GAME SEMANTICS OF THE TRANSACTION ROLLBACK DATABASE OPERATION

Veronika SZABÓOVÁ, Csaba SZABÓ, Valerie NOVITZKÁ, Emília DEMETEROVÁ
*Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice, tel. 055/602 4120, E-mail: veronika.szaboova@tuke.sk, csaba.szabo@tuke.sk, valerie.novitzka@tuke.sk,emilia.demeterova@tuke.sk

ABSTRACT

This paper shows the power of game semantics when describing concurrent processes. First, we analyze the selected iterative model of database transaction execution and provide linear logic expression of this model. Next, game semantics for linear logic is used to express the semantics of the selected situation: transaction rollback. The developed strategy is being then proved using proof tree of the linear logic calculus. Finally, we discuss the presented power of game semantics of database transactions, and show future work directions in the field of expressing game semantics of software engineering problems.

Keywords: *database transactions; game semantics; linear logic; rollback.*

1. INTRODUCTION

Software engineering and the software industry often deal with the problem of ambiguous requirement definitions [11]. These requirements are being refined by analyzing questionnaires and/or during customer consultations. These activities develop a system of interacting processes that could be referred as the design plan of the software. When considering information processing systems, a subset of key processes is defined by database transactions. Our paper addresses this group of processes and their execution.

Information systems are distributed systems by their architecture. The most used mathematical modeling language for the description of distributed systems is a Petri net. In this paper, we build our linear logic description for database transactions also based on an existing Petri net model that was discussed by [3]. Linear logic is suitable to describe [4] Petri net syntax, and provides a tool to express sentences of the language defined by it using linear logic sequents.

To express semantics of action sequences represented by sentences of the mentioned language, game semantics [1, 2] is used. We use the Opponent-to-Proponent dialog games to achieve appropriate strategies, which describe semantics in our model, i.e. existence of a winning strategy for the proponent implies reachability and this strategy itself implies transaction life cycle. The life cycle stages are given by the sources and formulas used in the sentence describing the strategy.

We provide proponent payoff reachability proofs based on the sequents defined by the strategies. Proof trees are used when proving a sequent in the linear logic calculus. The sequent is proved, if the leaves of the proof tree contain only axioms.

The achieved results are prerequisites to prove selected internals of information systems such as self-properties and their execution strategies. The resulting model is also a pilot in the process of expressing self-properties of autonomous systems using linear logic and game semantics.

The organization of the paper is as follows. First, the theoretical basis is described in sections 2 and 3. Then, we introduce our linear logic model in section 4. Later in this section, the selected database transaction rollback is described and expressed in linear logic. For the resulting sequent game semantics is developed and proved. The last section discusses the results and shows future directions.

2. LINEAR LOGIC

Linear logic (LL) is a refinement of classical and intuitionistic logic. The emphasis is on resource depletion in the formula. New operators are introduced. It is one of the logical systems. It was introduced by Jean-Yves Girard in [5, 6].

About linear logic, one could say that it is an important part of interfaces between mathematics and informatics. Its main advantage is the applicability in real world environments. We could select typical examples of linear logic application in parallel process theory, in logical and/or functional programming, and in computing program execution [8, 10]. Other applications include description of Petri nets, Turing machines and proof nets.

Using:

- $0, 1, \bot, \top$ for constants,
- p for atomic expressions,
- $(,)$ as help symbols,
- $\multimap, \otimes, \oplus, \invamp, \&$ for logical operator symbols and
- $!, ?$ for modal operators,

the syntax of LL could be expressed by the following production rule as in [6]:

$$\begin{aligned}\varphi ::= \quad & 0|1|\bot|\top|p|\varphi_1 \multimap \varphi_2|\varphi_1 \otimes \varphi_2|\varphi_1 \oplus \varphi_2| \\ & \varphi_1 \invamp \varphi_2|\varphi_1 \& \varphi_2|\varphi^\bot|!\varphi|?\varphi\end{aligned} \tag{1}$$

2.1. Operators of linear logic

[5, 6] defines three categories of LL operators:

1. multiplicative ($\multimap$, $\otimes$, $\invamp$),
2. additive ($\oplus$,&), and
3. modal (!, ?) also called exponentials.

For each of these operators, the literature provides a description as follows:

Linear implication $\varphi_1 \multimap \varphi_2$ is used to express that the action φ_1 causes the (re)action φ_2. Execution of this implication consumes the source φ_1.

Multiplicative conjunction $\varphi_1 \otimes \varphi_2$ Both actions are executed in parallel. The constant 1 is the neutral value of this operation.

Multiplicative disjunction $\varphi_1 \parr \varphi_2$ Expresses alternative action execution: if the action φ_1 executes, then φ_2 does not, and vice versa. Neutral value for this operation is $\perp$.

Additive conjunction $\varphi_1 \,\&\, \varphi_2$ Only one of the actions will be executed, the decision is being made based on external circumstances (i.e. external non-determinism). Neutral value for this operation is $\top$.

Additive disjunction $\varphi_1 \oplus \varphi_2$ Only one of the actions will be executed, the decision is being made based on internal circumstances (i.e. internal non-determinism). Neutral value for this operation is 0.

Exponential $!\varphi$ "of course" the source φ can be used but will be not consumed.

Exponential $?\varphi$ "why not" is used to express potentiality of source φ.

Negation $(\varphi)^\perp$ expresses the duality between action φ and reaction $(\varphi)^\perp$. Negation is involute: $\varphi^{\perp\perp} \equiv \varphi$, meaning that De Morgan rules apply for LL operators too:

$$(\varphi_1 \otimes \varphi_2)^\perp \equiv \varphi_1^\perp \parr \varphi_2^\perp \tag{2}$$

$$(\varphi_1 \parr \varphi_2)^\perp \equiv \varphi_1^\perp \otimes \varphi_2^\perp \tag{3}$$

$$(\varphi_1 \,\&\, \varphi_2)^\perp \equiv \varphi_1^\perp \oplus \varphi_2^\perp \tag{4}$$

$$(\varphi_1 \oplus \varphi_2)^\perp \equiv \varphi_1^\perp \,\&\, \varphi_2^\perp \tag{5}$$

2.2. Deduction rules of linear logic

Girard describes the sequent calculus for linear logic in [6], where the main subject of interest is a sequent instead of LL formulas. A LL sequent consists of an antecedent (e.g. Δ), the turnstile symbol ($\vdash$), and a succedent (e.g. Σ) as shown in eq. (6).

$$\Delta \vdash \Sigma \tag{6}$$

Turnstile ($\vdash$) expresses causality in the sequent calculus and defines two sides of the sequent. Girard also expressed the right-hand-side sequent expression, where the turnstile symbol is separated on the left side as leading symbol for the sequent. This alternative expression allows combination of facts and usage of rules in proof trees from both antecedent and succedent part of the original sequent. The transformation is defined as follows:

$$\varphi_1, \ldots, \varphi_n \vdash \psi_1, \ldots, \psi_m$$

transforms to :

$$\vdash \varphi_1^\perp, \ldots, \varphi_n^\perp, \psi_1, \ldots, \psi_m \tag{7}$$

Using this notation, we present the deduction rules as defined by [6]:

- Identity (i.e. $\Delta \vdash \Delta$):

$$\frac{}{\vdash \Delta^\perp, \Delta}\,(\mathrm{id}) \tag{8}$$

- Cut, exchange:

$$\frac{\vdash \varphi, \Delta \quad \vdash \varphi^\perp, \Sigma}{\vdash \Delta, \Sigma}\,(\mathrm{cut}) \qquad \frac{\vdash \Delta, \varphi, \psi, \Sigma}{\vdash \Delta, \psi, \varphi, \Sigma}\,(\mathrm{ex}) \tag{9}$$

- Rules for logical operators:

$$\frac{\vdash \varphi, \Delta \quad \vdash \psi, \Sigma}{\vdash \varphi \otimes \psi, \Delta, \Sigma}\,(\otimes) \qquad \varphi \multimap \psi \equiv \varphi^\perp \parr \psi\,(\multimap) \tag{10}$$

$$\frac{\vdash \varphi, \psi, \Delta}{\vdash \varphi \parr \psi, \Delta}\,(\parr) \qquad \frac{\vdash \varphi, \Delta \quad \vdash \psi, \Delta}{\vdash \varphi \,\&\, \psi, \Delta}\,(\&) \tag{11}$$

$$\frac{\vdash \varphi, \Delta}{\vdash \varphi \oplus \psi, \Delta}\,(\oplus_1) \qquad \frac{\vdash \psi, \Delta}{\vdash \varphi \oplus \psi, \Delta}\,(\oplus_2) \tag{12}$$

- Rules for exponentials:

$$\frac{\vdash \Delta}{\vdash ?\varphi, \Delta}\,(\mathrm{w?}) \qquad \frac{\vdash ?\varphi, ?\varphi, \Delta}{\vdash ?\varphi, \Delta}\,(\mathrm{c?}) \tag{13}$$

$$\frac{\vdash \psi, ?\Delta}{\vdash !\varphi, ?\Delta}\,(!) \qquad \frac{\vdash \varphi, \Delta}{\vdash ?\varphi, \Delta}\,(\mathrm{der}) \tag{14}$$

Dereliction expresses the following:

$$\varphi \multimap ?\varphi \qquad !\varphi \multimap \varphi \tag{15}$$

Negation of exponentials:

$$(!\varphi)^\perp \equiv ?\varphi^\perp \qquad (?\varphi)^\perp \equiv !\varphi^\perp \tag{16}$$

- Expressions with constants:

$$\frac{}{\vdash 1} \qquad \frac{\vdash \Delta}{\vdash \perp, \Delta} \qquad \frac{}{\vdash \top, \Delta} \tag{17}$$

$$1^\perp \equiv \perp \qquad \perp^\perp \equiv 1 \qquad \top^\perp \equiv 0 \qquad 0^\perp \equiv \top \tag{18}$$

3. GAME SEMANTICS

Paul Lorenzen presented game semantics for logic. Later, his ideas were applied on linear logic by Blass in [2], and by Abramsky and Jagadeesan in [1]. In this type of semantics, the meaning of formulas is given by dialogs. Game semantics is based on game theory, which is a branch of applied mathematics. It analyzes and examines expected and real behavior of individuals in games.

The game or dialog is realized between two actors, which are denoted as P for Proponent or O for Opponent, respectively. The players (actors) express the input-output polarity of moves.

Luke Ong introduced the ideas included with Tab. 1 in [9] at the ICCL Summer School in 2004:

To express game semantics of a simple function call, we present a first-order function. We show a function example pred : $\mathbb{N} \to \mathbb{N}$.

1. O asks: "What is the output of this function?"

Table 1 Ong's ideas

Role	Point of view	Functional	Imperative	Concurrent
P	system	term	procedure	process
O	environment	program context	memory context	rest of the system

2. P asks: "What is the input to this function?"
3. O answers: "The input is 3."
4. P answers: "The output is 2."

Alternative way of expression:

	pred :	$\mathbb{N} \to$	$\mathbb{N}$
O			q
P		q	
O		3	
P			2

Advantage of the presented short example is that it is good understandable. More complicated real-world examples would need to have additional information introduced. The rows from top to down represent the order of questions and answers in the dialog, columns indicate the subject of these questions and/or answers. Additional information that could be necessary is the group of arrows representing relations between questions and the corresponding answers or between two consecutive questions, respectively.

4. THE LOGICAL SYSTEM OF DATABASE TRANSACTIONS

In our work, we implemented the logical system consisting of linear logic description of the iterative model of continuous transaction tracking and deadlock detection system presented by Chen in [3] originally using a stochastic Petri net (SPN) of 13 places and 17 transitions. Our solution uses linear logic for situation handling (state-action) description and game semantics for transaction life cycle description by strategies. To maintain correspondence to the original model, the same notation for the places and transitions is used. Only relevant ones are discussed in this paper (see Tab. 2).

In this paper, we present the solution of the example situation of transaction rollback. Rollback in our model is expressed in linear logic as follows:

$$\begin{aligned}
&!(p'_7 \multimap (p_2 \,\&\, p_5))\\
&\otimes\, !(p_2 \multimap p'_2)\\
&\otimes\, !(p'_2 \multimap (p_3 \,\&\, p_4))\\
&\otimes\, !(p_4 \multimap p'_4)\\
&\otimes\, !(p'_4 \multimap (p_6 \,\&\, p_{12}))\\
&\otimes\, !(p_6 \multimap p_1)\\
&\otimes\, p'_7\\
&\vdash p_1
\end{aligned} \tag{19}$$

The transitions will be not consumed, i.e. could be fired any time the prepositions are met, therefore each of them is expressed using the ! exponential. The actions could be

Table 2 Description of relevant places and transitions based on the original SPN model [3]

Notation	Description
p_1	Transaction is waiting for the first lock.
p_2, p'_2	Transaction is waiting for its next lock.
p_3	Transaction sets the lock.
p_4, p'_4	Transaction is waiting for the deadlock detection algorithm to finish.
p_5	Transaction unlocks its locks after successful commit.
p_6	Transaction unlocks its locks after being aborted (no success, e.g. rollback).
p'_7	Transaction is performing calculations after it gained access to data.
p_{12}	Transaction is waiting for other ones to unlock data items.
$!(p'_7 \multimap (p_2 \,\&\, p_5))$	Stochastic phenomenon of continuing or ending the transaction, respectively. [3] defines a probability of P_{exit} that depends on the state of the whole system.
$!(p_2 \multimap p'_2)$	Timed transition representing waiting for the next lock.
$!(p'_2 \multimap (p_3 \,\&\, p_4))$	Stochastic phenomenon of granting or rejecting the next lock for the transaction, respectively. [3] defines a probability of P_{g2} that depends on the state of the whole system.
$!(p_4 \multimap p'_4)$	Timed transition representing waiting for the deadlock detection algorithm to finish.
$!(p'_4 \multimap (p_6 \,\&\, p_{12}))$	Stochastic phenomenon of removing the transaction from the system (e.g. rollback) or setting its state to waiting for finishing of other transactions, respectively. [3] defines a probability of P_d that depends on the state of the whole system.
$!(p_6 \multimap p_1)$	Timed transition representing the time needed to unlock all locks set by the failed transaction and to rollback data items.

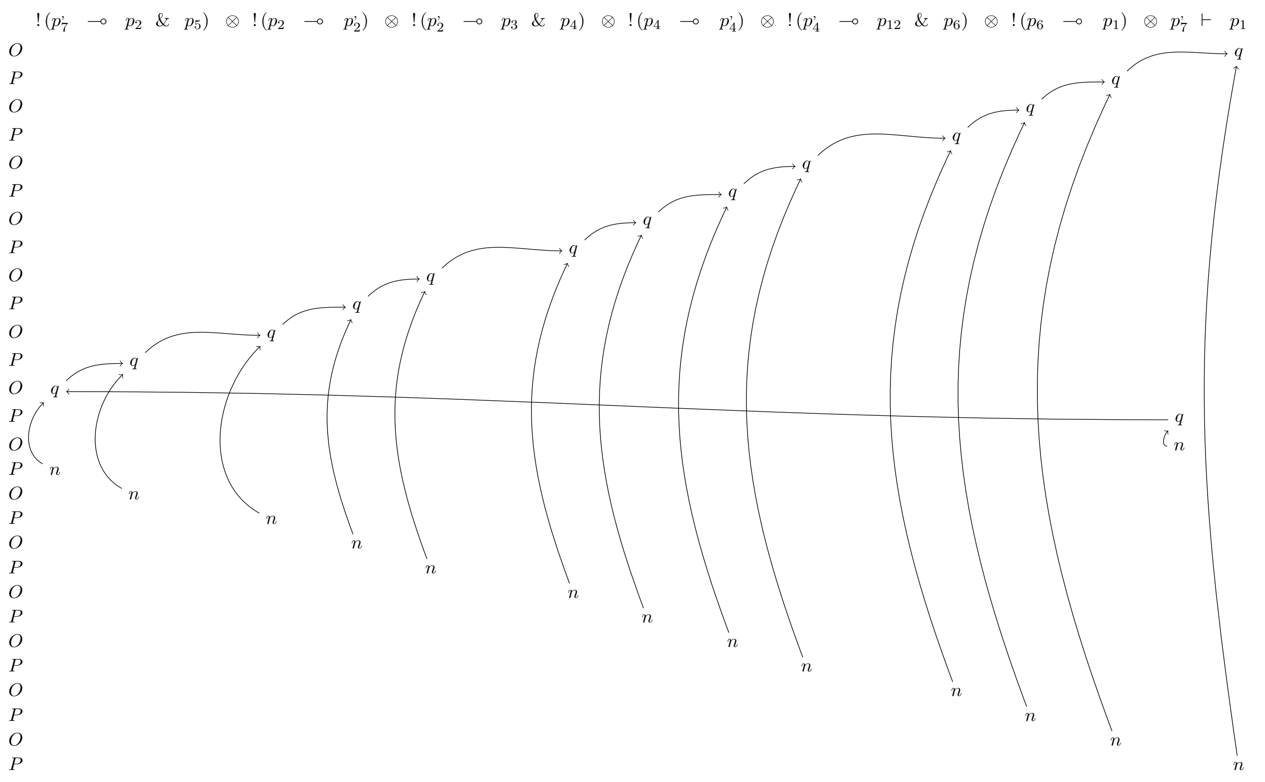
$!(p'_7 \multimap p_2 \,\&\, p_5) \otimes !(p_2 \multimap p'_2) \otimes !(p'_2 \multimap p_3 \,\&\, p_4) \otimes !(p_4 \multimap p'_4) \otimes !(p'_4 \multimap p_{12} \,\&\, p_6) \otimes !(p_6 \multimap p_1) \otimes p'_7 \vdash p_1$
O P O P O P O P O P O P O P O P O P O P O P O P O P O P
q
n

Fig. 1 Game semantics of the rollback operation

also performed in parallel that is indicated by the usage of multiplicative conjunction $\otimes$.

The facts p'_7 and p_1 represent initial and final states in the system (i.e. markings, in terms of Petri nets), respectively. The goal of this example is to follow transaction life cycle from the state it finished some calculations (p'_7) through asking for another data item lock (p'_2) to its rollback to the queue of waiting transactions (p_1).

Rollback strategy based on our model is shown in Fig. 1, where the game semantics of this operation is expressed. The goal of the opponent is to ask questions until the proponent can answer them. The first question is aimed on the existence of the result, i.e. if the transaction could be in the queue of waiting transactions. The dialog continues by mutually exchanging the positions of interviewer between O and P to get closer to a specific information. Possible branches in the dialog (i.e. alternative ways) are determined by values of probability defined by [3] and are chosen by the authors to model the selected rollback situation. The presented strategy is finite and has all questions answered. The order of answering the questions is given by the arrows between the letters q in Fig. 1, the letters n represent corresponding answers, the order of asking the answers is given by the order of rows starting with O respectively P (for opponent or proponent) from top to down.

The proof of the found strategy is presented in Fig. 2, where the evaluation is given by the proof tree in the sequent calculus for linear logic.

The root of the tree contains the sequent presented in (19). In the first step, the sequent is transformed that the turnstile appears on the left hand side of it. Then, deduction rules are applied, which application order is denoted by the found strategy. All leaves of the proof tree contain the identity axiom (8) meaning the sequent (19) is proved.

5. DISCUSSION AND FUTURE WORK

Both computer science and software engineering aim to describe large software systems to ease their development, usage and maintenance. This description was formerly based on structural modeling and derivation of results as a completely new software but, with the emerging demand on software modifications (software update), behavioral description of software comes into foreground. Such a description of existing systems allows their deep understanding, detailed evaluation, maintenance, adaptation and these processes lead to a better software system.

This work has been devoted to the solution of a practical example from the field of informatics, in particular database systems. In the example, linear logic was used to describe the system of transactions monitoring and game semantics was used to express the examples of transaction life cycle. This way a logical system for game semantics of database transactions was created.

The found strategy and the reachability of its result were proved using a proof tree. The resulting strategy and the proof is quite complicated even when a relatively simple assignment is given. The rollback strategy is straightforward, following its unique goal. Other operations are more complicated due to their concurrency.

The presented system provides much greater opportunity to express the game semantics of transaction processing tasks. One such task could be the game semantics of

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash p'^{\perp}_7, p'_7}\,(\text{id}) \quad \dfrac{\dfrac{}{\vdash p_2^{\perp}, p_2}\,(\text{id}) \quad \dfrac{\dfrac{}{\vdash p'^{\perp}_2, p'_2}\,(\text{id}) \quad \dfrac{\dfrac{}{\vdash p_4^{\perp}, p_4}\,(\text{id}) \quad \dfrac{\dfrac{}{\vdash p'^{\perp}_4, p'_4}\,(\text{id}) \quad \dfrac{\dfrac{}{\vdash p_6^{\perp}, p_6}\,(\text{id}) \quad \dfrac{}{\vdash p_1^{\perp}, p_1}\,(\text{id})}{\vdash p_6^{\perp}, (p_6 \otimes p_1^{\perp}), p_1}\,(\otimes)}{\vdash p'^{\perp}_4, (p'_4 \otimes p_6^{\perp}), (p_6 \otimes p_1^{\perp}), p_1}\,(\otimes)}{\vdash p_4^{\perp}, (p_4 \otimes p'^{\perp}_4), (p'_4 \otimes p_6^{\perp}), (p_6 \otimes p_1^{\perp}), p_1}\,(\otimes)}{\vdash p'^{\perp}_2, (p'_2 \otimes p_4^{\perp}), (p_4 \otimes p'^{\perp}_4), (p'_4 \otimes p_6^{\perp}), (p_6 \otimes p_1^{\perp}), p_1}\,(\otimes)}{\vdash p_2^{\perp}, (p_2 \otimes p'^{\perp}_2), (p'_2 \otimes p_4^{\perp}), (p_4 \otimes p'^{\perp}_4), (p'_4 \otimes p_6^{\perp}), (p_6 \otimes p_1^{\perp}), p_1}\,(\otimes)}{\vdash (p'_7 \otimes p_2^{\perp}), (p_2 \otimes p'^{\perp}_2), (p'_2 \otimes p_4^{\perp}), (p_4 \otimes p'^{\perp}_4), (p'_4 \otimes p_6^{\perp}), (p_6 \otimes p_1^{\perp}), p'^{\perp}_7, p_1}\,(\otimes,\text{ex})}{\vdash (p'_7 \otimes (p_2^{\perp} \oplus p_5^{\perp})), (p_2 \otimes p'^{\perp}_2), (p'_2 \otimes (p_3^{\perp} \oplus p_4^{\perp})), (p_4 \otimes p'^{\perp}_4), (p'_4 \otimes (p_6^{\perp} \oplus p_{12}^{\perp})), (p_6 \otimes p_1^{\perp}), p'^{\perp}_7, p_1}\,(\oplus)}{\vdash (p'_7 \otimes (p_2 \,\&\, p_5)^{\perp}), (p_2 \otimes p'^{\perp}_2), (p'_2 \otimes (p_3 \,\&\, p_4)^{\perp}), (p_4 \otimes p'^{\perp}_4), (p'_4 \otimes (p_6 \,\&\, p_{12})^{\perp}), (p_6 \otimes p_1^{\perp}), p'^{\perp}_7, p_1}\,(4)}{\vdash (p'^{\perp}_7 \parr (p_2 \,\&\, p_5))^{\perp}, (p_2^{\perp} \parr p'_2)^{\perp}, (p'^{\perp}_2 \parr (p_3 \,\&\, p_4))^{\perp}, (p_4^{\perp} \parr p'_4)^{\perp}, (p'^{\perp}_4 \parr (p_6 \,\&\, p_{12}))^{\perp}, (p_6^{\perp} \parr p_1)^{\perp}, p'^{\perp}_7, p_1}\,(3)}{\vdash (p'_7 \multimap (p_2 \,\&\, p_5))^{\perp}, (p_2 \multimap p'_2)^{\perp}, (p'_2 \multimap (p_3 \,\&\, p_4))^{\perp}, (p_4 \multimap p'_4)^{\perp}, (p'_4 \multimap (p_6 \,\&\, p_{12}))^{\perp}, (p_6 \multimap p_1)^{\perp}, p'^{\perp}_7, p_1}\,(\multimap)}{\vdash (p'_7 \multimap (p_2 \,\&\, p_5))^{\perp} \parr (p_2 \multimap p'_2)^{\perp} \parr (p'_2 \multimap (p_3 \,\&\, p_4))^{\perp} \parr (p_4 \multimap p'_4)^{\perp} \parr (p'_4 \multimap (p_6 \,\&\, p_{12}))^{\perp} \parr (p_6 \multimap p_1)^{\perp} \parr p'^{\perp}_7, p_1}\,(\parr)}{\vdash ?(p'_7 \multimap (p_2 \,\&\, p_5))^{\perp} \parr ?(p_2 \multimap p'_2)^{\perp} \parr ?(p'_2 \multimap (p_3 \,\&\, p_4))^{\perp} \parr ?(p_4 \multimap p'_4)^{\perp} \parr ?(p'_4 \multimap (p_6 \,\&\, p_{12}))^{\perp} \parr ?(p_6 \multimap p_1)^{\perp} \parr p'^{\perp}_7, p_1}\,(\text{der})}{!(p'_7 \multimap (p_2 \,\&\, p_5)) \otimes !(p_2 \multimap p'_2) \otimes !(p'_2 \multimap (p_3 \,\&\, p_4)) \otimes !(p_4 \multimap p'_4) \otimes !(p'_4 \multimap (p_6 \,\&\, p_{12})) \otimes !(p_6 \multimap p_1) \otimes p'_7 \vdash p_1}\,(7)
$$

Fig. 2 Proof tree for the rollback strategy

parallel execution of more independent or competing transactions, respectively.

The results respectively failures found in the existing LL model, e.g. failing to find a good strategy would help to identify the problematic parts of the model that need to be changed.

In the future, we plan to create a LL description of processes inside a large information system based on its specification to provide a tool for system's self-reflection.

REFERENCES

[1] ABRAMSKY, S. – GHICA, D. R. – MURAWSKI, A. S. – ONG, C. H. L.: (2004). *Applying Game Semantics to Compositional Software Modeling and Verification.* In: Proceedings of the International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS), LNCS 2988, (2004), pp. 421–435. Springer.

[2] BLASS, A.: (1992). *A game semantics for linear logic.* Annals of Pure and Applied Logic 56(1992), pp. 183–220.

[3] CHEN, I.-R.: (1995). *Stochastic Petri Net Analysis of Deadlock Detection Algorithms in Transaction Database Systems with Dynamic Locking*. The Computer Journal, Vol. 38, No. 9, 1995, pp. 717–733.

[4] DIMOVSKI, A.: (2007). *Compositional Software Verification Based on Game Semantics.* PhD thesis, University of Warwick, 188 p., July 2007.

[5] GIRARD, J-Y.: (1987). *Linear logic.* In: Theoretical Computer Science, ISSN 0304-3975, 1987, vol. 50, no. 1, pp. K 1–102.

[6] GIRARD, J-Y.: (1995). *Linear logic: Its Syntax and semantics.*, 1995, pp. K 1–42

[7] GIRARD, J-Y. – TAYLOR, P. – LAFONT, Y.: (1989). *Proofs and types.* Cambridge Tracts in Theoretical Computer Science, 7., Cambridge University Press, Cambridge-New York, 1989

[8] NOVITZKÁ, V. – MIHÁLYI, D. – SLODIČÁK, V.: (2006). *Linear Logical Reasoning on Programming.* Acta Electrotechnica et Informatica, vol. 6., no. 3., 2006, pp. 34–39, ISSN 1335-8243

[9] ONG, L.: (2004). *Game Semantics and its Applications.* In: ICCL Summer School, Dresden, Germany, June 2004.

[10] SLODIČÁK, V. – MACKO, P.: (2011). *How to apply linear logic in coalgebraical approach of computing* In: Proceedings of the 22nd Central European Conference on Information and Intelligent Systems, Varaždin, Croatia, pp. 373–380.

[11] SZABÓ, Cs. – SLODIČÁK, V.: (2011). *Software Engineering Tasks Instrumentation by Category Theory.* In: SAMI 2011, Proceedings of the 9th IEEE International Symposium on Applied Machine Intelligence and Informatics, Smolenice, Slovakia, pp. 195–199.

Received February 27, 2015, accepted April 27, 2015

BIOGRAPHIES

Veronika Szabóová PhD. student: graduated (MSc.) in the field of Informatics at the Dept. of Computers and Informatics of the Fac. of Electrical Engineering and Informatics (FEEaI) at Technical University of Košice in 2012. The title of her thesis was "Logical System for Game Semantics." Currently she is a PhD. student in the field of Informatics at the FEEaI at Technical University of Košice. Her research is focused on behavioral description of internal processes of information systems, autonomous computing, and software maintenance.

Csaba Szabó Assistant Prof.: graduated (MSc.) with distinction at the Dept. of Computers and Informatics of the

Fac. of Electrical Engineering and Informatics (FEEaI) at Technical University of Košice in 2003. He obtained his PhD. in Program- and Information Systems at the FEEaI at Technical University of Košice in 2007. Since 2006 he is affiliated with the Dept. of Computers and Informatics, FEEaI, Technical University of Košice. Currently is involved in research in the field of behavioral description of software, information systems and web services, software and test evolution, and testing and evaluation of software.

Valerie Novitzká Full Professor: defended her PhD thesis entitled "On semantics of specification languages" at the Hungarian Academy of Sciences in 1989. She works at Department of Computers and Informatics of the Fac. of Electrical Engineering and Informatics (FEEaI) at Technical University of Košice from 1998, firstly as Assistant Professor, from 2004 as Associated Professor, since 2008 as Full Professor in the field of Informatics. Her research areas cover category theory, categorical logic, type theory, classical and linear logic and theoretical foundations of program development.

Emília Demeterová PhD. student: graduated (MSc.) in the field of Informatics at the Dept. of Computers and Informatics of the Fac. of Electrical Engineering and Informatics (FEEaI) at Technical University of Košice in 2011. The title of her thesis was "Linear term calculus in informatics." Currently she is a PhD. student in the field of Informatics at the FEEaI at Technical University of Košice. Her research is focused on predicate linear logic, category theory, and linear logic programmig language – Lygon.