JAM CM Journal of Applied Mathematics and Computational Mechanics 2015, 14(1), 91-100
www.amcm.pcz.pl p-ISSN 2299-9965
DOI: 10.17512/jamcm.2015.1.09 e-ISSN 2353-0588

LINEAR LOGIC IN COMPUTER SCIENCE

William Steingartner, Andrea Poláková, Peter Pražňák, Valerie Novitzká

Department of Computers and Informatics, Technical University of Košice
Košice, Slovakia
william.steingartner@tuke.sk, andrea.polakova@student.tuke.sk, praznak@minv.sk
valerie.novitzka@tuke.sk

Abstract. Linear logic has many properties that make it suitable for application in various areas of computer science. It is able to describe dynamic processes, non-determinism, parallelism on syntactic level. In our paper we try to discuss resource oriented character of linear logic, its possibility to deal with such important resources for computer science as space (memory) and time. Handling with resources takes place in deduction system of linear logic. We show how special form of proofs, called designs, is constructed and we show the relationship between space and time in designs.

Keywords: *linear logic, resources, proofs*

Introduction

Linear logic (LL) was introduced in 1987 by Jean-Yves Girard and published in special issue of a prestigious journal Theoretical Computer Science [1]. The author suggested this logic as an extension of classical propositional and first order logic, but its properties ensure a wider spectrum of applications in various disciplines of computer science that are not expressible in classical logics. LL enables to describe by formulae dynamics of processes, non-determinism, parallelism, resources as time and space, exhaustibility and inexhaustibility of these resources and many other useful aspects used in computer science [2]. Intuitionistic LL is the subject of Curry-Howard correspondence [3] enabling to consider programs-as-proofs. Formulae of LL correspond with Petri nets as it was published in [4, 5]. Every formula of classical logic can be transformed to LL formula using special operators called exponentials [6].

Semantics of LL is defined in several ways. The author of LL defined semantics using phase spaces and coherent spaces. In [7] is defined categorical semantics of LL using symmetric monoidal closed categories. Another categorical model with *-autonomous categories is in [8, 9]. Blass formulated in [10] also game semantics for LL.

LL consists of several fragments that can serve for different purposes as it is discussed in [11].

LL can be extended by modal operators of various non-classical logics to enable describing of important properties of real program systems. We extended coalgebraic modal logic [12] to linear coalgebraic modal logic in [13] that enables to express behavior of program systems modeled as coalgebras. For describing and modeling Intrusion Detection System we extended LL with epistemic operators [14].

In our paper we concentrate on resource oriented character of LL. We come out from published results [15-17] and our previous research [18, 19]. The aim is to express how the dealing with resources as space (memory) and time can be treated by LL in its deduction system and to show the relationship between these most important resources for computer science.

In the next section we introduce some basic notions of LL needed for our approach and then we concern with resources in LL.

1. Basic notions

The formulae of LL can be considered as actions, processes or resources. Every formula is interpreted as a type. Logical connectives of LL enable to express dynamic, parallelism, non-determinism and inexhaustibility of resources on syntactic level.

Let φ and ψ be linear formulae. Linear implication $\varphi \multimap \psi$ expresses dynamics of processes, i.e. after an action φ the action ψ follows. If we consider formulae as resources, a resource φ is consumed after linear implication, what can be described by linear negation $\varphi^{\perp}$.

LL has two conjunctions and two disjunctions in accordance with multiplicative and additive fragments of LL. Multiplicative conjunction $\varphi \otimes \psi$ describes that the actions φ and ψ can be performed simultaneously. The neutral element of multiplicative conjunction is 1. Additive conjunction $\varphi \,\&\, \psi$ expresses external non-determinism, i.e. only one of the actions can be performed, but we can know which from the context or environment. Its neutral element is $\top$. Multiplicative disjunction $\varphi \; par \; \psi$ is similar to operation *xor*, i.e. if φ is not performed then ψ is performed and vice versa. Its neutral element is $\perp$. Additive disjunction $\varphi \oplus \psi$ expresses internal non-determinism, i.e. only one action is performed but we cannot predict which one. The neutral element of additive disjunction is 0.

LL has two modal operators called exponentials. The formula $!\varphi$ expresses non exhaustibility of φ and the formula $?\varphi$ expresses potential non exhaustibility of φ.

Resources in LL are handled in its deduction system LL uses Gentzen's sequent calculus. A sequent has a form:

$$\Gamma \vdash \varphi$$

where Γ is a final list of linear formulae $\psi_1, \ldots, \psi_n$ serving as assumptions and φ is a linear formula derivable from them. Γ is often called context. Every deduction rule consists of a set of sequents called hypothesis written above a horizontal line and a single conclusion, also a sequent, written below the line of a deduction rule. The general scheme of deduction rule is as follows:

$$\frac{hypothesis_1 \qquad hypothesis_2}{conclusion}$$

We present only those deduction rules that are necessary for illustrating examples in this paper. The complete deduction system of LL can be found in many publications, e.g. in [1, 2, 6, 7, 16, 17].

$$\frac{}{\varphi \vdash \varphi}(id) \qquad \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi \& \psi, \Delta}(\& - r)$$

$$\frac{\Gamma_1 \vdash \varphi, \Delta_1 \qquad \Gamma_2 \vdash \varphi, \Delta_2}{\Gamma_1, \Gamma_2 \vdash \varphi \otimes \psi, \Delta_1, \Delta_2}(\otimes - r) \qquad \frac{\Gamma \vdash \psi, \Delta}{\Gamma \vdash \varphi \oplus \psi, \Delta}(\oplus - r_2)$$

$$\frac{\Gamma \vdash \varphi, \psi, \Delta}{\Gamma \vdash \varphi \; par \; \psi, \Delta}(par - r)$$

Deduction rule (*id*) of identity is an axiom. Deduction rule (&-*r*) introduces additive conjunction on the right side of sequent. Rule ($\otimes$-*r*) introduces multiplicative conjunction on the right side and the rule ($\oplus$-r_2) is one of the rules introducing additive disjunction on the right side and (par-r) introduces multiplicative disjunction on the right side. Formulae can be moved from the left side of sequent to the right side and vice versa using linear negation $()^{\perp}$. Using this possibility is marked in proof tree by ($\perp$-*r*) and ($\perp$-*l*). A proof of LL formula is a tree, where proved formula is in the root of tree. A formula is proved if all leaves of tree are axioms of identity.

Deduction system of LL does not contain structural deduction rules of weakening and contraction. This property relates with resource character of LL, i.e. every formula (resource) is meaningful, no resource can be added (by weakening) or contracted (by contraction).

In the next examples we show how resources can be treated in LL.

Example 1. We have 5 euros in our wallet. We denote this amount as a resource φ. We buy a lunch for 5 euros that we denote by a formula ψ. The process of spending a resource φ and obtaining a new resource ψ can be expressed by linear implication

$$\varphi \multimap \psi.$$

After performing this linear implication, we have no money in our wallet but we have a lunch.

Example 2. This example is taken from [2] because it illustrates how real processes, not only in computer science, can be expressed by LL formulae.

We would like to express by LL formula the process of creating water from hydrogen and oxygen. We denote these resources as φ for a molecule H_2 of hydrogen and ψ for a molecule O_2 of oxygen. Chemical equation of creating two water molecules H_2O denoted by ϑ from hydrogen and oxygen requires two molecules of hydrogen and one molecule of oxygen as resources

$$2H_2 + O_2 \rightarrow 2H_2O.$$

We can express this process by linear implication as

$$\varphi \otimes \varphi \otimes \psi \multimap \vartheta \otimes \vartheta .$$

2. Resources in LL - time

LL handles with resources within its deduction system. Logical connectives are divided into two groups depending on their polarity. The connectives $\oplus, \otimes, 0, 1, !$ are positive connectives whereas &, par, $\perp$, $\top$, ? are negative ones. The positive connectives can be considered as algebraic style and negative connectives as logical style. The polarity of LL formulae depends on their outermost connective. Negative formulae are reversible, i.e. their decomposing from bottom of proof to leaves of proof is deterministic. On the other side, positive formulae can be characterized as true non-determinism.

Change of polarity within the proof of a linear formula characterizes incrementation of time. The steps of a proof with the same polarity can be enclosed into a cluster. The actions in a cluster can be performed simultaneously.

Example 3. Let α be a formula

$$\alpha = (((\varphi^{\perp} \,\&\, \psi^{\perp}) \,\&\, \vartheta^{\perp}) \otimes \omega^{\perp}).$$

Let φ, ψ and ϑ be positive formulae and let Δ be a set of linear logic formulae. The proof tree of α is the following:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\varphi \vdash \Gamma}(id)}{\vdash \varphi^{\perp}, \Gamma}(\perp_{-r}) \quad \dfrac{\dfrac{}{\psi \vdash \Gamma}(id)}{\vdash \psi^{\perp}, \Gamma}(\perp_{-r})}{\vdash \varphi^{\perp} \,\&\, \psi^{\perp}, \Gamma}(\&_{-r}) \quad \dfrac{\dfrac{}{\theta \vdash \Gamma}(id)}{\vdash \theta^{\perp}, \Gamma}(\perp_{-r})}{\vdash (\varphi^{\perp} \,\&\, \psi^{\perp}) \,\&\, \theta^{\perp}, \Gamma}(\&_{-r}) \quad \dfrac{\dfrac{}{\omega \vdash \Delta}(id)}{\vdash \omega^{\perp}, \Delta}(\perp_{-r})}{\vdash (((\varphi^{\perp} \,\&\, \psi^{\perp}) \,\&\, \theta^{\perp}) \otimes \omega^{\perp}), \Gamma, \Delta}(\otimes_{-r})$$

In this proof we observe the change of polarity, first from positive to negative and then from negative to positive. We enclose this proof to the clusters and we get:

$$\dfrac{\dfrac{\varphi \vdash \Gamma \qquad \psi \vdash \Gamma}{\vdash (\varphi^{\perp} \,\&\, \psi^{\perp}), \Gamma} \qquad \theta \vdash \Gamma \qquad \omega \vdash \Delta}{\vdash \alpha, \Gamma, \Delta}$$

We can rewrite the formula α using De-Morgan's laws:

$$(((\varphi^{\perp} \,\&\, \psi^{\perp}) \,\&\, \vartheta^{\perp}) \otimes \omega^{\perp}) \equiv (((\varphi \oplus \psi) \oplus \upsilon)\ par\ \omega)^{\perp}.$$

There exist three equivalent proofs of this formula depending of the used deduction rules for connectives of additive disjunction. Here we present only one of them:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \theta, \omega, \Gamma}\,(id)}{\vdash (\varphi \oplus \psi) \oplus \theta, \omega, \Gamma}\,(\oplus_{-r_2})}{\vdash ((\varphi \oplus \psi) \oplus \theta) \parr \omega, \Gamma}\,(\parr_{-r})}{\alpha \vdash \Gamma}\,(\perp_{-l})$$

Enclosing the proof steps with the same polarity into clusters we get the following simplified proof:

$$\dfrac{\vdash \theta, \omega, \Gamma}{\dfrac{\vdash ((\varphi \oplus \psi) \oplus \theta) \parr \omega, \Gamma}{\alpha \vdash \Gamma}} \quad \equiv \quad \dfrac{\dfrac{\vdash \theta, \omega, \Gamma}{\vdash \alpha^{\perp}, \Gamma}}{\alpha \vdash \Gamma}$$

Proofs with clusters are not only simpler but also indicate time incrementation. The actions enclosed in clusters can be performed simultaneously. Expressing time incrementation in proof can help the programmers in paralellization of computing.

3. Resources in LL - space

In the linear logic we consider about space in the terms of locations. Every LL formula has a location, an address. Using this idea we remove the content of proof and we replace formulae by their locations. Proofs containing only locations are named designs.

A location of proved formula in the design is denoted by ξ we call its location address. If the formula has direct subformulae, their locations are called biases and they are appended to an address as natural numbers. The structure of space occupied by a formula is uniquely identified by finite sequence of biases. Finite set of

biases is called ramification and it is needed for proof steps of multiplicative connectives. In these rules exist two conditions. The sequent can be one-sided if all formulas in Γ are positive. Then sequent has the form:

$$\vdash \Gamma$$

But if the sequent consists of two positive and one negative formulae it can be written in the form:

$$\varphi^{\perp} \vdash \psi, \theta$$

where φ is a negative formula. Construction of design has two steps.

The first step is construction of the LL proof in such manner that in the root has to be a positive linear logic formula because of non-deterministic behavior of negative formulae. To ensure it, we need some new connectives (shifts) changing the polarity of formulae. The connective ↑ changes polarity from positive to negative and the connective ↓ changes negative polarity to positive one. Applying shift $\uparrow\varphi$ to negative formula φ and $\downarrow\psi$ to positive formula ψ has no effect.

The second step of constructing design is removing content of formulae using three rules: positive, negative and daimon rules [16, 17].

1. *Positive rule* - if the outermost connective of formula is positive

$$\frac{\ldots \quad \xi * i \vdash \Lambda i \quad \ldots}{\xi \vdash \Lambda} \ (+, \xi, I)$$

2. *Negative rule* - if the outermost connective of formula is negative

$$\frac{\ldots \quad \vdash \xi * I \quad \ldots}{\xi \vdash \Lambda} \ (-, \xi, N)$$

3. *Daimon* - if we cannot use positive or negative rule, we use the daimon rule (for example if an error occurs).

$$\frac{}{\vdash \Lambda} \ \dagger$$

Designs provide information about needed space for data structures. Every LL formula can be represented as a type. If a formula has subformulae, representation is a type with internal structure, e.g. record, list or variant record. An address of a formula together with all biases provide complete information about space needed for data structures of this type. We show the construction of designs on several examples

Example 4. Let α be a formula of linear logic

$$\alpha = (\varphi \,\&\, \psi) \oplus \theta^{\perp}.$$

Its outermost connective is positive. As a first step we construct linear logic formula:

$$\dfrac{\dfrac{\vdash \varphi \qquad \vdash \psi}{\vdash \varphi \,\&\, \psi}\ \{(\varphi \,\&\, \psi,\ \varphi), (\varphi \,\&\, \psi,\ \psi)\}}{\vdash (\varphi \,\&\, \psi) \oplus \theta^{\perp}}\ \{(\varphi \,\&\, \psi) \oplus \theta^{\perp},\ \varphi \,\&\, \psi\}$$

Now we can construct design. The first used rule is a positive rule and the second is negative one:

$$\dfrac{\dfrac{\vdash \xi * 11 \qquad \vdash \xi * 12}{\xi * 1 \vdash}\ \{(-, \xi 1, \{1\}), (-, \xi 1, \{2\})\}}{\vdash \xi}\ (+, \xi, \{1\})$$

After constructing a design we can determine the address of formula α as ξ together with the addresses of its subformulas φ, ψ and θ, e.g. φ is located on address ξ*11.

Example 5. Let β be a formula of linear logic

$$\beta = (\varphi \,\&\, \psi).$$

The formula β contains only negative connective &.
First, we must modify β, because it is a negative formula and for construction of design we need a positive formula.

negative formula β $\quad\rightarrow^{\text{modify}}\quad$ positive formula $\beta' \downarrow (\uparrow\varphi \,\&\, \uparrow\psi)$

In the root of proof tree is now a positive LL formula β' so we can construct the following proof in linear logic.

$$\dfrac{\dfrac{\vdash \varphi \qquad \vdash \psi}{\vdash (\varphi \,\&\, \psi)}\ \{(\varphi \,\&\, \psi,\ \varphi), (\varphi \,\&\, \psi,\ \psi)\}}{\vdash \downarrow (\uparrow \varphi \,\&\, \uparrow \psi)}$$

Now we can construct design. At first we use a positive rule and then a negative rule.

$$\dfrac{\dfrac{\vdash \xi * 11 \qquad \vdash \xi * 12}{\xi * 1 \vdash}\ \{(-, \xi 1, \{1\}), (-, \xi 1, \{2\})\}}{\vdash \xi}\ (+, \xi, \{1\})$$

After constructing a design we can determine the address of formula α as ξ and the addresses of the subformulae of formula α, e.g. ψ is on the address ξ*12).

4. Space-time relationship

In the previous sections we showed how time and space can be treated in LL proof. First we construct a proof of positive formula with clusters and then we construct design to obtain information about space. Now we discuss the relationship between time and space in designs.

A design can have one or more branches and it expresses two relationships: time and/or space one. The addresses in the same branch of design are comparable addresses and they have time relationship. The addresses in different branches are incomparable, i.e. they have space relationship.

Comparable addresses of subformulae are subjects of sequential computing with corresponding data structures. Incomparable addresses can be used in independent parallel computing without danger of errors.

The Figure 1 shows the relationships in designs. The arrow on the left side shows incrementation of time. The clouds in design illustrate address space. The signs + and – represent used positive rule or negative one, respectively.

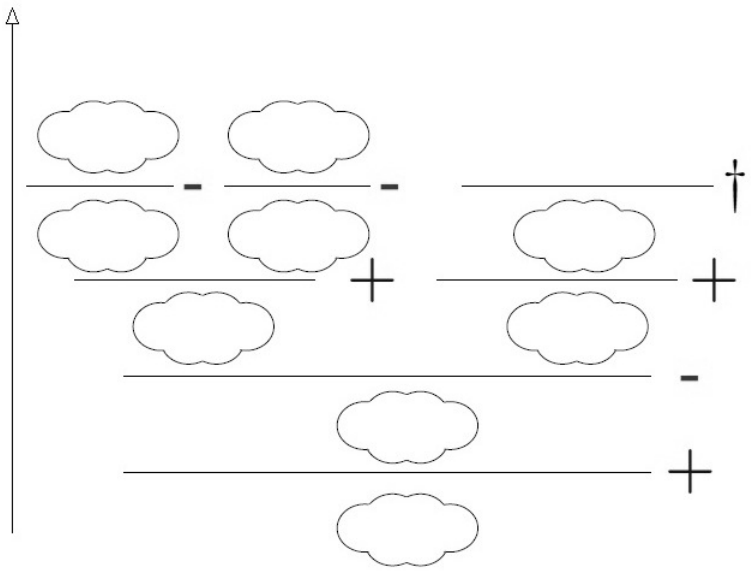

Fig. 1. Time and space in design

Conclusion

In this article we tried to characterize shortly interesting properties of LL with emphasis on its resource oriented character. Possibility of handling with resources within deduction system of LL enables in verified manner predict memory needed for performing computation. Using polarity of formulae and consequent enclosing into clusters give information of time incrementation and suitable paralellization of

computation. Handling with resources on abstract level may help to avoid many failures hardly predicted during preparing programs.

Representation of LL formulae by real types, basic or simple types, the fact that bises are abstract information gives the possibility in the future to quantify in categorical model of LL memory occupation.

Acknowledgment

This work has been supported by KEGA grant project No. 050TUKE-4/2012: "Application of Virtual Reality Technologies in Teaching Formal Methods".

References

[1] Girard J.-Y., Linear logic, Theoretical Computer Science 1987, 50, 1-102.

[2] Girard J.-Y., Linear logic: its syntax and semantics, [in:] Advances in Linear Logic, eds. J.-Y. Girard, Y. Lafont, L. Regnier, London Mathematical Society Lecture Note Series, No. 222, Cambridge University Press, 1995.

[3] Sorensen M., Urzyczyn P., Lectures on the Curry-Howard isomorphism, Techn. Report 98/14, DIKU, 1998.

[4] Mihályi D., Novitzká V., Slodičák V., From Petri nets to linear logic, Proc. of the CSE 2008 International Scientific Conference on Computer Science and Engineering, Stará Lesná, September 24-26, 2008, 48-56.

[5] Cardoso J., Pradin-Chézalviel B., Logic and fuzzy Petri nets, Workshop on Manufacturing and Petri Nets on XVIII International Conference on Applications and Theory of Petri Nets, 1997, 1-18.

[6] Girard J.-Y., Taylor P., Lafont Y., Proofs and Types, Cambridge University Press, 1989.

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

[8] Blute R., Scott P., Category theory for linear logicians, [in:] Linear Logic Summer School, eds. P. Ruet, T. Ehrhard, J.-Y. Girard, P. Scott, Cambridge University Press, 2003.

[9] Mellies P.-A., An innocent model of linear logic, [in:] Category Theory in Computer Science, ed. L. Birkedal, Electron. Notes in Theoretical Computer Science 2005, 122, 171-192.

[10] Blass A., A game semantics for linear logic, Annals of Pure and Applied Logic 1992, 56, 151--166.

[11] Mihályi D., Novitzká V., What about linear logic in computer science? Acta Polytechnica Hungarica 2013, 10, 4, 147-160.

[12] Kurz A., Coalgebras and modal logic, Theoretical Computer Science 2001, 260, 1-2, 119--138.

[13] Mihályi D., Duality between formal description of program construction and program behaviour, Information Sciences and Technologies Bulletin of the ACM Slovakia 2010, 2, 1, 1-5.

[14] Mihályi D., Novitzká V., Towards the knowledge in coalgebraic model of IDS, Computing and Informatics 2014, 33, 1, 61-78.

[15] Andreoli J.M., Logic programming with focusing proofs in linear logic, Journal of Logic and Computation 1992, 2, 297-347.

[16] Curien P.L., Introduction to linear logic and ludics, part 1, Advances in Mathematics (China) 2005, 34, 513-544.

[17] Curien P.L., Introduction to linear logic and ludics, part 2, Advances in Mathematics (China) 2006, 35, 1-44.

[18] Macko P., Novitzká V., Slodičák V., The rôle of designs in linear logic, Electrical Engineering and Informatics 2012, 3, 620-623.

[19] Macko P., Novitzká V., The resource character of linear logic, Electrical Engineering and Informatics 2011, 2, 1-4.