Introduction.

In the last years, many authors have investigated the problem of the synthesis of programs according to the following schema:

Problem: given a motivation defined in some language, build up an algorithm expressed in some formalism, i.e. a procedural definition having the same "meaning" as the non-procedural one represented by the motivation.

A precise formulation of the above problem involves the following questions:

A. One has to define the motivation language \( M \) and to assign a meaning to motivations

B. One has to choose the formalism \( Q \) to write down the algorithms and has to assign a meaning to the latter

C. One has to investigate the synthesis-maps, intended as computable functions

\( S : M \rightarrow Q \).

In this introduction, we will briefly review the most significant papers on the subject; we will distinguish two different attitudes:

I. The informal attitude: points A and B are not clearly defined and are analyzed informally, starting from the well known notion of function computed (or defined) by an algorithm and from an intuitive interpretation (a "common sense" interpretation) of the motivations; point C is usually investigated by euristic methods. There are many examples in literature of such an attitude [7, 8].

II. The formal attitude: it requires a well defined (formal) frame in which points A and B can be precisely stated and which allows the development of some criteria and techniques in order to systematically investigate point C. Here, the formalism adopted to define the motivations and to state the meaning of algorithms is generally inspired by formal logic, while there is a wide range of choices for the algorithmic language \( Q \). In this attitude, we may distinguish two different ways to write down
motivations, from which different developments of point C arise:
(a) for the first one, the typical frame is the predicate calculus \([2, 3, 11]\) or any other formalism suitable to specify properties in a context less restrictive than the one of a formal theory describing a specific structure; e.g., the set theoretic formalism used in \([5]\) (see also \([4]\)) here, one defines the problem (or the motivation), to be solved, essentially by means of a description of its properties in the chosen formalism, and a motivation is something as: "build up an algorithm which computes a function satisfying the given properties".

(b) for the second point of view, the frame is defined by a formal theory describing a specific but powerful enough structure (for instance a number theory \(T_N\), which allows induction proofs), where one expresses the function to be computed by a formula of the theory; here a motivation is something as: "construct an algorithm to compute the function expressed by the formula" \([4, 6, 10, 13, 14]\)

The aim of our paper is to describe a synthesis-procedure based on a Gentzen-like calculus, to which special "construction-rules" (synthesis-rules) are added. More precisely, the proposed synthesis-procedure arises from a merging of the synthesis-method in \(T_{NI}\) expounded in \([6]\) (\(T_{NI}\) is Kleene's intuitionistic number theory) and of the assertion-method to verify the correctness of programs.

The synthesis-method in \(T_{NI}\) is outlined in the following schema:
- the synthesis-problem ("motivation") is expressed by a formula of the language of \(T_{NI}\) a formula such as \(\exists z \varphi(x, z)\), which is to be interpreted in the following way:  
  "for every \(\hat{x}\) such that \(\models \exists z \varphi(\hat{x}, z)\) holds, find a value \(\hat{z}\) such that \(\models \varphi(\hat{x}, \hat{z})\)" (NOTE 1)
- the synthesis is obtained by applying a standard proof-procedure, in order to verify the i.w.c.-ness property of the formula \(\exists z \varphi(x, z)\) (for the definition of "i.w.c.-ness", we send to \([6]\)); starting from such a verification, one can automatically construct a program to compute, for every \(\hat{x}\) such that \(\models \exists z \varphi(\hat{x}, z)\), an appropriate \(\hat{z}\) such that \(\models \varphi(\hat{x}, \hat{z})\) (there is, indeed, an implementable algorithm to do so, expounded in \([6]\)).

On the other hand the assertion method requires (as it is well known) to associa-

NOTE 1. "\(\models\)" means "validity" on the structure of the natural numbers (according to the definition of validity for the classical theories - i.e., we do not attach any sophisticated meaning to our "motivations", even if our procedure is based on intuitionism- ); \(\hat{x}\) denotes a natural number and \(\check{x}\) denotes the corresponding numeral in the language of the theory.