データベース・システム　79－1
プログラミング言語　26－1
（1990．　9．　7）

新しいソフトウェアデータベースを目指して

冨永 和人　　徳田 雄洋

東京工業大学 工学部 情報工学科

プログラムの内容についての質問に答えることはソフトウェアデータベースにとって重要な機能である。その際にプログラムの持つ情報をどのような形で格納するかは問題である。本論文では、元のプログラムの情報を失うことなく、問合せに対する答を効率よく得るために、問合せのためにプログラムを一度別の表現に変換する方法を提案する。この表現 — プログラム問合せ中間表現 — は、対象となるプログラミング言語の構文規則に問合せを行いたい情報を計算する意味規則を追加した属性文法によって記述される。そのプログラミング言語で書かれたプログラムに対してこの文法で作られる属性つきの木が、そのプログラムの問合せ中間表現である。

Towards New Software Databases

Kazuto Tominaga and Takehiro Tokuda

Department of Computer Science, Tokyo Institute of Technology

One of the most important facilities of software databases is software inquiry. A software inquiry facility allows users to ask questions about the programs stored in the database. This paper proposes a new approach based on intermediate forms. The intermediate form is an attributed tree suitable for efficient software inquiry. The attributed tree has sufficient information to represent the original source text as well as necessary information to process designated types of queries from the users. The intermediate form of a given program is produced from the description of program syntax and query semantics written in an attribute grammar.

1 はじめに

ハードウェアの性能が向上するにつれてソフトウェアに対する要求が大規模になりつつある。そのため、このようなソフトウェアの開発は複数の人間で行われる傾向にある。さらに、既に存在するソフトウェアの維持及び再利用を行うために必要な作業の量は、ソフトウェアが蓄積されればされるだけ大きくなる。現在に至るまでに開発された、またこれからも開発される多くのソフトウェアの維持と再利用を合理的に扱える範囲とするためには、個々の維持や再利用の効率を上げる必要がある。

これらの問題を解決しようとする研究の中に、ソフトウェアデータベースに関する研究がある。ソフトウェアデータベースとはソフトウェアのあらゆる活動を支援するデータベースである。すなわち、プログラムやデータ、ドキュメントといったソフトウェアに関するさまざまな情報を格納し、それらを利用する手段を提供するものである。

ソフトウェアデータベースに求められる機能としては、情報の出し入れ（変更や追加・削除、再利用）や情報に対する検索・質問に対しての回答、バージョンやコンフィギュレーションの管理などが挙げられる。また、ソフトウェアの開発・維持を行う環境としての機能には、複数人で同時に利用することを考えたトランザクションの制御やユーザの権限に関する情報へのアクセスの制御、また、通常のデータベース管理システムと同様にデータの内部一貫性の保持、障害からの回復、情報の共有などの能力を持っていることが望ましい。

本論文では、以上に挙げたさまざまな機能のうちでも重要であると思われる、プログラムに関する照会を行う機能に焦点を当て、プログラムについての問合せに答えるのに適したソフトウェアデータベースのための中間表現を提案する。

はじめに、上に挙げたようなさまざまな機能を実現するソフトウェアデータベースへのアプローチとして行われているファイルシステム、従来のデータベース、ハイパーテキスト、オブジェクトベースの4つの現状を、実際のシステムをいくつか取り上げて概観し、次にそれらのシステムについてプログラムに関する照会を行う機能を比較する。続いて本論文での提案『プログラム問合せ中間表現』について述べ、これを用いた簡単な例を示し、最後にこの方法の評価と将来への展望について述べる。

2 ソフトウェアデータベースの現状

ソフトウェアの活動に関わる情報を蓄えておく方法としては、現在いくつかのアプローチがある。利用するものによってこれらを4つに大きく分類することができる。すなわち、既存のファイルシステムを利用する方法、既存のデータベースを用いる方法、ハイパーテキストによる方法、オブジェクトベースを利用する方法である。

2.1 ファイルシステムを利用する方法

ファイルシステムのファイルやディレクトリ構造を用いて情報を蓄える方法には、代表的なものにUNIXにおけるRCS[1]やSCCS[2]がある。これらはいずれもテキストファイルを扱うバージョン管理システムであり、ファイルを情報の基本単位として扱う。この際に全てのバージョンをそのまま蓄えるのではなく、隣合ったバージョン間の差分を保存することによって記憶領域を

節約していることが特徴である。

これらのシステムが扱うファイルにはテキストファイルであるという以外には何の制限もないが、ファイルの内容に応じた繊細な処理を行うことはできない。すなわちファイルの内容がプログラムであろうとドキュメントであろうと同様に扱ってしまう。

2.2 従来のデータベースを用いる方法

ソフトウェアに関する情報には、関係として見なし得るものがある。例えば、手続き呼び出しにおける呼び出し部分と手続きの関係やプログラム全体を構成する部分モジュール同士の関係などである。これらを関係データモデルを用いたデータベースシステムで処理するというアプローチがいくつかある。

Dittrich と Lorie は、複数個の部分プログラムから構成される大きなプログラムについて、部分プログラムのバージョンの更新によって生じる異なったコンフィギュレーションの情報を関係データベースに表現する方法を示している [3] 。ただし、ここで扱われる情報はそれぞれのプログラムを識別するために与えた識別子とその組合せに関するものであり、プログラムそれ自身ではない。従って、プログラムの内容に関わる処理には向かない。

これに対して、Chen らによる CIA[4] は言語 C のプログラムを構成するソースファイル群から、型・外部 (global) 変数・関数・マクロとそれらを含むファイルという情報を抽出して関係データベースに蓄える。その結果、そのデータベース上の問合せ言語を用いることにより、これらの情報の範囲でプログラムの内容に関する質問を行うことができる。

SQUDER[5] では同じく言語 C を対象とし、変数の定義と使用の関係、バージョン情報、モジュールの包含関係などのプログラムに関する情報を E-R(Entity-Relationship) データモデルでとらえ、それらの情報をプログラムから取り出し、その情報について質問をするという方法を提案している。

2.3 ハイパーテキストを用いる方法

通常のテキストは基本的に始まりと終わりがある一次元の構造を持っている。このためこれを読む時には普通は最初から最後まで順番に（あるいは前後しながら）読むことになる。このテキストを意味のある断片に分割し（この断片をノードという）、断片間に参照のための有向リンクを持たせると、これを読む時にはもはやはじめから終わりまで順序よく読む必要はなく、必要な時にリンクをたどって離れたノードの情報を参照することができる。テキストにこのような構造を持たせようという考えがハイパーテキストの概念である。

本来ハイパーテキストは人間が読むためのものを扱う。これはソフトウェアの活動においては仕様書や設計書、マニュアルといったものにあたる。DIF[6] はソフトウェアの開発や維持に関わるこれらのドキュメントを管理するためのハイパーテキストシステムであるが、通常のドキュメント以外にもソースプログラムやオブジェクトプログラム、テスト情報や図形などといったものをもノードに格納することができる。

2.4 オブジェクトベースを用いる方法

オブジェクト指向のデータモデルを持つオブジェクト指向データベース（あるいはオブジェクトベース）を用いてソフトウェア開発環境での情報を扱おうという動きもある（オブジェクトベースの機能を提供するシステムとしてはCactis[7] やWorlds[8] などがある）。

Marvel([9],[10]) はオブジェクトベースを用いてソフトウェアの開発環境を提供するシステムである。プログラム、モジュール、型、変数、関数といったクラスを持ち、プログラム全体はモジュールの集まり、モジュールはそこに含まれる型や変数、関数に対応している。それぞれのクラスには属性として名前や型など、そしてプログラムやモジュールにはそれ自身の実行形式を属性として持っており、これらの情報をオブジェクトベースを用いて管理している。このシステムのもう一つの特徴は、ソフトウェア開発の知的な支援をルールを記述することによって行うという枠組を与えていることである。例えばこれを用いると、モジュールの変更などによって既に生成されているプログラムが最新のものでなくなってしまったような場合に自動的に新しいプログラムをコンパイルして生成するような環境を設定することができる。

3 プログラム問合せ

ソフトウェアの開発・維持において扱われる情報の中で特に複雑な構造を持ったものはプログラムである。しかしながら、これらの情報のうちで我々がもっとも主要なものとして取り扱うものもプログラムである。従って、プログラムに関して何らかの照会を行うことができれば、それは扱うプログラムに対する我々の理解を助けてくれるであろう。その点でソフトウェアデータベースにおいてプログラムに対する問合せに答える機能は重要である。ここでは前節で見たいくつかのアプローチにおいて、それぞれがプログラムに関する問合せにどのように答えることができるかを見る。

はじめに、ファイルシステムを用いたアプローチでは、プログラムは単なるテキストとして扱われる。従ってそれに対する問合せは通常のテキストに対しての問合せと同様になり、「何行目の何文字目は何という文字であるか」「XXXXX という文字列は何行目に現れるか」といった類の質問か、それらの組合せになると考えられ、プログラムの構造を意識するほど高いレベルの問合せに答えるのは難しいであろう。

次に従来のデータベースによる方法では、CIA 及びSQUDER の例を見るとある程度プログラムの内容に踏み込んだ情報を得ており、その内容についての質問であればそれに対して答えることはできる。しかし、いずれの場合も対象となるプログラムからあらかじめ必要と思われる情報を抽出しておくという方法であるので、それ以外の情報についての質問に対しては全く無力である。

ハイパーテキストはデータを持ったノードが有向リンクで結合されているわけであるから、これを用いると任意の有向グラフが構成できる。さらにそのハイパーテキストの構造の中で、ノードに持たせるデータの形やその操作をソフトウェアデータベース向けに自由に決めることができるので、有向グラフで自然に表せるようなプログラムの情報には適しているかもしれない。

オブジェクトベースでの処理は、基本的には操作する対象のオブジェクトのメソッドを起動することによって行う。従って（オブジェクトベース上での問合せ言語がないとすると）問合せに答える処理もメソッドとして記述することになり、ソフトウェアデータベースを設計するときには問合せを受けるであろう全てのオブジェクトに問合せに答えるためのメソッドを持たせなけれ

ばならない。

また、オブジェクトベースを用いてソフトウェアデータベースを構築する時には、何をオブジェクトと見なすかが問題になる。Marvel では大はプログラムやモジュールから小は変数や関数までがオブジェクトである。Cactis[7] では、プログラムを構成するモジュールひとつひとつをオブジェクトとして扱った場合と、構文木に現れる記号をオブジェクトとして扱った場合という、プログラムに関する情報の細かさについて大小両方のデータが扱えることが示してある。つまり、ソフトウェアデータベースを設計する時には何をオブジェクトとして選ぶのかを決定しなければならない。

すなわち、オブジェクトベースは柔軟で何でも記述できるがゆえにソフトウェアデータベースの設計を行うに際して何の枠組も与えないといえる。ハイパーテキストについても、ノードやリンクが自由に設定できノードに与えるデータを自由に決められるという点で同様である。

以上より、次のように考えられる。

- プログラムを扱い、それについての問合せに答える機能はソフトウェアデータベースにとって重要である
- 元の情報が失われていない形でプログラムを持つことが望ましい
- 効率的かつ自然にプログラムを扱うことができる何らかの枠組を示すことはソフトウェアデータベースの設計を楽にする

これらを踏まえ、プログラムをソフトウェアデータベースに格納する表現方法を提案する。

4 プログラム問合せ中間表現

まず、情報が失われない形でプログラムを格納するという点について考えてみる。もちろんプログラムをテキストのまま格納してしまえば情報は失われないが、問合せに対して答えるときにはそのテキスト上で何らかの処理を行わなければならない。これは効率が悪い。

ここで対象となるプログラミング言語が文脈自由文法で記述できるとする。すると、その言語で書かれたプログラムの構文は木をなす。あるプログラムのテキストによる表現よりも、構文木による表現の方がそのプログラムに対する構文に関する質問に容易に答えられる。構文木をテキストに戻すことで元のプログラムを得ることができるので、構文木にすることによってもともとのプログラムの情報は失われない。

さらに、この構文木の節に現れるシンボルそれぞれに意味の情報を与えることによって、プログラムの意味に関する問合せにも答えられるようにする。

こうしてできた問合せに答えるための木を『プログラム問合せ中間表現』と呼ぶことにする。

プログラム問合せ中間表現は、対象となるプログラミング言語の構文規則に従った木であって、その木の節の各シンボルに意味の情報として属性がついているわけであるから、このプログラム問合せ中間表現を作るには属性文法が適している。すなわち、扱うプログラミング言語の構文規則に、問合せ用の属性を各シンボルに与えるための意味規則を付け加えると、それがプログラム問合せ中間表現を構文木として生成するための文法規則となる。

プログラムに対する問合せに対しては、このプログラム問合せ中間表現の上で計算を行って答を求める。この表現が構文を反映しているので構文についての問合せに対して構文解析を行う必

要がなく効率がよい。意味に関する質問に対しては、計算された属性の値が答となる場合はそれを答とすればよく、直接どれかの属性値が答とならない場合でも、既に得られている木と属性値を用いて答を得ればよい（なぜならプログラム問合せ中間表現は元のテキストで表現されたプログラムの情報を失っていないので、元のプログラムで分かることは全てプログラム問合せ中間表現でも分かるはずだからである)。

5 プログラム問合せ中間表現の記述

属性文法を用いてプログラム問合せ中間表現を記述するには、まず、対象となるプログラミング言語の構文規則中に現れるシンボルに、問合せに答えるために必要であると思われる属性を与える。そして、それらの属性を計算する意味規則を構文規則に付け加える。

ここでは、簡単なプログラミング言語の構文規則に問合せのための意味規則を与えた属性文法を用いて、そのプログラミング言語で書いた小さなプログラムに対するプログラム問合せ中間表現を構成する例の一部を示す。

対象とした言語はPascalに似た文法を持った小さなものである。変数の型がひとつしかないことや、関数を持たないこと、手続きに引数がないことなどがPascalとの大きな違いであるが、ブロックの階層構造と変数などの識別子の静的なスコープはPascalと同様である。ここで示した中間表現の記述は、変数や手続きなどの宣言と使用について問い合わせるのに適した属性や意味規則を与えたものである。左側には言語の構文規則、それに対応した問合せのための意味規則を右側に示している。なお、< > 囲まれたものは非終端記号であり、意味規則において非終端記号の後に . で区切られて続いているのは属性の名前である。

構文規則	意味規則
<prog>::=<block> .	<block>.pos=1
	<block>.env↓=*null*
	<prog>.use↑=<block>.use↑
<block>::=<decls> ; <statement>	<decls>.pos=<block>.pos·1
	<statment>.pos=<block>.pos·2
	<decls>.env↓=<block>.env↓
	<statment>.envx=<block>.env↓
	<statment>.envl=<decls>.env↑

（以下略）

この記述にしたがってプログラムから属性つきの木が構成されたときには、変数や手続きの宣言の部分の属性には、言語の持つスコープを反映してそれらが使用されている木における位置のリストが値として与えられ、変数や手続きが使用されている部分にも同様にそれらの宣言されている位置が属性として与えられる。

6 評価とまとめ

問合せのためにプログラムをソフトウェアデータベースに格納する表現形式として、属性つきの木であるプログラム問合せ中間表現を提案し、属性文法を用いることによってその中間表現を記述できることを述べた。この属性文法の記述は、構文規則の部分は対象とするプログラミング言語の構文規則であり、意味規則の部分は問合せに必要な属性についての計算規則である。従って、同じ言語に対して別々の意味規則を与えれば、ある表現は定義と使用の関係についての問合せを受けるのに適したもの、また他の表現は手続きの呼び出し関係についての問合せを受けるのに適したもの、などというように場合に応じた表現形式を与えることができる。このとき、いずれの表現も元々テキストとしてプログラムが持っていた情報を失っていないので、問合せに用いるインタフェースや問合せ言語、あるいは答えを得るための手続きによってはいろいろな照会に耐え得ると思われる。

参考文献

[1] Walter F. Tichy. RCS – A System for Version Control. *Software – Practice and Experience*, 15(7):637–654, July 1985.

[2] Marc J. Rochkind. The Source Code Control System. In *Proc. on 1st National Conf. of Software Engineering*, pages 37–43, 1975.

[3] Klaus R. Dittrich and Raymond A. Lorie. Version Support for Engineering Database Systems. *IEEE Transactions on Software Engineering*, 14(4):429–437, April 1988.

[4] Yih-Farn Chen, Michael Y. Nishimoto, and C. V. Ramamoorthy. The C Information Abstraction System. *IEEE Transactions on Software Engineering*, 16(3):325–334, March 1990.

[5] 石原博史, 徳田雄洋. E-R モデルに基づくソフトウェアデータベースの設計と実現. 電子情報通信学会研究報告, SS-89-27, February 1990.

[6] Pankaj K. Garg and Walt Scacchi. A Hypertext System to Manage Software Life Cycle Documents. In *Proceedings of the 21st Annual Hawaii International Conference on System Sciences, Vol.2*, pages 337–346, 1988.

[7] Scott E. Hudson and Roger King. The Cactis Project: Database Support for Software Environments. *IEEE Transactions on Software Engineering*, 14(6):709–719, June 1988.

[8] David S. Wile and Dennis G. Allard. Worlds: an Organizing Structure for Object Bases. In *Proceedings of the Second ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environmerits*, pages 16–25, December 1986.

[9] Gail E. Kaiser, Peter H. Feiler, and Steven S. Popovich. Intelligent Assistance for Software Development and Maintenance. *IEEE Software*, 40–49, May 1988.

[10] Gail E. Kaiser and Naser S. Barghouti. Database Support for Knowledge-Based Engineering Environment. *IEEE EXPERT*, 3(2):18–32, 1988.