

制約指向の概念モデルによるプログラム仕様の抽出

横田和久 岡本克巳 橋本正明 佐藤正和
ATR 通信システム研究所

筆者らは仕様の再利用によるソフトウェア自動作成について研究中である。この研究の考え方は、再利用の対象としてモデルシステム仕様を作成しておき、その仕様をカスタマイズして、開発対象の目的システム仕様を得る。次に、その目的システム仕様からプログラム仕様を抽出して、その仕様をプログラムへ変換する。この研究の一環として、ユーザがプログラムの入出力データを指定することによって、そのプログラム仕様を目的システム仕様から抽出するための方法を研究中である。この抽出方法を可能とするため、本稿では目的システム仕様から作成された有向グラフの上で、ERモデルにおける関連型の関連数に着目した有向道の選択方法を提案する。

Extraction of Program Specification by Constraint Oriented Conceptual Model

Kazuhiya Yokota Katsumi Okamoto Masaaki Hashimoto Masakazu Sato
ATR Communication Systems Research Laboratories

The authors have been studying an automatic software synthesis method using reusable software specifications. In the method, model system specifications are constructed for reusing, and target system specifications are obtained by customizing the model system specifications. Then, program specifications are extracted from the target system specifications, and the program specifications are transformed into programs. Therefore, the authors have been studying the program specification extraction from the target system specifications by specifying the program input and output data. This paper proposes a directed path selection method based on the relationship type cardinalities in ER model for the program specification extraction.

1 はじめに

プログラムの再利用技術は既存のプログラムを修正して何度も利用するので、開発費が少なく、信頼性もよいといった効果を上げてきた。最近の再利用技術の研究は、プログラム作成時に現れる知識を再利用する動向になっており、その動向に沿った研究の一ステップとして、筆者らはプログラム設計時に現れる知識の一つである仕様を再利用して、プログラムを作成するための仕様再利用技術を研究中である。

この研究の考え方では、適用領域ごとに再利用の対象となるモデルシステム仕様を作成しておく。次に、その仕様をカスタマイズして目的システム仕様を得て、この仕様から多くのプログラム仕様を抽出する。それから、プログラム仕様をプログラムへ変換する。この研究の課題には1)仕様記述言語、2)仕様プログラム変換、および3)仕様再利用支援とがあげられる。1)と2)については、仕様記述言語 PSDL(Program Specification Description Language)[1]と、PSDLからCプログラミング言語への変換[2]とを研究中である。3)に関してはモデルシステム仕様の構成[3]と、本稿のプログラム仕様抽出について研究中である。

PSDLで表される目的システム仕様は、そのシステムで処理される対象世界の構造的側面を、概念データモデルの一つであるERモデルで表し、動的側面を制約で表している。このように目的システム仕様は、システムを実現するためのプログラム群、すなわちモジュール構造が決まっていなくても、システムへの要求を表すことができる。このため、目的システムの実現にあたって、目的システム仕様からモジュール構造を導出して、プログラム仕様を抽出することが必要となる。その研究の一ステップとして、プログラムの入出力データに着目したプログラム仕様の抽出について研究中である。

本稿ではERモデルと制約からできる有向グラフの上で、入出力データを端点とする有向道群を選択して、その道群上に記述されている仕様をプログラム仕様として抽出する方法を提案したい。道の選択に関しては、データベースのアクセス性能に着目した研究[4]や、意味の近接性に着目した研究[5]などがあるが、本稿ではERモデルにおける関連型の関連数(Cardinality)に着目した道の選択方法を提案する。以下、本稿の第2章と第3章でPSDLと仕様再利用技術を概説し、第4章でプログラム仕様抽出について述べ、第5章に今後の課題をあげる。

2 PSDL

PSDLではプログラムの入出力データの性質に着目して、プログラム仕様を以下の3階層に分けて記述する。

- 1) プログラムの入出力データに表される対象世界の

情報について、その枠組を定めた情報層。

- 2) 帳票のような入出力データ形式を定めたデータ層。
- 3) 入出力ファイルのアクセス方法を定めたアクセス層。

本章では、図1のPSDLプログラム仕様を用いてPSDL文を概説する。このプログラムはファイルproductとsaleを入力して、売上毎に計算した売上額をファイルaccountへ出力する。

2.1 情報層

情報層は、図1では01行目のINFORMATION文と19行目のDATA文の間に記述する。

- (1) 実体型、属性、主キー、実体数

各々の実体型を図1の02行目のE(Entity type)文で記述する。それに続けて主キー属性を04行目のA(Attribute)文と05行目のK(Key)文で記述し、非主キー属性を06行目のA文で記述する。04行目のSTR(STRing)は属性値の定義域が文字列であることを示し、06行目のNUM(NUMber)は数値であることを示す。実体型に含まれる実体数は03行目のEN(Entity Number)文で記述する。03行目の“-50”は実体型productに含まれる実体が最大50個であることを示している。

- (2) 関連型、実体型の対応づけ、関連数

各々の関連型を07行目のR(Relationship type)文で記述する。それに続けて、関連型で対応づけられた実体型を08行目と10行目のC(Collection)文で指定する。この文には実体型とその役割をRole.EntityTypeの形で指定する。それらの実体型が相互に異なる場合は図1のように役割を省略してもよい。実体型を指定したC文の後には、その実体型の1つの実体につながる関連の個数を09行目のRN(Relationship Number)文で記述する。09行目の“M”(Many)は個数がゼロ以上であることを示し、11行目の“1”は1個であることを示している。なお、“-1”は1個ないしはゼロ個であることを示す。

- (3) 属性値従属性制約

この制約は非主キー属性の値を得るために用いる。図1では、値が得られる非主キー属性のA文に続けて、18行目の=(equal)文で制約を記述する。この制約から参照される属性はattributeまたはrole1.relationshipType.role2.entityType.attributeの形で記述する。ここで前者は、値の得られる実体を持っている他の属性を参照するのに用い、一方その実体へ関連で対応づけられた他の実体の属性を参照するのに後者を用いる。ところで、上記のrole1は値が得られる方の実体の役割を示し、role2は他方の実体の役割を示している。

- (4) 関連存在従属性制約

この制約は関連を得るのに用いる。この例は図1にないので以下に示す。たとえば、実体personが持っている

```

01 INFORMATION
02   E product
03     EN -50
04     A name STR
05     K
06     A price NUM
07   R sold
08     C .product
09       RM M
10     C .sale
11       RN 1
12   E sale
13     EN -100
14     A number NUM
15     K
16     A quantity NUM
17     A amount NUM
18     = .sold..product.price * quantity
19 DATA
20   I product_data
21     IX product_id
22     G product_record ON EndOfFile(product_data)
23     O product_record
24     %12s product_name
25     = product.name
26     %8d product_price
27     = product.price
28   I sale_data
29     IX sale_id
30     G sale_record ON EndOfFile(sale_data)
31     O sale_record
32     %4d sale_number
33     = sold..sale.number
34     %12s sale_product
35     = sold..product.name
36     %4d sale_quantity
37     = sale.quantity
38   I account_data
39     IX account_id
40     G account_record ON PEntityNumber(sale)
41     O account_record
42     %4d sale_number
43     = sale.number
44     %8d sale_amount
45     = sale.amount
46 ACCESS
47   D product INPUT 20 product_data
48   D sale INPUT 20 sale_data
49   D account OUTPUT 12 account_data

```

図 1: PSDL プログラム仕様

属性 skill の値が、実体 section に適した属性 skill の値に等しければ、その場合にのみ両実体が関連 belong.to で対応づけられるものとする。この関連存在条件は R 文と C 文に続けて、RC (Relationship existence Condition) 文で“RC (.person.skill == .section.skill)”と記述する。ここで、条件式から参照される属性は role.entityType.attribute の形で記述している。

(5) 実体存在従属性制約

この制約は実体を得るのに用いる。この例も図 1 にないので以下に示す。たとえば、実体 customer の属性 total が正値であれば、その場合にのみ、その実体に関連 demand で対応づけられる実体 account が存在するものとする。この時、得られた実体の主キー属性値を決めなければならないので、実体が得られる実体型の K 文に続けて、主キー属性値を決めるための計算式を属性値従属性制約と同じ = 文で“= .demand..customer.name ON (.demand..customer.total > 0)”と記述する。この例では account の主キー属性値は customer の主キー属性 name の値に等しいものとした。また、total の値が正か否かを判定するための条件式は ON 句で記述する。

2.2 データ層とアクセス層

(1) データ層

データ層は、図 1 の 19 行目の DATA 文と 46 行目の ACCESS 文との間に記述する。入出力データの構造は基本データ型や、連接集団データ型、繰返し集団データ型、選択集団データ型で階層的に木構造で定める。これらのデータ型は各々 % 文や、O (sequence Order) 文、I (Iteration) 文、S (Selection) 文で記述する。

% 文は基本データ型のデータ形式を C 言語と同様

に定める。たとえば、%12s は 12 文字の文字列を定めている。I 文には続けて指標も IX 文で記述する。上記の集団データ型は他のデータ型から構成されるが、構成要素のデータ型の中にさらに集団データ型があれば、それを G (Group) 文で指定する。繰返し集団データ型には繰返し終了条件が必要であり、22 行目の ON 句は End Of File 条件を示し、40 行目の ON 句は実体 sale の個数だけ繰り返すことを示している。

ところで、本稿では入出力データは実体や、その属性値、関連を表すものと見なしているの、まず実体については実体型の主キー属性を図 1 の 25 行目の = 文で基本データ型と結合する。属性値については属性を 27 行目の = 文で基本データ型と結合する。関連については、関連で対応づけられた実体型の主キー属性を 33 行目と 35 行目の = 文で基本データ型と結合する。なお、実体のないところには関連もないので、入出力データには関連とともに実体も表される。上記の結合をとることは情報層とデータ層の結合制約と呼ぶ。

(2) アクセス層

図 1 の 46 行目の ACCESS 文に続けて各々のファイルをデータセット型として 47 行目の D (Dataset Type) 文で記述する。この文にはファイル名や、INPUT と OUTPUT の区別、レコード長を記述する。データ層とアクセス層の結合制約として product_data のようにデータ型を指定する。

3 仕様の再利用技術

筆者らは以下のステップで仕様を再利用して、プログラムを作成する考え方をとっている。

1) 適用領域の中の典型的なシステムの事例を反映し

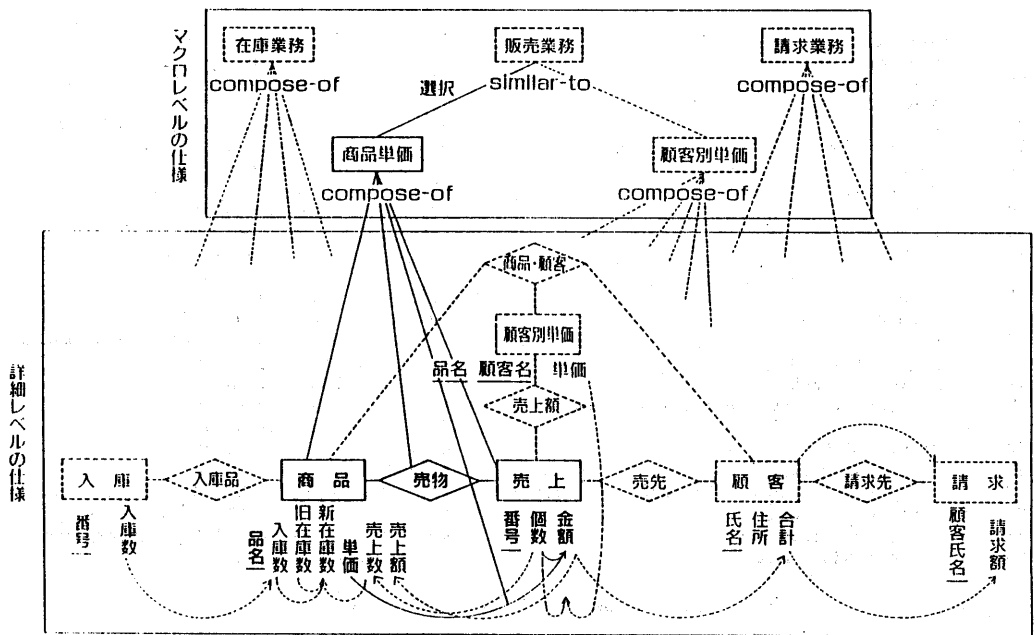


図 2: モデルシステム仕様

て、再利用の対象となるモデルシステム仕様を作成する。

- 2) 開発対象の目的システムに合わせてモデルシステム仕様をカスタマイズして、目的システム仕様を得る。
- 3) 目的システム仕様からプログラム仕様を抽出する。
- 4) プログラム仕様をプログラムへ変換する。

上記のうち、本稿と密接に関係するモデルシステム仕様と目的システム仕様、プログラム仕様抽出について以下に述べる。

3.1 モデルシステム仕様と目的システム仕様

これらの仕様は一つのプログラム仕様を表すものではなく、多くのプログラムから実現される一つのシステムについて、その要求を表すものである。そのため、プログラム仕様と比較して記述量が非常に多くなる。そこで、仕様の階層的な理解を支援するため、図2に示すように、PSDLの情報層で記述された仕様の上部構造を階層的に定義できる概念スキーマを設けた。

概念スキーマはいくつもの実体型や属性、関連型、制約、さらに他の概念スキーマも含むことができ、それらをまとめたマクロを定義するものである。概念スキーマの内部と外部とは、その概念スキーマに含まれている実体型や属性、関連型を介して制約でつながれている。

概念スキーマは仕様の理解を支援するほか、図2の“similar-to”の例で示すように、仕様の部分的な類似性や同一性も定義できる。この類似性や同一性は、ユーザが選択できる仕様をモデルシステム仕様を持たせ、モデルシステム仕様の内容を豊富にするものである。なお、カスタマイズの段階でユーザによる選択が実施されるので、目的システム仕様は概念スキーマは含んでいても、類似性や同一性の定義を含まないものとする。

3.2 プログラム仕様抽出

目的システム仕様からプログラム仕様を抽出するのに以下の2つの方法がある。

- 1) 目的システム仕様中の概念スキーマが一つのプログラムの仕様を表している場合、ユーザがそのスキーマを指定して、その中に含まれている仕様を得ることによってプログラム仕様を抽出できる。
- 2) 概念スキーマには表されていないプログラムの仕様を得たい場合は、目的システム仕様から作成された有向グラフ上で、ユーザによって指定されたプログラムの入出力データを端点とする有向道群上の仕様を抽出すればよい。

本稿の主題である上記2)について次章に詳しく述べる。

4 プログラム仕様の抽出

以下の順序でプログラム仕様を抽出する。

4.1 有向グラフの作成

グラフの節点は PSDL 文にしたがって集めるが、表 1 に示すように節点を 10 種設け、それらを構造節点と制約節点とに分ける。なお、データ型節点には木構造の根となっているデータ型のみを反映させる。さらに、そのデータ型毎に情報層とデータ層の結合制約を集約して 1 つの節点で表す。

グラフの有向枝は構造節点と制約節点の間に張る。枝の方向は情報層では実体や、属性値、関連が制約から参照される方向と、制約から得られる方向に合わせ、データ層とアクセス層ではデータが入力用データセット型から流出する方向と、出力用データセット型へ流入する方向に合わせる。図 3 に有向グラフの例を示す。

表 1: 節点

節点	
構造節点	実体型節点
	属性節点
	関連型節点
	データ型節点
	データセット型節点
制約節点	属性値従属性制約節点
	関連存在従属性制約
	実体存在従属性制約
	情報層とデータ層の結合制約
	データ層とアクセス層の結合制約

4.2 有向道の選択

3.2 節で述べたように、有向道による仕様の抽出にあたってはユーザが入出力データを指定するが、その指定の中に誤りが混入しうる。そこで、出力データの方を重視して、出力データのデータセット型節点から逆にたどって有向道群を検出するものとする。その場合、有向道上で 2 本以上の有向枝が合流する節点が出現するが、その枝の全てが必ずしも必要ではない。このため、以下に示すように関連型の関連数に着目して、入力データの指定を補助情報に用いながら有向枝の選択を行う。

4.2.1 制約節点の有向枝選択

制約を表す式(または関数)の中に属性値の参照が 2 つ以上あると、制約節点へ有向枝が合流する。この場合、式(または関数)の計算にどの属性値も欠かすことが出来ないため、全ての有向枝を選択する。

4.2.2 実体型節点の有向枝選択

同じ実体型の実体が 2 つ以上の実体存在従属性制約や入力データから得られる場合、実体型節点に有向枝が合流する。以下、合流有向枝が 2 本の場合について選択法を示す。なお、3 本以上への拡張は容易である。2 本の合流有向枝の関連数を (m, n) で表現する。

(1) 関連数が $(1, 1)$

一方の有向枝から得られる実体他方の有向枝からも得られるので、片方の有向枝しか必要としない。このため、指定された入力データにつながっている道上の有向枝のみを選択する。

(2) 関連数が $(1, -1)$

関連数が -1 の有向枝から得られる実体は全て、関連数が 1 の有向枝からも得られるが、その逆は成り立たない。このため、関連数が 1 の有向枝のみを選択する。

(3) 関連数が $(-1, -1)$

一方の有向枝から得られない実体が、他方の有向枝から得られることがあるので、両方の有向枝を選択する。

(4) 関連数が M の場合の補正

この有向枝から得られない実体が、他の有向枝から得られることがあるので、関連数が -1 の場合と同様な選択を行なう。

4.2.3 属性節点の有向枝選択

属性の値が 2 つ以上の属性値従属性制約や入力データから得られる場合、属性節点へ有向枝が合流する。以下、前節と同様に合流有向枝が 2 本の場合について選択法を示す。

(1) 関連数が $(1, 1)$

同じ属性値が両有向枝から得られるので、片方の有向枝しか必要としない。このため、入力データにつながっている道上の有向枝のみを選択する。

(2) 関連数が $(1, -1)$

関連数が -1 の有向枝から得られる属性値は全て、関連数が 1 の有向枝からも得られるが、その逆は成り立たない。このため、関連数が 1 の有向枝のみを選択する。

(3) 関連数が $(-1, -1)$

一方の有向枝から得られない属性値が、他方の有向枝から得られることがあるので、両方の有向枝を選択する。

(4) アグリゲーション関数による関連数の補正

合流有向枝の始点となっている制約節点が、例えばサンメンション(総和)のようなアグリゲーション(集合)関数の場合、その枝の関連数が -1 か M であっても必ず属性値が得られる。このため、その有向枝の関連数が 1 の場合と同様な選択を行なう。

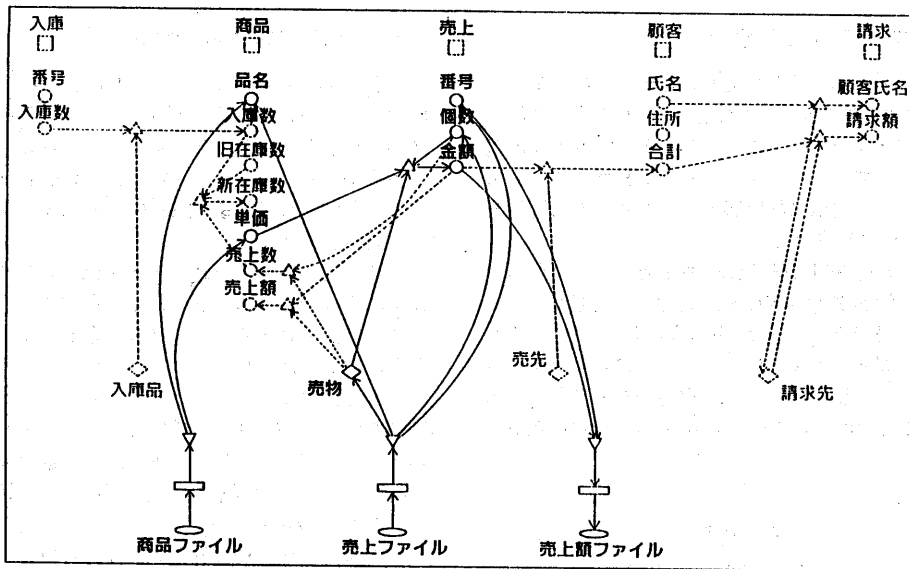


図 3: 有向グラフ

4.2.4 関連型節点の有向枝選択

関連が関連存在従属性制約と、入力データの両方から得られる場合に有向枝が合流する。この場合、必要な有向枝は自動的に判断できないので、ユーザの選択に任せる。

4.3 支援系の作成

筆者らは SUN 上で ER 図を操作して仕様を編集できる、PSDL エディタを実現した。現在、そのエディタへ階層的な概念スキーマの機能や本抽出法を組み込み中である。また、社会保険システムのモデルシステム仕様を事例として作成中である。

5 おわりに

ER モデルと制約に基づく仕様記述言語 PSDL で表された目的システム仕様から、関連型の関連数に着目した道の選択によって、一つのプログラム仕様を抽出する方法を提案した。

本稿の抽出法は必要最小限な仕様を包含した冗長な仕様が抽出されるので、その冗長性を除去するため、ユーザの介入が必要になっている。今後は現在作成中の仕様再利用支援系を用いて実験を行い、ユーザの介入を減らすための知能について解明を図りたい。

さらに将来はこの研究を発展させて、目的システム仕様からソフトウェア・モジュール構造を決める方法や、カスタマイジング後の目的システム仕様の完全性や無矛盾性を検査する方法について研究する予定である。

謝辞

日頃ご指導いただく葉原会長、山下社長、竹中室長に深謝いたします。また、ご討論いただいた研究室の諸氏、ならびに支援系の作成にご協力いただく日本電子計算株式会社の諸氏に感謝いたします。

参考文献

- [1] K. Okamoto and M. Hashimoto: On real-time software specification description with a conceptual data model-based language, In *Proc. of ICCI'90*, 1990.
- [2] M. Hashimoto and K. Okamoto: A set and mapping-based detection and solution method for structure clash between program input and output data, In *Proc. of COMPSAC'90*, 1990.
- [3] 岡本克巳, 橋本正明: 制約指向の概念モデルを用いた高次部品化によるプログラム合成, 電子情報通信学会技報ソフトウェアサイエンス 89-18, 1989.
- [4] T. K. Sellis and L. Shapiro: Optimization of Extended Database Query Languages, In *Proc. of ACM-SIGMOD*, Vol. 14, No. 4, 1985.
- [5] S. Y. W. Su and S. Puranik and H. Lan: Heuristic Algorithms for Path Determination in a Semantic Network, In *Proc. of COMPSAC'90*, 1990.