CHURCH'S TYPES IN LOGICAL REASONING ON PROGRAMMING

Valerie NOVITZKÁ
Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic, E-mail: Valerie.Novitzka@tuke.sk

SUMMARY

In our previous paper [1] of this series we defined the basic types as the startpoint of scientific problem solving by help of logically and mathematically founded programming of mathematical machines. In this paper we extend the type system with Church's types that enable the first step of problem solving during logical reasoning.

***Keywords*:** *category theory, categorical logic, type theory, Church's types, λ-calculus*

1. INTRODUCTION

In our paper [1] we have introduced the basic concepts and facts for scientific problem solving by help of mathematical machines, i.e. by logical reasoning about programming of these machines. These fundamental concepts were: category, cartesian closed category, diagram and limit, topos and elementary topos, but the most important was the concept of basic types. Basic types actually form the starting point in the process of scientific problem solving by mathematical machines. The main purpose of this paper is to introduce a bit extended type system, the so called Church's type system as a further step of the scientific problem solving process by mathematical machines.

2. MANY-TYPED SIGNATURES

We begin the extension of our system of basic types with introducing the well-known notion of universal algebra: many-typed signature. A *many-typed signature* is important not only for the type system but also for some aspects of logical reasoning. In the following we use only the word signature for the notion of many-type signature.

A *signature* $\Sigma = (T, \mathcal{F})$ consists of a finite set T of (the names of) *basic types* denoted by letters $\sigma, \tau, \upsilon, \ldots$ and a finite set $\mathcal{F}$ of function symbols. Every function symbol $F \in \mathcal{F}$ is of a form $F: \sigma_1, \ldots, \sigma_n \rightarrow \sigma_{n+1}$, for some natural number n. A function symbol F takes inputs of types $\sigma_1, \ldots, \sigma_n$ and yields an output of type σ_{n+1}. A signature morphism $\phi: \Sigma \rightarrow \Sigma'$ from a signature $\Sigma = (T, \mathcal{F})$ to a signature $\Sigma' = (T', \mathcal{F})$ is a pair $(u, (f_\alpha))$, where $u: T \rightarrow T'$ is a function between underlying sets of types and (f_α) is a family of functions between corresponding sets of function symbols, where $\alpha = ((\sigma_1, \ldots, \sigma_n), \sigma_{n+1})$. Then for a function symbol $F: \sigma_1, \ldots, \sigma_n \rightarrow \sigma_{n+1}$

$$f_\alpha(F): u(\sigma_1), \ldots, u(\sigma_n) \rightarrow u(\sigma_{n+1}).$$

We can construct a *category of signatures* ***Sign*** containing:

– as objects: many-typed signatures,
– as morphisms: signature morphisms between them.

Sign is a category, because for every object Σ there is an identity $id_\Sigma: \Sigma \rightarrow \Sigma'$, $id_\Sigma = (id_T, (id_F))$ and composition of morphisms is inherited from the composition of signature morphisms. The forgetful functor $U:$ ***Sign*** $\rightarrow$ ***Set*** from the category of signatures to the category ***Set*** of sets and functions assigns to every signature $\Sigma = (T, \mathcal{F})$ from ***Sign*** its underlying set of types T from ***Set*** and to every morphism $(u, (f_\alpha))$ from ***Sign*** the function $u: T \rightarrow T'$ from ***Set***. The forgetful functor 'forget' the structure of signatures and it is a split fibration [2].

3. TERMS FOR MANY-TYPED SIGNATURES

In the following text we assume a many-typed signature $\Sigma = (T, \mathcal{F})$ defined as above. To introduce terms we need a set $Var = \{v_1, v_2, \ldots\}$ of *term variables*. Every variable has exactly assigned one type from the set T by a *variable declaration* $v:\sigma$. A finite sequence of variable declarations

$$\Gamma = (v_1:\sigma_1, \ldots, v_n:\sigma_n)$$

is called a *context*.

Terms are defined with respect to a fixed finite sequence of term variables which receive their types from contexts. We denote a *term* M by a sequent

$$\Gamma \mid\!- M: \tau$$

which expresses that a term M is of a type τ in context Γ, i.e. a term M may contain only typed variables from Γ and its value is of type τ.

Terms are constructed by successive applications of the following two basic rules and three structural rules. The *basic rules* describe construction of terms:

$$\frac{}{v:\sigma \mid\!- v: \sigma} \quad \text{- identity}$$

for $F: \sigma_1, \ldots, \sigma_n \rightarrow \sigma_{n+1}$

$$\frac{\Gamma \vdash M_1: \sigma_1, \ldots, \Gamma \vdash M_n : \sigma_n}{\Gamma \vdash F(M_1, \ldots, M_n) : \sigma_{n+1}} \text{ - function symbol}$$

The *structural rules* are

$$\frac{v_1:\sigma_1,\ldots, v_n:\sigma_n \vdash M:\tau}{v_1:\sigma_1,\ldots, v_n:\sigma_n , v_{n+1}: \sigma_{n+1} \vdash M:\tau} \text{ - weakening}$$

$$\frac{\Gamma, v_i:\sigma , v_{i+1}:\sigma \vdash M:\tau}{\Gamma, v_i:\sigma \vdash M:\tau} \text{ - contraction}$$

$$\frac{\Gamma, v_i:\sigma_i , v_{i+1}:\sigma_{i+1} , \Delta \vdash M:\tau}{\Gamma, v_{i+1}:\sigma_{i+1} , v_i:\sigma_i , \Delta \vdash M:\tau} \text{ - exchange}$$

The structural rule of weakening allows to add a redundant variable declaration, the contraction rule enables to replace two variables of the same type by a single one and the exchange rule expresses that variable declarations in contexts can permute. The following *substitution rule* enables to substitute in a term $M:\tau$ a variable $v:\sigma$ by a term $N:\sigma$ of the same type, where N contains only variables from Γ:

$$\frac{\Gamma, v:\sigma \vdash M:\tau \quad \Gamma \vdash N:\sigma}{\Gamma \vdash M[N/v]:\tau} \text{ - substitution}$$

We remember that the types in contexts and of terms are only basic types from the signature Σ. Basic rules, structural rules together with the substitution rule determine the *term calculus $\lambda(\Sigma)$ over a signature Σ*. We note here that $\lambda(\Sigma)$ calculus will play very important role in the construction of proof nets [3,9] (with various semantics) in the complicated logical reasoning in the framework of one complete theory. So, this calculus serves as a foundation for solving scientific problems by mathematical machines.

4. CLASSIFYING CATEGORY AND ITS MODEL

Contexts and typed sequences of terms over a signature Σ form a category. In this construction we use *terms-as-morphisms* approach [9] based on the following idea. A term in the form of the following sequent

$$v_1:\sigma_1, \ldots, v_n:\sigma \vdash M:\tau$$

may be regarded as an operation mapping input values $a_i:\sigma_i$, $i=1,\ldots,n$ on the left side of the sequent to an output value $M[a_1/v_1, \ldots,a_n/v_n] :\tau$ of a type τ on the right side of the sequent. Therefore we can consider a term as a morphism between types

$$M: \sigma_1 \times \ldots \times \sigma_n \to \tau .$$

By regarding terms as morphisms between types from the set T of basic types of signature Σ we construct the *classifying category Class(Σ) over a signature Σ* as follows:

- objects are contexts $\Gamma =(v_1:\sigma_1 , \ldots, v_n:\sigma)$ as defined above,
- morphisms between contexts $\Gamma \to \Delta$, where $\Delta = (w_1:\tau_1,\ldots,w_m:\tau_m)$ are m-tuples $(M_1, \ldots,M_m)$ of terms $\Gamma \vdash M_i:\tau_i$, for $i=1,\ldots,m$,
- an identity morphism id_Γ on every object Γ is the n-tuple of variables $(v_1,\ldots,v_n)$ from Γ, and
- composition of morphisms

$$\Gamma \xrightarrow{(M_1,\ldots,M_m)} \Delta \xrightarrow{(N_1,\ldots,N_k)} \Theta$$

is the k-tuple $(L_1,\ldots,L_k)$ of terms defined by substitution

$$L_i = N_i[M_1/w_1,\ldots,M_m/w_m].$$

for $i=1,\ldots n$.

The classifying category *Class(Σ)* introduces Gentzen's sequent calculus into categorical logic. For every two objects Γ, Δ from *Class(Σ)* a *binary product* $\Gamma\times\Delta$ can be defined as context concatenation

$$(\Gamma,\Delta)= (v_1:\sigma_1,\ldots,v_n:\sigma_n ,w_1:\tau_1,\ldots,w_m:\tau_m)$$

with two projections

$$\pi_1=(v_1:\sigma_1,\ldots,v_n:\sigma_n) \text{ and } \pi_2=(w_1:\tau_1,\ldots,w_m:\tau_m)$$

as it is illustrated on Fig. 1.

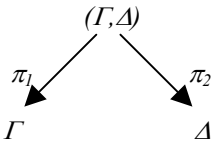

Fig. 1 Product of contexts

The category *Class(Σ)* has as *terminal object* the empty context $\varnothing = ( )$, because for every object Γ there is just one morphism from $\Gamma\to\varnothing$. Because the category *Class(Σ)* has finite binary products and a terminal object, it is *cartesian category* [6].

We define set-theoretical model of the classifying category *Class(Σ)* as follows:

- to every basic type $\sigma\in T$ we assign its *carrier set* A_σ ,
- to every function symbol $F\in\mathcal{F}$, such that $F: \sigma_1,\ldots,\sigma_n \to \sigma_{n+1}$ we assign a function

$$[\![F]\!]: A_{\sigma 1}\times\ldots \times A_{\sigma n} \to A_{\sigma n+1}$$

between corresponding carrier sets.

A *Σ-model* (or Σ-algebra) is a pair

$$((A_\sigma)_{\sigma\in T}, [\![-]\!])$$

which consists of a T-indexed family of carrier sets and of a collection $[\![-]\!]$ of actual functions for every function symbol from $\mathcal{F}$.

Set-theoretic models of classifying categories form the category ***SModel*** such that

- objects are three-tuples $(\Sigma, (A_\sigma)_{\sigma\in T}, [\![-]\!]_\Sigma)$, such that the last two members form a Σ-model,
- morphisms are pairs

$$(\phi, (h_\sigma)):$$
$$(\Sigma, (A_\sigma)_{\sigma\in T}, [\![-]\!]_\Sigma) \to (\Sigma', (A'_\sigma)_{\sigma\in T'}, [\![-]\!]_{\Sigma'})$$

where $\phi: \Sigma \to \Sigma'$ is a signature morphism and (h_σ) is a corresponding model homomorphism

$$(h_\sigma): ((A_\sigma), [\![-]\!]) \to ((A'_\sigma), [\![-]\!]').$$

5. INTRODUCING CHURCH'S TYPES

Until here we have considered only basic types from a signature Σ. Now we introduce Church's types constructed from basic types by constructors '$\to$', '$\times$' and '+' . Applying the constructor '$\to$' on basic types $\sigma, \tau \in T$ we can construct *arrow types* (function types) $\sigma\to\tau$, applying the constructor '$\times$' we can construct *product types* $\sigma\times\tau$, and by using the constructor '+' we can construct *coproduct types* (sum types) $\sigma+\tau$. In accordance with this construction we successively extend $\lambda(\Sigma)$ term calculus to the term calculus over Church's types.

First, we introduce arrow types. Let T_1 be the least set containing the set T closed under morphisms between types, i.e. if $\sigma, \tau \in T$ then $\sigma\to\tau \in T_1$. Term calculus $\lambda 1(\Sigma)$ built over a signature Σ with arrow types has all the rules as $\lambda(\Sigma)$calculus and the following rules for abstraction and application:

$$\frac{\Gamma, v:\sigma \mid- M:\tau}{\Gamma \mid- \lambda v:\sigma.M:\sigma\to\tau} \text{ - abstraction}$$

$$\frac{\Gamma \mid- M:\sigma\to\tau \quad \Gamma \mid- N:\sigma}{\Gamma \mid- M\,N:\tau} \text{ - application}$$

The abstraction rule introduces term $\lambda v:\sigma.M:\sigma\to\tau$ as a function assigning to a value $a:\sigma$ of type σ the result value $M[a/v]:\tau$ of the type τ .The application rule is an elimination rule which describes the application of a function $M:\sigma\to\tau$ to an argument term $N:\sigma$. These rules we complete with the following type conversion rules [4]:

$$\frac{\Gamma, v:\sigma \mid- M:\tau \quad \Gamma \mid- N:\sigma}{\Gamma \mid- (\lambda v:\sigma.M)\,N = M[N/v]:\tau} \text{ - } \beta\text{- conversion}$$

$$\frac{\Gamma \mid- M:\sigma\to\tau}{\Gamma \mid- \lambda v:\sigma.Mv = M:\sigma\to\tau} \text{ - } \eta\text{- conversion}$$

$$\frac{\Gamma, v:\sigma \mid- M = M':\tau}{\Gamma \mid- \lambda v:\sigma.M = \lambda v:\sigma.M':\sigma\to\tau} \text{ - } \xi\text{ - conversion}$$

$$\frac{\Gamma \mid- M = M':\sigma\to\tau \quad \Gamma \mid- N = N':\sigma}{\Gamma \mid- M\,N = M'\,N':\tau} \text{ - translation}$$

β-conversion rule describes the evaluation of functions on their arguments, η -conversion rule describes extensionality of functions. ξ -conversion rule and translation rule extend conversion relation '=' into equivalence relations.

Over $\lambda 1(\Sigma)$ calculus we construct new classifying category $Class1(\Sigma)$ over a signature Σ as follows:

- objects are contexts $\Gamma = (v_1:\sigma_1, \ldots, v_n:\sigma_n)$, where the types $\sigma_i \in T_1$, for $i=1,\ldots,n$,
- morphisms $\Gamma \to \Delta$ for $\Delta = (w_1:\tau_1, \ldots, w_m:\tau_m)$ are m-tuples of equivalence classes (with respect to conversion rules above) of terms

$$([M_1], \ldots, [M_m]) .$$

Now we extend $\lambda 1(\Sigma)$calculus with product and coproduct types. First we add new types *0* and *1* that are not in the signature Σ to the set T_1. The type *1* serves for empty product type and the type *0* serves for empty coproduct type. Let T_2 be the least set containing T_1 closed under finite products and coproducts of types, i.e. if $\sigma, \tau \in T_1$ then also

$$\sigma \times \tau \in T_2 \quad \text{and} \quad \sigma + \tau \in T_2.$$

The corresponding $\lambda 2(\Sigma)$ calculus has all rules as the $\lambda 1(\Sigma)$ calculus and the following new rules for product and coproduct typed terms. We use for tuples of product type angle brackets '$\langle$' and '$\rangle$' , and for cotuples of coproduct types the square brackets '$[$' and '$]$'. First two rules are introduction rules of terms $\langle\rangle$ of empty type *1* and product typed terms $\langle M,N\rangle : \sigma\times\tau$. The next two rules are elimination rules of projections $\pi_1:\sigma\times\tau \to \sigma$, $\pi_2:\sigma\times\tau \to \tau$.

$$\frac{}{\langle\rangle : 1} \text{ - 1 - introduction}$$

$$\frac{\Gamma \mid- M:\sigma \quad \Gamma \mid- N:\tau}{\Gamma \mid- \langle M,N\rangle : \sigma\times\tau} \text{ - } \times\text{ - introduction}$$

$$\frac{\Gamma \mid- P:\sigma\times\tau}{\Gamma \mid- \pi_1 P:\sigma} \qquad \frac{\Gamma \mid- P:\sigma\times\tau}{\Gamma \mid- \pi_2 P:\tau} \text{ - projections}$$

We add also the corresponding type conversion rules for product types:

$$\frac{\Gamma \mid- M:1}{\Gamma \mid- M = \langle\rangle : 1} \qquad \frac{\Gamma \mid- M:\sigma \quad \Gamma \mid- N:\tau}{\Gamma \mid- \pi_1\langle M,N\rangle = M:\sigma}$$

$$\frac{\Gamma \mid\!- M:\sigma \quad \Gamma \mid\!- N:\tau}{\Gamma \mid\!- \pi_2\langle M,N\rangle = N:\tau} \qquad \frac{\Gamma \mid\!- P:\sigma\times\tau}{\Gamma \mid\!- \langle\pi_1 P,\pi_2 P\rangle = P:\sigma\times\tau}$$

For coproduct types, i.e. disjoint union types, we add to the $\lambda 2(\Sigma)$ calculus the rules for introduction coproduct by injections (coprojections) $\kappa_1:\sigma\to\sigma+\tau$ and $\kappa_2:\ \tau\to\sigma+\tau$:

$$\frac{\Gamma \mid\!- M:\sigma}{\Gamma \mid\!- \kappa_1 M:\sigma+\tau} \qquad \frac{\Gamma \mid\!- N:\tau}{\Gamma \mid\!- \kappa_2 N:\sigma+\tau}$$

For elimination rule we introduce new operation *unp* similar to *unpack* operation in [2]. It deals for a term $\Gamma \mid\!- P:\sigma+\tau$ of coproduct type as 'case'. Let $Q:\upsilon$ containing variable $x:\sigma$ and/or $x':\tau$. Then

- if P is in the type σ then do a term $Q:\upsilon$ with P for the variable $x:\sigma$,
- if P is in the type τ then do a term $Q':\upsilon$ with P for the variable $x':\tau$.

The operation *unp* binds variables $x:\sigma$ and $x':\tau$. Then the corresponding rule for elimination coproduct is

$$\frac{\Gamma \mid\!- P:\sigma+\tau \quad \Gamma, x:\sigma \mid\!- Q:\upsilon \quad \Gamma, x':\tau \mid\!- Q':\upsilon}{\Gamma \mid\!- unp\ P\ as\ [\kappa_1 x\ in\ Q\ ,\ \kappa_2 x'\ in\ Q']:\upsilon}$$

If the context contains a variable of empty coproduct type $z: 0$ then the term with such context is empty cotuple $[\ ]$:

$$\frac{}{\Gamma, z:0 \mid\!- [\ ]:\upsilon}$$

The following rules define type conversion for coproduct types

$$\frac{\Gamma \mid\!- M:\sigma \quad \Gamma, x:\sigma \mid\!- Q:\upsilon \quad \Gamma, x':\tau \mid\!- Q':\upsilon}{\Gamma \mid\!- unp\kappa_1 M\ as\ [\kappa_1 x\ in\ Q\ ,\ \kappa_2 x'\ in\ Q'] = Q[M/x]:\upsilon}$$

$$\frac{\Gamma \mid\!- N:\tau \quad \Gamma, x:\sigma \mid\!- Q:\upsilon \quad \Gamma, x':\tau \mid\!- Q':\upsilon}{\Gamma \mid\!- unp\kappa_2 N\ as\ [\kappa_1 x\ in\ Q\ ,\ \kappa_2 x'\ in\ Q'] = Q'[N/x']:\upsilon}$$

The last rule describes that if empty coproduct type variable is in the context, then every term with this context has to be convertible into empty cotuple

$$\frac{\Gamma, z:0 \mid\!- M:\upsilon}{\Gamma, z:0 \mid\!- [\ ]:\upsilon}$$

In constructing corresponding classifying category *Class2(Σ)* for $\lambda 2(\Sigma)$ calculus over Church's types we have the advantage that we can use types instead contexts. Finite product types ensure that any term $M:\tau$ with context

$$v_1:\sigma_1, \ldots, v_n:\sigma_n \mid\!- M:\tau$$

is in one-to-one correspondence with a term $N:\tau$ of the same type with a single variable of product type

$$v:\sigma_1\times\ldots\times\sigma_n \mid\!- N:\tau.$$

If $n = 0$ then $v:1$.

The category *Class2(Σ)* for $\lambda 2(\Sigma)$ calculus has then

- as objects Church's types $\sigma\in T_2$ constructed from basic types of signature Σ,
- as morphisms between types $\sigma\to\tau$ equivalence classes $[M]$ of terms with respect to conversion of types $v:\sigma \mid\!- M:\tau$.

It is easy to see that the empty coproduct type 0 is the *initial object* and the empty product type 1 is the *terminal object* of the classifying category *Class2(Σ)*. For every Church's type σ there is a term $z:0 \mid\!- [\ ]:\sigma$ and to every term $z:0 \mid\!- M:\sigma$ from the last conversion rule holds $z:0 \mid\!- M = [\ ]:\sigma$, so that the equivalence class

$$[M] = [[\ ]]: 0\to\sigma.$$

6. CHURCH'S FIBRATION

Until now we presented the sketch of a model of Church's type theory in usual, i.e. many-typed algebraic sense [7,8]. We intend to reason about scientific problem solving not only in the framework of many-typed algebras but also in the framework of Gentzen's logic, general type theory based on basic types in the framework of the language of categories. To do this we generalize the previous section in the theory of categories.

Let $\boldsymbol{B}$ be a category that is cartesian and has finite coproducts and which objects are set-theoretic structures as in section 4 above. We say that a model of classifying category *Class2(Σ)* is a functor

$$\mathcal{M}: Class2(\Sigma)\to\boldsymbol{B}$$

which assigns to every object σ in *Class2(Σ)* an object (carrier set) $[\![\sigma]\!]$ in $\boldsymbol{B}$ and to every morphism between Church's types $\sigma\to\tau$ in *Class2(Σ)* a morphism (function) between corresponding images.

Under proposition-as-types approach the Church's type theory corresponds to *proof theory* of propositional logic, where type constructors '$\to$' , '$\times$' and '+' correspond to logical connectives for implication '$\Rightarrow$ ' , for conjunction '$\wedge$' and for disjunction '$\vee$ ', respectively. Types 0 for empty coproduct type and 1 for empty product correspond with logical constants $\bot$ (bottom, false) and $\top$ (top, true), respectively. In this approach we can construct from propositions as objects and from propositional connectives as morphisms also cartesian category with finite coproducts.

In a fibred description of a type theory [5] the contexts form objects of a *base category* $\boldsymbol{B}$. These objects we can generalize as *indexing objects*. So, we can consider every set of Church's types $[\![T_2]\!]$, where T_2 is a set of Church's types over a signature Σ, as an indexing object in a base category of fibred category theory. Every indexing object of a base

category indexes objects of a *fibre category* $\boldsymbol{E}_I$ over this object in pointwise manner. Because the set of Church's types $[\![T_2]\!]$ of a signature Σ is actually a special case of an indexing object in a base category, we can also form Church's fibred category, whose objects are pointwise indexed by elements of this Church's type set $[\![T_2]\!]$.

Generally, let $\boldsymbol{B}$ be a cartesian category with finite coproducts. If we consider $\boldsymbol{B}$ as a base category, we construct a category $\boldsymbol{E}$ *of fibrations* which objects are indexed by objects (types) from $\boldsymbol{B}$ as follows:

- objects are pairs (I, X), such that I is indexing object and X is indexed object both from $\boldsymbol{B}$,
- morphisms $(u,f): (I, X) \to (J, Y)$ are pairs of morphisms $u: I \to J$ and $f: I \times X \to Y$ in $\boldsymbol{B}$,
- identity on object (I, X) is a pair (id_I , π_2), where $\pi_2: (I, X) \to I$ is second projection,
- a composition

$$(I, X) \xrightarrow{(u,f)} (J, Y) \xrightarrow{(v,g)} (K, Z)$$

is a pair $(v \circ u, g \circ \langle u \circ v, f \rangle)$, where

$$I \times X \xrightarrow{\langle u \circ \pi_1 , f \rangle} J \times Y \xrightarrow{g} Z$$

A projection functor

$$p: \boldsymbol{E} \to \boldsymbol{B}$$

from the category $\boldsymbol{E}$ of fibrations defined by

$$p(I, X) = I \text{ , and}$$
$$p(u, f) = u$$

is *Church's fibration on* $\boldsymbol{B}$. It is a fibration because for every object (J, Y) from $\boldsymbol{B}$ we can find cartesian lifting of $u: I \to J$ from $\boldsymbol{B}$ in the category $\boldsymbol{E}$ as a pair (u, π_2):

$$\begin{array}{ccc} \boldsymbol{E} & & (u, \pi_2) \\ \downarrow p & (I, Y) & \longrightarrow (J, Y) \\ \boldsymbol{B} & I & \xrightarrow{u} J \end{array}$$

For any fixed object I from the base category $\boldsymbol{B}$ the subcategory $\boldsymbol{E}_I$ of objects indexed by I is *fibre category over I*. Morphisms in $\boldsymbol{E}_I$ are *vertical morphisms*. This fibre category is also called *Church's slice category* and is denoted by $\boldsymbol{B}//I$. Its objects are objects X from $\boldsymbol{B}$ indexed by I and its morphisms $X \to Y$ are morphisms $I \times X \to Y$ in $\boldsymbol{B}$. These ideas we illustrate in Fig. 2.

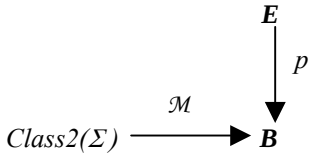

Fig. 2 Church's fibration

It is easy to see that such a classifying category and Church's slice category enable to start logical reasoning from *proposition-as-types* and *proofs-as-morphisms* approach [9]. Of course, this very simple logic allows to derive only simple results.

7. CONCLUSION

After introducing the Church's types we follow our research by defining such new type constructions in our logical reasoning that are able to capture not only the syntax of a logical language as it is excellently written in Gentzen's sequent calculus. We would like to construct these new type constructions in such a manner that they enable various semantics as algebraic topological, category theoretical and game semantics. We plain to use them in the development of an assistent system for scientific problem solving by mathematical machines.

This work was supported by VEGA Grant No.1/2181/05: Mathematical Theory of Programming and Its Application in the Methods of Stochastic Programming.

REFERENCES

[1] V.Novitzká: Logical Reasoning about Programming Mathematical Machines, *Acta Electrotechnica et Informatica,* 3,No.3,2005, pp.50-55

[2] B.Jacobs: *Categorical Logic and Type Theory,* Elsevier, Amsterdam, 1999

[3] J.-Y.Girard: Linear Logic, *Theoretical Computer Science,* 50, 1987, pp.1-102

[4] R.Hindley, J.P.Seldin: *Introduction to Combinators and λ calculus,* Cambridge University Press, 1990

[5] C.A.Hermida: *Fibrations, Logical Predicates and Indeterminates,* PhD. Thesis, Univ.Edinbourgh, 1993

[6] M.Barr, C.Wells: *Toposes, Triples and Theories,* Springer , 2002

[7] V. Novitzká,V. Novitzký: Metamathematical fundamental concepts of computer programming, In: Kátai, I.(Ed.): Annales Universitatis Scientiarum Budapestinensis de Rolando Eotvos Nominatae, Section Computatorica Vol.22, Budapest, Hungary, 2004, pp.193-212

[8] B.Ehrig, B.Mahr: *Fundamentals of Algebraic Specifications 1,2,* Springer , 1985, 1990

[9] J.-Y.Girard, P.Taylor, Y.Lafont: *Proofs and Types*, Cambridge University Press, 1990

BIOGRAPHY

Valerie Novitzká defended her PhD Thesis: On semantics of specification languages at Hungarian Academy of Sciences in 1989. She works at Department of Computers and Informatics from 1998, firstly as Assistent Professor, from 2004 as Associated Professor. Her research areas covers category theory, categorical logic, type theory, classical and linear logic and theoretical foundations of program development.