1. Introduction

In this paper we construct a domain of substitutions that may form the basis of a semantics for logic programming, since a computation involving a logic program and a query naturally results in a substitution. Our domain differs from usual domains in that \( \omega \)-continuity is defined with respect to both increasing and decreasing chains. This difference is a consequence of our principle of computational equivalence: The results of two computations should be given the same denotation precisely when they bear the same finitely computable information content.

In the following we discuss the principle of computational equivalence alternated with a stepwise construction of the substitution domain. Finally we present a denotational semantics based on the finished domain.

This paper is abstracted from a research report [2], where further details can be found. Concerning prerequisites in the areas of logic programming and Scott-domains the reader may consult [3] and [4] respectively.

2. Information content

We mentioned above that a computation involving a logic program and a query naturally results in some sort of substitution. In a semantic domain such substitutions should have the same denotation if and only if they are equivalent in an appropriate sense. Consider the following example:
Example 1.

**Program:** \( Q(f(z), g(z)) \leftarrow p(z). \)
\( P(a). \)

**Query:** \( Q(X, Y)? \)

Result of computation in the form of some possible syntactic answers:

1): \[ X \rightarrow f(z), Y \rightarrow g(z), z \rightarrow a. \]
2): \[ X \rightarrow f(w), Y \rightarrow g(w), w \rightarrow a. \]
3): \[ X \rightarrow f(a), Y \rightarrow g(a). \]

Underscore denotes an anonymous variable. All variables that do not occur in the query are made anonymous in the answer. By giving names to anonymous variables, we obtain the possibility of relating different undefined values. However, the information content of a substitution is independent of a specific choice of names: answers 1) and 2) are equivalent. In the case of ex. 1 we may completely remove anonymous variables and obtain the answer 3). The information contents of the three substitution answers are identical. Consequently the three substitutions should have the same denotation in a domain of substitutions.

We are going to build a domain based on Scott's information system framework [4]. Apparently concepts such as information content and anonymous variable are important, when dealing with computational equivalence. We will demonstrate, how these concepts can form the basis of a definition of substitution dataobject.

3. Substitution dataobject

Initially we provide some basic definitions. The syntactic basis consists of one countable set of identifiers and another of variables. We do not bother whether an identifier denotes a function or a predicate, neither do we deal with arities. On the basis of identifiers and variables we form the recursively defined set of terms and the set of substitutions.