用いて協調図の整合性を半自動的に検査できることがわかる.

## 6. 結　　び

本研究では, 推論機構が効率よく決定可能である記述論理を用いて,UML 協調図を形式化を行った. また, 実際に例を利用して, 形式化されたものについて RACER システムによる推論を行い, 協調図の矛盾を発見できることが確認できた.

## 謝　　辞

本研究の一部は文部科学省科学研究費補助金 (課題番号 14580392) の支援による.

## 文　　献

[1] Maria Agustina Cibran,Vanesa Mola,Claudia Pons,Wanda Marina Russo: "Rigorous description of the syntax and semantics of UML Collaborations", *ASSE2000*,Argentina
[2] Object Management Group: "OMG Unified Modeling Language Specification", 1999
[3] RACER,http://www.sts.tu-harburg.de/ r.f.moeller/racer/
[4] Donini, F. : Reasoning in Description Logics, in *Principle of Knowledge Representation*, CSLI Publication, 1996
[5] Calvanese, D. , Lenzerini, M. et al: Description Logics for Conceptual Modelling, in *Logics for Databases and Information Systems*, Kluwer, 1998
[6] Calvanese, D. : Finite Model Reasoning in Description Logics, *KR96*, 1996