LINEAR LOGICAL REASONING ON PROGRAMMING

Valerie NOVITZKÁ, Daniel MIHÁLYI, Viliam SLODIČÁK
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, Daniel.Mihalyi@tuke.sk, Viliam.Slodicak@tuke.sk

SUMMARY

In our paper we follow the development of our approach of regarding programming as logical reasoning in intuitionistic linear logic. We present basic notions of linear logic and its deduction system and we define categorical semantics of linear logic as a symmetric monoidal closed category. Then we construct linear type theory over linear Church's types involving linear calculus with equational axioms. We conclude with the interpretation of the linear type theory in symmetric monoidal closed category. Defined entities included in our whole linear logical system give us a possible mean for deduction and reduction of problem solving in the framework of mathematics and computer science.

Keywords: *linear logic, deduction system, linear type theory, symmetric monoidal closed category*

1. INTRODUCTION

Linear logic was introduced by Girard [6] and it becames a natural mean for research and applications in computer science. We can apply linear logic and use its methods in many places of direct, indirect and special linear reasoning. It is able to describe systems changeable during they are used. We can pick up many applications of linear logic in the theory of concurrent processes [2,4]. Because in linear logic the storage applies directly along the types [1] the garbage machanism can be simplified. Moreover, there are several applications of linear logic in logic and functional programming [8,10].

In our previous works [12,13] we have sketched our approach to logical reasoning about programming. We introduced basic types and Church's types in categorical terms and we described categorical propositional deduction system. In this paper we deal with linear logic, we introduce linear deduction system, linear term calculus, linear types and categorical semantics of linear type theory.

2. LINEAR CONNECTIVES AND DEDUCTION RULES

Formulas in linear logic describe (intuitively) *actions*. While classical and intuitionistic logics treat with the sentences that are always true or false, in linear logic the truth value of so called „facts" depends on the internal state of a dynamic system. Classical and intuitionistic logics are included in linear logic. Linear logic presented by Girard [6] is often called classical linear logic because it has an involutive „negation" reminiscent of the negation in classical logic. Because our aim is to describe a construction of problem solution, we consider only intuitionistic linear logic. First, we introduce the syntax of linear formulas with informal semantics, in the following section we deal with formal categorical semantics of it.

Let $Prop=\{ p_1 , p_2 , ...\}$ be a countably infinite set of *atomic propositions* $p_1 , p_2 , \dots$. *A linear formula* φ can be of the form written by the following BNF grammar:

$$\varphi ::= \mathbf{0} \mid \mathbf{1} \mid \top \mid p_n \mid \varphi_1 \otimes \varphi_2 \mid \varphi_1 \multimap \varphi_2 \mid \varphi_1 \,\&\, \varphi_2$$
$$\mid \varphi_1 \oplus \varphi_2 \mid \varphi_1 \parr \varphi_2$$

The symbol „$\multimap$ " denotes *linear implication*. Linear implication is causal, i.e. the action described by φ_1 is a cause of the action described by φ_2 but φ_1 does not hold after linear implication. For instance, let φ_1 be „we have some amount of money" and φ_2 be „we buy some thing". The linear implication $\varphi_1 \multimap \varphi_2$ can be read „if we have some amount of money then we buy some thing". But after this implication is performed we have no money, i.e. φ_1 does not hold. Linear implication of such form cannot be iterated because its condition is modified after its use.

The symbol „$\otimes$ " is read „times" and denotes *multiplicative conjunction*. Its neutral element is the constant $\mathbf{1}$. Multiplicative conjunction $\varphi_1 \otimes \varphi_2$ expresses that both actions described by linear formulas φ_1 and φ_2 will be performed.

The symbol „&" is read „with" and it denotes *additive conjunction*. Its neutral element is the constant $\top$ (top). Additive conjunction $\varphi_1 \,\&\, \varphi_2$ expresses that only one of the actions described by φ_1 and φ_2 will be performed and we shall decide which one.

The symbol „$\oplus$ " is read „plus" and denotes *additive disjunction* with the constant $\mathbf{0}$ as its neutral element. It expresses that only one of the actions described by φ_1 and φ_2 will be performed but we cannot decide which one. Additive disjunction is dual of additive conjunction.

Dual of multiplicative conjunction is *multiplicative disjunction* that uses the symbol „$\parr$"and is read as „par" with the constant $\perp$ (bottom) as neutral element. A linear formula $\varphi_1 \parr \varphi_2$ expresses that if the action described by φ_1 is not performed then the action described by φ_2 is done or if the action described by φ_2 is not performed then the action described by φ_1 is done. Because multiplicative disjunction requires multiple conclusions in deduction rules [11] we do not use it in our approach.

Basic components of a *linear deduction system* are sequents written in Gentzen's style. A *sequent* of linear logic has a form

$$\Gamma \vdash \varphi$$

where Γ is a finite (possibly empty) list $\psi_1, \ldots, \psi_n$ of linear formulas and φ is a linear formula. Γ is called a *context* and consists from the assumptions for φ.

The *deduction system* of intuitionistic linear logic consists of basic and structural deduction rules together with introduction rules for the linear connectives introduced above. We have only one basic rule, *identity* rule that is the axiom

$$\frac{}{\varphi \vdash \varphi}\;(ident)$$

expressing that from the context φ is derivable φ. Intuitionistic linear logic has only two structural rules: *cut* rule and *exchange* rule:

$$\frac{\Gamma_1, \varphi \vdash \psi \quad \Gamma_2 \vdash \varphi}{\Gamma_1, \Gamma_2 \vdash \psi}\;(cut)$$

$$\frac{\Gamma_1, \varphi_1, \varphi_2, \Gamma_2 \vdash \psi}{\Gamma_1, \varphi_2, \varphi_1, \Gamma_2 \vdash \psi}\;(exchange)$$

The cut rule expresses that if from a context Γ_1, φ is derivable ψ and from a context Γ_2 is derivable φ we can exclude φ, i.e. ψ is derivable from the context Γ_1, Γ_2. The exchange rule expresses that the order of linear formulas in contexts is not important.

Linear logic has only these two structural rules and it is often called a logic without *weakening* and *contraction*. (Later in this section we introduce the restricted versions of weakening and contraction.) In linear reasoning it is important *which* assumptions and *how many times* is an assumption asserted. Girard noted in [7] that the undecibility and nonconstructivity of the classical predicate calculus follows just from the contraction rule.

Now we write the introduction rules for linear connectives. The constant ***1*** as the neutral element of multiplicative conjunction can be introduced on the left and/or right side of the sequent:

$$\frac{\Gamma \vdash \varphi}{\Gamma, \mathbf{1} \vdash \varphi}\;(\mathbf{1}\text{-}l) \qquad \frac{}{\vdash \mathbf{1}}\;(\mathbf{1}\text{-}r)$$

Multiplicative conjunction has two introduction rules the first for introducing $\otimes$ on the left and the second on the right side of the sequent:

$$\frac{\Gamma, \varphi_1, \varphi_2 \vdash \psi}{\Gamma, \varphi_1 \otimes \varphi_2 \vdash \psi}\;(\otimes\text{-}l) \qquad \frac{\Gamma_1 \vdash \varphi \quad \Gamma_2 \vdash \psi}{\Gamma_1, \Gamma_2 \vdash \varphi \otimes \psi}\;(\otimes\text{-}r)$$

Similarly, linear implication can be introduced on the left and/or right side of the sequent as it express the following two rules:

$$\frac{\Gamma_1 \vdash \varphi_1 \quad \Gamma_2, \varphi_2 \vdash \psi}{\Gamma_1, \varphi_1 \multimap \varphi_2, \Gamma_2 \vdash \psi}\;(\multimap\text{-}l) \qquad \frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \multimap \psi}\;(\multimap\text{-}r)$$

Multiplicative disjunction has two introduction rules for introducing $\&$ on the left side and one rule for introducing this connective on the right side of a sequent:

$$\frac{\Gamma, \varphi_1 \vdash \psi}{\Gamma, \varphi_1 \& \varphi_2 \vdash \psi}\;(\&\text{-}l1) \qquad \frac{\Gamma, \varphi_2 \vdash \psi}{\Gamma, \varphi_1 \& \varphi_2 \vdash \psi}\;(\&\text{-}l2)$$

$$\frac{\Gamma \vdash \varphi \quad \Gamma \vdash \psi}{\Gamma \vdash \varphi \& \psi}\;(\&\text{-}r)$$

Additive constants ***0*** and $\top$ has the following introduction rules:

$$\frac{}{\Gamma, 0 \vdash \varphi}\;(0\text{-}l) \qquad \frac{}{\Gamma \vdash \top}\;(\top\text{-}r)$$

We note that the constant ***0*** has only left introduction rule while the constant $\top$ has only right introduction rule.

Additive disjunction has two right introduction rules and one left rule as follows:

$$\frac{\Gamma, \varphi_1 \vdash \psi \quad \Gamma, \varphi_2 \vdash \psi}{\Gamma, \varphi_1 \oplus \varphi_2 \vdash \psi}\;(\oplus\text{-}l)$$

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \varphi \oplus \psi}\;(\oplus\text{-}r1) \qquad \frac{\Gamma \vdash \psi}{\Gamma \vdash \varphi \oplus \psi}\;(\oplus\text{-}r2)$$

Example 2.1: Using deduction rules of intuitionistic linear logic defined above we can prove some simple formulas as: $\varphi \otimes (\psi \& \upsilon) \vdash (\varphi \otimes \psi) \& (\varphi \otimes \upsilon)$. A proof is the following tree:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\varphi \vdash \varphi} \quad \dfrac{\dfrac{}{\psi \vdash \psi}\,(ident)}{\psi \& \upsilon \vdash \psi}\,(\&\text{-}l1)}{\varphi, \psi \& \upsilon \vdash \varphi \otimes \psi}\,(\otimes\text{-}r)}{\varphi \otimes (\psi \& \upsilon) \vdash \varphi \otimes \psi}\,(\otimes\text{-}l) \quad \dfrac{\dfrac{\dfrac{}{\varphi \vdash \varphi} \quad \dfrac{\dfrac{}{\upsilon \vdash \upsilon}\,(ident)}{\psi \& \upsilon \vdash \upsilon}\,(\&\text{-}l2)}{\varphi, \psi \& \upsilon \vdash \varphi \otimes \upsilon}\,(\otimes\text{-}r)}{\varphi \otimes (\psi \& \upsilon) \vdash \varphi \otimes \upsilon}\,(\otimes\text{-}l)}{\varphi \otimes (\psi \& \upsilon) \vdash (\varphi \otimes \psi) \& (\varphi \otimes \upsilon)}\,(\&\text{-}r)$$

□

So defined linear logic is extremely weak and we can prove only a few assertions by this deduction system. It seems reasonable to allow repeating of some linear formulas. Such repeating is denoted by the symbol „*!* " called *exponential* and the formula $!\varphi$ describes the action that can be repeated more times, i.e. this formula has stable truth value. For instance, in the linear implication $!\varphi \multimap \psi$, φ has the same truth value after the implication as before it (we have enough money to buy more things). The exponential enables to reintroduce the structural rules for weakening and contraction but in controlled manner. We say that these rules are used only for modal formulas $!\varphi$. We denote by $!\Gamma = !\varphi_1,...,!\varphi_n$ for $\Gamma = \varphi_1,..., \varphi_n$. The introduction rules for the exponentials are

$$\frac{\Gamma,\varphi \vdash \psi}{\Gamma,!\varphi \vdash \psi}\ (!\text{-l}) \qquad \frac{!\Gamma \vdash \psi}{!\Gamma \vdash !\psi}\ (!\text{-r})$$

Restricted weakening and contraction rules are then

$$\frac{\Gamma \vdash \psi}{\Gamma, !\varphi \vdash \psi}\ (weak) \qquad \frac{\Gamma, !\varphi, !\varphi \vdash \psi}{\Gamma, !\varphi \vdash \psi}\ (contr)$$

This means that we can extend a context of a sequent only with a modal (repeatable) formula and we can exclude one of two same formulas from a context only if they are modal (repeatable).

3. CATEGORICAL SEMANTICS OF INTUITIONISTIC LINEAR LOGIC

In the literature there are many approaches to the semantics of linear logic. The first and the simplest is the phase semantics by Girard [7], further there are consequence algebras, quantales [11], coherence spaces, resource semantics [5] and many others. We prefer *symmetric monoidal closed categories* as a semantics for introduced linear intuitionistic logic for the following facts:

- In the indirect reasoning presented in [13] we constructed from basic types the Church's types representable as objects in cartesian closed categories. Symmetric monoidal categories are a generalisation of them.

- Every consequence algebra and quantale is a symmetric monoidal closed category when it is viewed as a category.

- For any symmetric monoidal closed category there is a linear type theory whose model is this category [2].

Because we have introduced the exact definition of symmetric monoidal closed categories in [12], we mention here only the essence of it.

A *symmetric monoidal closed category* is a sixtuple

$$(\mathbf{C}, \otimes, \mathbf{I}, a, l, c, hom(-,-))$$

where

- $\mathbf{C}$ is a category;

- $\otimes : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ is a tensor functor;

- $\mathbf{I}$ is an object in $\mathbf{C}$, the neutral element of the tensor product;

- $a_{X,Y,Z} : (X \otimes Y) \otimes Z \to X \otimes (Y \otimes Z)$ is a natural isomorphism expressing *(left) associativity* of tensor product, where X, Y, Z are objects in $\mathbf{C}$;

- $l_X : \mathbf{I} \otimes X \to X$ is a natural isomorphism expressing *(left) neutral element* of tensor product;

- $c_{X,Y} : X \otimes Y \to Y \otimes X$ is a natural isomorphism expressing *commutativity* of tensor product;

- for components of this category hold *coherence axioms* presented by commutative diagrams in [12];

- closedness is defined by the following property: for every object X in $\mathbf{C}$ the functor $- \otimes X$ has a right adjoint hom-functor $hom\ (X,-)$ with natural transformations

$$\varepsilon_{X,Y} : Hom(X,Y) \otimes X \to Y$$

$$\delta_{X,Y} : X \to Hom\ (Y, X \otimes Y).$$

Example 3.1: If $\mathbf{C}$ is a category with finite products then the functor $\otimes$ is given by cartesian category product, $\mathbf{I}$ is a terminal object of the category $\mathbf{C}$ and natural isomorphisms are given by appropriate combinations of projection morphisms and pairing. □

4. LINEAR TYPE THEORY

In this section we formulate linear type theory and in the following one we interpret this theory in symmetric monoidal closed categories.

Let $\mathbf{B} = \{X, Y, Z, ...\}$ be a set of basic types and let I be the unit type. From basic types we construct *Church's linear types* denoted by $A, B, C, ...$ as follows

$$A ::= I \mid X \mid A \otimes B \mid [A,B]$$

i.e. all basic types and unit type I are Church's linear type, $A \otimes B$ is a product linear type and $[A, B]$ is a function linear type, that is the set of functions from a type A to the type B. We denote by $\mathbf{ChT}$ the set of all linear Church's types.

Let $\mathbf{F}$ be a set of function symbols of the form $f: A \to B$, where $A, B \in \mathbf{ChT}$. We introduce the special function symbols, so called *combinators* as follows:

$$Id_A : A \to A$$

$$assl_{A,B,C} : A \otimes (B \otimes C) \to (A \otimes B) \otimes C$$

$$assr_{A,B,C} : (A \otimes B) \otimes C \to A \otimes (B \otimes C)$$

$$swap_{A,B} : A \otimes B \to B \otimes A$$

$$open_A : A \to I \otimes A$$

$$close_A : I \otimes A \to A$$

$$eval_{A,B} : [A,B] \otimes A \to B$$

From function symbols and combinators we can construct generalized combinators by the following rules of *composition, product* and *abstraction*. Let $\alpha, \beta, \gamma, \ldots$ denote function symbols (or combinators):

$$\frac{\alpha: A \to B \quad \beta : B \to C}{\beta \circ \alpha: A \to C} \quad (composition)$$

$$\frac{\alpha: A \to B \quad \beta : C \to D}{\alpha \otimes \beta : A \otimes C \to B \otimes D} \quad (product)$$

$$\frac{\alpha: A \otimes B \to C}{\Lambda(\alpha) : A \to [B,C]} \quad (abstraction)$$

To define linear calculus, we assume for every type $A \in$ ***ChT*** a countably infinite set *var(A)* of variables of the type *A*. For every linear Church's type *A* let *preterm(A)* be a set of all preterms of type *A* defined inductively as follows:

- $() \in preterm\ (I)$ is the empty preterm;
- if $x \in var(A)$ then $x \in preterm(A)$;
- if $s \in preterm\ (A)$ and $t \in preterm\ (B)$ then $(s,t) \in preterm(A \otimes B)$;
- if $s \in preterm(A)$ and $\alpha: A \to B$ is a function or combinator then $\alpha(s) \in preterm(B)$.

A preterm $s \in preterm(A)$ defined as above is a *term* of a type *A* if no variable occurs more than once in *s*.

Example 4.1:
a) A preterm s of the form

$$s = (f(x), eval_{A,B}(f, y))$$

where $f: A \to B$, $x, y \in var\ (A)$, $A, B \in$ ***ChT*** is a term of the type $B \otimes B$;

b) A preterm t of the form

$$t = swap_{A,A}(x, x)$$

where $x \in var\ (A)$, for any $A \in$ ***ChT*** is not a term, only a preterm of type $(A \otimes A)$ because it has two occurrences of the variable *x*.

□

We denote by *term(A)* the set of all terms of type *A*. A term *s* is a *basic term* if it contains no combinators, i.e. it is built only from variables, function symbols and brackets „ ("and„) ".

Example 4.2:
a) A variable $x \in var\ (A)$ is a basic term $x \in term(A)$;

b) A pair of variables (x, y), where $x \in var\ (A)$ and $y \in var\ (B)$ is a basic term (variable of product type)

$$v = (x , y) \in term\ (A \otimes B).$$

c) A term $s = (f(x), eval_{A,B} (f, y))$ is not a basic term because it contains the combinator $eval_{A,B}$.

□

Variables in preterms can be *substituted* by the preterms of the same type. Let *s* be a preterm, $s \in preterm\ (A)$ and *x* be a variable of the same type, $x \in var\ (A)$. We denote by $t[s / x]$ a preterm *t* where all occurrences of *x* are replaced by *s* using the following rules:

- $() [s / x] = ()$;
- $y [s / x] = \begin{cases} s, & \text{if } x = y \\ y, & \text{otherwise} \end{cases}$
- $(t, u) [s / x] = (t [s/ x], u [s/ x])$;
- $\alpha (t) [s / x] = \alpha (t[s / x])$;

where *t, u* are preterms of any type and α is a fuction symbol or combinator between appropriate types.

By the induction on the structure of a term *t* it can be proved that if the terms *t* and *s* have no common variables then $t [s / x]$ is a term.

To construct linear type theory we have to introduce an equivalence relation on linear terms which enables us to define axioms of linear type theory. We denote such equivalence relation by the symbol „$\equiv_A$ " and define that terms $s, t \in term(A)$ are equivalent, written

$$s \equiv_A t$$

if and only if s and t have the same variables of the same types. If it does not lead to confusion, we omit the type subscript.

A set ***E*** of (equational) axioms of linear type theory consists of the following axioms. Let *s, t, u* be the terms of appropriate linear types and α, β, γ be function symbols, then

$$Id_A\ (s) \equiv s$$

$$(\gamma\ o\ \alpha\ )\ (s) \equiv \gamma\ (\alpha\ (s\ ))$$

$$assl_{A,B,C}\ (s,\ (t,\ u)) \equiv ((s,t\ ),u)$$

$$assr_{A,B,C}\ ((s,\ t),\ u) \equiv (s,\ (\ t,u\ ))$$

$$swap_{A,B}\ (s,\ t) \equiv (\ t,\ s)$$

$$open_A\ (s) \equiv (\ (\ ),\ s)$$

$$close_A\ (\ (\ ),\ s\ ) \equiv s$$

$$eval_{A,B}\ (\ \Lambda\ (\alpha\ )(s),\ t) \equiv \alpha\ (s,\ t)$$

We denote by ***LinTT(B,F,E)*** the linear type theory over the set ***B*** of basic types, where ***F*** is a set of function symbols with combinators and where ***E*** is a set of linear axioms. Linear Church's types are generated from basic types, linear terms are constructed from variables of linear Church's types and function symbols and the equational axioms of linear type theory introduced above hold.

5. INTERPRETATION OF LINEAR TYPE THEORY

The linear type theory ***LinTT(B,F,E)*** introduced in the previous section *we interpret* in a symmetric monoidal closed category $(\mathbf{C}, \otimes, \mathbf{I}, a, l, c, hom(-,-))$ defined in Section 3 as follows.

First we define a *type interpretation function* i as

$$i : \mathbf{B} \rightarrow Obj(\mathbf{C})$$

which assigns to every basic type $X \in \mathbf{B}$ an object $i(X)$ in $\mathbf{C}$. We extend this type interpretation function to the linear Church's types as follows:

$$i\ (I) = \mathbf{I}$$

$$i\ (\ A \otimes B) = i\ (A)\ \otimes i\ (B)$$

$$i\ ([A,B]) = hom\ (\ i\ (A),\ i\ (B))$$

where I on the left side of the first equation is the unit type and $\mathbf{I}$ on the right side of it is a terminal object in $\mathbf{C}$.

Interpretation of function symbols and combinators in the category $\mathbf{C}$ requires another *function interpretation mapping* j, which assigns to every function symbol $f \in \mathbf{F}$ of the form $f\colon A \rightarrow B$ a category morphism

$$j\ (f)\colon i\ (A) \rightarrow i\ (B)$$

in $\mathbf{C}$. We can naturally extend this function interpretation mapping to the combinators as follows:

$$j\ (Id_A) = id_{i(A)}$$

$$j\ (\ assr_{A,B,C}) = a_{i(A),\ i(B),\ i(C)}$$

$$j\ (assl_{A,B,C}) = a^{-1}_{i(A),\ i(B),\ i(C)}$$

$$j\ (swap_{A,B}) = c_{i(A),\ i(B)}$$

$$j\ (open_A) = l_{i(A)}$$

$$j\ (close_A) = l^{-1}_{i(A)}$$

$$j\ (eval_{A,B}) = \varepsilon_{i(A),\ i(B)}$$

where $id_{i(A)}$ is the identity morphism on an object $i(A)$ in the category $\mathbf{C}$.

If $\alpha : A \rightarrow B$ and $\beta : B \rightarrow C$ are composable combinators then

$$j\ (\ \beta\ o\ \alpha\ ) = j\ (\beta\ )\ o\ j\ (\alpha\ )$$

and

$$j\ (\ \alpha \otimes \beta\ ) = j\ (\alpha\ ) \otimes\ j\ (\ \beta\ ).$$

If $\Lambda(\alpha)\colon A \rightarrow [B,\ C]$ then $j\ (\Lambda\ (\alpha\ ))$ is the transpose of $j\ (\alpha\ )$ across the adjunction

$$— \otimes i\ (B) \dashv hom\ (i\ (B),\ —),$$

that is

$$j\ (\Lambda\ (\ \alpha\ ))\colon i\ (A) \rightarrow i\ (\ [\ B,\ C\ ]).$$

We can conclude: an interpretation of the linear type theory ***LinTT(B,F,E)*** in a symmetric monoidal closed category $(\mathbf{C}, \otimes, \mathbf{I}, a, l, c, hom(-,-))$ is a pair of functions i, j defined above

$$(\ i,\ j\ ) : \mathbf{LinTT(B,F,E)} \rightarrow (\mathbf{C}, \otimes, \mathbf{I}, a, l, c, hom(-,-))$$

with the following property: every basic term t such that the equivalence

$$\alpha\ (t) \equiv \beta\ (t)$$

is derivable from the set ***E*** of linear axioms implies that the interpretation of combinators is the same morphism

$$j\ (\ \alpha\ ) = j\ (\ \beta\ )$$

in the symmetric monoidal closed category $(\mathbf{C}, \otimes, \mathbf{I}, a, l, c, hom(-,-))$.

6. CONCLUSION

In our paper we introduce linear types, linear calculus and formulate linear axioms and so we include these entities into our whole linear logical system as a possible means for deduction and reduction of problem solving in the framework of mathematics and computer science.

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] S. Abramsky: Computational interpretations of linear logic, Tech.Report DOC 90/20, Imperial College, London 1990

[2] S. J. Ambler: First order linear logic in symmetric monoidal closed categories, PhD.Thesis, University of Edinburgh, 1991

[3] R. Blute, P. Scott: Category theory for linear logicians, Proc. Linear Logic Summer School, Cambridge Univ. Press, 2004

[4] C. T. Brown: Linear logic and Petri nets, PhD.Thesis, University of Edinburgh, 1990

[5] M. Dam: Relevance logic and concurrent composition, PhD. Thesis, University of Edinburgh, 1990

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

[7] J. Y. Girard: Linear Logic: Its Syntax and Semantics, In: J. Y. Girard, Y. Lafont, and L. Regnier, editors, Advances in Linear Logic, Cambridge, 1995, pp. 1-42

[8] J. Harland, M. Winikoff: Some applications of the linear logic programming language Lygon, Proc. 19th Australian Comp.Sc.Conference, melbourne, 1996, pp.262-271

[9] J. M. E. Hyland, V. C. V de Paiva: Full intuitionistic linear logic, Annals of Pure and Applied Logic, 64,3, 1993, pp.273-291

[10] I. Mackie: Lilac – a functional programming language based on linear logic, Journal of Functional Programming, 4,4,1993, pp.395-433

[11] C. J. Mulvey: „&", Suppl. Ai Rend. Del Circ. Mat. Di Palermo, Serie II, 12, Palermo, 1986, pp.99-104

[12] V. Novitzká: Logical reasoning about programming of mathematical machines, Acta Electrotechnica et Informatica, 5,3,2005, Košice, pp.50-55

[13] V. Novitzká: Church's types in logical reasoning on programming, 2005, (submitted)

BIOGRAPHIES

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.

Daniel Mihályi is an researcher at the Department of Computers and Informatics since 1989. Now he works on his PhD. Thesis. His main areas of research activities and interests are categorical logic, linear logic and logical reasoning in applied mathematics and programming. He is interested also in analysis and methodology design of program systems security in Unix operating systems environment and internet services technologies.

Viliam Slodičák was born in 1981. He graduated at Technical university of Košice, Slovakia. He is working on his PhD. degree at the Department of Computers and Informatics FEEI, Technical university of Košice, Slovakia. His scientific research area are topos theory, categorical logic and linear logic.