LE STUDIUM
Loire Valley
Institute for Advanced Studies

LE STUDIUM Multidisciplinary Journal
ISSN 2669-7807
www.lestudium-ias.com

FELLOWSHIP FINAL REPORT

A domain-theoretic model construction for Coquand/Huet's calculus of constructions

Dieter Spreen[1, 2.3]

[1] LE STUDIUM Institute for Advanced Studies, 45000 Orléans, France
[2] Université d'Orléans, LIFO, 4500 Orléans, France
[3] Department of Mathematics, University of Siegen, Siegen, Germany

REPORT INFO

Fellow: **Dieter Spreen**
From University of Siegen, Germany
Host laboratory in region Centre-Val de Loire: LIFO, University of Orléans
Host scientist: **Jérôme Durand-Lose**
Period of residence in region Centre-Val de Loire: April-September 2023

Keywords :
Domain theory, type theory, proof assistant, logic, effectively given structures, computability

ABSTRACT

The Calculus of Constructions is one of the most powerful systems of constructive logic. It consists of three logical levels: proofs, propositions, kinds. The kinds are the types of operations, i.e., constructions of propositions. Both, the level of propositions and the level of kinds, are closed under general rules of quantification. A model construction involves giving meaning to the objects of each of the three levels in such a way that the quantification rules are satisfied. The aim is to give a construction that uses ordered structures, known as domains. In the present work domain constructions used for the interpretation of the quantification constructs are presented in a general way and computability issues are discussed.

1- Introduction

Software is produced by humans and thus error-prone. To guarantee that a software product works correctly, that is, according to the given specification, it is normally tested against a variety of possible scenarios. As follows from daily experience not all errors will be found this way, leading to back-up cycles. For applications in safety-critical situations such as autonomous mobile robots or car driving systems this approach is thus inefficient: the correctness has to be mathematically proven. Given the size of such software packages, proving its correctness by hand is a tedious task. To this end interactive proof assistants have been developed that automatically generate large parts of the proof. A popular such assistant is the Coq package. It is based on the Calculus of Constructions (CC) introduced by T. Coquand and G. Huet (1988), still one of the most powerful systems of constructive logic.

Constructive logics have been developed as a way out of the foundational crisis after the discovery of Russel's paradox. The central notion of the new approach is that of a *construction* or *proof*. In the Brouwer-Heyting-Kolmogoroff interpretation the meaning $[A]$ of a logical formula A is suggested to be the collection of all its proofs. In particular, an implication $A \to B$ is interpreted by the set of all maps t that transform proofs p of A (formally written $p : A$) into proofs $t(p)$ of B. Various systems of higher-type constructive logic have been introduced in the literature. At least in rudimentary form they all include the rule

$$\frac{A : \mathbf{Prop} \qquad B(x) : \mathbf{Prop} \;\; (x : A)}{(\prod x : A.\, B(x)) : \mathbf{Prop}}$$

(Read: If A is a proposition and for every $x : A$, $B(x)$ is a proposition, then also $\prod x : A.\, B(x)$ is a proposition.) In the spirit of the Brouwer-Heyting-Kolmogoroff interpretation $\prod x : A.\, B(x)$ will be interpreted as a set of functions t that map every x in $[A]$ onto some element in $[B(x)]$, that is, a proof that $B(x)$.

2- The calculus of constructions

Well-formed expressions of CC are divided into three levels: proofs, propositions, kinds. Propositions and kinds are closed under the formation

of expressions of the form $\prod x : A.\ B(x)$, where A is either a proposition or a kind, and similarly, $B(x)$ is either a family of propositions or kinds. If A is a proposition, then the family is indexed by the proofs of A; in the other case it is indexed by the operations of kind A. Operations can, e.g., be thought of as being constructions of propositions. There is a kind constant **Prop**. An expression of the form A : **Prop** then means that A is a proposition.

The rules of the calculus allow the derivation of statements of the form

$$\Gamma \vdash s : A \text{ and } \Gamma \vdash s = t : A,$$

where Γ is a c*ontext* declaring the range of the variables appearing in *s, t*, and *A.*

The model we are constructing will give meaning to the expressions of an extension of CC allowing the expressions of all three levels to be defined recursively, and in addition to being closed under the formation of expressions $\prod x : A.\ B(x)$, both propositions and kinds are closed under the formation of expressions $\sum x : A.\ B(x)$. Such expressions will be interpreted as sets of pairs (a, b) where a is an object with property A and b a proof that $B(a)$, which is a constructive reading of the statement "There exists some a in A so that $B(a)$."

3- The model construction

As follows from results of Reynolds (1984), there is no set-theoretic model of Girard's system F, which is a subsystem of CC containing the constant **Prop** but no other kinds. For system F domain models have been constructed. However, the constructions quickly went into foundational mathematical problems and had to use tricks to avoid them.

Domains have been introduced by Dana Scott (1970) in order to provide a mathematical basis for a denotational semantics of programming languages. Berry (1978) refined the theory by introducing stable domains which are better suited for giving semantics to functional programs.

Domains are ordered structures that satisfy requirements on the existence of least upper bounds of certain subsets. In the model construction we will follow the approach by Coquant, Gunter and Winskel (1988) and use Berry's dI-domains for the interpretation of propositions and proofs. However, in order to avoid the foundational difficulties of previous model constructions for system F a suitable subclass of dI-domains will be used. Moreover, for the interpretation of propositions we will not use the domains but their canonical representation: event structures, which were introduced by Winskel (1980). They can be ordered by a natural substructure relation leading to a stable bifinite domain. Such domains, or better, their representations, will be used to interpret kinds.

As is well-known from the proof of a statement "For all x in A there is some y in B" in a constructive logic an algorithm can be extracted for the computation of y from x such that the result y has property B.

Therefore, the domains we are using for the construction should be effectively given and all constructions should be computable.

4- Results

The present research concentrated on two aspects of the research plan: the development of a suitable computability theory and the search for a class of domains general enough to include the above-mentioned two types of domains and special enough to allow for the derivation of the results needed in the further development of the construction. The class we found are the countably algebraic distributive L-domains with Berry's Property I. Locally it has all the good properties that the special domain classes we are using later have globally. Moreover, the category **dLI** of these domains with rigid embedding-projection pairs as morphisms has the central properties we need: existence of co-limits of directed diagrams and of pullbacks. As a consequence, we can define what is to be understood by a continuous and in particular, by a stable functor. The two main results are:

Theorem 1. Let D be a domain in **dLI** and F: $D \rightarrow$ dLI be a stable functor. Then also

$$\sum(D, F) = \{ (x, x') \mid x \text{ in } D \text{ and } y \text{ in } F(x) \},$$

ordered by $(x, y) \leq (x', y')$ if $x \leq_D x'$ and $F[x\ x'](y) \leq_{F(x)} y$,

is a domain in **dLI**.

Call a function $f: D \to \bigcup_{x \in D} F(x)$ F-function if for all x in D, $f(x)$ in $F(x)$ and let $\prod(D, F)$ be the set of all stable F-functions, ordered by the stable ordering.

Theorem 2. Let D be a domain in **dLI** and F: $D \to$ dLI be a stable functor. Then $\prod(D, F)$ is a distributive L-domain.

An example is given showing that in general $\prod(D, F)$ is not countably algebraic. Property I is only defined for countably algebraic domains.

5- Conclusion

The aim of the proposed project is to present a domain model of (an extension of) the calculus of constructions that avoids the difficulties appearing in earlier constructions of domain models for system F, a subsystem of the calculus. Moreover, the construction will be more transparent and comprehensible, the domains effectly given and the domain constructions computable.

The first part of the research has now been finished: a general class of domains was singled out encompassing the domain classes that will be used in the interpretation of the two object levels of the calculus, that is, propositions and kinds. Central results needed in the construction could be derived in great generality. They ensure that the corresponding domain categories contain colimits of directed diagrams and pullbacks. Moreover, a general construction of dependent sums and products has been presented which easily specializes to the domain subcategories to be considered.

The second part of the research will study the special types of domains mentioned above, in particular their canonical representations. Dependent products and sums of representations will have to be introduced corresponding to the dependent products and sums, respectively, of the represented domains. Moreover, the structure of the respective sets of representations with respect to a substructure relation needs be investigated.

On the basis of these results the interpretation of CC can be defined and studied. In the last step, eventually, the computability of the construction will be shown.

The author had the opportunity to present an overview of the construction to members of the hosting group at the University of Orléans in the Seminar: "Calculus of constructions: at the foundations of Coq", organized by Professor Jérôme Durand-Lose, on the 15th of May, 2023.

6- Perspectives of future collaborations with the host laboratory

To deepen the cooperation with LIFO, work is undertaken on a research proposal to be submitted to funding agencies in Germany and France to obtain support for further visits.

7- Articles published in the framework of the fellowship

Spreen, D. (2023). How much partiality is needed for a theory of computability? 52 pages. arXiv:2305.06982, under review for publication in *Computability.*

8- Acknowledgements

The author visited the University of Orléans within the framework of the ATHENA – European University programme. His stay was supported by the German Academic Exchange Service, and by cooperative agreement 2022-2004-261 as Le Studium / ATHENA Visiting Researcher, from Loire Valley Institute of Advanced Studies, Orléans, France, hosted by LIFO, University of Orléans in 2023. He is grateful to the colleagues at LIFO for their hospitality and nice and interesting discussions.

Furthermore, he wants to point out that the support of Le Studium Loire Valley Institute was really special. The peaceful and pleasant environment in which researchers are hosted as well as the friendliness and helpfulness of the entire team at the Institute contributed greatly to the success of his research.

9- References

1. Berry, G., 1978. Stable models of typed lambda-calculi. In: *Proc. ICALP*,

Springer Lect. Notes in Comput. Sci. 62.
2. Coquand, T., Gunter, C. and Winskel, G., 1988. Domain theoretic models of polymorphism. *Information and Computation*, 81:123-167.
3. Coquand, T. and Huet, G., 1988. A calculus of constructions. *Information and Computation,* 76:95-120.
4. Girard, J.Y., 1972. Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieur. Thèse d'Doctorat d'État, Université de Paris VIII.
5. Reynolds, J., 1984. Polymorphism is not set-theoretic. In: *Semantics of Data Types*, Springer Lect. Notes in Comput. Sci. 173.
6. Scott, D., 1970. Outline of a mathematical theory of computation. In: *4th Annual Princeton Conference on Information Sciences and Systems*, pages 169-176.
7. Winskel, G., 1980. Events in Computation. PhD thesis, University of Edinburgh.