Formal REA model at operational level

Ito, Sohei and Vymetal, Dominik

Tokyo Institute of Technology, Silesian University, School of Business Administration

16 November 2011

Formal REA model at operational level.

Sohei Ito

Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology

ito@fmx.cs.titech.ac.jp

Dominik Vymětal

Silesian University in Opava, School of Business Administration in Karvina

vymetal@opf.slu.cz

Abstract

Despite a lot of attention gained by the Resource-Event-Agent (REA) framework among researchers in enterprise modeling, it still lacks comprehensive formal description. Most of the formalization approaches to REA use only UML or other graphical representation. This paper aims to define REA ontology at operational level using formal logic tools. The general approach to formal logic description of REA was motivated by L_{TAP} introduced by Ito, Hagihara and Yonezaki. After basic REA concepts are presented, semantics and logical language L_{REA} are defined including axioms for the REA operational level. Future research is shortly described in conclusion.

Keywords: REA framework, formal models, modal logic.

JEL classification: C600, L860, O210

Introduction

The Resource-Event-Agent (REA) framework and ontology have gained a lot of attention both among researchers in accounting and later, enterprise modeling. Proposed as generalization of accounting with the aim to solve some double-entry booking problems by McCarthy (McCarthy, 1982) and expanded later by Geerts and McCarthy (Geerts and McCarthy 2002,2006) into enterprise ontology it presents application neutral data model with some potential to be used in new designed ERP systems (Vandenbosche and Wortmann, 2006). However, the application of the REA model in present ERP systems meets some problems. First, present ERP systems are built around double-entry bookkeeping procedures and the change to REA model would cause major rewriting of the ERP code, which is not accepted due to costs. Second, REA lacks some clearly defined concepts needed for operational use. There is also formalization required that would be useful for ontology completeness testing and use in practice. Third, REA is in principle static description of the Enterprise Business process domain. To gain completeness needed by application analysts and programmers the behavior patterns are to be defined. Borch and Stephansen, 2004 analyzed the central pattern of the REA ontology – the economic exchange from operational point of view and proposed three points important for operational success: rigorous instance/type distinction, clearness in cardinality issues among modeling objects and thorough analysis between the tradeoffs between size and fit. The dynamic aspect of the REA modeling has been also thoroughly studied recently. Batra and Sin, 2008 proposed the UML sequence diagrams as modeling tool for operational aspects of REA. We proposed so called dynamization of REA static models at the operational level in (Vymetal, 2009; Vymetal et al., 2010) based on state diagrams, complemented by UML activity and sequence diagrams. The formalization of REA model was treated by several authors recently. Gaily and Poels, 2005

proposed three steps for the formal representation of domain ontology: Business Domain Ontology Formulation, Graphical Representation in UML and Formal Representations in RDS(S) or OWL. Buder, Koschital and Felden (2009) proposed Semantic based Planning Approach (SEMPA – Heinrich et al., 2008) to overcome some drawbacks of present REA ontology approaches. Murthy and Wiggins (2004) proposed an Object-oriented REA model using typical OO approach in order to capture the behavioral aspects of the REA modeling. However, most of the formalization approaches mentioned with the exception of Gaily and Poels (ibid) use UML or other graphical representation of REA only.

This paper aims to define REA ontology outline of the operational level using formal tools of normal modal logic. The motivation for this approach is based on the Ito, Hagihara and Yonezaki paper (Ito, Hagihara and Yonezaki, 2007) describing formal language L_{TAP} for another business process model – the Tasks-Agents-Products model. The paper is structured as follows. After the fundamental REA concepts presentation the formal description of REA ontology at operational level is described. First, the semantic structure, followed by logical language for REA operational level axiomatization is defined in section 2. Next, the axioms for operational level ontology are formulated. In section 4, the REA process and some REA theorems are presented followed by conclusion and discussion on next research steps regarding the REA policy level.

1 Basic description of REA fundamental concepts.

REA ontology can be seen upon as a two layer model consisting of the operational level containing instances of REA concept types and their relationships ("what has happened") and of the policy level layer where the abstract types of the REA concepts including intended and /or planned modeled objects ("what should happen") together with corresponding relations are placed. Following three basic semantic concepts are used at the operational level.

Economic resource (resource herein after) is the basic static concept characterizing resource reserved at disposal and under control of the enterprise. Resources are scarce, have their own value and are subject to control and monitoring by the enterprise personnel (users). Resources are related to economic events.

Economic event (event herein after) is a central notion in REA ontology. It can be described as a class of events reflecting changes of resource values. These changes comprise exchange, production, consumption, usage and distribution. Events are principal entities of enterprise information system describing inflow and outflow of resources.

Following relations can be named as examples of associations among resources and events: *outflow, inflow, consumption, usage, production* etc.

The last primary concept of REA is *Economic agent (agent herein after).* The agents can be represented either by individuals taking part in the enterprise processes, group of individuals playing some specified roles there, or complete enterprises. The agents in REA schemes are connected to events similarly like the resources. There are two associations (relations) defined, namely either *provide* or *receive*. The relation name represents the semantics of the relation: the agent either provides the resource to the event or receives the resource as a result of the event. In this paper, the enterprise is an economic agent from whose perspective we create the REA model of exchange and a perspective of some representative of the enterprise (e.g. foreman) is used for conversion. (Exchange and conversion definition is provided herein under).

Each REA business process has at least two coupled events: one decrement event consuming input resources and one increment event reflecting increase of resources. One or more decrement events are coupled with at least one increment event. The coupling of events in REA language is called *duality*.

REA framework distinguishes two basic event dualities, namely *exchange* and *conversion*.

During the process of *exchange* we can look upon the resource as a collection of some rights associated to the resource. Such rights can be e.g. property rights, usage rights, author rights etc...The purpose of the event in the process of exchange is to transfer some rights associated to a resource from agent providing the rights in question to another agent receiving the rights transferred. The central exchange idea stipulates that the providing agent provides his resources (e.g. goods) to receiving agent in order to increase his other resources (e.g. cash).

The purpose of the *conversion* process is the production of new resources or change of their properties by means of usage or consumption of other (input) resources. While the exchange duality takes place in simple rights exchange pertaining to resources connected to a company and a customer, the conversion duality represents the essence of production during which some new product is produced, or some properties of some existing product are changed in course of an increment event coupled with consumption and usage of the input resources in the decrement event.

Up to now, general REA model at the operational level presents the entities and relations "at the end of times". This view would not matter, if we were modeling the events as if they occurred simultaneously and do not take some time to be realized. In order to model this situation we could introduce the time factor into our formalism. However, for programming the application based on the REA model, it is more practical not to take the time factor in consideration and to look upon the events as running simultaneously.

The REA formalism in this paper describes both exchange and conversion process at the operational level. The principle of conversion duality is presented in Fig.1. The production is accomplished by three *agents* – the Foreman, the Warehouse clerk and the Worker. The production of the output *resource* Product is carried out by *increment event* Assembly &control which is coupled with several *decrement events*: Material issue, Tools usage, Labor consumption and Schedule knowledge consumption by conversion duality. The *decrement events* consume or use *input resources* Material, Tools, Labor and Schedule knowledge. Schedule knowledge *resource* carries necessary planning information and Bill of Material data necessary for the production order accomplishment. The relations among *resources*, *events* and *agents* complete the scheme presented.

The REA exchange process is schematically shown in Fig. 2. Here, the *agent* Enterprise provides its *resource* Product to the *decrement event* Sale. In course of this event the *agent* Customer receives the Product and provides *resource* Cash to the *increment event* from which the Enterprise receives the money. The value of the *resource* Product diminished from the Enterprise point of view while the value of the *resource* Cash was increased by means of the cash reimbursement.

Basic axioms of the REA ontology at operational level stipulate following:

- Each *increment event* must fulfill the *exchange duality*, or *conversion duality* with at least one *decrement event* and vice versa (duality axiom);
- Each *increment event* must have *inflow* relation for exchange or *production* relation for conversion with at least one *resource*;
- Each *decrement event* must have *outflow* relation for exchange and *consume or usage* relation for conversion with at least one *resource*;

These axioms result in creation of value chain strings where the input of one *decrement event* results in the output of an *increment event*. The original REA ontology does not provide a possibility to connect *resources* directly or to define loops in REA value chain. The connection is provided via dual economic events. As we show later, this idea requires some re-consideration at least for conversion process.

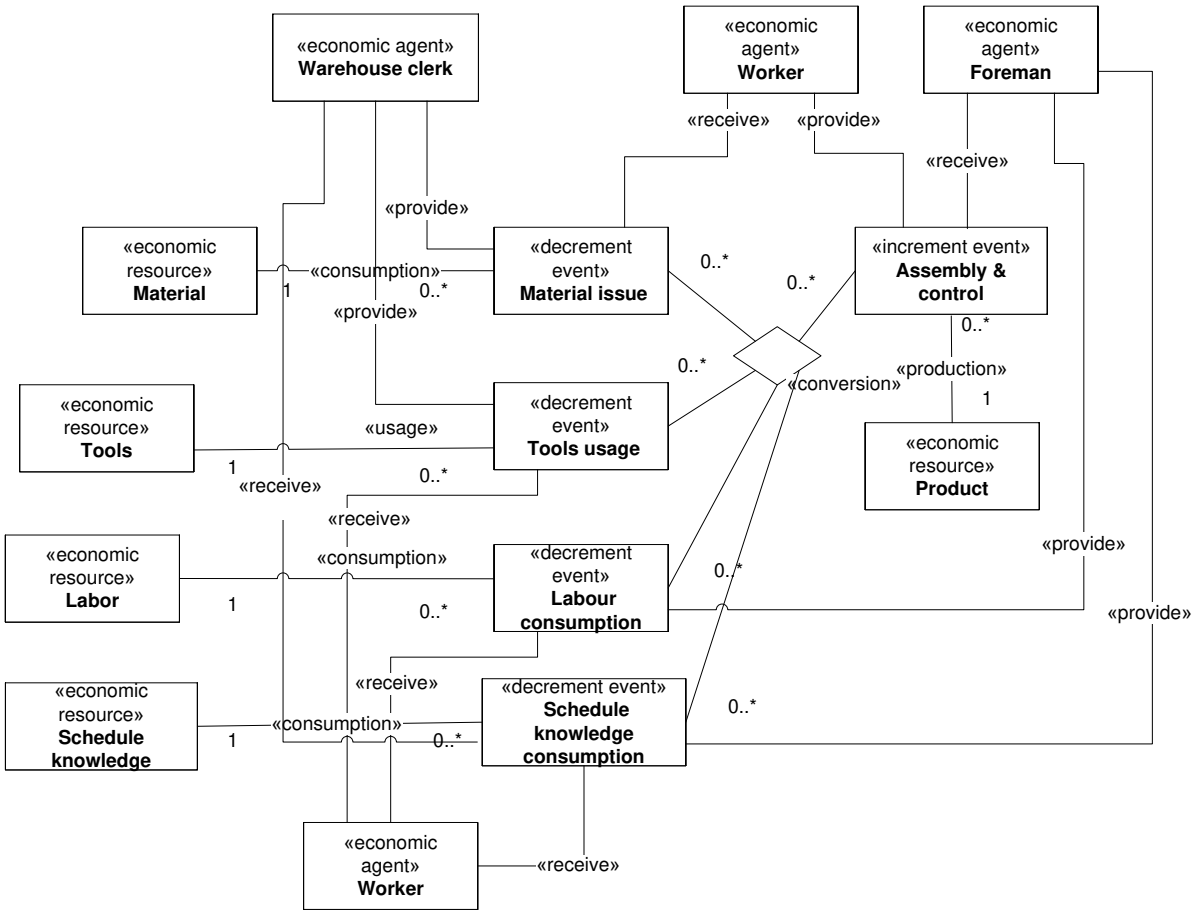

Fig. 1: REA model of conversion - a simple production order

Source: own

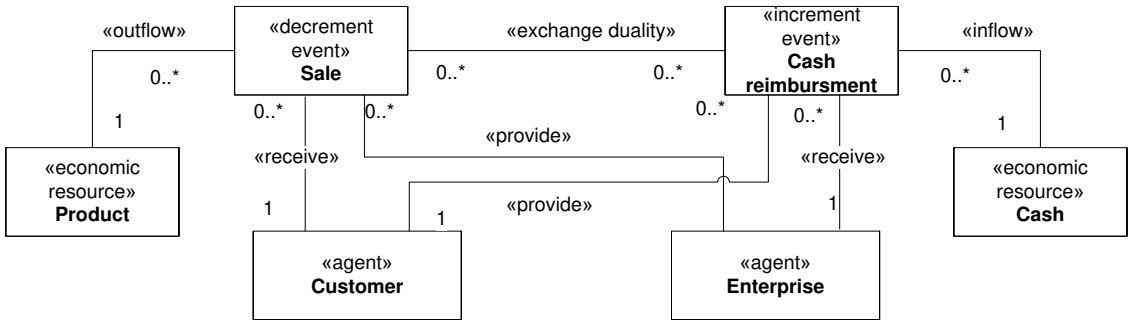

Fig. 2: REA model of exchange

Source: own

The application model must of course specify some attributes of the basic REA notions. So e.g. for events, the important attributes could be: date, quantity used, amount used etc., for the resource the important attributes would be e.g. quantity at disposal, unit of measure, and so

on, while for agents it could be the address data and others. The domains of attributes must be specified after the general model structure has been defined.

The extended REA ontology complements the operational level by the policy level. REA policy level can be defined as an infrastructure (layer) describing what should, could or must be occurring in future. The inputs to this layer result from planning and control activities. There exist three types of policy definitions (Geerts and McCarthy, 2006):

- Knowledge intensive descriptions (e.g. pricing rules)
- Validation rules (e.g. specifications of permitted values)
- Target descriptions (e.g. budgets)

However, in this paper we treat only the operational level and leave the axiomatization of policy level out of scope.

2 Operational level axiomatization

In this section we describe the axiomatization of REA basic concepts at the operational level.

2.1 Semantic structure

We assume a set of sort $\Sigma = \{R, E, A, Q, I\}$. An **operational frame** is a quintuple $\langle D, \rho, \delta, \boldsymbol{R}, \boldsymbol{F}\rangle$ where $D = \{D_\sigma\}_{\sigma\in\Sigma}$ is a set of domains of objects, $\rho = \rho[0]\rho[1]$ ... is a (possibly infinite) sequence of states, $\delta: D_E \to \rho$ is a total function, $\boldsymbol{R}$ is a set of intensional relations and $\boldsymbol{F}$ is a set of intensional functions. An intensional relation of arity $\sigma_0 \times ... \times \sigma_n$ is a total function from $\rho \to \mathcal{P}(D_{\sigma_0} \times ... \times D_{\sigma_n})$ and an intensional function of arity $\sigma_0 \times ... \times \sigma_n \to \sigma_{n+1}$ is a total function from $\rho \to D_{\sigma_0} \times ... \times D_{\sigma_n} \to D_{\sigma_{n+1}}$ where $\mathcal{P}(D)$ is a powerset of D. Moreover we assume D_Q and D_I are partially ordered by order relations $\leq_Q$ and $\leq_I$ respectively. Intuitively, D_R, D_E and D_A are domains for Resources, Events and Agents. D_Q and D_I are domains for attributes Quantity and Right of agents for each resource. One can imagine that the quantity domain is the set of (maybe nonnegative) real numbers or integers, so it is ordered. We think rights can be represented as e.g. percentage, so it is ordered too.

2.2 Logical language for REA operational level axiomatization

We introduce logical language L^O_{REA} (here O means "operational"). We need to mention, that "a relation holds at some time/always in future" or "an event occurs". The corresponding operators are therefore defined for the logical language proposed.

Let $\Sigma = \{R, E, A, Q, I\}$ be a sort. The symbols of L^O_{REA}consists of the followings:

1. The predicate symbols of arity E: $IncEv, DecEv$.
2. The predicate symbols of arity $E \times R$: $produce, consume, use, inflow, outflow$.
3. The predicate symbols of arity $A \times E$: $provide, receive$.
4. The predicate symbols of arity $E \times E$: $conversion, exchange$.
5. The predicate symbols of arity $Q \times Q$: $\leq_Q, =_Q$.
6. The predicate symbols of arity $I \times I$: $\leq_I, =_I$.
7. The function symbol of arity $R \to Q$: $quantity$.
8. The function symbol of arity $R \times A \to I$: $right$.
9. The sets of constant symbols $\{Con_\sigma\}_{\sigma\in\Sigma}$.
10. The sets of variable symbols $\{Var_\sigma\}_{\sigma\in\Sigma}$.

Terms and atoms are defined as usual in many-sorted logic (we assume that the set of atoms include the special symbol $true$). Formulas in L^O_{REA} are defined as follows:

1. Atoms are formulas.
2. If φ and ψ are formulas then $\neg\varphi, \varphi \wedge \psi, \Box\varphi, \overline{\Box}\varphi$ and $\circ\varphi$ are formulas.
3. If $e \in Con_E \cup Var_E$ then $\mathbf{Occurs}(e)$ is a formula.
4. If φ is a formula and $x \in Var_\sigma$ then $\forall_\sigma x\varphi$ is a formula, where $\sigma \in \Sigma$.

We introduce usual abbreviations: $false \stackrel{\text{def}}{=} \neg true, \varphi \vee \psi \stackrel{\text{def}}{=} \neg(\neg\varphi \wedge \neg\psi), \varphi \rightarrow \psi \stackrel{\text{def}}{=} \neg\varphi \vee \psi, \Diamond\varphi \stackrel{\text{def}}{=} \neg\Box\neg\varphi, \overline{\Diamond}\varphi \stackrel{\text{def}}{=} \neg\overline{\Box}\neg\varphi$ and $\exists_\sigma x\varphi \stackrel{\text{def}}{=} \neg\forall_\sigma x\neg\varphi$. We assume that $\wedge$ and $\vee$ binds more strongly than $\rightarrow$ and unary connectives bind more strongly than binary connectives.

Intuitively, $\Box\varphi$ means φ holds in any future and $\overline{\Box}\varphi$ means φ holds in any past. The abbreviation $\Diamond\varphi$ means φ holds at some future state and $\overline{\Diamond}\varphi$ holds at some past state. $\circ\varphi$ means φ holds at the next state. $\mathbf{Occurs}(e)$ means that event e occurs at this state. Now we formally define the semantics of L^O_{REA} with respect to operational frames.

Let $\mathfrak{O} = \langle D, \rho, \delta, \boldsymbol{R}, \boldsymbol{F}\rangle$ be an operational frame. An **interpretation** I of symbols with respect to $\mathfrak{O}$ is a function satisfying the followings:

1. $I(c) \in D_\sigma$ if $c \in Con_\sigma \cup Var_\sigma$ for $\sigma \in \Sigma$.
2. $I(p) \in \boldsymbol{R}$ if p is a predicate symbol where the arity of p and $I(p)$ are the same.
3. $I(f) \in \boldsymbol{F}$ if f is a function symbol where the arity of f and $I(f)$ are the same.

An **operational structure** is a pair $\langle\mathfrak{O}, I\rangle$. The semantics of L^O_{REA} is defined with respect to operational structures. Let $\mathfrak{W} = \langle\langle D, \rho, \delta, \boldsymbol{R}, \boldsymbol{F}\rangle, I\rangle$ be an operational structure. The interpretation of terms $(\mathfrak{W}, i)$ where i is an integer is the function satisfying the followings:

1. $(\mathfrak{W}, i)(c) = I(c)$ if $c \in Con_\sigma \cup \mathrm{Var}_\sigma$ for all $\sigma \in \Sigma$.
2. $(\mathfrak{W}, i)(f(t_1, \ldots, t_n)) = I(f)(\rho[i])((\mathfrak{W}, i)(t_1), \ldots, (\mathfrak{W}, i)(t_n))$.

The satisfaction relation $\models$ is defined inductively as follows:

1. $\mathfrak{W}, i \models p(t_1, \ldots, t_n)$ iff $\langle(\mathfrak{W}, i)(t_1), \ldots, (\mathfrak{W}, i)(t_n)\rangle \in I(p)(\rho[i])$.
2. $\mathfrak{W}, i \models t_1 \sim t_2$ iff $(\mathfrak{W}, i)(t_1) \sim (\mathfrak{W}, i)(t_2)$ where $\sim \in \{=_\sigma, \leq_\sigma \mid \sigma \in \Sigma\}$.
3. $\mathfrak{W}, i \models \neg\varphi$ iff $\mathfrak{W}, i \not\models \varphi$.
4. $\mathfrak{W}, i \models \varphi \wedge \psi$ iff $\mathfrak{W}, i \models \varphi$ and $\mathfrak{W}, i \models \psi$.
5. $\mathfrak{W}, i \models \forall_\sigma x\varphi$ iff $\mathfrak{W}[x \mapsto d], i \models \varphi$ for all $d \in D_\sigma$.
6. $\mathfrak{W}, i \models \Box\varphi$ iff $\mathfrak{W}, j \models \varphi$ for all $j \geq i$.
7. $\mathfrak{W}, i \models \overline{\Box}\varphi$ iff $\mathfrak{W}, j \models \varphi$ for all $j < i$.
8. $\mathfrak{W}, i \models \circ\varphi$ iff $\mathfrak{W}, i + 1 \models \varphi$ and $\rho[i + 1]$ exists.
9. $\mathfrak{W}, i \models \mathbf{Occurs}(e)$ iff $\delta((\mathfrak{W}, i)(e)) = \rho[i]$.

The notation $\mathfrak{W}[x \mapsto d]$ is the same as $\mathfrak{W}$ except that it has $I[x \mapsto d]$ as the interpretation of symbols which is the same as I except that it maps x to d. We simply write $\mathfrak{W} \models \varphi$ when $\mathfrak{W}, i \models \varphi$ for all $i < length(\rho)$ where $length(\rho)$ is the length of sequence $\rho = \rho[0]\rho[1]$ ... if ρ is finite and ω (i.e. the least infinite ordinal) if ρ is infinite.

Recall that δ was a *function*. This implies that $\mathbf{Occurs}(e)$ can be true at one time point at most. Readers might think that this definition is strange. In our opinion, events represent actual occurring of some economic event (e.g. transaction, transformation, etc.) in operational view of REA. Thus their occurrences are *unique*.

2.3 The axioms for REA operational level ontology

The purpose of axioms for REA operational level ontology is to answer the question "what is REA process". If we define a set of L^O_{REA} formulas Γ as the set of axioms for REA operational level ontology, then the answer to the question above can be formally described as $\{\mathfrak{M} \mid \mathfrak{M} \models \Gamma\}$. Thus, the set of models of axioms is *the* REA processes. To characterize REA processes

we use logical language L^{O}_{REA} introduced herein above. In this way we try to answer the question "what is REA process" formally in order to avoid ambiguous understanding of REA.

Axiom 1 (Rigidity)

This axiom stipulates that event types, duality relationships and provide/receive relationships are invariant, that is to say, they do not change in the course of a process. For example, if x is an increment event in a process, it is always an increment event through the whole process. It does not change to other type of events in that process.

$$\forall_E x\big(XXXEv(x) \rightarrow \Box XXXEv(x)\big)$$

$$\forall_E x \forall_E y(duality(x,y) \rightarrow \Box duality(x,y))$$

$$\forall_A a \forall_E x(p(a,x) \rightarrow \Box p(a,x))$$

Here $XXXEvent$ represents $IncrementEvent$ or $DecrementEvent$, $duality$ means $conversion$ or $exchange$ and p represents $provide$ or $receive$.

The reason why we introduced this axiom is that events in a process are not physical objects, but they are abstract objects existing in modelers' minds. Thus, the events are neither created nor disappear in course of process execution, that is to say, they exist all the time. The same is valid for duality relationships. Providing or receiving agents are associated to events, thus they are also rigid relationships.

Axiom 2 (typing for events)

The first formula says that there is no event which is both increment and decrement. The second is an artificial axiom which stipulates the notational convention about $duality$ predicate. It just says that in $duality$ predicate, the first argument is a decrement event and the second argument is an increment event.

$$\forall_E x \neg(IncEv(x) \wedge DecEv(x))$$

$$\forall_E x \forall_E y(duality(x,y) \rightarrow DecEv(x) \wedge IncEv(y))$$

Axiom 3 (duality)

There is at least one increment event necessary for one decrement event as its dual and vice versa.

$$\forall_E x\big(IncEv(x) \rightarrow \exists_E y\ duality(y,x)\big)$$

$$\forall_E x(DecEv(x) \rightarrow \exists_E y\ duality(x,y))$$

Axiom 4 (event occurring)

In conversion, at least one decrement event must happen before an increment event, and at least one decrement event must end at or after an increment event ends (i.e. resources cannot be produced from nothing). An example is using oven: it must be started before the cake is baked, and has to cool down after the cake is finished. In exchange duality, however, dual increment events may not have happened before decrement events occur since e.g. buyers may get goods before payment. The following formula describes this axiom:

$$\forall_E x \forall_E y(conversion(x,y) \wedge \mathbf{Occurs}(y) \rightarrow \exists_E z(conversion(z,y) \wedge^{\diamond} \mathbf{Occurs}(z) \wedge^{\diamond} \exists_R r\ con\_use(z,r))$$

Here con_use represents $consume$ or use.

Axiom 5 (events need resources)

Each increment event must have inflow or production relation with at least one resource and each decrement event must have outflow, consume or usage relation with at least one resource.

$$\forall_E x(DecEv(x) \wedge \mathbf{Occurs}(x) \rightarrow\diamond \exists_R r(inflow(x,r) \vee produce(x,r)))$$

$$\forall_E x(IncEv(x) \wedge \mathbf{Occurs}(x) \rightarrow\diamond \exists_R r(outflow(x,r) \vee consume(x,r) \vee use(x,r))$$

Axiom 6 (events need agents)

Each event must have at least one providing agent and one receiving agent.

$$\forall_E x(XXXEv(x) \rightarrow \exists_A a\ provide(a,x) \wedge \exists_A b\ receive(b,x))$$

Axiom 7 (inflow, produce, outflow, consume and use)

After the production, the quantity of the resource increases. Similarly, the quantity of the resource decreases after consumption. The quantity will not change when the resource is used.

$$\forall_E x \forall_R y \forall_Q z(in\_pro(x,y) \wedge quantity(y) =_Q z \rightarrow\circ quantity(y) >_Q z)$$

$$\forall_E x \forall_R y \forall_Q z(out\_con(x,y) \wedge quantity(y) =_Q z \rightarrow\circ quantity(y) <_Q z)$$

$$\forall_E x \forall_R y \forall_Q z(use(x,y) \wedge quantity(y) =_Q z \rightarrow\circ quantity(y) =_Q z)$$

Here in_pro represents $inflow$ or $produce$ and out_con represents $outflow$ or $consume$. Moreover, $x < y$ is the abbreviation of $x \leq y \wedge \neg x = y$.

One may think that this axiom is strange. For example, let us consider a process of aging wine or cognac. Quantity of them typically decreases during aging due to alcohol evaporation. So a vintage will decrease after the production process "aging." However, in operational REA view, we can distinguish as different resources the vintage before aging and after aging. This can be rephrased as that "aging" consumes, say, 10-year-old wine and produces 11-year-old wine. The quantity of "10-year-old wine" decreases and "11-year-old wine" increases. To be precise, this "aging" production process is "aging 10-year-old wine" process. Therefore in this production process, the resource "11-year-old wine" does not decrease due to the execution of "aging 11-year-old wine" process. Since the notion of time in our semantic structure is logical rather than physical, "aging" process can be decomposed to several "aging *-year-old wine" processes and they do not simultaneously occur at the same (logical) time in our semantic abstraction.

Axiom 8 (provide and receive)

Provide and receive relationships represent resource rights transferring between agents. That is to say, the right of the resource of the provider will decrease after the event and conversely the right of the resource of the receiver will increase after the event. This can be written as follows.

$\forall_E x \forall_R y \forall_A a \forall_A b \forall_I m \forall_I n(er(x,y) \wedge provide(a,x) \wedge receive(b,x) \wedge rights(a,y) =_I m \wedge rights(b,y) =_I n \rightarrow\diamond (rights(a,y) <_I m \wedge rights(b,y) >_I n))$ Here er represents $produce, consume, use, outflow$ or $inflow$.

Axiom 9 (totality of functions $quantity$ and $rights$)

Attribute $quantity$ is defined for any resource. Similarly, $rights$ is defined for any pair of agents and resources.

$$\forall_R x \exists_Q y(quantity(x) =_Q y)$$

$$\forall_A x \forall_R y \forall_I z(rights(x,y) =_I z)$$

3 REA process, theory and theorems.

3.1 REA process and theory.

Now we can define REA process at operational level. Let Γ^O be the set of axioms listed in section 2.3. Models of Γ^O are called **REA processes**. **REA theory** is the set of formulas which are true in any REA processes. Formulas in REA theory are called **REA theorems**.

3.2 REA theorems

Based on the presented L^O_{REA} language we can present some examples of REA theorems. Further research is needed to introduce a sound deductive system on L^O_{REA} therefore we present proofs of the theorems in examples purely semantically. The development of deductive system and proving of other REA theorems is a topic of next research.

The first theorem states that there is no event which is the dual event of itself.

Theorem 1 $\forall_E x \neg duality(x,x)$.

Proof. Immediate from axiom 2. ■

The next theorem states that there is no resource which is both consumed and used at the same time.

Theorem 2 $\forall_E x \forall_R r \neg\big(consume(x,r) \wedge use(x,r)\big)$.

Proof. Assume that $consume(x,r) \wedge use(x,r)$ holds in some REA process at some state. From axiom 9, $quantity(r) =_Q z$ holds for some z at this state. Then from axiom 7 both $quantity(r) <_Q z$ and $quantity(r) =_Q z$ must hold at the next state. However, it is impossible. ■

The third theorem states that there is no agent who provides to and receives from the same event.

Theorem 3 $\forall_E x \forall_A a \neg\big(provide(a,x) \wedge receive(a,x)\big)$.

Proof. The argument is similar as in theorem 2 except it is derived from axiom 8. ■

This theorem is an interesting result for REA researchers. This fact is accepted by REA researchers as true for exchange processes, but many models of conversion processes allow an agent to be both provider and recipient in the same event. In our axiomatization, however, this holds for any event (see axiom 8). This result implies that we might have to reconsider from the bottom how conversion processes are modeled in REA (note that our axiomatization in *provide* and *receive* relationships are modeled in view of rights transfer).

4 Conclusions

In this paper we presented formalized approach to REA ontology at the operational level. It could be shown, that based on defined L^O_{REA} language and elements of modal and temporal logic, the formal description of basic REA notions is possible and some REA theorems can be derived and proved by the logics proposed. By formalizing REA ontology, we found that the view of conversion processes in REA might be reconsidered.

Some interesting research topics are still open. First, in our axiomatization, value chain loops are still possible. Since value chains are naively characterized as transitive closure of single step association between resources through dual events, and transitive closures cannot be expressed in first-order logic, we cannot directly describe value chains in our language L^O_{REA} which is first-order. Therefore we need to extend L^O_{REA}, or to devise some "trick" to avoid

directly mentioning value chains to exclude value chain loops from REA processes. Second, extended REA ontology including policy level was not treated at all. REA policy level formal description could reveal some ambiguities in the REA ontology, which remained hidden at this stage. Third, a sound deductive system is to be built in order to enable syntactical proofs of REA. Fourth, a direct link to application design activities is to be established to prove the usefulness of the formalism proposed.

Acknowledgement

The authors wish to express their gratitude to dr. Pavel Hruby from REA technology DK for invaluable hints and remarks on the REA logical structure.

References

Batra, D., Sin, T. (2008). The READY Model: Patterns of Dynamic Behavior in REA-Based Accounting Applications. *Information Systems Management. 25:200/210.*ISSN 1934-8703. [online], [accessed 9.9.2010].

Borch, S.,E., Stephansen, C. (2004). Evaluating the REA Enterprise Ontology from an Operational Perspective. In: *Proceedings of the enterprise Modeling and Ontologies for Interoperability Workshop 2004 (EMOI-INTEROP 2004)* [Online], [accessed 7.12.2010], Available at: http://www.stefansen.dk/papers/ontology-paper-corrected.pdf

Buder, J-., Koschital, C., Felden, C. (2009). Formalization of REA Ontology. *Americas Conference on Information Systems (AMCIA 2009) Proceedings.* [online], [accessed 27.7.2010],available at: http://fak6.tufreiberg.de/fileadmin/Wirtschaftsinformatik/Publikatione/formalization_of_REA_ontology.pdf

Gailly, F. Poels, G. (2005). Development of a formal REA-Ontology representation. [online], [accessed 28.8.2010], available at: http://www. *citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.88.8152*

Geerts, G.,L., McCarthy, W.,E. (2002). An ontological analysis of the economic primitives of the extended-REA enterprise information architecture. *International Journal of Accounting Information Systems,* Vol. 3,Issue 1(March), pp. 1-16.

Geerts, G. L., McCarthy, W. E. (2006) Policy-Level Specification in REA Enterprise Information Systems. *Journal of Information Systems*. Vol 20, No. 2 pp. 37-63.

Heinrich, B., Bewernik, M., A., Hennberger, M., Krammer, A., Lautenbacher, F. (2008) SEMPA – Ein Ansatz der semantischen Prozessmanagements zur Plannung der Prozessmodellen. *Wirtschaftsinformatik (50), 6,* 445-460. [online], [accessed 12.9.2010], available at: http://www.uni-augsburg.de/exzellenz/kompetenz/kernkompetenzzentrum_fim/Forschung/paper/paper/wi-193.pdf.

Ito, S., Hagihara, S., Yonezaki, N. (2007): A Formal Ontology for Business Process Model TAP: Tasks-Agents-Products, In *Proceedings of the 17th European-Japanese Conference on Information Modeling and Knowledge Bases EJC2007*, pp294-301.

McCarthy, W.E. (1982). The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. *The Accounting Review* (July 1982).: 554-578

McCarthy, W.E. (2003). The REA Modeling Approach to Teaching Accounting Information Systems. *Issues in Accounting Education, Vol.18, No.4. pp.427-441.*

Murthy, U., Wiggins, C. (2004) OOREA: An Object-Oriented Resources, Events, Agents Model for Enterprise Systems Design, *Proceedings of the 25th ICIS*, December 12-14, New York, NY, USA Paper 16.[online], [accessed 22.1.2011], available at: http://aisel.aisnet.org/icis2004/1

Vandenbossche, P.,E.,A., Wortmann, J.,C., (2006). Why accounting data models form research are not incorporated in DERP systems. *Paper presented at the 2nd International REA Workshop, June 25, 2006, Fira Santorini Island, Greecel.*[online], accessed[12.11.2010],

Vymětal, D., Hučka, M., Huňka, F., Kašík, J. (2009). Process and Value Chain Oriented Modeling: Combining both Perspective into One. In Huzar, Z., Nawrocki, J., Szpyrka, M. (eds.) *Software Engineering Techniques in Progress.* Krakow: AGH University of Science and Technology Press, 2009: pp. 43-52.

Vymětal , D. , (2009). Value Chain and Process Models: is the Gap between them real? In *Distance Learning Simulation and Communication Proceedings.* Brno: Univerzita obrany Brno, 2009. s. 198-205. ISBN 978-80-7231-638-0.