

“柔らかい” 部品参照を用いるプログラミング 言語に関する一考察

小林 弘明 岡本 秀輔 曾和 将容

電気通信大学大学院 情報システム学研究科, 曾和研究室

〒182 東京都調布市調布が丘 1-5-1

一般にシステムは複数の部品から構成される。この時、用いる部品を特定の部品へと固定せずに、部品の集合として『柔らかく』参照する事で、状況変化に対する適応性が向上する可能性が有る。本稿ではそのような、部品を柔らかく参照する方法に関して考察を行う。考察の結果に基づき、プログラミング言語の基本要素のレベルで利用可能な、柔らかい部品参照のための仕組みを提案する。

A basic study for programming language with Indirect Predicate Reference to Module Candidates.

Hiroaki KOBAYASI, Shusuke OKAMOTO and Masahiro SOWA

Graduate School of Information Systems,
University of Electro-Communications, Tokyo, 182 Japan

In general, a system is constructed as a collection of modules. If we can specify each module used in the system as a set of candidates, then whole system will get more flexibility. In this study, we discuss how we should represent the candidates, and propose a notation to specify the candidates through an expression consists of symbols for concept.

1 はじめに

情報処理技術の発達に伴い、情報システムの適用範囲も拡大の一途をたどって来た。システムに対する要求は多様化し、全ての要求を事前に予想し完全に分析する事は困難になりつつ有る。このため、事前の分析に依存したシステムではなく、状況の変化に適応する能力を持った『柔軟いシステム』[1]の必要性が増大している。またシステム運用中の柔軟さのみならず、プログラムモジュールの仕様記述のような知識記述に関しても、要求仕様の非単調的な増加に適応させるために、『柔軟い形式的仕様記述』[2]に関する研究が行われている。

この、システム運用前または運用中の状況変化に対して適応性を得るには、システムもしくは知識記述の全体的な姿を、変化に応じて適切に変更できる必要が有る。このための方法のうち最も容易な方法は、システムを構成する一部の部品を他の適切な部品と入れ換える事である。この方法で得られる柔軟さは、部品そのものを変更する方法と比べて限度が有るが、部品の量や個々の部品の汎用性が十分に有る場合は、変化に適応可能である。しかし、部品の入れ換えを自動化するためには、どの部品をどの部品へと入れ換えれば良いか、自動的に判断する仕組みが必要である。

この判断を行うためには、この部品が入る、と直接的に部品への参照を記述するのではなく、その場所に入り得る部品(候補)の集合に対する参照を記述しておく事が必要である。我々は、候補の集合への参照を記述するとは、極論すれば概念を記述する事に他ならないと考えた。本研究は、概念を記述するための仕組みをプログラミング言語の基本要素(識別子)のレベルで提供する方法に関する基礎的問題を扱うものである。本稿では運用前(コンパイル時までの)記述に柔軟さを与える方法に関して、部品の内部には触れずに、部品の入れ換えに関する部分に焦点を絞って考察を行う。この考察の結果に基づき、概念を記述するためのプログラム要素として、概念制約式を提案する。

2 柔軟さの定義と『概念』を記述する仕組みの必要性

2.1 柔軟いシステムの定義と、本研究の前提

ここでは[1]の定義と分類に準じて、柔軟い¹システムを以下のように定義する。

¹[1]では『やわらかい』、[2]では『軟らかい』と表記しているが、本稿では『柔軟い』に統一する。

定義 2.1 システムの利用環境の変化やシステム内部の変化に対し、事前に標準として用意された方法に拘束されることなく、利用者へ必要なサービスを提供し、同時にシステム機能の低下を防ぐように対処可能なシステムは、柔軟いシステムである。

柔軟さは、その外部の変化の認識方法と、変更の制御の方法に応じて次の3種に分けられる[1]。

定義 2.2 内的あるいは外的変化を認識した人間が、システム(あるいは部品)の変更を制御することにより得られるシステム(部品)を、『受動型』の柔軟いシステム(部品)と呼ぶ。一方、外的変化や内的変化を観測し、それらの意味を認識し、その変化に外部の人間やシステムの介在無しに自身を変更する事により対応するシステム(部品)を『自律型』の柔軟いシステム(部品)と呼ぶ。また外的変化や内的変化に対して、人間とシステム(部品)が協調してシステム(部品)を変更する場合を『協調型』の柔軟いシステム(部品)と呼ぶ。

本研究では、上記の分類における『受動型』を前提として議論を行う。システムに対する要求の変化は、運用開始前に与えられるとする。

2.2 本研究における、柔軟いシステムの構成法

一般にシステムは『部品』とそれらの『接続関係』からなる。本研究における部品とは、プログラマが再利用する価値を見出した、一つのまとまった記述であるとする。完結したプログラムの場合も有れば、一種のスケルトンプログラムの場合もある。典型的には、OOP(オブジェクト指向プログラミング)[3]におけるクラスが部品となる。

システムが柔軟いためには、その構成部品と接続関係が柔軟い必要がある。図1(a)は柔軟い部品と接続関係からなる柔軟いシステムを表している。運用時も

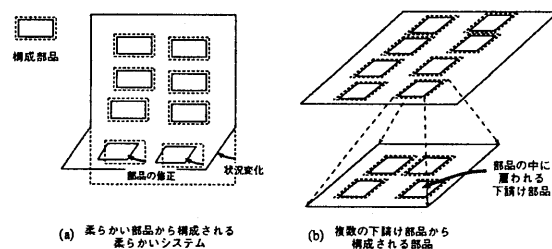


図1: 部品と接続関係からなる柔軟いシステム

しくは運用前に、仕様の変更の要求が発生した時に、要求に対応する部品あるいはそれと接続関係を持つ部品を修

正する事により、変化した仕様に対応したシステムへと修正できる可能性が有る。

同図 (b) は、部品が複数の部品 (下請け部品と呼ぶ) の集まりから構成されている場合を表した図である。この場合、変更の要求に対して、下請け部品の入れ換えのみで対処できる可能性が有る。そこで、上位の部品は、直接特定の部品を指すのではなく、その場所に入り得る下請け部品の候補の集合を指すようにする方法が考えられる。状況に応じて候補の中の適切な部品を選ぶ仕組みが有れば、これにより変化に適応する事が可能になる。図 2 は、部品の各部分が、下請け部品の候補の集合を指す様子を表した図である。

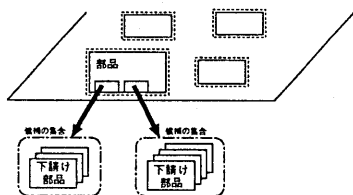


図 2: 部品の候補の集合

2.3 マッチングによる、下請け部品との結び付き

部品の中に下請け部品の候補集合を直接列挙的に書き入れる事は、下請け部品の種類が増えた場合に関して柔軟性を欠いている。そこで、下請け部品のデータベース²を用意し、候補を直接列挙する代わりにデータベースから部品を選び出すための条件式を書き入れる、という方法が考えられる。下請け部品の側も、同じ仕組みの条件式を用いて自分が持つ特性をデータベースに登録しておき、両者の間でマッチングを行う。

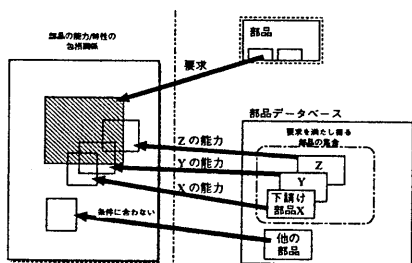


図 3: 判定空間を介した、候補集合の決定

この方法の特徴は、部品のプログラマが、下請け部品の候補の集合を直接的に制限する事が出来なくなる事で

²実際には、下請けを利用する上位の部品も、更に上位の部品に対しては下請け部品となるため、部品は全てこのデータベースに登録する必要がある。

ある。プログラマに、その場所に入り得る下請け部品の条件について熟慮する事を強制できるという利点がある。この点は、プログラマの負担が増えるという欠点でもある。

2.4 条件式を、概念としてまとめる必要性

条件式をそのまま記述に用いる事は問題があるため、これを一つのまとまりを持った『概念』として名前を付けて扱う必要が有る。これは以下の理由による。

部品データベースから下請け部品を選び出すための条件式を考えるという作業は、プログラムのその個所に本質的に入り得る物が何であるかを分析する作業である。

下請け部品を利用する箇所が複数有る時に、それぞれの候補集合がほぼ同じである場合がある。これは、プログラマには一つのまとまった概念として把握されるはずである。にも関わらず毎回条件式を書かせる事は、しばしばプログラマに同じ分析を何度も強いる事になる。これはプログラマのミス (同じ条件式が入れられるべき所に、プログラマが間違えて異なる条件式を書いてしまう) を誘発する可能性もある。

さらに、プログラマに一つ概念として把握されているものに関して、候補集合を決めるための条件式に訂正または追加を行おうとした場合、プログラムの中に直接条件式を書いていた場合、その概念を参照する全ての部品について条件式を書換えを行う必要が生じる。

そこで、条件式を個々の部品に直接埋め込む事はせず、部品を越えた共通の分類表に登録し、そこで、集中管理する事が望ましいと考えた。

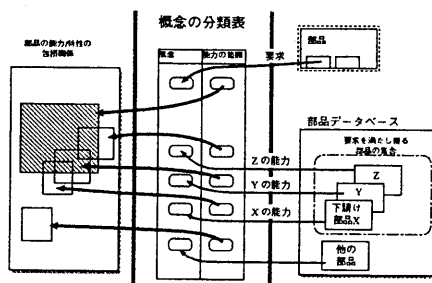


図 4: 概念の分類表を介した、判定空間上の範囲の決定

3 概念を扱うための言語的サポート

3.1 プログラム記述の中で識別子が果たす機能

一般的に識別子は、局所的な識別子(局所変数名、仮引数名)と、大域的な識別子(クラス名、大域変数名、関数名、publicにされたメソッド名)に分けられる。局所的な識別子の名前は、その識別子の指す要素が現在の部品の中で担う役割を表現するように付けられる事が通例である。また、大域的な識別子の名前は、その識別子の指す部品が、一般的に持つ性質を表現するように付けられる。

大域的な識別子を参照する事は、他の部品(もしくはその一部)を参照する事であり、その本質は、部品を雇い入れる事であると考えられる。そのために使われる識別子は、その部品(もしくは部品の属性、メソッド)に付けられた名前であり、その特性を表現した名前になっている事が普通である。

したがって、概念を表現する記述要素を、従来の言語における識別子に替わるものと位置付ける事は、現状の自然な拡張になると考えられる。

3.2 概念制約と概念制約式の定義

概念を表現し、概念の具体性を調整するための記述要素として、『概念制約式』を定義する。これは従来の言語における識別子³に替わる記述要素である。

定義 3.1 概念制約式 ::= 識別子

| 概念制約式 ' $\leq$ ' 概念制約式
| ' $?$ ' 識別子 { 概念制約式 } ' $?$ '

○ [識別子] と 識別子 とは、等価。

○ 概念制約式 $\leq$ 概念制約式 と

[概念制約式 概念制約式] とは等価とし

3.4節で定める『マスク付き単一化』を意味する

概念制約式の中で用いられる識別子は、必ず概念分類表に登録する。概念分類表は、識別子と、以下に定義する概念制約(の集合)との対応表である。

定義 3.2 部品の性質に関する命題⁴の集合 Q (質問集合と呼ぶ) と、それに対する答の集合 $A = \{T, *, u, F\}$ を考える。この時、写像 $C_Q: Q \rightarrow A$ を概念制約と呼ぶ。

答集合 A 中の各記号の意味は、次のように表される。すなわち必然性を表す様相記号を $\square$ 、可能性を表す様相記

³現段階では大域識別子を対象とする。局所識別子を概念制約式により表現する方法については今後の課題である。

⁴今後、述語へと拡張する可能性がある

号を $\Diamond$ 、ブール代数における真と偽を t, f で表すとして、 $T = \square t$, $* = \Diamond t$, $u = \Diamond f$, $F = \square f$ である。特に $*$ は一種のワイルドカード、 u は未知を意味する。例えば要求側が T , $*$ を使った場合、それぞれ『必ずその能力が必要である』、『必ずしも必要ではない』を意味する。また供給側の場合は、『(すると言われても) 必ずその仕事を達成してしまう』、『達成できるが、すると言われたら、しないで済ませることができる』という意味になる。

概念制約上には通常のブール代数のように論理和(=一般化)や論理積(=単一化)の演算が定義される。(図5)。

	単一化				一般化			
	F	u	$*$	T	F	u	$*$	T
F	F	$?$	F	$\perp$	F	$?$	$*$	$*$
u	$?$	u	$?$	$?$	$?$	u	$?$	$?$
$*$	F	$?$	$*$	T	$*$	$?$	$*$	$*$
T	$\perp$	$?$	T	T	$*$	$?$	$*$	T

図 5: 演算の一例

図中、 $\perp$ は、単一化失敗を意味する。また記号 u と他の記号との間に演算が生じた時(図5における“ $?$ ”)は、演算の結果として未知の概念が作られる。これは、原則的に単一化失敗として扱う。ただし、通常の部品検索のためのマッチングではなく、プログラマが明示的にマッチングを指示した場合は、“ $?$ ”を生じた概念が妥当かどうかプログラマに対して問い合わせる。

概念分類表と部品データベースの例を図6に表す。図では、部品データベースには quicksort を実現する部品と、それが用いる下請け部品だけが登録されている。概念分類表には、部品データベース内に記述された全ての概念制約式(の中で使われた識別子)に関する、概念制約が登録されている。

3.3 識別子を概念制約の集合に対応させる理由

一つの識別子は、一つ概念制約ではなく、概念制約の集合と対応付けられる。その理由は以下の通りである。

一つ概念制約は、具体的な概念か、もしくは抽象的な概念か、いずれか一方だけ表現するための物である。これに対し、概念の抽象的な大枠と、その細部の一部を一つのまとまりとして扱えると便利な場合がある。例えば、『文字列同士の比較』に関して、『大文字と小文字を同一視する』という概念は、『比較』という概念の一部分に相当していると考えられる。これらを単一化して、『大文字小文字を同一視するような、比較』という概念を式により表現できれば便利である。そこで、『比較』という概念

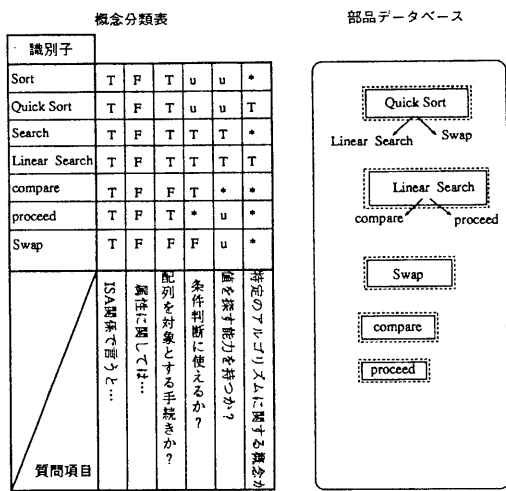


図 6: 概念分類表と部品データベース

に、その大枠を定める概念制約と、細部を定める概念制約を持たせる事を可能にするため、識別子に概念制約の集合を対応させる事にした。

3.4 マスク付き単一化

概念制約式は具体性を調整された概念を表現するための式であり、抽象概念を指す識別子と補助的な条件設定を指す識別子とを組み合わせ用いられる。具体性を調整するための演算として、以下に定義する『マスク付き単一化』を用いる。

定義 3.3 マスク付き単一化 $[X \ Y \Leftarrow Z]$ は概念制約上の演算であり、演算結果の概念制約が指す部品候補集合は、 X , Y , Z の指す部品候補集合を x , y , z とした時、 $(x - y) \cup (x \cap y \cap z)$ である。すなわち、識別子 X の指す概念制約の集合のうち、概念制約 Y との単一化に成功した概念制約のみを概念制約 Z と単一化し、これを、 Y との単一化に失敗した概念制約の集合に加えた集合である。

図 7は、マスク付き単一化を用いて概念の具体性を調整する例である。図では、以下の 1.~ 3. の概念制約式とその概念分類表の関係が示されている。

1. [compare]
- ↓
2. [compare normalization $\Leftarrow$ [ignore digit]]
- ↓
3. [compare normalization $\Leftarrow$ [ignore digit]
normalization $\Leftarrow$ [ignore case]

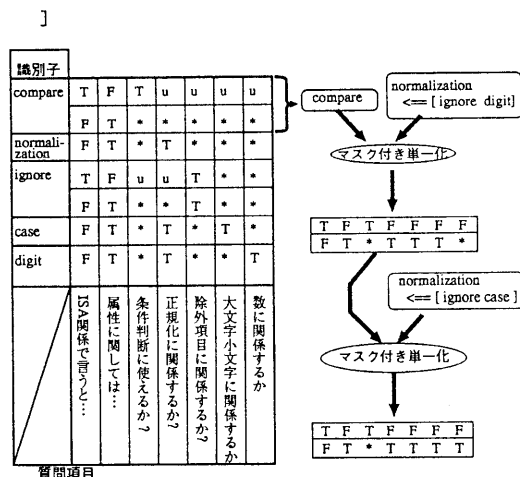


図 7: 概念の具体性の調整

3.5 概念制約式を用いる言語処理系の全体像

概念制約式を用いて抽象的に記述された部品の集まりから、実行可能なプログラムを作り出す方法は未完成である。そこで、その大まかな全体像を述べるに止める。図 8は、概念制約式を用いる言語において、実行可能なプログラムを得るために、言語処理系が行うべき作業のフローチャートである。

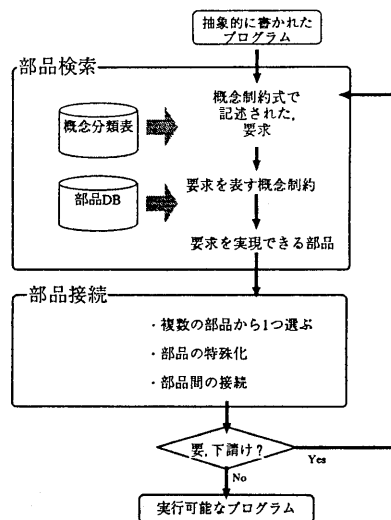


図 8: 概念制約式を用いる言語処理系の全体像

処理系は、(1. 部品検索フェーズ): 抽象的に (細部を

省略して) 書かれたプログラムをプログラマから受け取ると、プログラムの中の概念制約式を取り出し、これを概念分類表を参照しながら概念制約へと変換する。次にこの概念制約で表された要求にマッチする能力を持った部品を部品データベースから探し出す。(2. 部品接続フェーズ): 検索フェーズにより得られた部品群を利用状況に合わせて特殊化し、相互に接続⁵する。一つの要求に対して複数の部品が見付かった場合の処理も、この段階で行う必要が有る。

以上の処理 (1,2) を巾優先戦略に基づき繰り返し、実行可能なプログラムを出力する。

4 おわりに

システムの構成要素となる部品を、特定の部品に固定せずに、部品の集合として『柔らかく』参照する事で、状況に応じて入れ換える事を可能にするという問題に関して考察を行った。その結果に基づき、プログラミング言語の基本要素のレベルで利用可能な柔軟な参照の仕組みを提案した。

今後の課題としては、今回扱わなかった部品の内部の詳細な姿を決定する事、および、部品どうしを組み合わせる事で実行可能な一つのプログラムを得る方法を完成させる事が挙げられる。

参考文献

- [1] 白鳥則郎, 菅原研次: やわらかいネットワークの開発に向けて — 知識型設計方法論 —, 電子情報通信学会 技術研究報告, AI93-46, 1993.
- [2] 蓬萊 尚幸: モジュール間協調に基づく軟らかい形式的仕様記述, 第 47 回 情報処理学会 全国大会論文集, 第 5 分冊, pp25-26, 1993.
- [3] Bertrand Meyer: Object-Oriented Software Construction. Hertfordshire, England: Prentice Hall International, 1988.
- [4] 小林, 有田, 川口, 曾和: 概念制約式を用いたプログラミングとプログラム合成, 第 47 回 情報処理学会 全国大会論文集, 第 5 分冊, pp19-20, 1993.
- [5] 小林弘明, 岡本秀輔, 曾和将容: 部品データベースへの問い合わせをアルゴリズム記述要素として用いるプログラミング手法についての一考察, 第 50 回 情報処理学会 全国大会論文集, 第 5 分冊, pp235-236 (1994).

⁵ 引数の受渡し等の解決を行う