3R-8

オブジェクト指向言語における名前づけとタイプ

青柳龍也・河野真治・田中英彦
(東大工学部)

1. NameMasterのねらい

我々は、並列オブジェクト指向計算機システムORAGA[1]の名前管理、モジュール管理サブシステムNameMasterの開発を進めている。NameMasterは、プログラム中に現れる名前を管理し、それにより、プログラムの生産性、直接的には、プログラムの再利用性の向上を実現する。

プログラムの再利用性を向上させるためには、二つのことを実現しなければならない。一つはプログラムへの到達可能性の向上であり、一つは、プログラムの読みやすさの向上である。

到達可能性の向上には、モジュールの名前付けが重要である。ライブラリ中にあるモジュールを再利用するためには、まず、そのモジュールに到達しなければならない。人間は名前をたよりにモジュールを探し出す。

読みやすさの向上にも、名前付けが重要である。ORAGAシステムでは、プログラムを「アイデアを記述した文書」としてとらえている。文書は読めなければならない。モジュールをブラックボックスとして扱うのではなく、中身の読めるホワイトボックスとして扱う。プログラムが読めるかどうかは、プログラム中の名前付けに依存する。

従来、名前付けに関しては、精神論が説かれるだけであった。NameMasterのねらいは、重要な名前付けに関して、明確な理論を構築し、計算機で支援することである。

2. 名前付けの方法

オブジェクト指向言語のプログラムに出現する名前には、クラス名、メソッド名、変数名等がある。今回は変数名の名前付けについて議論する。

変数名は、次の二つのことを表現するように付けなければならない。

1)変数が保持するオブジェクトのクラス

2)変数がプログラムのモデルの中で果たす役割

例えば、収入と支出を記録するFinancialHistory[2]というプログラムを考えてみよう。このプログラムのモデルは出納帳である。出納帳には、収入を記録する欄(incomes)と支出を記録する欄(expenditures)がある。それぞれに対応する変数が必要となる。

一方、それらの変数として適当なクラスをライブラリの中から探すと、Dictionaryというクラスが見つかり、それを再利用することにする。

このようにして、変数は二つの側面から特徴付けられる。一つは、変数に保持され再利用されるクラス(Dictionary)としての特徴であり、一つは、そのプログラムのモデルの中で果たす役割(incomes,expenditures)である。

言いかえれば、変数は、Dictionaryという抽象的なクラスをincomes,expenditures向きに具体化したものである(実際、Dictionaryの各項目には金額を表わす整数のみが入ることになる)とともに、incomes,expendituresというものを、Dictionaryで実現したものである。(図1)

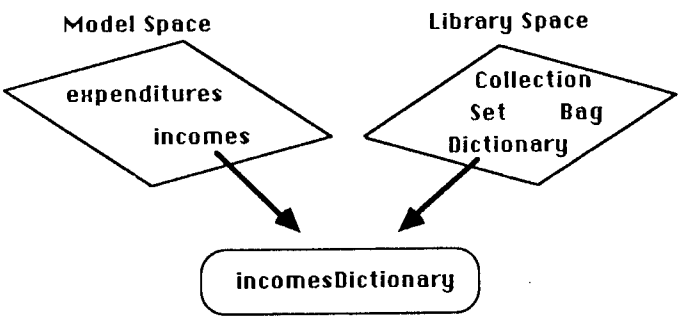

図1 変数名の名前付け

実際の変数名は、二つの特徴を表わす単語を並べて構成される。上の例では、incomesDictionary 等の名前となる。

名前が表わすべき二つの特徴の内、2)の特徴は個々のプログラムのモデルにより異なるので、今回は扱わない。一方、1)の特徴はタイプ推論を応用することでどんなプログラムでも統一的に扱うことができる。以下、その方法について述べる。

3. オブジェクト指向言語のタイプ推論

関数型言語や論理型言語に関するタイプ推論の研究は数多く存在する[3,4]が、オブジェクト指向言語に関するものはあまりない[5]。

ここでは、ORAGA システムのプログラミング言語DinnerBell[6]のタイプ推論を説明する。DinnerBellは並列オブジェクト指向言語であるが、並列言語かどうかはタイプ推論には直接関係はない。

Smalltalk と同じく、DinnerBellも言語仕様としてはタイプのない言語である。そこで、まず、タイプを定義しなければならない。

Names and Types in Object Oriented Language
Tatsuya AOYAGI, Sinji KONO and Hidehiko TANAKA
Univ. of TOKYO

ここでは、タイプを次のように定義する。

すべてのクラスの集合をCとして、

1. Cの任意の部分集合はタイプである。
2. タイプ変数はタイプである。タイプ変数にはCの部分集合のドメインが付随している。
3. a，bがタイプのとき、a×b，a←bもタイプである。

DinnerBellのメッセージ送信式は、新しい変数を導入することにより、すべて次の形に変換できる。

C←R msg: A

ここで、msg:がメッセージセレクタ、Aが引数、Rがメッセージを受け取る変数（オブジェクト）、Cが返ってきた値を受け取る変数（存在しないこともある）である。引数がない場合は、：のかわりに！が後置される。

例えば、次の二つのメッセージ送信式を考えよう。

Y←X p! ①

X q: Z ②

タイプ推論の仮定として、メッセージ送信式中のメソッドのタイプはすべてわかっているとする。この例では、p!とq:について、

p! a←a a：{A，B} ③

q: {A}×{C} ④

を仮定する。ここで、aはタイプ変数で、そのドメインはクラスAとクラスBである。

タイプ推論はドメインつきの単一化[7]によって行う。ドメインつきの単一化は、単一化される二つの変数のドメインの共通部分を単一化された変数の新しいドメインとする単一化である。この例では、まず、①と③の単一化により、変数XとYが、同じタイプ{A，B}を持つことがわかる。さらに、②と④の単一化により、Xのタイプは{A，B}と{A}の共通部分{A}となる。同時に、Yのタイプも{A}となり、結局、X，Y，Zのタイプはそれぞれ、{A}，{A}，{C}となる。

単一化に失敗したときは、タイプエラーを起こしていることになる。

4．タイプ推論に基づく名前付け

一般にプログラマは、個々の変数に保持されるオブジェクトのクラスを一意に想定しながらプログラムを組む。

オブジェクト指向言語におけるタイプのポリモルフィズムはインヘリタンスによってのみ実現される。すなわち、ある変数にいくつかのクラスのオブジェクトが保持されるとすれば、それらすべてのクラス間にはインヘリタンスの関係がなければならない。

変数の名前にはそれらのクラスの中で最も抽象的なクラスの名前を付ければ良い。

タイプ推論により変数のタイプ、具体的にはクラスの集合が求まる。タイプ推論に基づく名前付けは、次のようにして行う。

---

1. 得られたタイプの集合をインヘリタンスに関して同値類に分ける。代表元として最も抽象的なクラスを選ぶ。
2. プログラマの想定するクラスが、どの代表元のサブクラスにもなっていないときはタイプエラー。
3. 得られた代表元が一つで、プログラマの想定するクラスと一致すれば、そのクラス名から変数名を作る。
4. 一致しないときは、プログラマの判断により次のいずれかを選ぶ。
   - 4.1 タイプ推論が充分タイプを絞り込めなかったとして、プログラマの想定するクラスから変数名を作る。
   - 4.2 プログラマの想定するクラスが必要以上に具体的であったとして、代表元となっているクラスのクラス名から変数名を作る。
   - 4.3 本来あるべき抽象クラスが存在しなかったとして、代表元のいくつかをサブクラスとする抽象クラスを作り、そのクラス名から変数名を作る。

4.1 は例えば、プログラマの想定するクラスがIntegerのときに、十分な情報が存在しなかったためにタイプ推論の結果が{Object}となったようなときである。4.2 は、Integer という想定に対して、{Number}という推論がなされそのほうが適当だと判断されたような場合である。4.3 は、Integer に対して、{Integer,float} という推論がなされ、{Number}という抽象クラスが必要であったと判断される場合である。

＜参考文献＞

[1]神田、”並列オブジェクト指向計算機システムORAGA ”、東京大学情報工学博士論文、1986

[2]A.Goldberg and D.Robson、"Smalltalk-80 The Language and its Implementation"、1983

[3]R.Milner 、"A Theory of Type Polymorphism in Programming"、JCSC17,348-375、1978

[4]P.Mishra 、"Towards a theory of types in Prolog" 、Proc.IEEE Internat.Symp.Logic Programming 、1984

[5]N.Suzuki 、"Inferring Types in Smalltalk"、Proc.ACM 8th POPL 、1981

[6] 河野他、”並列オブジェクト指向システムORAGA-単一代入則に基づくオブジェクト指向プログラミング”、情報処理学会第３３回全国大会、5D-2、1986

[7] 川村他、”CS-Prolog:拡張単一化に基づくCONSTRAINT SOLVER”、ICOT Logic Programming Conference 、1987