SYNTACTICALLY CORRECT MULTILEVEL STRUCTURED
PROGRAM DESIGN

L. I. Nagornaya

UDC 681.3

The design of syntactically correct programs is considered in the multilevel structured program design framework. An interactive design algorithm for syntactically correct programs is described in SAA/1 (DSP-set).

One of the stages in program development after the program has been described in some high-level language involves the use of compilers or interpreters to generate a syntactically correct description of the algorithm in this language. The compiler systems accordingly include a syntactic analysis block (a parser), which performs this function. However, for sufficiently simple design languages (specification languages) it would be useful to supplement this traditional approach with tools that produce syntactically correct algorithms and programs already in the design stage. The solution of this problem is closely linked with the development of modern program design methods and appropriate support tools. Such support tools include specification and program design languages with their support systems.

The paper describes a basic approach to the construction of syntactically correct programs, using the multilevel structured program design (MSPD) method and its support environment — the MUL'TIPROTSESSIST system [1]. We describe an algorithm for interactive design of syntactically correct programs (DSP-set), written in the language of SAA schemas, and examine an example of its application to construct a particular program.*

The basic idea of the DSP-set involves level-by-level construction of programs from top down by expansion of the language constructs. In each step of the construction process, the system (interacting with the developer) displays for selection only those language constructs whose substitution in the SAA schema will produce a syntactically correct description of the algorithm in the SAA language.

The distinctive features of the MSPD method and its support tools (in comparison with standard technologies) include reliance on the algebraic-grammatical formalism, traceable to Glushkov’s system of algorithmic algebras (SAA). This ensures conceptual integrity of the design, conciseness, and transparency of the description of the algorithms and programs [1].

MUL'TIPROTSESSIST is a system for structured generation of programs and algorithms from their formal specifications. The source language for this system is SAA/1 — the language of construction of algorithm schemas intended for multilevel design of algorithms and programs (both sequential and parallel). The expressive tools of SAA/1 include composition A * B (sequential execution of the operators A and B); distributive structures of the form IF V THEN A ELSE B; loop structures WHILE NOT V DO A; and also tools for construction of compound conditions — conjunction V AND W, disjunction V OR W, negation NOT V.

The multilevel structure of SAA/1 schemas reflects the top-down expansion of the algorithm in the process of design. Given the SAA-schemas of the program design developed in SAA/1 and a library of elementary modules defining the particular application domain, the system automatically generates the programs in one of the supported object languages. The representation of program designs by schemas allows the following actions: interpretation — specification of the basis for the orientation of the schemas to the chosen application domain; reinterpretation — replacement of the basis in order to obtain algorithms of the same structure from the given or another application domain; transformation — modification (in particular optimization) of algorithms by the apparatus of relationships. Combination of these qualities corresponds to the notion of

*The basic idea of the DSP-set is due to G. E. Tseitlin.

algorithmic portability introduced in [2]. Thus, which is particularly important, the presentation of program designs in the language of SAA schemas and the use of the MUL'TIPROTSESSIST system for program generation not only produces portable programs but also ensures portability of the algorithms realized by these programs [2].

The MUL'TIPROTSESSIST system is syntax-controlled. The SAA-schema of the algorithm design is checked by the parser for syntactic correctness; if the design violates the source language syntax, the system produces error diagnostics, and so on until the SAA-schema is made syntactically correct. The parser is followed by the generator, which is parametrically tunable to a particular object language and generates a program from SAA-schemas and implementations of the basic concepts.

These features of the MSPD method, the SAA-schema language, and the MUL'TIPROTSESSIST system make it possible to design syntactically correct program schemas in SAA/1.

Let us consider in more detail the DSP-set algorithm. In accordance with the MSPD method, the program design process is an automatic multilevel stepwise transformation of a given problem into a program in some object language. Each level of the design process is open to the developer. The first main level of the program schema is the initial "blank," to which new levels are added, specializing various parameters of the language constructs and expanding the functions of the constructs in which they are included.

Each new level is an equality whose left-hand side is an explanatory identifier that appears in the right-hand side of the preceding level. The identifiers that do not enter the left-hand sides of any level are called elementary.

The SAA/1 source language includes the following constructs.

1. Operator constructs
   1. A * B (sequential execution of operators)
   2. WHILE NOT V DO A (loop structure)
   3. IF V THEN A
   4. IF V THEN A ELSE B (distributive structure), where A, B are operators, V, W are conditions.

The SAA-schema of a first-level program ("blank") has the form

SCHEMA schema-name;
OPERATOR 0 0;
END;

In the MUL'TIPROTSESSIST source language syntax, OPERATOR 0 0 may be one of the operator constructs. The DSP-set displays them on the screen and allows the user to select the required operator by pointing with the cursor. For example, the developer selects the composition of operators A * B. The DSP-set substitutes this construct into the SAA-schema, which thus takes the following form:

SCHEMA schema-name;
"OPERATOR 0 0" =
"OPERATOR 1 1" * "OPERATOR 1 2";
END;

Interacting with the DSP-set, the developer may add a comment to the equality or replace OPERATOR 0 0, OPERATOR 1 1, OPERATOR 1 2 with explanatory identifiers; otherwise the DSP-set substitutes its own identifiers.

The next step of the design process is to refine the syntactic structure of all the parameters in the equality, in this case the parameters OPERATOR 1 1 and OPERATOR 1 2.

It the parameters entering the equality are elementary, the design ends. If the parameters are nonelementary, then the DSP-set displays to the developer the language constructs that may be substituted for the parameters, and so on. The designed SAA-schema is written to an output file and is passed to the MUL'TIPROTSESSIST system for program generation.

A special feature of the DSP-set algorithm is the use of queue memory. The queue is a list of memory fields which are written into from one side (the end of the queue) and read from at the other side (the beginning or top of the queue). Reading from the top of the queue releases the top memory field and shifts the queue top pointer to the next field.