What about Linear Logic in Computer Science?

Daniel Mihályi, Valerie Novitzká

Department of Computers and Informatics, Technical University of Košice
Košice, Slovakia
E-mail: Daniel.Mihalyi@tuke.sk, Valerie.Novitzka@tuke.sk

Abstract: In this paper we discuss several useful features of linear logic especially from the viewpoint of computing science. We start with a short overview of linear logic with an emphasis on the special properties of linear implication and exponential operators. We present our idea of the possible fragmentation of linear logic and the usefulness of particular fragments in various areas of computing science. Finally, we consider possible extensions of linear logic and we illustrate how an extension with epistemic operators can serve for obtaining knowledge and belief about an intrusion attempt.

Keywords: linear logic; type theory; behavioral theory; modal logics

1 Introduction

Linear logic was introduced by J. Y. Girard in 1987 [6] as a non-classical logic of actions and resources enabling one to describe dynamics of processes and resource handling. This logic can be considered as a suitable interface between logic and computing science because it can manipulate with the events of real world in natural way. Linear logic is a new logic, but the whole classical logic can be translated into linear formulae [3].

From the computing science's point of view, for intuitionistic fragment of linear logic the Curry-Howard correspondence [21] is valid, i.e. the formulae of linear logic correspond with the types of data structures. Similarly, the proof trees of the sequent calculus of linear logic correspond with programs [7]. If we consider formulae as resources, within the realization of a proof they are distributed in time and space [5] in some model of the real world, e.g. a computer machine in a precise and controlled manner. During our research, we have recognized several interesting properties and possibilities of linear logic:

- We have used an intuitionistic fragment of linear logic to formally describe program execution [17], [20], [22], [24];
- We have used linear logic to define linear type theory [15];

- In the sense of Curry-Howard correspondence, functional programming can be regarded as logical reasoning in linear logic. Linear proofs enable us to anticipate computability and correctness of computing [18], [19];
- Formulae are equivalent with some paterns of Petri nets [8], [9], [15];
- Extending linear logic with modal operators of necessity and possibility we have used modal linear logic for reasoning about the observable behavior of programs [13];
- Extending linear logic with epistemic operators of knowledge and belief we obtained epistemic linear logic useful for achieving experiences about incomming network intrusions based on a natural manner of causalities [13], [14].

The aim of this paper is to discuss several interesting features of linear logic and the possible applications of linear logic in several areas of computing science. We consider propositional linear logic. The second section contains a short introduction to linear logic with special emphasis on its modal operators and on the static and dynamic nature of linear implication. In the third section, we present our view of linear logic fragmentation that can serve for different purposes in various areas of computing science. In the fourth section, we show how classical logic can be expressed by linear logic. The fifth section shows the correspondence between linear logic and linear type theory. In the sixth section, we show how an extension of linear logic with epistemic modalities of knowledge and belief can provide useful information about the behaviour of programs.

2 Linear Logic Overview

In this section we introduce the basic notions of linear logic. Let $Props=\{p_1, p_2...\}$ be a countable set of atomic propositions denoted by the letters $p_1,p_2....$ Any proposition can be considered in two ways: as an action or as a resource. A linear formula φ can be of the form defined by the following BNF rule:

$$\varphi ::= p_n \mid 0 \mid 1 \mid \bot \mid \top \mid !\varphi \mid ?\varphi \mid \wedge\varphi \mid \vee\varphi \mid \varphi^{\perp} \mid \varphi_1 \otimes \varphi_2 \mid \varphi_1 \,\&\, \varphi_2 \mid \varphi_1 \oplus \varphi_2 \quad (1)$$

$$\mid \varphi_1 \wp \varphi_2 \mid \varphi_1 -\circ \varphi_2$$

Linear logic has two conjunction operators and two disjunction operators. We describe an informal meaning of linear connectives:

- *Linear implication* $\varphi_1 \multimap \varphi_2$ is causal; - it expresses that an action described by φ_1 is a cause of the (re)action described by φ_2. If we consider resources, a resource φ_1 is consumed after linear implication, i.e. it becomes a linear negation $\varphi_1^{\perp}$;

- *Multiplicative conjunction* ("*times*") $\varphi_1 \otimes \varphi_2$ has the neutral element **1**. It expresses that both actions φ_1 and φ_2 will be performed simultaneously or that we have both resources φ_1 and φ_2 at once.
- *Additive conjunction* (*"with"*) $\varphi_1 \,\&\, \varphi_2$ has the neutral element $\top$. It expresses that only one of the actions described by φ_1 and φ_2 will be performed. But we can deduce or anticipate from an environment which of them will be performed. This formula can be considered as an analogy with the statements *if-then-else* and *case* in programming languages. Somethimes it is called *external nondeterminism* (dependent choice);
- *Additive disjunction* (*"plus"*) $\varphi_1 \oplus \varphi_2$ has the neutral element **0**. It expresses that only one of the actions described by φ_1 and φ_2 will be performed (or only one of these resources is available), but we cannot anticipate which one. It can be considered *internal nondeterminism* (free choice);
- *Multiplicative disjunction* (*"par"*) $\varphi_1 \parr \varphi_2$ has the neutral element $\bot$ and its meaning can be expressed as follows: if an action φ_1 is not performed, then an action φ_2 is done or vice versa; if an action φ_2 is not performed, then an action φ_1 is done. Multiplicative disjunction can be regarded as an allegory of the well-known construct *xor* in programming;
- *Linear negation* $\varphi^{\perp}$ denotes a reaction of an action φ or a consumption of a resource φ. Linear negation is involutive, i.e.

$$\varphi^{\perp\perp} \equiv \varphi \tag{2}$$

2.2 Linear Exponentials

Another special property of linear logic represents two unary operators called exponentials. These operators can be considered from two points of view: concerning resources or concerning modalities. If we consider resources, then

- the operator "!" expresses potential resource inexhaustibility and
- the operator "?" expresses the actuality of potential resource inexhaustibility. .

Exponentials are dual, i.e.

$$(!\varphi)^{\perp} \equiv ?(\varphi^{\perp}) \tag{3}$$

Duality between exponentials can be considered as the difference between actual and potentional infinity [26]. The formula *(!φ)* expresses an unexhausted store of a resource φ and the formula $?(\varphi^{\perp})$ expresses potentional replenishment of exhausted resources. For instance, if we consider a resource φ to be a part of computer memory, we can indicate the potentional need to extend it. The resource character of exponentials are in the Table 1.

Table 1
Linear exponentials dealing with resources

Operator	Resource view	Modal view
!	unexhaustibility	*of course*
?	potential unexhaustibility (depending on actual replenishment)	*why not*

In terms of modality:

- the operator "!" ("*of course*") expresses obviousity and
- the operator "?" ("*why not*") expresses polemic.

Linear exponentials can be considered as linear alternatives of traditional modalities of necessity ("□") and possibility ("◊"), respectively, as is shown in the Table 2.

Table 2
Modal nomenclature in linear logic manner

	∇_1	∇_2
Modal logic	◊	□
	Possibility	Necessity
Linear logic	?	!
	Polemic	Obviousity

Linear exponentials are necessary also for translating classical propositional logic into linear logic. We consider this translation in the Section 4. The exponential "*of course*" can also serve for expressing the repeating of some actions [14].

2.3 Static and Dynamic Nature of Implication

Classical logic has an obvious implication $\varphi_1 \Rightarrow \varphi_2$ with a static character. Ituitionistic propositional logic knows also weaker implication called partial implication $\varphi_1 \Rightarrow_p \varphi$ corresponding with linear partial functions under the Curry-Howard correspondence [4]. Both these implications can be translated to linear formulae $!\varphi_1 \multimap \varphi_2$ and $\varphi_1 \multimap ?\varphi_2$, respectively, thanks to exponentials, as we show in Table 3. Traditional linear implication $\varphi_1 \multimap \varphi_2$ has a dynamic character; its premise φ_1 is consumed after performing the linear implication. If we consider formulae as actions, we can say that an action φ_2 follows an action φ_1. From Table 3 we can see that linear logic has more forms of implication, and so linear logic has greater expressive power. In addition, if we combine translated forms we can get a generalized form of linear implication $!\varphi_1 \multimap ?\varphi_2$ that can be particularly useful for programming languages with recursion [4].

Table 3
Forms of linear implications

Classical view	Linear view	Kinds of linearity
	$\varphi_1 \multimap \varphi_2$	Linear implication
$\varphi \Rightarrow \varphi$	$!\varphi_1 \multimap \varphi_2$	Unrestricted linear implication
$\varphi \Rightarrow_p \varphi$	$\varphi_1 \multimap ?\varphi_2$	Partial linear implication
	$!\varphi_1 \multimap ?\varphi_2$	Generalized linear implication

In linear logic we can choose whether we would like to work in static mode or in dynamic mode. If we translate classical implication $\varphi_1 \Rightarrow \varphi$ into $!\varphi_1 \multimap \varphi_2$, we work in static mode. We also note that classical implication $\varphi_1 \Rightarrow \varphi_2$ is equivalent with disjunction:

$$\varphi_1 \Rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2 \tag{4}$$

Translating the left and right parts of the previous formula into linear logic, we get the following equivalence:

$$\varphi_1 - \circ \varphi_2 \equiv \varphi_1^{\perp} \oplus \varphi_2 \tag{5}$$

but this is not valid by [6] because there exist two proof trees for the linear formula on the right side.

If we would like to consider dynamically, linear implication $\varphi_1 \multimap \varphi_2$ can be understood that an action φ_2 follows an action φ_1 , i.e. an action φ_2 starts after an action φ_1 . In contrast to the previous case, the following equivalence of linear formulae is valid:

$$\varphi_1 - \circ \varphi_2 \equiv \varphi_1^{\perp} \parr \varphi_2 \tag{6}$$

3 Linear Logic Fragmentation

Linear logic can be used as a whole, but in some cases it is appropriate to consider only a fragment of linear logic. In this section, we present an overview of how linear logic can be fragmented into several blocks according to the a nature of the particular fragments [12]. We illustrate our ideas of possible fragmentations in Figure 1.

First, we consider the vertical ellipses. The left one contains the multiplicative fragment of linear logic, and the right one contains the additive fragment of linear logic together with the corresponding constants. Linear implication and linear negation are neutral, they play important role in both fragments.

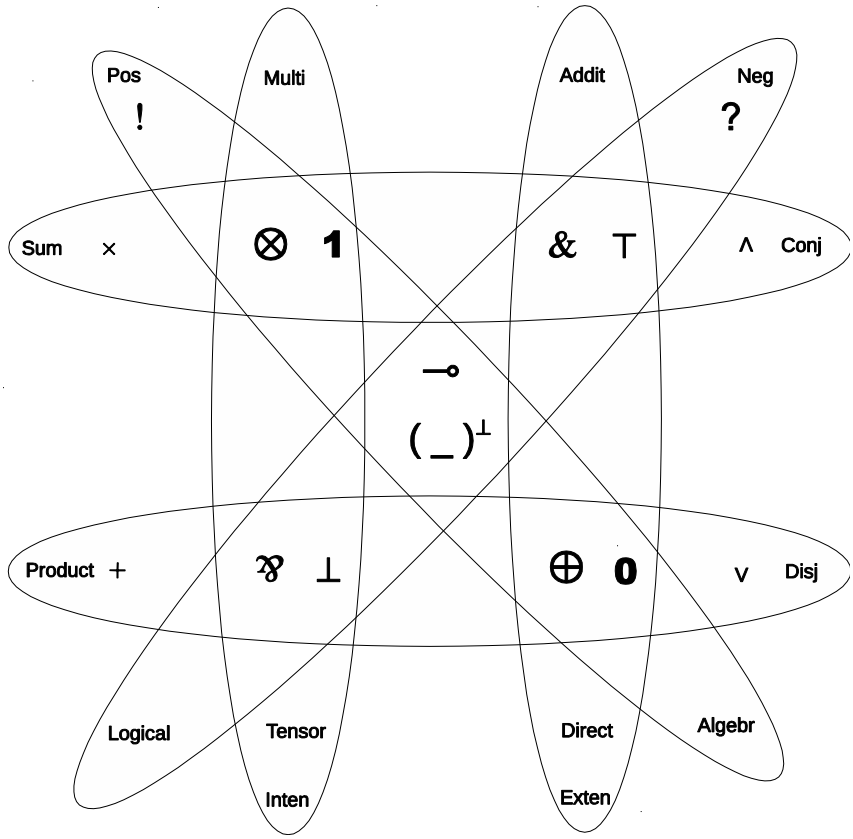

Figure 1
Linear logic fragmentation

From the semantic point of view we can consider the left ellipse as the intensional frangment and the right one as the extensional frangment. This arises from the semantical notions of extension and intension [2]. Whereas the extension of a given concept is its subject or the family of subjects included within it, the intension is the content of it. The extension of a given action is a truth value in the Tarski tradition; – the intension is an idea (sense) expressing it, – in the Heyting tradition. Extension we understand as a denotation and intension we identify as a sense. Traditionally, atomic propositions in (the Tarski tradition) are assertions that have exactly true or false truth values. In the extensional fragment of linear logic we assign to linear formulae the truth values (**1** or ⊥). But in the intensional fragment we consider their sense or nonsense (⊤ or **0**). For instance, if we have atomic proposition *Snowing*, it can be valid (**1**) and it has also sense (⊤). But the atomic proposition *Spowing* has no sense (⊥) and neither can it be valid (**0**). This also demonstrates the greather expressive power of linear logic, which is able to diferentiate between denotation and sense already at the syntactic level.

Vertical fragments play their role also in linear type theory. The left fragment contains the tensor product and sum while the right fragment contains the direct product and sum.

Now we consider the diagonal ellipses, which reflect another kind of fragmentation based on the idea of polarity. All logical connectives and neutral elements can be split into the following groups with:

1 Positive polarities: **0**, **1**, $\otimes$, $\oplus$,!;

2 Negative polarities: $\perp$, $\top$, &, $\wp$,?;

3 Dependent polarity: —o;

4 Turn over polarity: $(.)^{\perp}$;

These fragments can be considered from algebraic/logical point of view: the connectives with positive polarities correspond with the *algebraic style* and the connectives with negative polarities correspond with the *logical style*.

Linear negation causes the polarity to be turned over. This means that if an action is positive, its negation becomes negative, and vice versa. Linear implication is neutral again with respect to polarity; it causes the polarity of implication premise to be changed. An action (formula) of linear logic is positive if its outermost logical connective is positive; it is negative if its outermost logical connective is negative.

Finally, we consider the horizontal fragments of linear logic. If we work with the translation of propositional logic into linear logic, the upper fragment contains two linear conjunctions corresponding with classical conjunction and the lower fragment contains two linear disjunctions corresponding with classical disjunction. From the point of view of linear type theory we can regarded the upper fragment as the product type's constructors and the lower fragment as the sum type's constructors.

4 From Classical Logic to Linear Logic

As we mentioned above, linear logic can be considered as a generalization of classical logic. Every formula of classical logic can be unambiguously translated into linear formula. The static character of classical implication in linear logic ensures the exponential "!":

$$\varphi_1 \Rightarrow \varphi_2 \rightarrow !\varphi_1 - \circ \varphi_2 \tag{7}$$

Table 4 consists of the corresponding connectives for translating propositional logic into linear logic.

Table 4

Aristotelian logic to linear logic translation overview

CL to LL	$\wedge$	$\vee$	$\Rightarrow$	$\neg$	True	False
	&	$\oplus$	—o	$(.)^{\perp}$	T	**0**

Aristotelian logic based on the Tarski semantical tradition can be translated into the fragment of linear logic in the sense of Table 4. In this fragment the linear additive conjunction (&) is a generalized classical conjunction ($\wedge$), the linear additive disjunction ($\oplus$) is a generalization of classical the disjunction ($\vee$), the linear implication (—o) is a generalized classical implication ($\Rightarrow$) and classical negation ($\neg$) is expressed by linear negation ($(.)^{\perp}$). Aristotelian truth values, *True/False*, correspond to neutral elements, T**/ 0** of additive conjunction and disjunction, respectively.

Table 5

Intuitionistic logic to linear logic translation overview

CL to LL	$\wedge$	$\vee$	$\Rightarrow$	$\neg$
	&	$\oplus$	—o	!_ —o 0

When we come out from the Heyting semantical tradition, such generalization leads to intuitionistic linear logic (Table 5). For example, intuitionistic formulae can be translated into linear formulae using the following equivalences:

$$\begin{aligned} \varphi_1 \wedge \varphi_2 &\equiv \varphi_1 \,\&\, \varphi_2 \\ \varphi_1 \vee \varphi_2 &\equiv \varphi_1 \oplus \varphi_2 \\ \varphi_1 \Rightarrow \varphi_2 &\equiv !\varphi_1 - \circ \varphi_2 \end{aligned} \tag{8}$$

5 From Linear Logic to Linear Type Theory

Due to the Curry-Howard correspondence between intuitionistic linear logic and type theory [1], any formula φ of linear logic can be interpreted as a linear type denoted e.g. by *A*. Using linear connectives we can formulate a linear type theory in the sense of Table 6. According to the selected fragment of linear logic we can work with tensor fragment and/or direct fragment.

Table 6

Type theory nomenclature in Linear logic manner

Linear type theory			***Linear logic***	
Tfrag	Tensor product	$\otimes$	Multiplicative conjunction	Mfrag
	Tensor sum	$\wp$	Multiplicative disjunction	

Linear type theory			*Linear logic*	
Dfrag	Direct product	&	Additive conjunction	Afrag
	Direct sum	⊕	Additive disjunction	

Every programming language has a collection of predefined types. These types can be considered as basic types forming a set *Btypes={X, Y,...}*. Let *I* be a unit type. We can construct linear Church's types over basic types and unit type using type operators corresponding with linear logic connectives. Then the syntax of the linear types can be defined as:

$$A ::= I \mid X \mid A_1 \otimes A_2 \mid A_1 \oplus A_2 \mid A_1 - \circ A_2 \mid A_1 \,\&\, A_2 \mid A_1 \wp A_2 \tag{9}$$

In this grammar, *I* denotes a linear unit type and *X* denotes a linear basic type. The following constructions are linear Church's types:

- $A_1 \otimes A_2$ is product linear type;
- $A_1 \oplus A_2$ is coproduct (sum) linear type, and
- $A_1 \multimap A_2$ is function linear type as a set of functions from type A_1 to a type A_2.

Binary product/coproduct linear types can be generalized to

- Finite product linear types of the form $A_1 \otimes A_2 \otimes \ldots \otimes A_n$ together with the projections π_i: $A_1 \,\&\, A_2 \,\&\, \ldots \,\&\, A_n \to A_i$, $i=1,\ldots,n$;
- Coproduct linear types of the form $A_1 \oplus A_2 \oplus \ldots \oplus A_n$ together with the coprojections (injections) κ_i: $A_i \to A_1 \wp A_2 \wp \ldots \wp A_n$, $i=1,\ldots,n$.

Correspondence between traditional type theory and linear type theory is shown in Table 7. In linear type theory, any variable can appear in a term only once [1]. Product types (⊗) together with projections (&) are illustrated in the upper horizontal ellipse in Figure 1. Coproduct types (⊗) together with coprojections (℘) are illustrated in the lower horizontal ellipse in Figure 1.

Table 7

Traditional and linear Type theory

Type manipulation	**Type theory**		
	Traditional	***Linear***	***Comment***
Product	×	⊗	constructor
		&	selector
Coproduct	+	⊕	deconstructor
		℘	integrator
Function type	→	—o	constructor

6 Behavior, Knowledge and Belief

The expressive power of linear logic can be increased by various extensions, for instance with modal operators. If we consider the vertical fragments in Figure 1, we can construct various modal extensions of linear logic that enable additional useful applications in computing science.

Firstly, consider the intensional fragment of linear logic (left vertical ellipse). If we extend this fragment with modal operators for necessity and possibility (◊, □), we achieve a new logical system constructed over coalgebra [10], [11] as a resource oriented modal coalgebraic linear logic suitable for describing the observable behavior of running programs [12].

Now we consider the extensional fragment of linear logic (right vertical ellipse). We extend this fragment by epistemic objective knowledge operator *K* and by epistemic rational belief operator *B*. Assuming an agent *c* a formula $K_c\varphi$ expresses that an agent *c* has a knowledge φ and a formula $B_c\varphi$ expresses that an agent *c* has belief about φ. This fusion between epistemic and linear logic we have used to construct a Kripke model for acquiring knowledge and empirical belief about incoming network intrusions [12], [13] based on [25]. We shortly describe the main ideas of our approach. We use the following extensional fragment of linear logic extended with epistemic operators:

$$\varphi ::= p_n \mid \varphi_1 \,\&\, \varphi_2 \mid \varphi_1 \oplus \varphi_2 \mid \varphi_1 -\!\circ \varphi_2 \mid \varphi^{\perp} \mid !\varphi \mid K_c\varphi \mid B_c\varphi \tag{10}$$

Assume a signature based Intrusion Detection System and three possible types of intrusions: *A, B* and *C.* Every type of intrusion attempt has several symptoms that can be described as elementary propositions.

Let *007* be an rational agent, e.g. some program. Let the symptoms of an intrusion attempt of a type *A* be denoted by elementary propositions a_1, a_2, a_3 and a_4. The symptoms of an intrusion of a type *B* are b_1, b_2 and b_3 and the ones of a type *C* are c_1, c_2 and c_3. An intrusion attempt of a particular type occurs only if all its symptoms have occurred. Using additive conjunction we can describe the knowledge about all mentioned types of intrusion attempts by the following formulae:

$$\begin{aligned} K_{007}\varphi &\equiv K_{007}a_1 \,\&\, K_{007}a_2 \,\&\, K_{007}a_3 \,\&\, K_{007}a_4 \\ K_{007}\psi &\equiv K_{007}b_1 \,\&\, K_{007}b_2 \,\&\, K_{007}b_3 \\ K_{007}\theta &\equiv K_{007}c_1 \,\&\, K_{007}c_2 \,\&\, K_{007}c_3 \end{aligned} \tag{11}$$

Let $K_{007}\tau$ be a formula describing the knowledge about a sender, e.g. its IP address. A formula

$$\left(\underbrace{K_{007}\varphi \,\&\, \ldots \,\&\, K_{007}\varphi}_{300} \right) \& K_{007}\tau \tag{12}$$

describes that we have the knowledge that an intrusion attempt of type *A* occurred three hundred times from the same sender. The following epistemic linear formula

$$K_{007}\chi \equiv \left(\underbrace{K_{007}\varphi \,\&\, \ldots \,\&\, K_{007}\varphi}_{300} \right) \& K_{007}\tau - \circ \left(\left(K_{007}\psi \,\&\, K_{007}\theta \right) \& K_{007}\tau \right) \tag{13}$$

expresses the situation when after three hundred attempts of type *A* the attempts of types *B* and *C* follow immediately. The same situation exists in real IDS, e.g. vertical portscan [23]. If this situation repeats, we can state that our agent *007* has achieved a rational belief about the intrusion attempt expressed by the formula

$$!\left(K_{007}\chi \right) - \circ B_{007}\chi \tag{14}$$

and we can realise some protective actions. Exponential *!* enable us to describe the repetition of attempts, i.e. a real behavior of intrusions by the principle "*Repetitio est mater studiorum*".

In [16] we explained our approach in detail together with a construction of a Kripke model and a definition of the semantics of our epistemic linear logic.

Conclusions

In our paper we presented a few inventions regarding possible areas of applying linear logic in various disciplines of computing science. We considered several criteria for the fragmentation of linear logic and we discussed the known applications of these fragments in type theory and behavioral theory. We also discussed the special properties of linear connectives and exponentials. The static and dynamic properties of linear logic we illustrated in various forms of linear implication. Classical and intuitionistic logic can be translated into linear logic using exponentials. The dynamic character of linear logic enables it to definine linear type theory. Linear logic can be extended by new operators, e.g. modal operators, epistemic operators, etc. These extensions allow for increasing of the expressive power of linear logic and for opening new application domains. The modal intensional fragment of linear logic can be useful for describing the observable behavior of programs, and the epistemic extensional fragment enables us to obtain knowledge and beliefs about intrusion attempts.

The dynamic/static resource oriented character of linear logic destines it for wide usage in computing science. In this paper, we presented only a few possible applications of it based mainly on our research results. We believe that the presented inventions can lead to the discovery of further applications in computing science.

Acknowledgement

This work was supported by the Slovak Research and Development Agency under the contract No. APVV-0008-10 "Modelling, simulation and implementation of GPGPU-enabled architectures of high-throughput network security tools."

This work is the result of the project implementation: Center of Information and Communication Technologies for Knowledge Systems (ITMS project code: 26220120030) supported by the Research & Development Operational Program funded by the ERDF.

References

[1] Ambler S. J.: First Order Linear Logic in Symmetric Monoidal Closed Categories, PhD. Thesis, University of Edinburgh, 1991

[2] Avron A.: The Semantics and Proof Theory of Linear Logic, Theoretical Computer Science, Vol. 57, 1988, pp. 161-184

[3] Braüner T.: Introduction to Linear Logic, BRICS LS-96-6, Aarhus, 1996

[4] Chang, E. B.-Y., Chaudhuri, K., Pfenning, F.: A Judgmental Analysis of Linear Logic, Carnegie Mellon University, Report CMU-CS-03-131R, 2003

[5] Girard, J.-Y. From foundations to ludics. Bulletin of Symbolic Logic 9, 2 (2003), 131-168

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

[7] Girard J.-Y.: P. Taylor, Y. Lafont, Proofs and Types, Cambridge University Press, New York, NY, USA, 1989

[8] Korečko Š, Sobota B.: Using Coloured Petri Nets for Design of Parallel Raytracing Environment, Acta Universitatis Sapientiae. Vol. 2, No. 1, 2010, pp. 28-39

[9] Korečko Š, Sobota B., Szabó Cs.: Performance Analysis of Processes by Automated Simulation of Coloured Petri Nets, Intelligent Systems Design and Applications: Proceedings of the 10th international conference: 29 Nov.-1 Dec. 2010, Cairo, Egypt

[10] Kurz A.: Coalgebras and Modal Logic, CWI, Amsterdam, Netherlands, 2001

[11] Mihályi D.: Duality Between Formal Description of Program Construction and Program Behaviour, Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 1, No. 2, 2010, pp. 1-5

[12] Mihályi D., Jenčík M.: Few Inventions about Utilising Linear Logic in Computer Science, ICTIC 2012 - Information and Communication Technologies – International Conference, Žilina, 19. – 23. 3. 2012, 2012

[13] Mihályi D., Novitzká V., Ľaľová M.: Intrusion Detection System Epistème, Central European Journal of Computer Science, Vol. 2, No. 3, 2012, pp. 214-221

[14] Mihályi D., Novitzká V., Ľaľová M.: Intrusion Detection System Epistème, Proceedings of the International Scientific Conference Informatics'2011, Rožňava, 16.-18.11.2011, Košice, Equilibria, 2011, 11., pp. 61-65

[15] Mihályi D., Novitzká V., Slodičák V.: From Petri Nets to Linear Logic, CSE'2008, Fifth International Scientific Conference on Electronic Computers and Informatics, Vysoké Tatry - Stará Lesná, 24. - 26. 9. 2008, Košice, 2008, pp. 48-56

[16] Mihályi D., Novitzká V.: Towards to the Knowledge in Coalgebraic Model of IDS, Computing and Informatics, 2012 (accepted)

[17] Novitzká V., Mihályi D., Slodičák V.: Categorical Models of Logical Systems in the Mathematical Theory of Programming, Journal of Pure Mathematics and Applications, 17, 3-4, 2006, pp. 367-378

[18] Novitzká V., Mihályi D., Slodičák V.: How to Combine Church's and Linear Types, ECI'2006 - Seventh International Scientific Conference on Electronic Computers and Informatics, Košice - Herľany, 20.-22.9.2006, Košice, 2006, pp. 128-133

[19] Novitzká V., Mihályi D., Slodičák V.: Linear Logical Reasoning on Programming, Acta Electrotechnica et Informatica, Vol. 6, No. 3, 2006, pp. 34-39

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

[21] Sørensen M. H., Urzyczyn P.: Lectures on the Curry-Howard isomorphism, DIKU Rapport 98/14, 1998

[22] Slodičák, V.: Some Useful Structures for Categorical Approach for Program Behavior, Journal of Information and Organizational Sciences, Vol. 35, No. 1, 2011, pp. 99-109

[23] Snort web site. Availaible on: http://www.snort.org

[24] Szabó Cs., Slodičák V.: Software Engineering Tasks Instrumentation by Category Theory, Proceedings of the 9th IEEE International Symposium on Applied Machine Intelligence and Informatics SAMI 2011, 27-29.1.2011, Košice, Elfa s.r.o., 2011, pp. 195-199

[25] Vokorokos L. Baláž A.: Distributed Detection System of Security Intrusions Based on Partially Ordered Events and Patterns, Towards Intelligent Engineering and Information Technology, Studies in Computational Intelligence, Vol. 243, Springer, 2009, pp. 389-403

[26] Zlatoš P.: Ani matematika si nemôže byť istá sama sebou, Iris, Bratislava, 1995