BIBLIOTHEQUE DU CERIST

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

Laboratoire d'Informatique

STRUCTURE ALGÉBRIQUE, COMPILATION, ET PROGRAMMES STRUCTURÉS :

2ème Partie

Vers une technologie de construction de compilateurs ayant une base théorique

Charles RATTRAY & Teodor RUS

R.R. n° 40 Aôut 1976

BIBLIOTHEQUE DU CERIST

Une structure mathématique hétérogène est suggérée comme pouvant servir de base et de soutien à une technologie de construction des compilateurs.

Les possibilités de définition et d'application de cet outil de soutien et son emploi comme dispositif de spécification de systèmes, sont illustrés en décrivant le chemin qui, du langage de description, conduit au langage d'implémentation, en passant par un langage d'analyse.

°

Chacun de ces rapports de recherche est un chapitre d'un ouvrage plus important sur les sujets suivants : basic mathematical theory, programming language syntax and semantics, theory of compiling, virtual machine structure, structured programs and structured programming, structure and construction of a practical compiler writing system.

°

BIBLIOTHEQUE DU CERIST

Towards a Software Construction Technology having a Theoretical Basis

A heterogeneous mathematical structure is suggested as a basis, a scaffolding, for a compiler construction technology. The definitional capacity and applicability of the scaffolding, and its use as a system specification device, are illustrated by tracing the path from language description, through language analysis, to language implementation.

For presentation to the EUROCOMP Conference on Software Engineering, ONLINE, London, 14 - 16 September 1976.

1.0. INTRODUCTION

1.0.1. General problems

Two areas worth investigating, in the study of software, can be identified :

(1) development of a theoretical basis for software, i.e. a means for studying and understanding computer programs;

(2) development of a methodology for the construction of software, i.e. a means for defining a problem and for the construction of the corresponding program such that :

(a) the program reflects problem structure while emphasizing program maintainability, transportability, and generality;

(b) the program is "correct".

A third area exists, the so-called "transfer of technology" area, which is essentially the educational one of bringing the discoveries of (1) and (2) to the practitioners of computing.

In a wider context the major problem with computing systems is effective communication with the system and within the system. To be able to discuss, in a unified way, such a diversity of devices as constitute a computing system, each object can be considered an expression in some language but such that the expression itself is an operation over some sets of objects.

1.0.2. Total language system

We characterize a language system, LS, by a triple of subsystems

$$LS = (Syn, Sem, Phy; ER),$$

where

BIBLIOTHEQUE DU CERIST

Syn = (BLSyn, SSyn; TR) | BLSyn ≡ Base-level Syntax, SSyn = Surface Syntax;

Sem = (BLSem, SSem; TR) | BLSem ≡ Base-level Semantics, SSem = Surface Semantics;

Phy = (BLPhy, SPhy; TR) | BLPhy ≡ Base-level Physical Representation

SPhy ≡ Surface Physical Representation

and

TR ≡ Transformational Rules,

ER ≡ Expression Rules.

LS, itself, is a subsystem of a larger system involving extra-linguistic references and the "real world".

Syntactic, semantic, and physical representations are regarded as having their own autonomous structures and their own conditions of well-formedness; the transformation rules lead from the "base-level" to "surface" structure in each representation, with the relation between representations defined by expression rules. This has the effect of distributing language complexity throughout the various aspects of the language system, and suggests a pleasing symmetry in the overall language structure.

The strength of the integrated view is that it allows the abstraction of a common structure from language syntax, semantics, and physical realization, so permitting the transfer to semantics, for instance, of techniques of analysis which have proved successful with other aspects of language. The grammarian, the semanticist, and equally important, the implementor - separately - have introduced and developed tools which seem to be "natural" within their fields of discourse. What we now seek is some device, a scaffolding, with the aid of which each of the three can construct the descriptions, or systems, that they want; or which can be superimposed upon their existing descriptions, or systems,

BIBLIOTHEQUE DU CERIST

so as to permit better understanding of those systems. It is this structuring, descriptive device, of wide applicability, which is the "natural" structure for programming languages, and which forms a theoretical basis for a software construction technology.

Every act of computing system communication, externally or internally, involves "translation". The schematic model of translation is one in which a message from a source language passes into a receptor (target) language via a transformational process. Exactly the same model - this is rarely stressed - operates within a single language. Within or between languages, communication equals translation. A study of translation is a study of language. Compiling is the most obvious case of translation, while program execution, involving the translation of the compiled program into a language of internal machine actions, is another.

1.0.3. Aims

The purpose of this paper is to display the scaffolding device, by formally defining it and showing its naturalness in terms of the abovementioned general language model, and to illustrate the philosophy hinted at above by describing the design of a compiler-generating system, based directly on the developed theory of such a device. A practical version of this system is currently being implemented by the author; a similar system has been implemented by the co-author, Teodor Rus, I.T.C., Cluj, Rumania.

On a basic homogeneous structure Σ (called a "scheme") is constructed a heterogeneous structure $\underline{A}$ (called a "Σ-model") such that objects, operations, relations of $\underline{A}$ are realizations of those

BIBLIOTHEQUE DU CERIST

of Σ. $\underline{A}$ is then the basic unit of the scaffolding device; scaffold extension is defined in terms "translation" between basic units. Section 1.1 finishes with some hints on how the scaffolding relates to syntax, semantics, compiling, and a compiler. The aim of this section is to use the scaffolding in a definitional capacity.

Any language L, with a Σ-model structure, has the following properties :

(i) it can be decomposed into a hierarchy of levels $L_0, L_1, \ldots, L_m$, m determined by Σ, such that $L_0 \subseteq L_1 \subseteq \ldots \subseteq L_m = L$;

(ii) L_0 is a finite set, given by Σ, of constants of the model;

(iii) for $i = 1, 2, \ldots, m-1$, L_i is generated by L_{i-1} and freely generated by L_0.

Section 1.2 finishes by utilizing the language decomposition to develop a cascade of syntax analysis algorithms to be applied in an order determined by the decomposition. Ambiguity no longer presents problems. The aim of this section is to show the applicability of the scaffolding.

The definitional capacity and applicability of the scaffolding suggest a construction for a compiler-generating system; the resulting system ensures that the compiler produced satisfies (2a) and (2b). This has been achieved by abstracting the "compiler algorithm". Section 1.3 describes something of the system's design and performance.

Most systems do not automate the construction of compilers in general but deal with the generation of compilers for particular classes of languages or particular classes of machine operating in a certain kind of environment. In principle, our system deals with the general compiler construction problem although, as with all engineering projects, it is likely to be particularized to certain classes of languages or certain classes of machines in practice.

BIBLIOTHEQUE DU CERIST

The aim of this section is to use the scaffolding as a system specification device.

BIBLIOTHEQUE DU CERIST