永続的データを安全に扱うための階層的な型管理系

7R−2

電気通信大学 情報工学科
鈴木 卓治・小林 光夫

1．はじめに

計算機で処理されるデータは，巨大化，複雑化の一途をたどっている．これを安全かつ便利に扱うための研究として，データベースなどに代表される永続的データ(persistent data)に対し，任意のデータ型を付加して扱えるようにするという永続的データ管理技術の研究が，現在盛んに進められている[1]．永続的データの型情報の厳密な管理は，複数の人間による大規模なソフトウェア開発の現場では必須となろう．

小論では，データ型がどのようにして構成されるのかを表わす型情報の依存関係と，データ型の管理の階層的な構造を対応させ，型情報を安全かつ便利に扱う方式を提案する．また，この方式を，永続的データを管理するためのやわ物部品群[2,3]の型管理系に適用し，実現をすすめている．これについても報告する．

2．型管理系の設計方針

2.1 一括管理方式の採用

型情報の管理の方式には，大きく分けて，型情報をデータと切り放して一ヶ所で管理する一括管理方式と，数カ所で分散して管理する分散管理方式の二つがある．

一括管理方式では，データの格納効率がよく，正確な型検査が比較的容易で検査の手間も軽減できるが，データの可搬性が悪くなる．逆に，分割管理方式では，データの可搬性はよくなるが格納効率は悪く，また型の一致を正確に検査することが困難になる．

文献[2,3]では，データの可搬性を考慮して，PS-Algolと同様に，複数の記憶領域が利用できるようになっている．この場合，PS-Algolのように分散管理方式を採用することが望ましい[4]と思われるが，われわれは，特に型一致判定の困難さ(部品群を組み込む対象となる言語の，型の扱いの貧弱さによる)という点から，一括管理方式を採用した．永続的データを型管理系の管理範囲外へ持ち出すときは，データを操作するプログラムの側に型情報を付随させ，プログラムの改変を禁じるといった運用を行なうことにより，可搬性の問題を解決する．

Hierarchical and Safe Type Management System for Persistent Data
SUZUKI Takuzi, KOBAYASI Mituo
University of Electro-Communications

2.2 型情報の依存関係

データ型は一般に，すでに定義された型を利用して構成される．これにより，型情報の依存関係を定義することができる．たとえば，次の(Pascal語の)型 X はすでに定義された型 Y を利用しているので，X は Y に依存していることになる．

```
type X = record ...; x:Y; ... end;
```

図2.1の実線部分は，型情報とその間の依存関係を有向グラフで表わしたものである(破線部分は後述)．このグラフを $T=\langle V_T, E_T \rangle$ とする．(V_Tは節の集合を，E_Tは有向辺の集合を表わす．) 節は型情報を表わし，節を結んでいる有向辺(矢線)は依存関係を表わす．たとえば図2.1は，型情報 i は c,e を利用しており，e は a を利用していることを表わす．

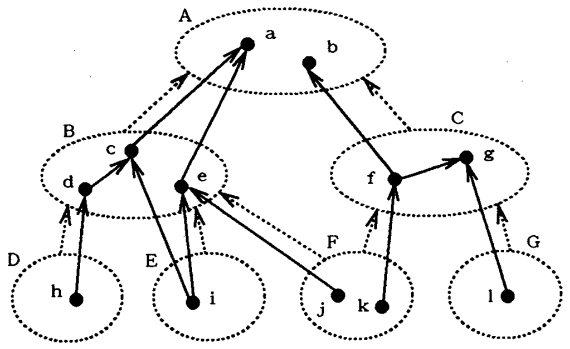

図2.1 型情報と型管理域との関係

なお，グラフ T は閉路を含むことがある．たとえば，次の(Pascal語の)型 Z と型 ZP の依存関係は，図2.2の閉路で表わされる．

```
type ZP = ^Z;
type Z = record ...; n:ZP; ... end;
```

図2.2 閉路の例

2.3 型管理域と型情報の公開範囲

相互に依存関係のあるデータ型群を，何人かの人間で作成したり利用したりする場面を考えよう．

ある利用者が作成したデータ型の情報を，いきなり他の利用者に公開する(利用可能とする)のは危険であり，テストや虫とりといった段階を経て，初めて公開できる．また，いったん公開された型の情報がたびたび変更されるようでは困る．

このことは，永続的データの型管理系を考える場合，型情報の公開範囲を考慮する必要があることを示している．

ここでは，型情報の管理単位として“型管理域”(型情報の集合，以下管理域と略記)というものを考え，型情報の公開範囲を，管理域間の階層関係によって定めよう．

図2.1の破線部分は，管理域と，管理域で管理されている型情報の公開範囲の包含関係を有効グラフで表わしたものである．このグラフを $M=<V_M, E_M>$ とする．破線の楕円は管理域を表わしており，管理域の間に引かれた破線の矢線は公開範囲の包含関係を表わしている．いま，管理域が型情報を管理しているという関係を関数 $Manage: V_M \rightarrow \mathcal{P}(V_T)$ ($\mathcal{P}(V_T)$はV_Tのべき集合を表わす)およびその逆関数 $Member: V_T \rightarrow V_M$ で表わすことにすると，たとえば図2.1は，Manage(B)={c,d,e}，すなわち管理域 B は c,d,e の３つの型情報を管理していることを表わしている．また管理域 D から利用できる型情報は，管理域 D,B,A で管理されている型情報 a,c,d,e,h (つまり Manage(D)∪Manage(B)∪Manage(A)の要素)であることを表わしている．

管理域間の階層関係はおおむね木構造になる(一般にはDAG)であろう．

2.4 型情報の依存関係と公開範囲の整合性の管理

われわれの方式では，次の８つの操作を組み合わせて，型の管理を行なう：管理域の作成，削除，公開範囲の設定，取消し，型情報の追加，変更，削除，管理域変更．

ここで，管理域変更について簡単に論じておこう．たとえば，図2.1で型情報 i の管理を管理域 E から B に変更すると，i は B,D,E の範囲で公開されることになる．これにより，型情報をより広い範囲に公開することができ，しかも自分の作成した型の情報が使えなくなる，ということがなくなる．ただし，このとき型情報の依存関係と公開範囲の間に矛盾が生じてはならない．たとえば，型情報 g の管理を管理域 C から G に変更したとすると，G の利用者が g の内容を改変するたび，それを利用している f および k の意味が代わる可能性があり，g を安心して利用できない．また j の管理を F から C へ変更した場合も，j が利用している e の変化が，C から把握できないため，やはり j を安心して利用できない．

このような矛盾があるかどうかは，グラフ T および M から容易に検査することができる．すなわち

「$\forall a,b \in V_T$ について，a から b への有向経路が存在するとき，Member(a) から Member(b) への有向経路が存在する」

という簡単な条件を調べればよい．

3．型管理系の実現

われわれは現在，永続的データを管理するための部品群をPascal語の環境に対して開発している[3]．そこでは型管理系は部品群の一つとして提供される．型管理系は，一つのファイルに納められた定数，型，変数，手続きおよび関数の集まりを，一つの型情報とみなして管理を行なう．型情報の登録時には，情報のうち外部から参照できる名前について，管理系内での一意性を検査する．これにより，型情報の解釈が一意に定まり，利用者は安心して型情報を利用することができる．

この実現方式は，Pascal語だけでなく，C語やC++語(部品群は当初C++語用として作成された[2])などの手続き型言語に広く適用できるが，Pascal語のようにモジュール化の機能の弱い言語では特に有効に働き，抽象データ型の実現をやりやすくし，便利に利用できる環境を提供してくれる．

4．おわりに

データ型の依存関係と型管理の階層的な構造との対応に注目した型管理法を提案し，永続的データの型管理系に適用できることを示した．この方式の特徴は，以下の通りである．

- 型情報を一括して管理する．
- 型情報の依存関係を考慮している．
- 型情報の管理単位として型管理域を考え，型情報の公開範囲を，管理域間の階層関係によって定めている．
- 型情報や管理域のもつ関係をグラフとしてとらえることにより，管理の過程にしたがって発生するさまざまな矛盾をグラフの構造から発見し，解決している．

型情報の依存関係と管理域により定まる型情報の公開範囲とが作るこのような構造は，互いに依存関係をもつ対象が，階層的に管理される場合に，広く一般に見られるものである．したがって，この手法は一般的なプログラムモジュールの管理・保守に対しても応用できるであろう．これの問題については機会を改めて報告したい．

参考文献

[1] Hull,R., Morrison,R., Stemple,D.(eds.): Proceeding of the 2nd International Workshop on Database Programming Language, Morgan-Kaufmann, 1990.

[2] 鈴木卓治，小林光夫：永続的データを管理するためのやわ物部品群，電気通信大学紀要，Vol.3, No.1, pp.49-58 (1990).

[3] 鈴木卓治，小林光夫：複数の記憶領域を利用するときのデータ型と永続性の直交化，電気通信大学紀要，Vol.3, No.2(1990)(掲載予定).

[4] Atkinson,M., Buneman,P., Morrison,R.: Binding and Type Checking in Database Programming Languages, The Computer Journal, Vol.31, No.2, pp.99-109(1988).