A PROGRAM MODEL AND KNOWLEDGE BASE FOR COMPUTER AIDED PROGRAM SYNTHESIS

Richard J. Wood

Department of Computer Science
University of Maryland
College Park, Maryland 20742

Introduction

Program synthesis is a complex task comprising many interacting subactivities and requiring access to a variety of knowledge sources. Recent investigations have discovered the inadequacies of current synthesis techniques to keep pace with the increasing difficulties of managing large intricate problem solutions. An alternative approach to software methodologies is the development of intelligent computer systems that manage the vast amount of information assimilated and accessed during this process. The system's "intelligence" is characterized not by an innate ability to invent solutions, but by the incorporation of an internal model of the problem domain and corresponding program solution.

Overview

This project examines a style of programming, called the client-consultant paradigm, in which a domain expert (the client) and an adaptable programmer (the consultant) interact to formulate a program. By casting this investigation in the client-consultant paradigm, the techniques and knowledge general to programming in varying domains can be isolated and examined. Within this cooperative framework of program synthesis the following four major categories of activities have been identified:

- Requirement Acquisition: the attainment of a description of a task and its domain from the client.
- Specification Formulation: the transformation, compilation, and refinement of the problem requirements into terms recognized by the system.
- Algorithm Construction: the selection of known techniques for solving a task's subproblems and the combination of these solution fragments to form an overall task solution.
- Code Production: the instantiation of language construct schemata that correspond to steps of the solution and whose execution will achieve the overall program behavior.

A system capable of supporting computer aided synthesis must have components corresponding to each of the above activities. Additionally, an automated consultant must access a rich knowledge base of programming information in constructing a model of the evolving program. This model must explicitly represent the effects of the activities occurring during synthesis and must be accessed and manipulated by the consultant during the construction of the final program.

The Interactive Program Synthesizer (IPS) is a system designed to fulfill the role of a consultant during the construction of a program. While similar in theme to other investigations of automatic programs e.g., SAGE [Baizer 80], PSI (Green 79), the IPS differs in several aspects; the most important concerning the nature of the role of the consultant while interacting with a client. The consultant assimilates information and queries the client for clarification of unrecognized terminology. This role is active in a way in which the client is directed to elaborate abstract operations and objects using commonly shared terminology. This focusing of the dialogue helps to eliminate the introduction of extraneous information by the client and shortens the time before terms are identified. Other systems produce complete process descriptions, but at a cost of additional processing for referent identification.

This report focuses on the nature of the program model and the programming knowledge base required for a successful synthesis system. Specifically, the architecture of the Interactive Program Synthesizer, under current development, is described.

The Program Model

The central data structure of the IPS system is the program model which represents the developing program during synthesis. The organization of the program model must accommodate operations that include the introduction of new terms from the user's problem description, the refinement and further definition of existing terms, the detection of inconsistencies in the description, and the efficient retraction of the inconsistent assertions. These activities occur in a client-consultant programming and correspond to the initial problem description by the client, the explanations and clarifications required by the consultant and the rejection of unfruitful partial solutions. The program model is a record of all the assertions, inferences, and deductions made during the synthesis and the justifications for each assertion.

The IPS program model is encoded as a semantic network, a data structure which facilitates the processing of synthesis activities. The nodes of the program model act as repositories for the descriptions of domain objects (both operands and operations) while the links of the model define the flow of information among the nodes. Included in the set of program model link types (and considered primitive to programming) are the following links: definition (*DEF), representation (*REP), refinement (*REP), and reduction (*RED). The first two link types (*DEF and *REP) describe the relationship between an object and its definition. The semantics of this link type differ from the more general ISA link that is in most semantic networks, in that the ISA link implies (usually) an inheritance of features from the superobject to an instance. Similarly, the *REP and *RED links describe the correspondence between an operation and the set of states requisite for its achievement.

When the sentence "The screen is thought of as a 40 by 80 byte array." is processed by the system two objects are introduced into the model, 1) an object whose name is "screen" and 2) an object whose...
which is an instantiation of the two dimensional array frame with the information presented in the sentence (e.g., the extents of each dimension and the type of the array frame). The two objects are linked via a *DEF link that reflects the client's decision to consider the abstract object "screen" as a two dimensional array. The processing of the sentence "To clear the screen store a blank in every position of the screen," introduces new subregions to the domain of expertise of user specification and links them via a *REF link. The *REP link introduces two objects corresponding to the domain of expertise of operation and links them via a *RED link. The *RED link is used in a similar manner, but represents system-generated decisions (inferences) as opposed to user specifications.

The division of link types parallels the distinction between the two domains of expertise of the client and the consultant. Clarifying requests to the user are expressed using terminology identified by *DEF links (e.g., objects "screen" and "clear," in the above example), while the system automatically infers information about *RED and *REP objects (e.g., 2-D array and array store). When an inimical interaction between two states in the model is detected, the system unrolls [Rieger&London, 1977] the current solution and selects alternative strategies for achieving the REP state before causing a retraction of that state. If the REP state is a *DEF state the system must appeal to the user for a restatement of the goal. These inferences of user-supplied decomposition and must turn to the user for an alternate decomposition.

Other link types exist in the program model (e.g., inheritance, feature-description, dependency) but are beyond the scope of this report. The reader is directed to other projects that investigate the foundations of semantic networks (e.g., [Brachman, 1977] and [Fahlman, 1979]).

The Programming Knowledge Base

The program model is constructed by instantiating a knowledge base of programming knowledge with problem-specific data presented by the user. The programming knowledge base consists of facts and program construction techniques that investigate the foundation of semantic networks (e.g., [Brachman, 1977] and [Fahlman, 1979]).

The IPS knowledge base is organized in a hierarchical frame system, an efficient organization of knowledge for two system activities: recognition and inference. Features presented during the user's behavioral task description suggest potential programming objects to represent the abstract domain objects. Identification of a particular programming object supplies information normally associated with an object, but not stated in the user's discourse. These inferences provide a basis for queries to the user about requisite information (e.g., is this a type description or a specific individual?), and selection of a particular object from a set of candidates.

The programming frames contain information describing characteristics, attributes, and potential roles of an object in a program. While processing the sentence "The screen is thought of as a 40 by 80 byte array," for example, the prototypical two dimensional array frame is retrieved and instantiated with the data presented in the sentence. Additionally, the array frame provides default information about some characteristics, such as, starting indices of the dimensions and commonly used terminology for referencing components of the array (e.g., "row" and "column" for the dimensions). It also suggests queries to the user about requisite information (e.g., is this type fixed, is this a type description or a specific individual?), and selection of two dimensional array (e.g., the ability to define sub-regions of an array are included in the frame but not processed immediately). If later assertions refer to these roles they can be retrieved from the prototypical frame and instantiated.

The knowledge base contains frames for both programming objects and operations. Object frames contain defining characteristics and potential roles of an object, while operations are described by the set of states requisite for their correct execution and the post conditions and side-effects of the action.

Conclusion

This note describes the nature of two components of a computer-aided program synthesis system: the internal program model and the programming knowledge base. These structures are part of a larger project [Wood, 1980] directed towards the development of a theoretical model of program synthesis and an implementation of a programming system that incorporates this model. The project is investigating assembly language programming on a simple microprocessor, in particular, the activities and knowledge used by a consultant during the construction of a software package that manages a video display buffer. By examining program synthesis in the concrete and uncluttered realm of assembly language programs (as compared to abstract high-level languages) progress towards a successful computer-aided programming system can advance in much the same manner that advances to general purpose problem solving resulted from investigations into the blocks-world domain.

Acknowledgements

Thanks to Chuck Rieger, Steve Small, and Handy Trigg for reading drafts of this paper and making helpful comments.

References

[Balzer, 1978]

[Brachman, 1978]

[Fahlman, 1979]

[Green, 1979]

[Rieger&London, 1977]

[Wood, 1980]