Process Ontology

Marie Duží[1], Martina Číhalová[1], Marek Menšík[1,2], and Lukáš Vích[1]

[1] Department of Computer Science FEI, VSB-Technical University Ostrava
17. listopadu 15, 708 33 Ostrava, Czech Republic
[2] Institute of Computer Science FPF, Silesian University Opava
Bezručovo nám. 13, 746 01, Opava, Czech Republic
m.tina.cihal@gmail.com, marie.duzi@vsb.cz, mensikm@gmail.com,
lukas.vich.st@vsb.cz

Abstract. In the paper we propose a method of building up ontology of processes. First, a summary of our approach to ontology building using Transparent Intensional Logic (TIL) is presented. Since TIL operates smoothly at three levels of abstraction, namely hyperintensional, intensional and extensional level, we have a logical machinery to explicate concepts as hyperintensions which we define as TIL closed constructions in their normal form. Constructions are *procedures* that produce intensions or extensions as their products. Thus TIL is apt for modelling ontology as an extensional *logic of (hyper-)intensions*. However, while ontologies of a given specific domain are frequently studied, ontology of *processes* has been rather neglected. The goal of this paper is to fill the gap and to extend our method to the specification of process ontology. Since *logical* analysis presupposes full linguistic competency, we intend to make use of the results of linguistic analysis, in particular of verb-valency frames. Each process can be specified by a verb (what is to be done), possibly with parameters like the agent/actor of the process (who), the object to be operated on, resources, etc. Since valency frames correspond to senses of verbs, we thus obtain a finer specification of the process/procedure. Particular properties of the actor and other parameters are then specified as requisites of the process or its typical properties.

Key words: til, ontology, valency frames, process

1 Introduction

The term 'ontology' has been borrowed from philosophy, where ontology is a systematic account of existence. In recent computer science and artificial intelligence a formal ontology is an explicit and systematic conceptualization of a domain of interest. Given a domain, ontological analysis should clarify the *structure* of knowledge on what exists in the domain. A formal ontology is, or should be, a stable heart of an information system that makes knowledge sharing, reuse and reasoning possible. As J. Sowa says in [9], "logic itself has no vocabulary for describing the things that exist. Ontology fills that gap: it is the

Petr Sojka, Aleš Horák (Eds.): Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2010, pp. 77–88, 2010.

study of existence, of all the kinds of entities — abstract and concrete — that make up the world".

Current languages and tools applicable in the area of an ontology design focus in particular on the *form* of ontological representation rather than what a *semantic content* of ontology should be. Of course, a unified syntax is useful, but the problems of syntax are almost trivial compared to the problems of developing a common semantics for any domain. Duží and Materna in [3] dealt with semantic ontology content in general. In this paper we focus on a process ontology design. Each process can be specified by a verb (*what* is to be done), with parameters like the agent/actor of the process (*who*), the object to be operated on, resources, etc. Since valency frames correspond to senses of verbs, we thus obtain a specification of the process. As a specification tool we apply the *procedural semantic* framework of Transparent Intensional Logic (TIL) which is briefly introduced in Section 2. Section 3 provides a summary of an ontology content in general, and in the main Section 4 we propose a process ontology based on valency frames. The proposal is illustrated by an example of the analysis of an accident.

2 TIL in Brief

Since the pioneering paper [5] logicians and semanticists have striven to define so-called *structured meanings* that would comply with the principles of compositionality and universal referential transparency. Various adjustments of Frege's semantic schema have been proposed, shifting the entity named by an expression from the extensional level of atomic (physical/abstract) objects to the intensional level of molecular objects such as sets or functions/mappings. Yet natural language is rich enough to generate expressions that talk neither about extensional nor intensional objects. Propositional attitudes are notoriously known as the hard cases that are neither extensional nor intensional, as Carnap in (1947) [1] characterized them. It has become increasingly clear since the 1970s that we need to individuate meanings more finely than by possible-world intensions, and the need for hyperintensional semantics is now broadly recognised. Our position is a plea for *hyperintensional* semantics, which takes expressions as encoding *algorithmically structured procedures* producing extensional/intensional entities (or lower-order procedures) as their products. This approach — which could be characterized as being informed by an algorithmic or computational turn — has been advocated by, for instance, Moschovakis in (1994) [8]. Yet much earlier, in the early 1970s, Tichý introduced his notion of *construction* and developed the system of Transparent Intensional Logic (TIL).[3]

Constructions, as well as the entities they construct, all receive a type. The ontology of TIL is organized in an infinite, bi-dimensional hierarchy of types. Since we strictly distinguish between a construction of an object and the object itself, and between a function and its value, construction must be always of a higher order than the object it constructs, and a function is of a higher degree

[3] See Tichý (1988 and 2004) [11,12].

than its value. Thus one dimension of the type hierarchy increases molecular complexity of functions, the other dimension increases the order of constructions. Our definitions are inductive, and they proceed in three stages. First, we define the simple types of order 1 comprising non-constructions. Then we define constructions and, finally, the ramified hierarchy of types.

Definition 1. *(*types of order 1*). Let B be a* base*, where a base is a collection of pair-wise disjoint, non-empty sets. Then:*

(i) Every member of B is an elementary type of order 1 over B*.*
(ii) Let $\alpha, \beta_1, \ldots, \beta_m (m > 0)$ *be types of order 1 over B. Then the collection* $(\alpha\beta_1 \ldots \beta_m)$ *of all m-ary partial mappings from* $\beta_1 \times \ldots \times \beta_m$ *into* α *is a functional* type of order 1 over B*.*
(iii) Nothing is a type of order 1 over B *unless it so follows from (i) and (ii).*

The types *ad* (ii) are *functional* types. They are sets of *partial functions*, i.e., functions that associate every m-tuple of arguments with at most one value. Thus total functions are a special kind of partial functions.

The choice of the base depends on the area and language we happen to be investigating. When investigating purely mathematical language, the base can consist of, e.g., two atomic types; o, the type of truth-values, and ν, the type of natural numbers. For the purposes of natural-language analysis, we are currently assuming the following base of ground types, which is part of the ontological commitments of TIL:

o the set of truth-values $\{\mathrm{T}, \mathrm{F}\}$;
ι the set of individuals (the universe of discourse);
τ the set of real numbers (doubling as discrete times);
ω the set of logically possible worlds (the logical space).

Since *function* rather than relation is a primitive notion of TIL, we model *sets* and *relations* by their characteristic functions. Thus, for example, the set of prime numbers is a function of type $(o\tau)$ that associates any number with **T** or **F** according as the given number is a prime.

Definition 2. *(*intension and extension*)*
(PWS) intensions *are entities of type* $(\beta\omega)$*: mappings from possible worlds to some type* β*. The type* β *is frequently the type of the* chronology *of* α*-objects, i.e., mapping of type* $(\alpha\tau)$*. Thus* α*-intensions are frequently functions of type* $((\alpha\tau)\omega)$*, abbreviated as '*$\alpha_{\tau\omega}$*'.* Extensions *are entities of a type* α *where* $\alpha \neq (\beta\omega)$ *for any type* β*.*

Examples of frequently used intensions are:
propositions (denoted by declarative sentences) are of type $o_{\tau\omega}$; *properties of individuals* (usually denoted by nouns or intransitive verbs like 'is a student', 'walks') of type $(o\iota)_{\tau\omega}$; *binary relations-in-intension* between individuals are of type $(o\iota\iota)_{\tau\omega}$; *individual offices/roles*(usually denoted either by superlatives like 'the highest mountain' or terms with built-in uniqueness, like 'The President of the USA') are of type $\iota_{\tau\omega}$.

Expressions which denote non-constant intensions (i.e. functions that take different values in at least two world-time pairs) are *empirical*. Note that some

extensions involve the set of possible worlds, but not as their domain. For instance, a *set* of propositions is an extensional entity of type $(oo_{\tau\omega})$. On the other hand, a *property* of propositions, like being true in a world w at time t, is an intensional entity of type $(oo_{\tau\omega})_{\tau\omega}$.

Quantifiers $\forall^\alpha$, $\exists^\alpha$ are extensions, viz. type-theoretically polymorphous functions of type $(o(o\alpha))$, for an arbitrary type α, defined as follows. The *universal quantifier* $\forall^\alpha$ is a function that associates a class A of α-elements with **T** if A contains all elements of the type α, otherwise with **F**. The *existential quantifier* $\exists^\alpha$ is a function that associates a class A of α-elements with **T** if A is a non-empty class, otherwise with **F**.

The *singularizer* $Sing^\alpha$ is a partial type-theoretically polymorphic function of type $(\alpha(o\alpha))$ that associates a class C with the only α-element of C if C is a singleton, otherwise the function $Sing^\alpha$ is undefined.

Where A v-constructs a truth-value, i.e. an o-object and x v-constructs an α-object, we will often use the abbreviated notation '$\forall xA$', '$\exists xA$' and 'ιxA' instead of '$[^0\forall^\alpha \lambda xA]$', '$[^0\exists^\alpha \lambda xA]$', '$[^0\iota^\alpha \lambda xA]$', respectively, when no confusion can arise.

Constructions are assigned to expressions as their algorithmically structured, context-invariant meanings. When claiming that constructions are algorithmically structured, we mean the following. A construction C consists of one or more particular steps, or *constituents*, that are to be individually executed in order to execute C. The objects a construction operates on are not constituents of the construction. Just like the constituents of a computer program are its subprograms, so the constituents of a construction are its sub-constructions. Thus on the lowest level of non-constructions, the objects that constructions work on have to be supplied by other (albeit trivial) constructions. The constructions themselves may occur not only as constituents to be executed, but also as objects that still other constructions operate on. Therefore, one should not conflate *using* constructions as constituents of compound constructions and *mentioning* constructions that enter as input/output objects into compound constructions. So we must strictly distinguish between using constructions as constituents and mentioning constructions as objects.

Mentioning is, in principle, achieved by *using* atomic constructions. A construction C is *atomic* if it does not contain any other construction as a used sub-construction (a 'constituent of C') but C. There are two atomic constructions that supply entities (of any type) on which compound constructions operate: *Variables* and *Trivializations*. *Compound* constructions, which consist of other constituents than just themselves, are *Composition* and *Closure*. Composition is the instruction to apply a function to an argument in order to obtain its value (if any) at the argument. It is *improper*, i.e., does not construct anything, if the function is not defined at the argument. Closure is the instruction to construct a function by abstracting over variables in the ordinary manner of λ-calculi. Finally, higher-order constructions can be used once or twice over as constituents of constructions. This is achieved by a fifth and sixth construction called *Execution* and *Double Execution*, respectively.

Definition 3. (construction).

(i) *The variable x is a* construction *that v-constructs an object O of the respective type dependently on a valuation v.*

(ii) Trivialization*: Where X is an object whatsoever (an extension, an intension or a construction),* ^0X *is the construction* Trivialization*. It constructs X without any change.*

(iii) *The* Composition $[X\, Y_1 \ldots Y_m]$ *is the following* construction*. If X v-constructs a function f of type* $(\alpha\beta_1 \ldots \beta_m)$*, and* $Y_1, \ldots, Y_m$ *v-construct entities* $B_1, \ldots, B_m$ *of types* $\beta_1, \ldots, \beta_m$*, respectively, then the* Composition $[X\, Y_1 \ldots Y_m]$ *v-*constructs *the value (an entity, if any, of type α) of f on the tuple argument* $\langle B_1, \ldots, B_m\rangle$*. Otherwise the* Composition $[X\, Y_1 \ldots Y_m]$ *does not v-construct anything and so is v-improper.*

(iv) *The Closure* $[\lambda x_1 \ldots x_m\, Y]$ *is the following* construction*. Let* x_1*,* x_2*, …,* x_m *be pair-wise distinct variables v-constructing entities of types* $\beta_1, \ldots, \beta_m$ *and Y a construction v-constructing an α-entity. Then* $[\lambda x_1 \ldots x_m\, Y]$ *is the construction* λ-Closure *(or* Closure*). It v-constructs the following function f of the type* $(\alpha\beta_1 \ldots \beta_m)$*. Let* $v(B_1/x_1, \ldots, B_m/x_m)$ *be a valuation identical with v at least up to assigning objects* $B_1/\beta_1, \ldots, B_m/\beta_m$ *to variables* $x_1, \ldots, x_m$*. If Y is* $v(B_1/x_1, \ldots, B_m/x_m)$*-improper (see iii), then f is undefined on* $\langle B_1, \ldots, B_m\rangle$*. Otherwise the value of f on* $\langle B_1, \ldots, B_m\rangle$ *is the α-entity* $v(B_1/x_1, \ldots, B_m/x_m)$*-constructed by Y.*

(v) *The* Single Execution ^1X *is the* construction *that v-constructs the entity v-constructed by X. Otherwise* ^1X *is v-improper.*

(vi) *The* Double Execution ^2X *is the following* construction*. If X v-construct a construction Y and Y v-construct an entity Z, then* ^2X *v-constructs Z. Otherwise* ^2X *is v-improper.*

(vii) *Nothing is a* construction*, unless it so follows from (i) through (vi).*

Notation and abbreviations.
– 'X/α' means that the object X is (a member) of type α;
– '$X \rightarrow_v \alpha$' means that the type of the object v-constructed by X is α. We use '$X \rightarrow \alpha$' if what is v-constructed does not depend on a valuation v;
– We will standardly use the variables $w \rightarrow_v \omega$ and $t \rightarrow_v \tau$;
– If $C \rightarrow_v \alpha_{\tau\omega}$, the frequently used Composition $[[Cw]t]$, the intensional descent of the α-intension v-constructed by C, will be written as 'C_{wt}';
– When using constructions of truth-value functions, namely $\wedge$ (conjunction), $\vee$ (disjunction) and $\supset$ (implication) of type (ooo), and $\neg$ (negation) of type (oo), we often omit Trivialisation and use infix notion;
– When using identity relations $=^\alpha$ $/(o\alpha\alpha)$, we often omit the superscript α and use infix notation, whenever no confusion arises.

As mentioned above, constructions themselves are objects and thus also receive a type. Only it cannot be a type of order 1, because a construction cannot be of the same type as the object it constructs. Constructions that construct entities of order 1 are *constructions of order 1*. They belong to a type of order 2, denoted by '$*_1$'. This type $*_1$, together with atomic types of order 1, serves as the base for the following induction rule: any collection of partial mappings,

type $(\alpha\beta_1 \ldots \beta_n)$, involving $*_1$ in their domain or range is a *type of order 2*. Constructions belonging to the type $*_2$, which identify entities of order 1 or 2, and partial mappings involving such constructions, belong to a *type of order 3*; and so on *ad infinitum*.

The definition of the ramified hierarchy of types decomposes into three parts. First, simple types of order 1 were already defined by Definition 1. Second, we define constructions of order n, and third, types of order $n + 1$.

Definition 4. *(*ramified hierarchy of types*).*

$\mathbf{T}_1$ (types of order 1). *See Definition 1.*

$\mathbf{C}_n$ (constructions of order n).

i) *Let x be a variable ranging over a type of order n. Then x is a* construction of order n over B.

ii) *Let X be a member of a type of order n. Then* ${}^0X$, ${}^1X$, ${}^2X$ *are* constructions of order n over B.

iii) *Let* X, X_1,..., X_m $(m > 0)$ *be constructions of order n over B. Then* $[X\, X_1 \ldots X_m]$ *is a* construction of order n over B.

iv) *Let* $x_1, \ldots, x_m$, X $(m > 0)$ *be constructions of order n over B. Then* $[\lambda x_1 \ldots x_m\, X]$ *is a* construction of order n over B.

v) *Nothing is a* construction of order n over B *unless it so follows from* C_n *(i)-(iv).*

$\mathbf{T}_{n+1}$ (types of order $n + 1$). *Let* $*_n$ *be the collection of all constructions of order n over B. Then*

i) $*_n$ *and every type of order n are* types of order $n + 1$.

ii) *If* $0 < m$ *and* $\alpha, \beta_1, \ldots, \beta_m$ *are types of order* $n + 1$ *over B, then* $(\alpha\, \beta_1 \ldots \beta_m)$ *is a* type of order $n + 1$ over B.

iii) *Nothing is a* type of order $n + 1$ over B *unless it so follows from* T_{n+1} *(i) and (ii).*

So much for the logical machinery we are going to apply in the next paragraphs in order to specify the process-ontology content.

3 Ontology Content

Formal ontology is a result of the conceptualization of a given domain. Typically, a formal ontology encompasses these parts:

(1) Conceptual (terminological) dictionary which contains:
   a) primitive concepts
   b) compound concepts (ontological definitions of entities)
   c) the most important descriptive attributes, in particular identification of entities

(2) Relations
   a) contingent empirical relations between entities, in particular the part-whole relation
   b) analytical relations between intensions, i.e., requisites and essence, which give rise to ISA hierarchy

(3) Integrity constraints
   a) analytically necessary rules
   b) nomologically necessary rules
   c) common rules of 'necessity by convention'

The process of an ontology design usually begins with the specification of primitive concepts, i.e., Trivializations of objects that are not constructions. These primitive concepts are supposed to be commonly understood and they are not further refined. For instance, primitive concepts of traffic-system ontology might be ${}^0Agent$, ${}^0Lane$, ${}^0Crossroads$, etc. Next we specify compound concepts as ontological definitions of entities of a given domain. For instance, a road can be defined as consisting of two or more lanes which pass from a crossroad to another crossroad.

When specifying relations between entities, we distinguish between empirical and analytical relations. The former are relations-in-intension, mostly between individuals, like the part-whole relation. Analytical relations are relations-in-extension between intensions, for instance, a requisite relation and a property typical for another property. These relations give rise to ISA taxonomies. For instance, that a driver is a person is analytically necessary proposition *TRUE* that takes the value **T** in all $\langle w, t\rangle$-pairs. The requisite relation of the type $(o(o\iota)_{\tau\omega}(o\iota)_{\tau\omega})$ between the property of being a driver and the property of being a person is defined as follows:

$$[{}^0Req\ {}^0Person\ {}^0Driver] =_{df} \forall w \forall t[\forall x[[{}^0Driver_{wt}\ x] \supset [{}^0Person_{wt}\ x]]]$$

*Gloss.*Being a person is a requisite of being a driver. In other words, necessarily for any individual x, if x instantiates the property of being a driver then x also instantiates the property of being a person.

On the other hand, a driver typically owns a car, unless he is a chauffeur or a chauffeuse working for somebody else without owning a car. We say that owning a car is a typical property of a driver:

$$[{}^0Typically\ {}^0OwnCar\ {}^0Driver\ {}^0Exception] =_{df}$$
$$\forall w \forall t[\forall x\,[\neg[{}^0Exception_{wt}\ x] \supset [[{}^0Driver_{wt}\ x] \supset [{}^0OwnCar_{wt}\ x]]]]$$

Requisites and typical properties can obtain between intensions of any type. Here we define these relations only between properties of individuals. The other kinds can be easily deduced from this one. Let $p, q, exc \rightarrow_v (o\iota)_{\tau\omega}$; $x \rightarrow_v \iota$; $True/(oo_{\tau\omega})_{\tau\omega}$: the property of a proposition of being true in a world w at time t. Then q is a *requisite* of p, if and only if

$$\forall w \forall t[\forall x[[{}^0True_{wt}\ \lambda w \lambda t[p_{wt}\ x]] \supset [{}^0True_{wt}\ \lambda w \lambda t[q_{wt}\ x]]]]$$

The relation of being a *typical property* is defined as follows:

$$\forall w \forall t[\forall x[\neg[{}^0True_{wt}\ \lambda w \lambda t[exc_{wt}\ x]] \supset$$
$$[[{}^0True_{wt}\ \lambda w \lambda t[p_{wt}\ x]] \supset [{}^0True_{wt}\ \lambda w \lambda t[q_{wt}\ x]]]]]$$

Gloss. p is typical of q unless *exc*(eption).

Note. Due to partiality, we must use the property of propositions of being true. It returns the value **T** if the given proposition takes the value **T** in a $\langle w, t\rangle$-pair, otherwise **F**. If we did not apply this property, then it might be the case that $[p_{wt}x]$ would be v-improper and thus the whole Composition $[[^0True_{wt}\,\lambda w\lambda t[p_{wt}\,x]]\ \supset\ [^0True_{wt}\,\lambda w\lambda t[q_{wt}\,x]]]$ would be v-improper as well, which means that the above Closure would construct **F**. This is wrong, for sure, because the relation of being a requisite, providing it is valid, then it is valid in all $\langle w, t\rangle$-pairs.

Example. The requisite of the property of having stopped smoking is the property of previous smoking. Thus for those individuals y who never smoked the Composition $[^0StopedSmoking_{wt}\,y]$ is v-improper, $StopedSmoking/(o\iota)_{\tau\omega}$; $y \to_v \iota$.

It is a well-known fact that hierarchies of intensions based on requisite relations establish *inheritance* of attributes and possibly also of operations. For instance, a driver in addition to his/her special attributes like having a driving license inherits all the attributes of a person. This is another reason for including such a hierarchy into ontology. This concludes our summary of the logic of intensions. Now we are going to investigate ontology of processes.

4 Ontology of Processes

The specification of processes is driven by the analysis of verbs that denote actions. For instance, the specification of the process of John's driving from Prague to Ostrava is given by the sense of the verb 'to drive' together with its arguments (*who* is driving: the actor, *from where*, *to where*, etc.).

Our analysis makes use of Tichý's (1980) [10], where such verbs are called emphepisodic verbs. They tell us what people *do* rather than what they *are*. Thus while attributive verbs like 'to be happy', 'to be a good pianist' ascribe to people empirical *properties*, John's driving from Prague to Ostrava is not a condition in which John may be. Rather, it is John's *behaviour*, a time consuming *process* consisting of a series of *events*.

Here we are not going to deal with an exact definition of an event and an episode/process. Referring for details to Tichý [10] (1980), we just briefly summarise. Tichý defines an event as a set of basic propositions (possibly with a time-shift) together with a proposition specifying time when the event occurs. Thus the type of an event is $(oo_{\tau\omega})$. Basic propositions are formed by the application of a basic property to an individual. Basic individual properties are the properties corresponding to pre-theoretical features of individuals which together constitute an intensional base of a given language. A series of events constitutes a process (Tichý's *episode*). Each process has assigned a time span when it occurs. Sure, John's driving from Prague to Ostrava on November 17 is another event than his driving on October 11. Moreover, each process has an *actor* who *does* the process. The *Does* relation is of type $(o\iota(o(oo_{\tau\omega})))_{\tau\omega}$, and it is

defined as follows:

$$\lambda w \lambda t[{}^0Does_{wt}\, x\, p] = \lambda w \lambda t \exists e[[p\, e] \wedge [{}^0Actor\, x\, e] \wedge [{}^0Occur_w\, e] \wedge [[{}^0Run\, e]\, t]]$$

Additional types: $x \rightarrow_v \iota$; $p \rightarrow_v (o(oo_{\tau\omega}))$: a process; $e \rightarrow_v (oo_{\tau\omega})$: an event; $Actor/(o\iota(oo_{\tau\omega}))$: the relation between an individual and an event in which this individual is involved;[4] $Occur/(o(oo_{\tau\omega}))_w$; $Run/((o\tau)(oo_{\tau\omega}))$: the function that given an event returns a time interval when the event occurs in w.

Gloss. An individual x does a process p in world w at time t iff there is an event belonging to the process such that the event occurs in world w and runs at time t, and x is an actor of the event.

In addition to the actor and the time span when the process occurs, many other parameters of a process are desirable to be followed. As stated above, the process specification is driven by episodic verbs. In order to determine the other parameters of a process, we make use of the results of linguistic analysis, in particular of verb-valence frames.

In linguistics, verb valence is characterized as the ability of a verb to be linked to other terms of the discourse.[5] This ability concerns the semantic level of a language that is the deep structure of a sentence. Since valence frames correspond to senses of verbs, we can obtain a finer specification of the process/procedure. From the logical point of view, a verb denotes a relation and the valence of the verb determines arity and the types of arguments of the relation. Thus each process can be specified by a verb (*what* is to be done), with the parameters like the agent/actor of the process (*who*), the objects to be operated on, resources, etc.

Referring for details to `http://ufal.mff.cuni.cz/vallex/2.5/doc/`, we quote:

> Within the *Functional Generative Description (FGD)* framework, valence frames in a narrow sense consist only of inner participants (both obligatory and optional) and obligatory free modifications. In VALLEX 2.5, valence frames are enriched with quasi-valence complementations. Moreover, a few non-obligatory free modifications occur in valence frames too, since they are typically related to some verbs (or even to whole classes of them) and not to others. (The other free modifications can occur with the given verb too, but they are not contained in the valence frame as their presence in a sentence is not understood as syntactically conditioned in FGD.)
>
> In VALLEX 2.5, a valence frame is modeled as a sequence of frame slots. Each frame slot corresponds to one (either required or specifically permitted) complementation of the given verb.
>
> Note on terminology: in this text, the term 'complementation' (dependent item) is used in its broad sense, not related to the traditional argument/adjunct (complement/modifier) dichotomy.

[4] Tichý calls this relation '*By*' and defines it as the relation that obtains between an individual a and an event e whenever a exemplifies the basic properties involved in e, that is those properties that generate basic propositions of which e consists of. [5] For details see, e.g., Lotko (2003) [7].

> The following attributes are assigned to each slot: *functor*, list of possible morphemic forms (realization), type of complementation.
> In VALLEX 2.5, functors (labels for 'deep roles'; similar to theta-roles) are used for expressing types of relations between verbs and their complementations. According to FGD, functors are divided into inner participants (actants) and free modifications (this division roughly corresponds to the argument/adjunct dichotomy). In VALLEX 2.5, we also distinguish an additional group of quasi-valence complementations.

Functors that occur in VALLEX 2.5 are devided into three groups, viz. *inner participants*, *Quasi-valence complementations* and *free modifications*. For our purpose, inner participants are the most important. They are as follows:

ACT (actor),
ADDR (addressee),
PAT (patient),
EFF (effect) and
ORIG (origin).

The other functors, if needed, will be explained when used. Consider, for instance, the simple sentence "Mary is sending a message to Tom". We have the process of *Sending* with three obligatory arguments: *Who* does the process (*ACT*: the actor of the process), *Whom* (emphADDR: addressee) and *What* (*PAT*: the message). Hence necessarily, whenever an actor does the process of *Sending* then there is an addressee and a patient of the process. We will say that *ADDR* and *PAT* are the *requisites* of the *Sending process*.

However, the requisite relation has been defined in Section 3 as the relation-in-extension between *intensions*, but a process is defined as a set of events, i.e., an *extension*, and *ADDR* is obviously an individual and *PAT* is a (hyper)proposition. Thus a question arises, in which sense can we say that the requisite relation obtains between a process and *ADDR*, *PAT*, respectively? As explained at the outset of this section, there are basic properties of an actor or other individuals involved in the process. The requisite relation between the *Sending* process and *ADDR* can thus be explicated as follows:

$$[{}^0Req_{pr}\, q\, {}^0Sending] =_{df}$$
$$\forall w \forall t \forall x [[{}^0Does_{wt}\, x\, {}^0Sending] \supset$$
$$\exists e [[{}^0Sending\, e] \wedge \exists p\, y [[e\, p] \wedge [p_{wt}] \supset [q_{wt}\, y]]]]]$$

Types: $Sending/(o(oo_{\tau\omega}))$; $x, y \rightarrow_v \iota$; $e/(oo_{\tau\omega})$; $p \rightarrow_v o_{\tau\omega}$; $q \rightarrow_v (o\iota)_{\tau\omega}$.

In our example the property q would be the property of being an addressee of the message. Similarly for *PAT*.

The last notion we need to introduce into our ontology is the *classification* of processes. Sure, the process of somebody's getting up or singing the Czech anthem is of a different kind than, for instance, the process of having a car crash. Accordingly we can then assign requisites to all the members of a particular class of processes.

As an example we adduce the processes of the class *Accident*. The obligatory attributes (requisites) of each accident are as follows:

- *ACT* – the actor(s) who caused the accident
- *PAT* – those who are involved in the accident
- *TWHEN* – time when the accident happened to occur

The other optional attributes (typical properties) are:

- *LOC* – where the accident occurred
- *RCMP* – damages
- *CAUS* – the cause of the accident
- *EFF* – the effects of the accident
- *BEN (benefactive)* – who is to be compensated

Having specified these requisites and typical properties of an accident, we can then specify ontological rules like

"The actor is responsible for the damage"
"The patient is the one to be compensated"
"The actor caused the accident"
"Typically, the actor is due to cover the accident compensation"
etc.

These rules make it possible to infer consequences of an accident, to calculate the damage, etc. In this way the process ontology can serve as an information resource for reasoning about the process.

5 Conclusion

In this paper we introduced the method of building up an ontology of processes. We applied the method of logical analysis of natural language expressions as provided within TIL, and made an attempt to exploit the results of linguistic analysis as provided by the so-called verb-valence frames.

Yet we wish to say that the results presented here are just the first proposal, and a lot of problems remain open. For instance, we only briefly tackled the problem of the classification of processes, the problem of inferring consequences of a process occurrence, etc. Thus the ontology of processes is still much work in progress.

Acknowledgements

This research has been supported by the Grant Agency of the Czech Republic, projects No. GACR 401/10/0792, 'Temporal aspects of knowledge and information' and 401/09/H007 'Logical Foundations of Semantics', and also by the internal grant agency of FEECS VSB-TU Ostrava – IGA 22/2009 'Modelling, simulation and verification of software processes'.

References

1. Carnap, R.: *Meaning and Necessity*, Chicago University Press (1947).
2. Duží, M., Číhalová, M., Menšík, M: 'Ontology as a logic of intensions'. In *European-Japanese Conference EJC 2010*, A. Heimburger, Y. Kiyoki, T. Tokuda, N. Yoshida (eds.), Jyvaskyla, Finland: University of Jyvaskyla, 9–28 (2010).
3. Duží, M., Materna, P.: 'Concepts and Ontologies'. In *Information Modelling and Knowledge Bases XX*, Y. Kiyoki, T. Tokuda, H. Jaakola, X. Chen and N. Yoshida (eds.), Amsterdam: IOS Press, (2009) 45–64.
4. Duží, M., Jespersen, B., Materna, P.: *Procedural Semantics for Hyperintensional Logic (Foundations and Applications of TIL)*. Berlin/Heidelberg: Springer, series Logic, Epistemology, and the Unity of Science (2010).
5. Frege, G.: 'Über Sinn und Bedeutung', *Zeitschrift für Philosophie und philosophische Kritik*, vol. 100 (1892), 25–50.
6. Hajičová, E.: 'What we are talking about and what we are saying about it'. In *Computational Linguistics and Intelligent Text Processing*, LNCS Springer Berlin/Heidelberg, vol. 4919, 241–262 (2008).
7. Lotko, E.: *Slovník lingvistických termínů pro filology*, Olomouc (2003).
8. Moschovakis, Y.N.: 'Sense and denotation as algorithm and value'. In J. Väänänen and J. Oikkonen (eds.), *Lecture Notes in Logic*, vol. 2, Berlin: Springer, pp. 210–249 (1994).
9. Sowa, J.: *Knowledge Representation.* Logical, Philosophical, and Computational Foundations. Brooks/Cole (2000).
10. Tichý, P.: 'The semantics of episodic verbs', *Theoretical linguistics*, 7, 263–296 (1980). Reprinted in: Tichý (2004, pp. 409–444).
11. Tichý, P.: *The Foundations of Frege's Logic*, Berlin, New York: De Gruyter (1988).
12. Tichý, P.: *Pavel Tichý's Collected Papers in Logic and Philosophy*, Dunedin: University of Otago Press; Prague: Filosofia, Czech Academy of Sciences (2004).