ソフトウェア基礎論　３２－１
(1989.11.13)

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

岡本克己，橋本正明

ＡＴＲ通信システム研究所

あらまし　プログラム合成について，対象世界の情報の枠組や，その枠組を記述するためのプログラム仕様記述言語，プログラム仕様の部品化などの観点から考察する．ＥＲモデルと制約を用いた概念モデルによって，人の認識方法に合致し，計算機の実働化に対する考慮は排除された対象世界の記述ができる仕様記述言語を示す．さらにこの言語で記述されたプログラム仕様の部品を用いた高次部品化技術を提案し，高次部品の再利用によるプログラム合成について述べる．

Program Synthesis by Specification Parts Technique Using A Constraint-based Conceptual Model

Katsumi OKAMOTO and Masaaki HASHIMOTO

ATR Communication Systems Research Laboratories

Sanpeidani Inuidani Seika-cho Soraku-gun Kyoto 619-02 Japan

Abstract This article discusses the program synthesis from the viewpoint of the framework for understanding a universe of discourse, the program specification description language, and the specification parts technique. The program specification description language based on the conceputual model using the ER-model and constraint is described. This language is suitable for how a programmer understand a universe of discourse, and does not need the consideration for the program implementation on a computer. The program synthesis by reusing the specification parts described by the language is proposed.

１．はじめに

今日，高品質，高信頼なソフトウェアをいかに効率良く作成するかが大きな問題となっており，この問題を解決する１つの方法としてソフトウェアの自動合成に関心が高まっている．

ソフトウェア自動合成の問題は，1)ソフトウェア化したい対象世界の認識の方法と，2)その認識結果に基づく計算機の動かし方，に大きく分けられる．そこで筆者らは，1)については対象世界の認識結果を記述するためのプログラム仕様記述法ＰＳＤＭ（Program Specification Description Method）[1),2),3),4)]を研究し，2)についてはプログラム構造の自動設計を含めたプログラム生成法[5),6)]を研究中である．

ＰＳＤＭはプログラムの入出力データに表されている情報についての枠組と，入出力データの形式に着目した宣言的な仕様記述法である．この情報の枠組を記述するのにＥＲ（Entity Relationship）モデルと制約を用いた概念モデルを用いている．

また，対象世界の認識結果から効率良くプログラムを合成するする方法として，この概念モデルを用いて記述されたプログラム仕様の部品化と，その部品の再利用によるプログラム合成の検討を行っている[7)]．

そこで本稿では，まず第２章でプログラム合成の考え方について対象世界の情報の枠組，対象世界の記述言語，プログラム仕様の部品化，などの観点から考察を加える．そして第３章で筆者らが対象世界の記述方法として研究中のプログラム仕様記述法ＰＳＤＭに基づいて規定されたプログラム仕様記述言語ＰＳＤＬ（Program Specification Description Language）の説明を行い，第４章でＥＲモデルに制約指向プログラミングの考え方を導入した概念モデルと，それを用いて記述されたプログラム仕様の部品である高次部品，高次部品の再利用によるプログラム合成の方法，今後の研究課題についてについて述べる．

２．プログラム合成の考え方

本章では，プログラム合成の考え方について対象世界の情報の枠組，対象世界の記述言語，プログラム仕様の部品化，などの観点から考察を加え，最後に筆者らの考えるプログラム合成の全体像について説明する．

２．１　対象世界の情報の枠組

プログラム合成においてソフトウェア化したい対象世界の認識方法や認識結果の記述方法が重要な問題である．

認識結果を記述するための枠組が満たすべき条件として，

1)人の認識方法との合致（線型性）

2)計算機の実働化に対する考慮の排除

3)適用分野との独立性

が上げられる．

条件1)については，対象世界を認識する主体が人であるために，対象世界記述の枠組は人が認識する方法に対して合致している必要がある．この条件が満足されておれば，多種多様な対象世界の記述のために生じる記述結果の修正や拡張の際に，人にとって本来無用な修正が他の部分へ波及することがなくなる．すなわち，記述の枠組が線型性を備えていることになる．

条件2)については，記述の対象がソフトウェア化したい対象世界であり，計算機の動かし方ではないので，例えば手続き型プログラミング言語のように計算機の動かし方を考慮しなければ記述できないものはこの条件に反している．すなわち，対象世界の言葉のみで記述できる必要がある．

条件3)については，もちろん適用分野に依存した枠組も重要であるが，対象世界の認識レベルとして，適用分野に独立なレベルを確立することは重要である．

以上に述べた条件を満たすものとして，筆者らは対象世界の情報の枠組を”事物の存在，及び，存在相互の関係”に基づいて認識するＥＲモデルに着目して研究している．

２．２　対象世界の記述言語

次に対象世界の認識結果を記述する言語の観点から考察すると，２．１節で述べた条件を満たす対象世界の記述言語が必要であることことがわかる．すなわち，線型性を持った（宣言的である）対象世界の言葉のみで記述できる（計算機の動かし方を考慮しなくてよい）言語が必要である．

そこで筆者らは，プログラムの入出力データに表されている情報についての枠組と，入出力データの形式に着目した宣言的な仕様記述法ＰＳＤＭを研究し，それに基づいて規定されたプログラム仕様記述言語ＰＳＤＬを研究中である．

２．３　プログラム仕様の部品化

プログラム合成を行うためには，合成に用いる個々の部品をどのような形で蓄積しておくかが重要である．

今日研究されている部品合成によるプログラムの自動生成で用いられているプログラム部品は，

1)ソースコードを部品化したもの

2)仕様レベルを部品化したもの

3)設計情報を部品化したもの

の３つに分類できる[8]．

ソースコードを部品化した場合，目標言語が既に固定されており，部品の抽象度が非常に低くい．さらに，ソースコード部品は当然ながら計算機の動かし方が含まれており，ソフトウェア化したい対象世界の認識結果を部品として蓄積していく上で適していない．

また設計情報の部品化は，現在の技術ではまだ困難である．

以上のことから仕様レベルを部品化することが現時点では最良と考え，筆者らは，ＰＳＤＬによって記述されたプログラム仕様の部品化を検討している．詳細については，第４章で述べる．

２．４　プログラム合成の過程

筆者らは，以上２．１節から２．３節において述べたような対象世界の認識結果記述の枠組を備えたプログラム仕様記述言語によって記述された，プログラム仕様の部品を用いた，プログラム合成を理想と考え，図１に示すようなプログラム合成過程を検討している．

まず，初期段階として問題領域の骨となる主な対象世界を記述し，それを初期部品として蓄積しておく．このときの対象世界記述は，対象世界の認識結果記述の枠組が２．１節で述べた３つの条件を満たすようなプログラム仕様記述法で記述されたプログラム仕様のレベルの部品として蓄積している．

このようにして蓄えられた対象世界記述の部品群の中から目的システムの仕様に合うように部品の選択，結合を行う．その際，既に蓄積されている部品がそのままの形だけしか使用不可能であれば，合成できるプログラムの種類は高々部品の組み合せに限られてしまう．そこで，部品結合された対象世界記述への追加，修正を行えるようにする．このように人に追加，修正されてできた対象世界記述は，既存の部品からは合成できない新しい概念を含んだものであるから，これも部品として新たに蓄えておく必要がある．

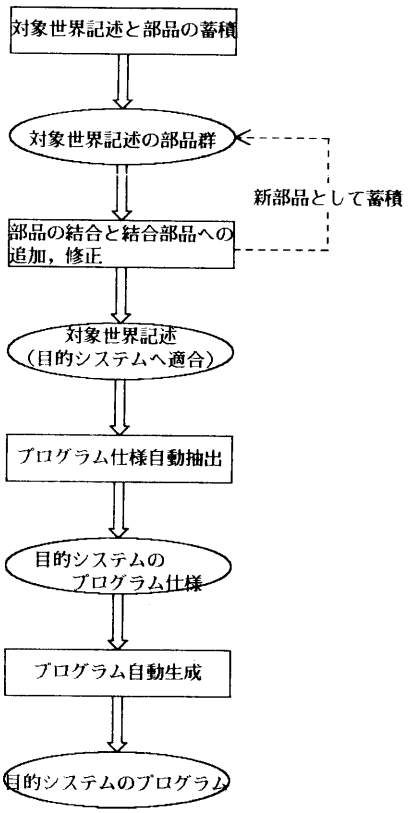


図１　プログラム合成過程

次に，部品結合によって得られた目的システムへ適合した対象世界記述に表れる事物の中で，目的システムでは何が入力として与えられ，何を出力として求めなければならないのかという情報を与えることによって目的システムのプログラム仕様を得る．

そうして，得られたプログラム仕様をプログラムジェネレータに入力として与え，目的システムのプログラムを得るというものである．

３．ＰＳＤＬ

本章では，筆者らが対象世界の記述言語として研究中のＰＳＤＬについて，その特徴と記述要素について述べる．

3.1 特徴

ＰＳＤＬは，プログラム仕様記述法ＰＳＤＭに基づいて規定されたプログラム仕様記述言語である．

ＰＳＤＭは，プログラムの入出力データの性質に着目した非手続き的なプログラム仕様記述法であり，入出力データの性質を，

1)入出力データが表す対象世界（プログラムが処理対象としている世界）に存在する事物や，その事物相互の関係を基準にして情報の枠組を定めた情報構造

2)入出力データ中のデータ項目の並び方や，データ項目中の記号の並び方，さらにデータ項目と情報構造の対応を定めたデータ表現方法

3)データ入出力について，ファイル等の性質や，入出力の区別などを定めたデータ・アクセス方法

という３つの側面から捉え，各々を情報層，データ層，アクセス層に分けて，プログラム仕様を記述する．さらにその際，各層をプログラム仕様に表れる対象をタイプとして記述する構造と，タイプに課す条件を記述する制約に分けて記述する．また情報層の記述には，２．１節で述べた対象世界の認識結果記述の枠組が満たすべき条件を満たすＥＲモデルが用いられている．

従ってＰＳＤＬは，仕様の理解性，記述性，拡張性，形式性に優れたプログラム仕様記述言語となっている．

情報層	・エンティティ・タイプ ・アトリビュート ・プライマリ・キー ・リレーションシップ・タイプ ・リレーションシップ存在従属性制約 ・アトリビュート値従属性制約 ・エンティティ存在従属性制約
データ層	・データ・タイプ ・データの情報制約 ・繰り返し制約 ・選択制約
アクセス層	・データセット・タイプ ・入出力制約 ・データ制約

表１　ＰＳＤＬの記述要素

3.2 記述要素

ＰＳＤＬには，表１に示すような記述要素がある．以下，各層ごとに構造と制約に分け，図２に示す例題のプログラム仕様をＰＳＤＬ記述した図３を用いて説明する．

3.2.1 情報層

情報層には情報構造をプログラム仕様として記述する．

情報層の構造は，エンティティ・タイプ，リレーションシップ・タイプ，アトリビュート，及び，プ

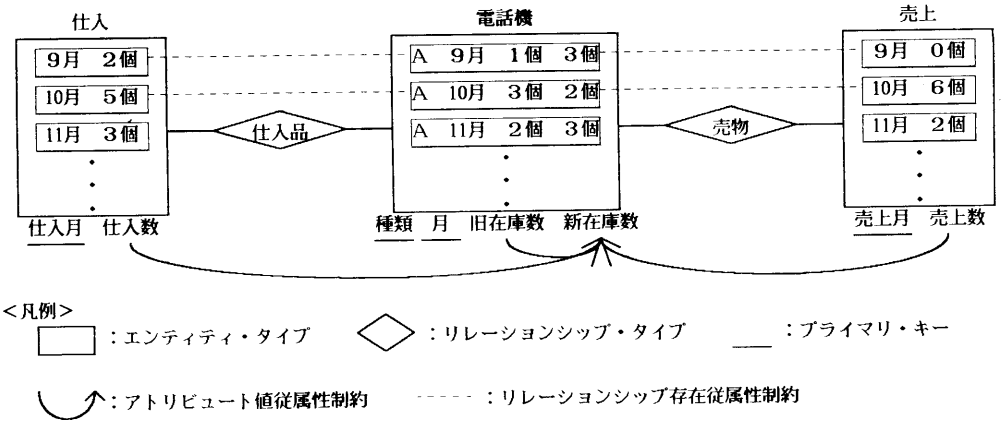


図２　情報層の構造と制約

```
00 INFORMATION
01   E shiire
02     K shiire-tuki     Num
03     A shiire-su       Num
04   E denwaki
05     K syurui          Str
06     A tuki            Num
07     A kyuu-zaiko-su   Num
08     A shin-zaiko-su   Num
09       = kyuu-zaiko-su + .shiire-hin..shiire.shiire-su
           - .urimono..uriage.uriage-su
10   E uriage
11     K uriage-tuki     Num
12     A uriage-su       Num
13   R shiire-hin
14     C .shiire
15     C .denwaki
16   R urimono
17     C .denwaki
18     C .uriage
19 DATA
20   IX s
21     {
22       0 shiire-r
23         {
24           %2d  s-shiire-tuki  = shiire.shiire-tuki
25           %2d  s-shiire-su    = shiire.shiire-su
26         }
27     }
28   IX z
29     {
30       0 zaiko-r
31         {
32           %1s  z-syurui     = denwaki.syurui
33           %2d  z-tuki       = denwaki.tuki
34           %2d  z-zaiko-su   = denwaki.shin-zaiko-su
35         }
36     }
37   IX u
38     {
39       0 uriage-r
40         {
41           %2d  u-uriage-tuki  = uriage.uriage-tuki
42           %2d  u-uriage-su    = uriage.uriage-su
43         }
44     }
45 ACCESS
46   D shiire-f input          4 shiire-r
47   D zaiko-f  input-output   5 zaiko-r
48   D uriage-f input          4 uriage-r
```

図3　ＰＳＤＬ記述

ライマリ・キーを用いて定める．

また情報層の制約は，与えられたエンティティそのアトリビュート値及びリレーションシップに基づいて，与えられたものとは異なるエンティティ，アトリビュート値またはリレーションシップを得るための計算方法を定めたものである．

図３では，00行目から18行目が情報層の仕様を記述した部分であり，00行目のINFORMATION文は情報層の仕様記述の始まりを示している．

３．２．１．１　構造

情報層の構造を構成する要素である，エンティティ・タイプ，リレーションシップタイプ，アトリビュート，プライマリ・キーについて，以下順に説明していく．

(1)エンティティ・タイプ

エンティティ・タイプは，対象世界における事物の存在を意味するエンティティを要素とする集合である．

エンティティ・タイプは，図３の01行目のようにE(Entity)文を用いて記述する．

(2)アトリビュートとプライマリ・キー

アトリビュートは，エンティティの性質を表し，属するエンティティ・タイプの記述の後に続けて，例えば03行目のように，A(Attribute)文を用いて記述する．

プライマリ・キーは，エンティティ・タイプの中で各々のエンティティを識別するために用いる特別なアトリビュートであり，02行目のように，K(Key)文を用いて記述する．

またエンティティ・タイプには，プライマリ・キーを自分自身で持つレギュラ・エンティティ・タイプと，他のエンティティ・タイプのプライマリ・キーで代用するウィーク・エンティティ・タイプとがある．

(3)リレーションシップ・タイプ

リレーションシップ・タイプは，対象世界に存在する事物（エンティティ）相互の対応付けを意味するリレーションシップを要素とする集合である．

リレーションシップ・タイプもエンティティ・タイプと同様に，レギュラ・リレーションシップ・タイプとウィーク・リレーションシップ・タイプに分かれる．前者は，レギュラ・エンティティ・タイプを相互に対応付けたものであり，後者は，少なくとも１つはウィークであるエンティティ・タイプを相互に対応付けたものである．

リレーションシップ・タイプは，13行目のようにR(Relationship)文を用いて記述する．

また，エンティティ・タイプ間の対応付けは，対応付けるR文の後に続けて，14行目のようにC((Collection)文を用いて記述する．

３.２.１.２　制約

情報層の制約は，与えられたエンティティ，そのアトリビュート値及びリレーションシップに基づいて，与えられたものとは異なるエンティティ，アトリビュート値またはリレーションシップを得るための計算方法を定めたものである．以下，各制約について説明していく．

(1)識別従属性制約

３.２.１.１(1)で述べたように，ウィーク・エンティティ・タイプは，そのプライマリ・キーを他のエンティティ・タイプのプライマリ・キーで代用している．そこで，どのエンティティ・タイプのプライマリ・キーで代用するかを指定する必要がある．その指定の仕方を定めた制約が識別従属性制約であり，図３の例ではないが，ID(IDentification)文を用いて記述する．

(2)アトリビュート値従属性制約

あるエンティティのアトリビュート値に基づいて，そのエンティティへリレーションシップで対応付けられた他のエンティティについて，そのアトリビュート値を得るための計算方法を定めた制約であり，求めるアトリビュートを定義したA文に続けて，09行目のように，=(attribute value)文を用いて記述する．

(3)リレーションシップ存在従属性制約

いくつかのエンティティに基づいて，そのエンティティの間に存在するリレーションシップを得るための計算方法を定めた制約であり，図３の例ではないが，RC(Relationship existence Constraint)文を用いて記述する．

(4)エンティティ存在従属性制約

あるエンティティに基づいて，そのエンティティとリレーションシップで対応付けられる他のエンティティを得るための計算方法を定めた制約であり，図３の例ではないが，=(entity existence)文を用いて記述する．

３.２.２　データ層

データ層には，データ表現方法をプログラム仕様として記述する．

図３では，19行目から44行目がデータ層の仕様を記述した部分であり，19行目のDATA文はデータ層の仕様記述の始まりを示している．

データ層の構造は，入出力データ中のデータ項目の並び方を定めるために，①それ以上分解すると意味がなくなるエレメント・データ・タイプ，②いくつかのデータ・タイプが順序を持って並んでできたシーケンス・データ・タイプ，③同じデータ・タイプのデータが１つ以上繰り返して並んでできたイテレーション・データ・タイプ，④いくつかのデータ・タイプのうちのいずれか１つだけが現れるセレクション・データ・タイプを定義し，これら４種類のデータ・タイプを用いて１つの入出力データを１つの木構造として表す．

それぞれデータ・タイプは各々，24行目の%(element)文，22行目のO(Order)文，20行目のIX(Iteration indeX)文，そして図３の例にはないが，S(Selection)文を用いて記述する．

またデータ層の制約は，データと情報層の対応，及び，シーケンス・データ・タイプの繰り返し終了条件やセレクション・データ・タイプの選択条件を定めるものであり，各々，24行目のように=(expression information)文，20行目のようにIX文，そして図３の例にはないが，S文を用いて記述する．

３.２.３　アクセス層

アクセス層にはデータ・アクセス方法をプログラム仕様として記述する．

図３では，45行目以降がアクセス層の仕様を記述した部分であり，45行目のACCESS文はアクセス層の仕様記述の始まりを示している．

構造としては，ファイルやプロセス入出力を表すデータセット・タイプを設け，その名前やアクセス単位長を定める．

制約としては，１つのデータセット・タイプをデータ層の１つの木構造に対応させるためのデータ制約，データセット・タイプの入力用，出力用を区別するための入出力制約を定める．

構造と制約は共に46行目のようにD(Dataset)文を用いて記述する．

４．高次部品化

本章では，プログラム合成の１つの手法として高次部品化を提案し，その特徴，及び，現時点で考えられる研究課題について述べる．

４.１　構想

筆者らは，２.４節で述べたプログラム合成の流れに合ったプログラム合成手法として，高次部品化によるプログラム合成を提案する．高次部品化とは，ＥＲモデルと制約を用いた概念モデルによる対象世界記述（プログラム仕様）の部品化と再利用のこと

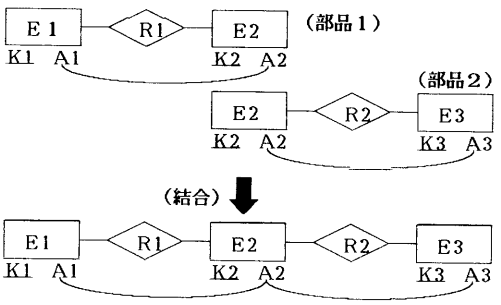


図4　高次部品の結合例

である．

高次部品化は，図1の流れに合わせ，以下に示す方法で実現される．なお，ＥＲモデルと制約を用いた概念モデルについては，4.3.1項で詳しく述べることとし，本節では詳述しない．

(1)対象世界の記述と部品の蓄積

ある対象世界，例えば図2で示した例の対象世界を，ＥＲモデルに制約指向の考えを導入した概念モデルによって認識し，その記述されたものを部品として蓄積する．

部品は，図4の上段に示すようにリレーションシップ・タイプを単位として蓄積できる．その部品の中にはリレーションシップ・タイプが対応付けているエンティティ・タイプとそのアトリビュート，制約も含まれる．

(2)部品の結合

蓄積された部品は原則として同じ名前を持つエンティティ・タイプ，例えば図4に示す"E2"を重ね合わせることによって相互に結合することができる．その時に，不足部品の追加や，部品の修正も行ってよい．

この際，部品結合によって得られたものへユーザが追加，修正することによって作り出された概念（対象世界記述）は，既存の部品からだけでは合成できない新しい概念を含んだものである．従って，これも新たな部品として蓄積しておく．

(3)プログラム仕様の抽出

結合された部品群のうち，プログラムの入力データから値が与えられるアトリビュートと，出力データへ値が渡されるアトリビュートを決定すると，アトリビュート値のデータフローの方向が決まる．それを基に目的システムに適合したプログラム仕様の抽出を自動的に行う．このプログラム仕様の抽出方法については，4.3.1項において詳述する．

(4)プログラムの生成

抽出されたプログラム仕様から記述要素を基に有向グラフを作成し，その有向グラフを解析することによってプログラムの構造設計を自動的に行い，目的システムのプログラムコードを自動生成する．このプログラム生成については別途研究中である．

4.2　特徴

高次部品の主な特徴を以下に上げる．

(1)対象世界の知識のみに依存する部品

前節までの説明からわかるように，高次部品の中に現れているのは対象世界の知識のみであり，プログラムの構造などのプログラムの実現方法に基づいた知識は現れていない．このため，高次部品は2.1節で述べた人の認識の枠組にあった部品であると言える．従って，高次部品の取り扱いにはプログラミングの専門家の必要性は小さくなると予想している．

(2)概念辞書としてのライブラリ

部品の中に現れるエンティティ・タイプやアトリビュート，リレーションシップ・タイプ，制約は，人の思考の中に現れる概念と同じものである．このため，高次部品化の研究は概念操作に着目した研究となり，知識工学の各分野の研究とつながりが取り易く，将来の発展が期待される．

4.3　研究課題

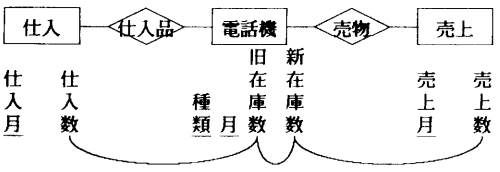
高次部品として様々な操作を行うためには，部品すなわち対象世界記述をどのような概念モデルによって記述するかが重要である．また，高次部品には様々な形態が予想される．

そこで本項では，まず，ＥＲモデルに制約指向の考えを導入した概念モデルについて述べる．そして次に，現時点で考えられる今後検討すべき高次部品に対する研究課題項目を上げる．

4.3.1　ＥＲモデル＋制約

既に述べたように，ＥＲモデルにおいて対象世界はエンティティとそのアトリビュート値，及び，リレーションシップで認識される．そこで，ＰＳＤＬでは，それぞれに応じた，①エンティティ存在従属性制約，②アトリビュート値従属性制約，③リレーションシップ存在従属性制約の3種の制約を用いている．

ここで，③の制約は条件式（関係式）で記述されているが，①と②の制約は計算式で記述されている．



条件式：/電話機/新在庫数 =
　/電話機/旧在庫数 + /仕入/仕入数 - /売上/売上数

計算式①：/電話機/新在庫数 ←
　/電話機/旧在庫数 + /仕入/仕入数 - /売上/売上数
計算式②：/電話機/旧在庫数 ←
　/電話機/新在庫数 - /仕入/仕入数 + /売上/売上数
計算式③：/仕入/仕入数 ←
　/電話機/新在庫数 - /電話機/旧在庫数 + /売上/売上数
計算式④：/売上/売上数 ←
　/電話機/旧在庫数 - /電話機/新在庫数 + /仕入/仕入数

図5　アトリビュート値の条件

高次部品化においては，その計算式の代わりに条件式の記述も許容する．例えば図２の例では，アトリビュート値従属性制約の記述に図５に示す条件式の記述を許すことになる．なお，制約は仕様に含めて記述される．

この図５に示された条件式からは図中の①から④の計算式を導き出すことが可能である．これは，４．１節(3)の高次部品化によるプログラム合成の流れにおけるプログラム仕様の抽出段階おいて，どのアトリビュートが入出力になるかが決まり，アトリビュート値のデータ・フローの方向が決まった時に，計算式の①から④の中から方向に合ったものを選んで条件式と置き換える．この制約解消の方法は，制約指向プログラミングの局所的伝播（Local Propagation）[9]に当る．

４．３．２　概念の抽象化

たくさんの概念（高次部品）を蓄積していくと，表された概念の大部分は同じであるが，残りの一部分だけが異なっている類似概念が数多く含まれる．そこで，これらの概念を抽象化した１つの概念で表すことによって，より体系化された概念辞書として蓄積することができる．

概念は，エンティティ・タイプとアトリビュート，及び，リレーションシップ・タイプと制約によって構成されている．またアトリビュートはエンティティ・タイプに，制約はリレーションシップ・タイプに付随するので，抽象化はエンティティ・タイプとリレーションシップ・タイプの両方に必要である．すなわち，共通のアトリビュートを持ったエンティティ・タイプを抽象化エンティティ・タイプ，共通の制約を持ったリレーションシップ・タイプを抽象化リレーションシップ・タイプとする．この概念の抽象化のイメージを図６に示す．

概念の抽象化において他に検討しなければならない項目として，概念をどの程度まで抽象化することを許すか，そして抽象化された概念をどのような方法で蓄積していくかなどがある．

４．３．３　概念の具象化

概念を抽象化して蓄積すると，当然その蓄積された概念を用いるときに概念の具象化ということを考えなければならない．

概念の具象化において検討すべき項目は，アトリビュートや制約のインヘリタンス，及び，具象化の際に生じる制約矛盾のチェックとその解消である．また，具象化，特に追加，修正によって新たに作成された部品の蓄積も必要である．

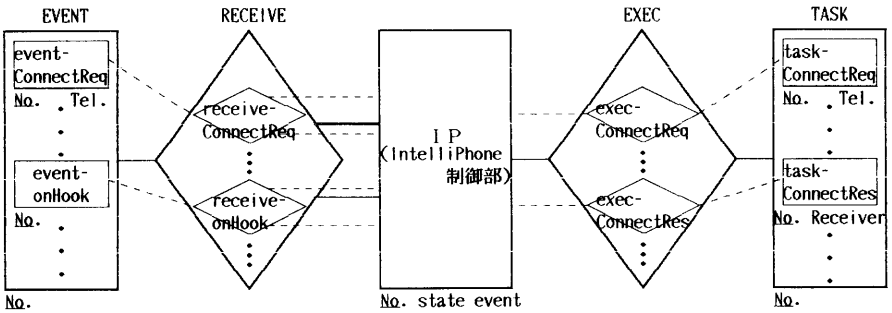


図6　概念の抽象化と具象化

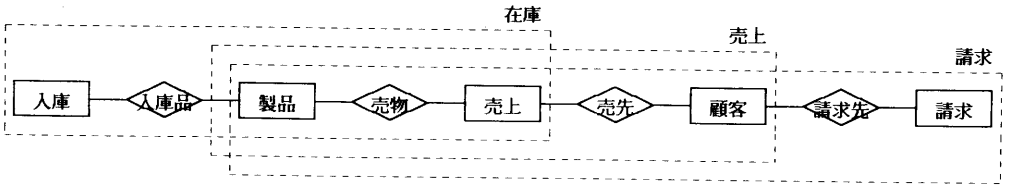


図7　マクロ概念

アトリビュートのインヘリタンス及び追加，修正の簡単な例を示すと，図６の"EVENT"という抽象化エンティティ・タイプに属するエンティティ・タイプの１つである"event-ConnectReq"を得るためには共通のアトリビュートである"No."を"EVENT"より継承し，さらに"Tel."を追加すれば良いわけである．

４．３．４　概念のマクロ化

４．２節の(1)で述べたように，高次部品の１つの単位は，１つのリレーションシップ・タイプとそれに対応付けられるエンティティ・タイプである．しかし，部品のいくつも結合したものが頻繁に定型的に現れれば，それを複合部品として取り扱った方が有効である．この複合部品の認識の枠組，すなわちマクロな認識の枠組について研究する必要がある．

図７にマクロな概念のイメージを示す．図中の破線で囲まれた，"在庫"，"売上"，"請求"といった一固まりがマクロな概念である．

また，マクロな概念を取り扱う場合に検討しなければならない項目として，取り扱うマクロ部品の大きさの基準やマクロ部品との部品結合などがある．マクロ部品の大きさがあまりにも大きすぎると元々の部品化の意味がなくなってしまう．そして，マクロ部品を結合する場合には，４．２節の(2)で述べたエンティティ・タイプ１ヶ所の重ね合わせだけでなく，数カ所に渡る重ね合わせや，リレーションシップ・タイプをも含んだ重ね合わせが必要となってくる．

４．３．５　概念の同一性

複数の人によって作成された多くの部品の中には，見かけは同じであるが実際に意味する内容は異なっていたり，また反対に見かけはまるっきり異なっているが同じ意味であるといった部品が存在する．これらの部品を蓄積，あるいは選択，結合していく際に，どの部品が同一であるかといった概念の同一性を判定する必要がある．

前者の，見かけは同じであるが意味が異なる例としては，エンティティ・タイプやリレーションシップ・タイプ，アトリビュートなどに同音異義語が用いられている場合である．

また後者の，見かけは異なるが意味が同じ例としては，エンティティ・タイプやリレーションシップ・タイプ，アトリビュートなどに同意語を用いられている場合，あるいは図８に示すようなエンティティ・タイプやリレーションシップ・タイプによる部品の構成方法が異なる場合などがある．

これらの問題を解決するためには，同意語辞書などを用意しておいたり，アトリビュートなどの依存関係を基に概念部品全体の意味をなんらかの形で表現し，その情報を基に概念の同一性の判定を行っていく必要がある．

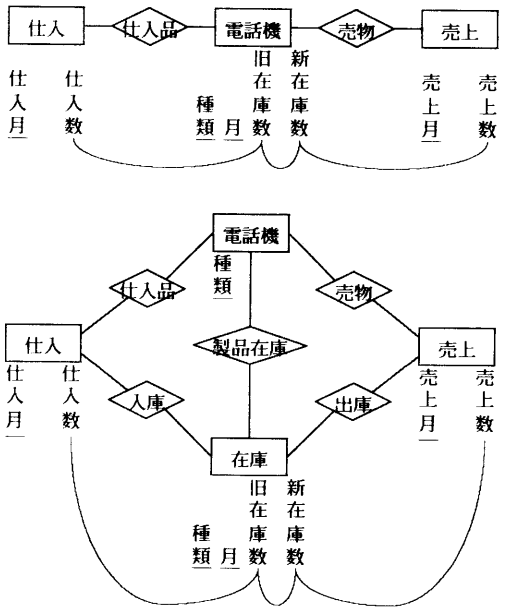


図8　概念の同一性（同値表現）

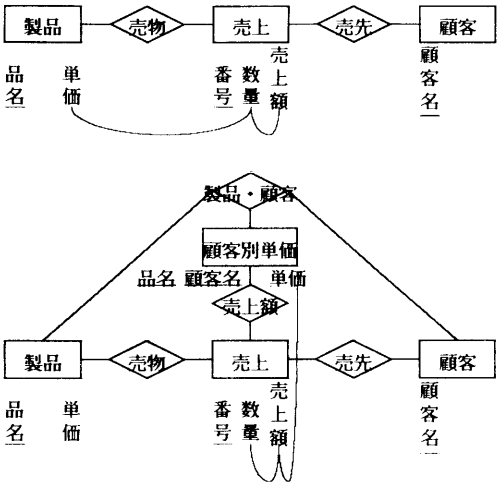


図9　概念の類似性

4.3.6　類推を用いた部品選択メカニズム

様々な概念部品を蓄積した後には，それらを選択するメカニズムが必要となってくる．また部品合成によるプログラム合成の手法では，合成できるプログラムの種類は，高々部品の組み合せに限られてしまう．そこで筆者らは，部品選択の推論メカニズムに類推を導入しようと考えている．

類推を用いた部品選択を行えば，最適な部品がなくても，それに類似した部品の導出も可能である．図9に類似した概念部品の例を示す．

またそのための重要な検討項目として，部品間の類似性の定義がある．これは，部品の蓄積及び同一性の判定段階でも重要な意味を持つが，部品選択の類推に必要不可欠な情報である．

さらに，その他にも部品選択，及び，結合時における各種のメタ知識（戦略知識）についても検討の必要がある．

5．おわりに

プログラム合成について，対象世界の認識の枠組，認識結果の記述言語，認識結果の部品化などの観点から考察し，人の認識方法に合致した線型性を持つ，計算機の実働化に対する考慮が排除された対象世界の認識の枠組として，"ＥＲモデル＋制約"を用いた概念モデルを提案し，その記述言語であるＰＳＤＬについて説明した．

そして，その概念モデルを用いて記述されたプログラム仕様の部品化，再利用である高次部品化によるプログラム合成の方法を提案した．

今後は，高次部品の様々な形態を明らかにするため，概念の取り扱いに関する上述の課題を中心にして研究していく予定である．

謝辞

日頃ご指導いただく，ＡＴＲ通信システム研究所の葉原会長，山下社長，竹中室長に深く感謝致します．また，ご意見をいただいた通信ソフトウェア研究室の諸氏に感謝致します．

参考文献

1）橋本正明：ＥＡＲモデルに基づく情報構造記述を用いたプログラム仕様記述法ＰＳＤＭ，情報処理学会論文誌，Vol.27，No.7（1986）．

2）橋本正明：データ中心のプログラム仕様記述法，P.210，井上書院，東京（1988）．

3）岡本克己，橋本正明：リアルタイム・ソフトウェアにおけるプログラム仕様記述法に関する一考察，情報処理学会ソフトウェア工学研究会資料，Vol.89，No.11，pp.73-80（1989）．

4）岡本克己，橋本正明，門田充弘：リアルタイム・ソフトウェア仕様の時間に関する一考察，情報処理学会第３８回全国大会講演論文集，7M-4，pp.1300-1301（1989）．

5）橋本正明：非手続き型言語と入出力データの構造不一致，情報処理学会論文誌，Vol.29，No.12（1988）．

6）橋本正明：プログラム構造設計の自動化について，情報処理学会「ＣＡＳＥ環境」シンポジウム論文集，pp.101-108（1989）．

7）岡本克己，橋本正明，門田充弘：ＥＲモデル＋制約を用いた対象世界の記述による高次部品化について，情報処理学会第３９回全国大会講演論文集，7S-2，pp.1603-1604（1989）．

8）古宮誠一，原田実：部品合成による自動プログラミング，情報処理，Vol.28，No.10，pp.1329-1345（1987）．

9）G.L.Steel JR.：The Definition and Implementation of a Computer Programming Language Based on Constraint，Ph.D. Dissertation，MIT-AI-TR 595（1980）．