Logical Operators for Ontological Modeling

Stefano BORGO [1], Daniele PORELLO [2], and Nicolas TROQUARD [3]
Laboratory for Applied Ontology, ISTC CNR, Trento, Italy.

Abstract. We show that logic has more to offer to ontologists than standard first order and modal operators. We first describe some operators of linear logic which we believe are particularly suitable for ontological modeling, and suggest how to interpret them within an ontological framework. After showing how they can coexist with those of classical logic, we analyze three notions of artifact from the literature to conclude that these linear operators allow for reducing the ontological commitment needed for their formalization, and even simplify their logical formulation.

Keywords. Logical operator, Ontological modeling, Linear logic, Artefact

1. The Logical Language in Ontology

If we look at the formalisations of ontologies, even of foundational ontologies, we notice that they usually embrace a standard logical system, like OWL 2 [22] and first-order logic [9], and stick to it, e.g. BFO [22], DOLCE [21] and GFO [14]. Note that the ontologies may be developed in more than one logical framework, for instance BFO has an OWL version and is developing a first-order version; DOLCE has both a first-order and an OWL-DL version. Nonetheless, these formalisms are disconnected in the sense that the user has to choose in which logic framework, among those made available, she wants to operate. The only considerable exception to this situation is the use of modal logic within first-order logic, see for instance the use of the necessity operator in DOLCE.

In this paper we put ourselves in the shoes of the ontologists that, after having carried out the study of the domain and set the ontological framework, is facing the challenge of formalizing the overall system. Her goal is to capture as faithfully as possible the identified notions and their interactions. We claim that if she is not aware of the different effects of modeling a constraint in one or another logical language, she cannot make an optimal choice and may be driven to develop unnecessarily complex systems in terms of (1) the number of axioms; (2) the types of entities that she needs to include; and even (3) the number of ontological choices she has to make. The result is a formalization that is harder to understand, and that could even obfuscate the truly ontological consequences of her original analysis.

[1] stefano.borgo@cnr.it
[2] daniele.porello@loa.istc.cnr.it
[3] nicolas.troquard@loa.istc.cnr.it

Today, when addressing formal languages, most ontologists have two issues in mind: the balance of expressivity and deduction properties, and the selection of the most appropriate non-logical primitives. This, we claim, is not enough. If the ontologist is unaware of the variety of the available logical operators, her choice of the language may easily fall short of being optimal leading to poor ontological formalizations as we will see.

Structure of the paper. Section 2 reports three definitions of artifacts that have been presented in the literature and that we will use to exemplify our logical proposal. Section 3 presents the logical framework of linear logic and describes its operators. The following section, Section 4, introduces a linear logic based language suitable for modeling the artifact definitions and, together with Sections 5 and 6, show how to distinguish these definitions with simple logical formulas. Section 7 concludes the paper.

2. Distinguishing technical artefacts

There is more than one way of being a technical artefact, and this diversity makes it a fitting class to illustrate our claims. We will do so by taking the three notions of artefact presented in [4] and showing that a choice of sub-structural logical operators can simplify their characterisations. We will work specifically with Linear Logic [10]. In this logical framework, we will argue that by using a logic that encompasses first-order logic as well as the operators of linear implication and non-commutative conjunction, we will be able to simplify the ontological characterization of the three notions. In this context, what will make a characterization "satisfactory" will essentially be its combined succinctness and intelligibility.

In [4] the authors discuss three definitions that aim to capture different uses and understandings of the notion of technical artefact. The definitions were developed by studying different communities: formal ontology, engineering design and philosophy of technology. The authors labeled these definitions *ontological*, *engineering* and *technological*, respectively. While there is no claim that these definitions are representative of the communities, it is argued that the elements they take at the core of each definition are essential for the understanding of the notion in those areas.

The first definition, called *ontological artefact*, is a specialization of the general notion of artefact introduced in [3]: "A[n ontological] *technical artifact* $\mathfrak{a}$ is a physical object which an agent (or group of agents) creates by two, possibly concurrent, intentional acts: the selection of a material entity (as the only constituent of $\mathfrak{a}$) and the attribution to $\mathfrak{a}$ of a technical quality or capacity." [4, p.5] The second definition, called *engineering artefact*, emerges from studies in the area of engineering design [17]: "A[n engineering] *technical artifact* $\mathfrak{a}$ is a physical object created by an intentionally performed production process. The process is intentionally performed by one or more agents with the goal of producing the object $\mathfrak{a}$ which is expected to realize intended behavior in some given generic technical situation." [4, p. 7] At about the same period, a definition of technical artefact was proposed by philosophers studying technology [15]. "A [technological] *technical artifact* $\mathfrak{a}$ is a physical object created by the carrying out by an agent (or by agents) of a make plan for an object with a physical description." [4, p. 9].

Our goal is to show that a suitable choice of logic operators highly simplifies the formalization of these notions. However, we do not aim to model all their aspects and for this reason we rephrase them in a simplified format. The reader can find in [4] a

presentation of their motivations, their analysis and their comparison, and even verify that our rephrasing captures the core of the definitions.

Restatement 1 (Ontological Artifact)**.** *A* technical artifact $\mathfrak{a}$ *is a physical object which an agent creates by means of two intentional events: the selection of a material entity and the attribution to* $\mathfrak{a}$ *of a technical property.*

Restatement 2 (Engineering Artifact)**.** *A* technical artifact $\mathfrak{a}$ *is a physical object which an agent creates by an intentional event consisting in: the transformation of a physical object into an object* $\mathfrak{a}$ *that is expected to manifest some technical property.*

Restatement 3 (Technological Artifact)**.** *A* technical artifact $\mathfrak{a}$ *is a physical object which an agent creates by an intentional event consisting in the execution of a make plan, terminating with a verification step, to obtain a physical object* $\mathfrak{a}$ *that satisfies a given description.*

A general definition pattern. One common element in these definitions is that every notion exclusively depends on an acting entity bringing about a specific intentional event: the artefact is the product of this specific event. The specificity of the event marks the specificity of the resulting technical artefact. Take $Crea_O(e,\mathfrak{a})$ (resp. $Crea_E(e,\mathfrak{a})$ and $Crea_T(e,\mathfrak{a})$) to mean that "e is an event that produces the ontological (resp. engineering and technological) artefact $\mathfrak{a}$".

We thus can see a pattern emerging

$$\begin{array}{lcl} Art_O(\mathfrak{a}) & \equiv & \exists e.\, Crea_O(e,\mathfrak{a}) \\ Art_E(\mathfrak{a}) & \equiv & \exists e.\, Crea_E(e,\mathfrak{a}) \\ Art_T(\mathfrak{a}) & \equiv & \exists e.\, Crea_T(e,\mathfrak{a}) \end{array}$$

It now suffices to define the three versions of being an event producing a technical artefact.

The simple case of ontological artefacts The case of ontological artefacts is formally developed within the DOLCE foundational ontology in [3] but is stated not to be ontologically dependent on it; it is then a normative notion justified by ontological considerations. As such, we can directly import the ontology. The central predicate for intentional selection ($IntentionalSel$) provides the conceptual handle needed to capture the event that is at work in the creation of ontological artefacts. $IntentionalSel(e,a,\mathfrak{a},y,q)$ is read in [3] as "e is the event of a obtaining the artefact $\mathfrak{a}$ by intentionally selecting y and attributing to it capacity q." Hence,

$$Crea_O(e,\mathfrak{a}) \equiv \exists a,y,q.\, IntentionalSel(e,a,\mathfrak{a},y,q)$$

We obtain effortlessly the exact same characterization of $Art_O(\mathfrak{a})$ as given in [3].

An important aspect from the ontology in [3] is that the selected object and the object to which a quality is attributed coexist. There is no consumption, nor creation. Thus, a naive attempt at characterizing ontological artefacts with FOL is rather satisfactory. Things will be different for engineering artefacts and for technological artefacts. Physical change as in transformations and make plans demand much care to capture their dynamic aspect.

3. Linear Logic

As said, we are not advocating new logical approaches to knowledge representation and ontology design; rather we are suggesting to take into account the possibilities that logic already provides us to better cope with a number of modeling needs and possibly reduce the ontological commitments. The formal ontologist is somehow familiar with this approach when she extends first-order logic with tools like modal operators and sorts. Since logical operators are much richer than is generally thought, we want to push their use in ontological theories a little further. We illustrate this by analyzing Girard's Linear Logic [10]. Linear logic is capable of classical reasoning, and its language (L_{LL}) provides more fine-grained operators. Linear logic, besides providing a well-behaved resource-sensitive reasoning method, allows indeed for reconstructing classical logic, a feature we will make extensive use of.

Since our goal is to extend the ontologist's toolkit, not to substitute it, our first aim is to show how classical and linear logics can coexist in the same logical framework. The simplest way to present this relationship is to argue in terms of proof-theory. We introduce some elements of the sequent calculus for classical and linear logic and use these to also present the different commitments and strengths of classical and linear operators. Moreover, sequent calculus can be viewed as the abstract theory that leads to developing logic programming, thus it gives the further advantage of defining, at least theoretically, a query answering system.

The reader interested in formal yet more familiar definitions of the set of theorems of Linear Logic can refer to the Hilbert axiomatization given in [25]. Also, semantics for Linear Logic exist in terms of phase semantics [10] and relational semantics [5]. The former is the original semantics designed by Girard; the latter is a recent approach that further supports modular extensions.

3.1. A constructive analysis of reasoning

The main motivation for employing non-classical, sub-structural, relevant, or linear logic operators [10,2,20] is typically to capture a form of resource-sensitive reasoning and cope with the paradoxes of material implication. In particular, classical implication $A \to B$ is not apt to model processes, transformations, causal entailments, etc. For example, if formula $E \to C$ is used to formalize the behavior of a coffee machine "before I have one euro (E), after I have one coffee (C)", we would like to distinguish this from formula $E \to E \wedge C$ ("before I have one euro, after I have one euro and one coffee") but classical implication does not. For another example, chemical reactions such as $H_2 + O \to H_2O$ are transformations that are found in nature [12] and should not be modelled as a material implication. Moreover, modeling by means of classical implication the transformation of a piece of clay into the statue that the artist is shaping, we would like to be free to assume whether the clay is still present alongside the statue or not depending on our ontological commitment and not on the type of logic we are using. This feature of classical logic follows from the fact that $A \to A \wedge A$ is a tautology in the system and it basically amounts to assuming that any formula is equivalent to the conjunction of an arbitrary number of its copies. Symmetrically, once we accept the classical logical entailment $A \to B$, this operator holds regardless of the context of application so that, e.g., $A \wedge C \to B$ holds for any C. This is a consequence of the monotonicity of the entailment of classical logic.

These observations show that by representing knowledge via classical logic, we are abstracting away from use, context, quantities of formulas occurring when reasoning. Namely, we are committed to view propositions as abstract, objective, and eternal thoughts in the Fregean sense [8], and not as actual contextual dependent information. Of course, one can represent temporal transformations, causal relationships and so on in classical logic and most ontological theories do so within this framework. However, in order to do it, one needs to introduce a number of non-logical elements, like time stamps or indexing of proposition occurrences, and *ad hoc* constraints to control them. In order to clarify the commitments of classical logic and to propose a more general treatment, e.g. to allow resource sensitive reasoning, below we recall some ideas from proof-theory, namely from Gentzen sequent calculus. Classical formulas remain compatible with this system; they can be even captured in it by means of a number of rules.

A sequent is an expression $\Gamma \vdash \Delta$ where Γ and Δ are sets of occurrences of formulas, the $\vdash$ symbol is the entailment relation: the semantic reading of a sequent is "the conjunction of the formulas in Γ entails the disjunction of the formulas in Δ". Sequents and rules of inferences model classical logic in the sense that sequent calculus is sound and complete wrt. the standard classical logic semantics. A crucial feature of classical logic sequent calculus is the presence of the so called *structural rules* of weakening (W), contraction (C) and exchange (E).

$$\frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta}\,\mathrm{W} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}\,\mathrm{W} \qquad \frac{\Gamma, A, A, \vdash \Delta}{\Gamma, A \vdash \Delta}\,\mathrm{C} \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\,\mathrm{C}$$

$$\frac{\Gamma, A, B, \Gamma' \vdash \Delta}{\Gamma, B, A, \Gamma' \vdash \Delta}\,\mathrm{E} \qquad \frac{\Gamma \vdash \Delta, A, B, \Delta'}{\Gamma \vdash \Delta, B, A, \Delta'}\,\mathrm{E}$$

Intuitively, the structural rules encode three important forms of abstraction from actual situated information: Weakening is responsible of the monotonicity and thus implies the context independence of the classical reasoning; Contraction is responsible of abstracting from the quantity of available resources; and Exchange amounts to abstracting from the order in which information is provided, exchange forces the commutativity of logical operators and abstracts away from any ordering, and in particular temporal ordering, of information.

Alternative approaches to classical logic that aim to define logical operators capable of capturing, for example, causal implications, context dependent reasoning etc., often place themselves in the so-called realm of the *sub-structural logics*, namely logics that reject the validity of (at least some) structural rules. This is motivated by the fact that structural rules determine the definition of logical operators; an important observation due to Jean-Yves Girard [10] and that grounds the analysis of inferences provided by linear logic. Take, for instance, the following two rules that together define the classical conjunction in the sequent calculus lingo

$$\wedge\,\frac{\Gamma \vdash A, \Delta \qquad \Gamma' \vdash B, \Delta'}{\Gamma, \Gamma' \vdash A \wedge B, \Delta, \Delta'} \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}\,\wedge$$

On inspection, the difference between the two formulations lies in the context of the entailments. The first one allows one to use different contexts (Γ and Γ' and Δ and Δ')

whereas the second formulation to be applied requires the same context in both premises. In the presence of the structural rules, the two formulations can be proved equivalent. By rejecting the structural rules of weakening and contraction, these are no longer equivalent. At this point we have a choice between two conjunctions with two different behaviors: one is called the multiplicative conjunction and is formally written $\otimes$ (read "tensor"), and the other the additive conjunction, written & ("with"). The rejection of these structural rules does not affect only conjunction. We can repeat the argument for the disjunction as well ending up with two different operators: a multiplicative disjunction $⅋$ ("parallel"), and an additive disjunction $\oplus$ ("plus"). The rejection of exchange has a different consequence. In this case a sequence of propositions behaves like an ordered lists of formulas, whereas the rejection of contraction and weakening leads a sequence to behave as a multiset of occurrences of formulas. By rejecting exchange, we are led to define a non-commutative multiplicative conjunction $\odot$ ("next") and a non-commutative multiplicative disjunction ∇ ("sequential") [7,1].

We have seen that sub-structural logic is a logical system obtained by rejecting some of the structural rules. Linear logic is not simply a sub-structural logic as the full fragment contains two more operators, namely, the *exponential operators* ! and ?. These operators locally license the structural rules and allow for retrieving the full expressive power of classical logic. Informally, formulas $!A$ and $?A$ are relieved from their linear resource sensitive status and the "quantity of A" no longer matters. We shall come back to the exponential operators later in order to present how classical logic can be retrieved from within linear logic.

3.2. Semantics of a fragment of LL

Here we present the intuitive meaning of a number of linear logic operators. For their proof-theoretic definition and for a precise presentation of linear logic see [25,13].

Firstly, it is important to note that atomic formulas of linear logic do not represent, strictly speaking, states of affairs; that is, they are not abstract and eternal truths (eternal thoughts in Fregean terminology). If they were abstract truths, it would not make much sense to argue whether the truth that two plus two makes four is equivalent or not to the truth that "two plus two makes four and two plus two makes four". The meaning of linear logic propositions can be better understood in terms of events, actions and processes performed or occurring in an environment and that depend on an amount of resources. In this sense, A and $A \otimes A$ are not equivalent as one thing is the occurrence of an instance of A, where A is an event, another thing is the occurrence of $A \otimes A$, i.e. the occurrence of two instances of A (perhaps at different times). It is this feature of linear logic that leads to see this framework as a further formal tool for the ontologist.

We provide some hints to the semantics of linear logic by focusing in particular on the commutative multiplicative conjunction $\otimes$, the non-commutative conjunction $\odot$, the linear implication $\multimap$, and the additive conjunction & and disjunction $\oplus$. The reason for this restriction is that such a fragment of linear logic allows for an adequate simplified semantics that is informative enough for the purpose of this paper[4]. The semantics for full linear logic is presented in [10,25]. The structure that we propose here is adapted from Urquhart semantics for relevant logics [26] and is known as *Kripke resource frame*,

[4]This simplified class of models is not adequate for full linear logic. In particular, an adequate semantic treatment of exponentials requires some algebraic constructions as in [10] or [5].

as it is a resource-sensitive version of Kripke frames. The resource sensitivity is given by viewing the set of worlds as a set of resources M that is structured by means of two operations of composition of resources: one commutative, the other non-commutative. Formulas are then evaluated wrt. resources, e.g. for $m \in M$, $m \models A$ means that resource m are relevant to make the process A occur. Moreover, resources are ordered by means of $\geq$ that represents a relevance order, i.e. $n \geq m$, n is more relevant than m. A valuation has to satisfy the *heredity* condition: if $m \in V(A)$ and $n \geq m$ then $n \in V(A)$.

A Kripke resource frame is a structure $\mathscr{M} = (M, e, \circ, \bullet, \geq)$, where $(M, e, \circ)$ is a commutative monoid[5] with neutral element e, $(M, e, \bullet)$ is a non-commutative monoid, and $\geq$ is a pre-order on M. The neutral element e represents the null resource and is used to model the independence of a process from the actual use of resources: a formula A holds in a model, written $e \models A$, if A does not depend on a specific amount of resources.

To obtain a *Kripke resource model*, a valuation on atoms $V : Atom \to \mathscr{P}(M)$ is added. The satisfaction conditions are thus defined as follows.

$m \models_{\mathscr{M}} p$ iff $m \in V(p)$.
$m \models_{\mathscr{M}} A \otimes B$ iff there exist m_1 and m_2 such that $m \geq m_1 \circ m_2$ and $m_1 \models_{\mathscr{M}} A$, $m_2 \models_{\mathscr{M}} B$.
$m \models_{\mathscr{M}} A \multimap B$ iff for all $n \in M$, if $n \models_{\mathscr{M}} A$, then $n \circ m \models_{\mathscr{M}} B$.
$m \models_{\mathscr{M}} A \odot B$ iff there exist m_1 and m_2 such that $m \geq m_1 \bullet m_2$ and $m_1 \models_{\mathscr{M}} A$, $m_2 \models_{\mathscr{M}} B$.
$m \models_{\mathscr{M}} A \,\&\, B$ iff $m \models_{\mathscr{M}} A$ and $m \models_{\mathscr{M}} B$.
$m \models_{\mathscr{M}} A \oplus B$ iff $m \models_{\mathscr{M}} A$ or $m \models_{\mathscr{M}} B$.

Entailment in linear logic can be understood as a check of whether a process of performing an action or reaching a particular event is achievable. For example, if A is a formula expressing an action, the axiom $\vdash A \multimap A$ expresses the truism that action A can be achieved by doing A. By definition of linear implication, $e \models A \multimap A$ iff for all $m \in M$, if $m \models A$ then $e \circ m = m \models A$.

The multiplicative (commutative) conjunction in formula $A \otimes B$ means that A occurs and B occurs, no more, no less. A bundle of resources makes $A \otimes B$ hold when m can be split into two bundles m_1 and m_2 that are relevant for A and B respectively. Thus, for example, we have $A \otimes A \nvdash A$, that intuitively means that if I spend two euro, I cannot infer that I have spent just one euro, the resources that make $A \otimes A$ hold may not be relevant for making A hold, as there may be too many of them. Moreover, $A \nvdash A \otimes A$, since, for instance, doing A twice is not the same as doing A once (compare getting a coffee once and getting a coffee twice).

Linear implications[6] $A \multimap B$ can be read "reacting to A, B happens" and models our initial coffee machine example $E \multimap C$ that can now be translated as "by consuming one euro, a coffee is produced". Thus, by means of the rules of linear logic (i.e. *modus ponens*) we can infer that we get a coffee from the hypothesis of spending one euro: $E, E \multimap C \vdash C$. Accordingly with our interpretation of occurrences of events, from the same assumption (E) we cannot infer that we get one coffee and one euro: $E, E \multimap C \nvdash E \otimes C$. Informally, the euro used to "produce" the coffee is "destroyed" in the deduction.

Operator $\odot$ is the non-commutative conjunction, i.e., the non-commutative version of the tensor operator, accordingly $A \odot B \nvdash B \odot A$. Non-commutativity allows for model-

[5] A monoid is an algebraic structure, namely a set equipped with a binary operation that satisfies associativity and has a neutral element.

[6] Implications are defined by means of (multiplicative) disjunction: $A \multimap B \equiv A^{\perp} \parr B$.

ing in purely logical way time sequentiality. For instance, in order to access a messages Me, one has to enter her login Lo and password Pa, in whatever order, and then she has to hit the enter button Bu, this is formalized by: $((Lo \otimes Pa) \odot Bu) \multimap Me$. Clearly, we cannot conclude, or achieve, Me by first hitting enter and then entering log in and password.

The additive conjunction $A \,\&\, B$ represent an agent's exclusive choice between A and B, so for example, $A \,\&\, B \vdash A$, however $A \,\&\, B \nvdash A \otimes B$ since only one between A and B is performed. Dually, the additive disjunction $A \oplus B$ represent the environment's choice between A and B.

In [13], an informal example to suggest the meaning of the linear logic operators is presented in terms of a menu of a restaurant.

$$Mo \multimap ((P \otimes M) \oplus (P \otimes F)) \odot (S \,\&\, G) \otimes !W$$

The formula above encodes the instruction contained in a menu. It states that, for a given amount of money (Mo), we can get: one of two alternatives, namely prosciutto (P) and melon (M), or prosciutto (P) and figs (F), next we can choose between Spaghetti (S) and Gnocchi (G), and we can get as much water as we like ($!W$). To formalize the same information in classical logic, we would have to add constraints on the fact that the money is consumed and it is no longer available, that any dish is available to the customer as a single unit, that the water is available in multiple units and so on.

3.3. Exponential operators and classical reasoning

Informally, the ontological presupposition of classical logic is that the content of propositions is available arbitrarily, whereas the ontological presupposition of linear logic is the opposite, namely, that such contents are available in single units. The standard logical perspective views linear logic as an alternative to classical logic, namely as a *non*-classical logic. From the ontological perspective, it is useful to view linear logic as a constructive analysis of classical (or intuitionistic) logical reasoning. In this section we present the main ideas of the analysis of the translation of classical logic into linear logic. This material will allow us to see how linear and classical operators can co-exist in the same logical system, and thus can be exploited together to capture rich ontological distinctions while simplifying the complexity of the axiomatization.

The exponential operators, "!" and "?", are important components of the language of full linear logic since they allow to retrieve the power of structural rules. Indeed, structural rules hold on exponentially marked formulas. Informally, the difference between "!" and "?" is that the first licenses structural rules on premises (the left hand side of the sequent) whereas the latter does so on the consequent (the right hand side). This means that on exponential formulas we can reason classically. In particular, the classical meaning of conjunction can be reconstructed in linear logic as follows.

$$!A \otimes !B \equiv !A \odot !B \equiv !(A \,\&\, B) \equiv A \wedge B$$

The same holds for disjunctions, and for the other operators of classical logic. In particular, classical implication can be defined in linear logic by $A \to B \equiv !A \multimap ?B$. This definition shows that the classical implication presupposes that arbitrarily many instances of the antecedent can be used to obtain arbitrarily many instances of the consequent. In

[25], Chapter 5, a translation of the formulas of classical logic into the language of linear logic, preserving provability and non-provability, is given.[7]

- $t(p) = p$, for p atomic.
- $t(\neg A) = !t(A) \multimap \bot$
- $t(A \wedge B) = !t(A) \& !t(B)$
- $t(A \vee B) = !t(A) \oplus !t(B)$
- $t(\forall x.A(x)) = \forall x.t(A(x))$
- $t(\exists x.A(x)) = \exists x.t(A(x))$

Henceforth, we can view linear logic as an extension of classical logic where classical formulas cohabit and coherently interact with linear resource-sensitive formulas. This means that an ontology that is written in classical first order logic, such as DOLCE, can be translated into linear logic and we can apply the reasoner for linear logic (e.g. the sequent calculus) in order to reason about DOLCE. Note that full linear logic is undecidable [19], but so is full classical first order logic. Moreover, by using linear logic the ontologist can represent situated truth as well as eternal truth. A formula A of L_{LL} that is provably equivalent to a formula of classical logic can typically be thought of as the representation of an eternal truth as it behaves classically. For atomic formulas, it is chiefly up to the modeller to decide what constitutes an eternal truth and what constitutes a "consumable" one. We shall exemplify and make use of this in the next section by focusing on a decidable fragment of linear logic. As we have seen, it is possible to design systems in which classical and linear formulas, suitably coded, coexist and freely interact, see for instance [11] and [13]. Although we do not have space for an exhaustive presentation, the discussion in this section gives an idea of how the logical system is structured: any definition or constraint given in classical logic is reformulated in the wider system based on linear logic as a specific subsystem or module. This means that the ontologist can write her definitions and constraints in classical logic when she wants or prefers to do so; these definitions and constraints are automatically added to the wider system via the mentioned translation(s) into linear logic.

4. A language for events, transformations and make plans

Let L_{NELL} be the system of *non-exponential linear logic*, i.e., the fragment of linear logic defined by means of atomic formulas, negations, and binary operators $\otimes$, $\multimap$, $\odot$, $\&$, $\oplus$, plus the first-order quantifiers. Let L_{EV} be the subset of the L_{NELL} formulas that refer to events, plans and transformation.[8] Note that here we are excluding exponentials since events are here seen as instances (particulars à la DOLCE) so it does not make sense to talk of an event that occurs twice: e.g. the very event 'The killing of Caesar by Brutus' can happen only once.

In order to separate the atomic formulas of L_{EV} from other atomic formulas in the ontology, let us assume that the atomic event formulas are given by means of a distinguished set of predicates, the "situated predicates". We decorate the situated predicates

[7] More precisely, this defines the translation from *intuitionistic* logic into linear logic. For classical logic, it suffices to apply the well known translations of intuitionistic logic into classical logic like Gödel's or double negation translation. Note that, by translating intutionistic logic into LL, our treatment can in principle include the ontology modeling based on intuitionistic logic and type theory developed by [6].

[8] For the interested reader, the proof-search complexity of the propositional L_{NELL} is PSPACE complete. Surprisingly, the first order version is decidable and it is NEXPTIME complete [19]

with a dot to mark their special meaning: e.g., $Pred^{\bullet}$. The choice of such predicates is domain specific, for example, they may include predicates for actions like giving, getting, spending and walking; and for physical or chemical properties like burning and melting. Those predicates are to be contrasted with "reusable predicates" or "abstract predicates" that are used for ontological definitions and express abstract information like "a is a physical object". Thus, L_{EV} is a *finite* subset of "situated formulas" in L_{NELL} such that

$$L_{EV} \subset \phi ::= Pred^{\bullet}(\bar{t}) | \phi^{\perp} | \phi \star \phi | \exists x.\phi | \forall x.\phi$$

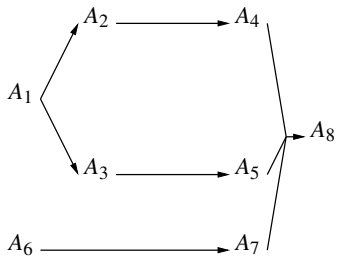
where $Pred^{\bullet}$ is a situated predicate, $\bar{t}$ is an n-tuple of terms, x is a term variable, and $\star \in \{\otimes, \multimap, \odot, \&, \oplus\}$. For example, L_{EV} may contain atomic propositions such as $G^{\bullet}(a,r)$ for "a grabs a rock r", that is situated and resource-sensitive: a cannot grab r more than once at a time. We do not characterize the language of events further [9].

The language of linear logic can specify complex events and behaviors in a simple way. For instance the following formula corresponds to a plan: $(A \otimes B \multimap C) \otimes (C \multimap D)$. The plan says that by performing A and B, one gets C, then by getting C, one gets D. The order of performing A and B is irrelevant in the formula above. If sequentiality matters, say, A must be performed before B, we can use the non-commutative operators to write the plan in this way: $(A \odot B \multimap C) \otimes (C \multimap D)$.

To show how a formula in L_{EV} can represent a transformation event, we provide a simple illustration of the use and interaction of a few linear operators that we have introduced earlier. Consider the formula:

$$\big((A_1 \multimap (A_2 \otimes A_3)) \odot ((A_2 \multimap A_4) \otimes (A_3 \multimap A_5)) \otimes (A_6 \multimap A_7)\big) \odot (A_4 \otimes A_5 \otimes A_7 \multimap A_8).$$

and its graphical representation in the following graph:



This formula stands for an event that takes two resources A_1 and A_6, and processes them in a few parallel and sequential steps to obtain the resource A_8. We can see that the 'left-upper side' of the graph, the part of the graph with A_1 as starting node and that corresponds to subformula $(A_1 \multimap (A_2 \otimes A_3)) \odot ((A_2 \multimap A_4) \otimes (A_3 \multimap A_5))$, can be performed in parallel with the 'left-lower side', corresponding to subformula $A_6 \multimap A_7$. Within the 'left-upper side', however, the parts corresponding to subformulas $A_2 \multimap A_4$ and $A_3 \multimap A_5$ (that are independent and can occur in parallel) both need to "wait" for a subevent to be completed, namely, the subevent corresponding to formula $A_1 \multimap (A_2 \otimes A_3)$. This con-

[9] Note that our treatment of events differs from Situation Calculus or Event Calculus. Events are here modeled as logical propositions (elements of L_{NELL}) and not as non-logical terms.

straint is enforced by the first occurrence of the non-commutative operator $\odot$. Analogously, the 'right side' of the graph, corresponding to subformula $A_4 \otimes A_5 \otimes A_7 \multimap A_8$, needs to "wait" for the initial subevent to be over (this is graphically represented by the whole 'left side' of the graph up to A_4, A_5 and A_7); this constraint is enforced by the second occurrence of the operator $\odot$ in the formula.

Let us assume that our ontology is written in the language of classical logic L_{CL} augmented with the dedicated set L_{EV} to separate eternal from situated truths. Given any formula ϕ in our ontology, for any sub formula ϕ' of ϕ, we can say (1) whether or not ϕ' is in classical or linear logic and (2) whether or not ϕ' is in L_{EV}. Let now LEV be a category for propositions in L_{EV} which we take to be obtained by reification of formulas on events,[10] and let us take LEV as the atomic propositions of the language L_{NELL}.

We constrain the relationship between formulas in LEV and the category of events EV in the ontology by means of a denotation relation *Denot* and the following principle:

$$\vdash \mathit{Denot}(e, \phi) \rightarrow \mathsf{EV}(e) \wedge \mathsf{LEV}(\phi)$$

meaning that the event e is denoted by the formula $\phi \in L_{EV}$.[11] The interpretation of the denotation relation may also be subject to a number of choices, for instance, one may decide that it should be a bijection, i.e. for every event there is a single formula that denotes it. In particular, one may want that logically equivalent formulas denote the same event. Again, we view these types of considerations as user specific and simply note that the approach can deal with a large number of ontological choices.

In what follows, we shall specify the category of LEV by viewing descriptions of transformations, TEV, and make plans, MP, as subsets of LEV. Note that by making them subcategories of LEV we are not committing to saying that make plans and transformations are ontologically distinct types of events. To indicate this we prefer to use the term 'transformation' instead of 'production', where only the latter is used by the authors in explaining their original definitions [4]. One could distinguish them from the ontological viewpoint, e.g., by saying that elements of MP must be associated to a given procedure, while those of TEV must not; or make the opposite assumption that make plans form the subclass of the transformation events that follow an explicit pre-existing description. These ontological characterizations are part of the underlying ontological system and we do not need to take a position. We are simply assuming, for the sake of the presentation, that they are described by specific subsets of formulas in L_{EV}. What we actually assume, instead, is that the user has introduced designated formulas to talk about them: $\vdash \mathsf{MP}(x) \rightarrow \mathsf{EV}(x)$ and $\vdash \mathsf{TEV}(x) \rightarrow \mathsf{EV}(x)$.

We now return to the definitions of technical artifact described in Section 2 to show how our language can help the ontologist to capture the core distinctions with minimal ontological commitments and a much simpler axiomatization.

We observed in Section 2 that the definition of ontological technical artifacts is already ontologically minimal: it relies on the existence of two events which are ontologically classified as selection and attribution. In [3] an explicit axiomatization in the foundational ontology DOLCE has been proposed with the detailed relationships between the

[10] For example, in an ontology like DOLCE, LEV can be seen as a subcategory of the category ABSTRACT which collects the abstract entities.

[11] From the language definition circular formulas like $\mathit{Denot}(e, \mathit{Denot}(e, \phi))$, cannot be generated.

different participants to these events. Some of these constraints can be modeled in a simpler way with our language but the discussion of the full system and its rewriting would take too much space to carry it out here. For a simple example, assume that an ontologist adopts this definition and wants to explicitly impose that the attribution act e_A, with $Denot(e_A, \phi_A)$, occurs not before the selection act e_S, with $Denot(e_S, \phi_S)$. This is usually formalized in first-order logic via reference to the temporal extensions of these acts. To do this the ontology needs to make some (perhaps minimal) choice in terms of temporal extensions and to quantify over these properties. In our language, this is not necessary as the result is obtained by the following formula $\neg(\phi_S \odot \phi_A)$, where $\neg$ is classical negation, or its rendering in linear logic: $!(\phi_S \odot \phi_A) \multimap \bot$. The formula states that it is never the case that e_S comes before e_A.[12]

5. Transformations and engineering artefacts

Regarding the definition of engineering technical artifact in Section 2, with linear implication at disposal it is easy to represent the requested process of transformation. Let $AGO(x)$ mean that x is an agentive object, $attr(x)$ that x is an attribute, $PC(x,e)$ that object x participates in event e, and $Has(x,q)$ that object x has attribute q. These notions are already referred to in the definition so we do not add further assumptions here nor require specific commitments regarding these predicates and relations. Of course, the logical formula depends on the ontology: if the underlying ontology models agency as, say, a role, we would have to use that role relation instead of AGO.

$$\begin{aligned} Crea_E(e,\mathfrak{a}) \quad \equiv \quad & \exists a,q,\phi.\ Denot(e,\phi) \wedge TEV(e) \wedge AGO(a) \wedge attr(q) \wedge \\ & PC(a,e) \wedge PC(\mathfrak{a},e) \wedge \\ & (\phi \multimap Has(\mathfrak{a},q)) \end{aligned}$$

Let us explain it in English. First notice that there is a part of the formula expressed in classical logic; this formalizes the mere ontological characteristics of the elements involved: event e is a transformation event (named ϕ in L_{EV}), x an agent and so on. In particular, the formula tells us that an event that creates an engineering artifact is a transformation event with some further constraints. The definition involves some agentive object a and some attribute q. The agent a and the artefact $\mathfrak{a}$ participate to the event e. Finally, the occurrence of the event e has the effect of causing the artefact $\mathfrak{a}$ to have the attribute q. Of course, the formula can be enriched with further constraints but it is clear that it already models a lot of information within a very simple formula structure. Furthermore, only the ontological elements (the predicates and relations like $TEV, AGO, attr, Has$, and PC) require *ad hoc* constraints. These constraints, however, should already be part of the underlying ontology to which the definition is being added via the $Crea_E(e,\mathfrak{a})$ relation.

6. Make plans and technological artefacts

Let us assume that the language of the ontology for the definition of technological artifacts, definition 3 of [4], includes a ternary relation $Check(e,\mathfrak{a},q)$. We take $Check(e,\mathfrak{a},q)$

[12] If we really aim to give a temporal connotation to the formula, we can use two further operators available in linear logic: the "non-commutative" implications called pre-implication " \ " and post-implication " / ". Because of space limitations we have not introduced them in Section 3.2.

to mean that the event e is a checking procedure (by some means, or agent) that the entity $\mathfrak{a}$ possesses the property q. Also, we take $P(x,y)$ to mean that x is part of y.

To formalize the definition of technological artifacts the ontologist can now use the following formula for the creation relation

$$\begin{aligned} Crea_T(e,\mathfrak{a}) \quad \equiv \quad & \exists e_1,e_2,a,q,\phi. \\ & Denot(e_1,\phi) \wedge MP(e_1) \wedge EV(e_2) \wedge AGO(a) \wedge attr(q) \wedge \\ & P(e_1,e) \wedge P(e_2,e) \wedge PC(a,e_1) \wedge PC(\mathfrak{a},e_1) \wedge PC(a,e_2) \wedge PC(\mathfrak{a},e_2) \wedge \\ & ((\phi \multimap Has(\mathfrak{a},q)) \odot Check(e_2,\mathfrak{a},q)) \end{aligned}$$

As before, there is a first subformula which is classical and that ontologically classifies the elements; the remaining subformula is characteristically linear. This time the creation event has at least two subevents: e_1 and e_2. The latter, e_2, is just a generic event while e_1 needs to be a make plan. Proposition ϕ in L_{EV} denotes this make plan. There is also an agentive object a and an attribute q involved. We also have that both the agent a and the artefact $\mathfrak{a}$ participate to e_1 and e_2. Finally, the occurrence of the make plan ϕ has the effect that the artefact $\mathfrak{a}$ has the attribute q, followed by the checking step, event e_2, aimed to verify whether $\mathfrak{a}$ has the attribute q. The atom $Check(e_2,\mathfrak{a},q)$ deserves some comments. Clearly, the event e_2 is different from the event of creation which we can identify with e_1. It is assumed that there is no creation in e_2, and it can be thought to be an achievement event. Practically, it is in an epistemic action necessary for the produced entity to be officially dubbed as artifact. The check must happen after the creation through the make plan: hence the non-commutative operator.[13]

7. Conclusions

Formal logic furnishes several operators that are interesting for ontological modeling although still today researchers exploit almost exclusively only the classical and modal ones. This paper makes clear that other operators can simplify the activity of the ontologist, improve the quality of formal ontologies, reduce the ontological commitment of the systems, and highlight the true ontological consequences of a formal ontology.

Notwithstanding these observations, today we lack a systematic study of these operators from the ontological viewpoint. We also have no development of methodologies for their use within the same ontological system. This paper is the first result of this new line of research. In the future, we plan to expand the list of logical operators that can have ontological interest, and to study their coexistence within the same logical system. Overall, we aim to develop methodologies for the ontological exploitation of these logical tools.

8. Acknowledgments

Borgo is supported by the Pro2Evo project funded by the "Progetto Bandiera la Fabbrica del Futuro"; Porello is supported by the VisCoSo project funded by the Autonomous Province of Trento ("Team 2011" funding programme); Troquard is supported by a Marie Curie fellowship (project "LASTS") under grant PCOFUND-GA-2008-226070.

[13] Depending on the notion of make plan one takes, one could as well reformulate the definition so to identify the creation event with the make plan, thus e_1 is e itself, and take e_2 to be the final part of the event.

References

[1] V. M. Abrusci. Non commutative logic: A survey. In Proc. *Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX 2003*, page 1. LNCS 2796, Springer, 2003.

[2] A. Ross Anderson and N. D. Jr. Belnap. *Entailment: The Logic of Relevance and Necessity*. Princeton University Press, 1975.

[3] S. Borgo and L. Vieu. Artifacts in Formal Ontology. In Anthonie Meijers, ed., *Handbook of the Philosophy of the Technological Sciences. Technology and Engineering Sciences*, vol. 9, pp. 273–307, 2009.

[4] S. Borgo, M. Franssen, P. Garbacz, Y. Kitamura, R. Mizoguchi, and P. E. Vermaas. Technical artifact: An integrated perspective. In *Formal Ontologies Meet Industry*, *FAIA* 229, pp. 3–15. IOS Press, 2011.

[5] D. Coumans, M. Gehrke, and L. van Rooijen. Relational semantics for full linear logic. *Journal of Applied Logic*, 12(1):50 – 66, 2014. Logic Categories Semantics.

[6] R. Dapoigny and P. Barlatier. Modeling ontological structures with type classes in coq. In *Conceptual Structures for STEM Research and Education*, pp. 135–152. Springer, 2013.

[7] P. De Groote. Partially commutative linear logic: sequent calculus and phase semantics. In *Proofs and Linguistics Categories–Applications of Logic to the analysis and implementation of Natural Language*, pp. 199–208, 1996.

[8] M. Dummett. *Frege: Philosophy of Language*. Harvard University Press, 1981. *2nd edition*.

[9] M. Fitting and R. L. Mendelsohn (1998). *First-Order Modal Logic*. *Synthese Library–Studies in Epistemology, Logic, Methodology, and Phil. of Sc.* vol. 277. Kluwer Academic Publishers, Dordrecht, 1998.

[10] J.-Y. Girard. Linear logic. *Theor. Comput. Sci.*, 50(1):1–101, 1987.

[11] J.-Y. Girard. On the unity of logic. *Ann. Pure Appl. Logic*, 59(3):201–217, 1993.

[12] J.-Y. Girard. Linear logic: Its syntax and semantics. In *Advances in Linear Logic*. Cambridge University Press, 1995.

[13] J.-Y. Girard. *Le point aveugle, cours de logique, tome 1 : vers l'imperfection.* Editions Hermann, collection, Visions des Sciences, 2006.

[14] H. Herre (2010). General formal ontology (GFO) : A foundational ontology for conceptual modelling. In R. Poli and L. Obrst, eds, *Theory and Applications of Ontology*, vol. 2. Springer, Berlin, 2010.

[15] W. Houkes and P. E. Vermaas (2009). Produced to Use: Combining Two Key Intuitions on the Nature of Artefacts, *Techne*, 13:123–136

[16] W. Houkes and P. E. Vermaas (2014). On What Is Made: Instruments, Products and Natural Kinds of Artefacts. In *Artefact Kinds: Ontology and the Human-Made World*. Springer, 2014.

[17] Y. Kitamura and R. Mizoguchi. Characterizing functions based on ontological models from an engineering point of view. In *Formal Ontology in Information Systems*, pages 301–314. IOS Press, 2010.

[18] P. A. Kroes and A. W. M. Meijers (2006). The dual nature of technical artifacts, *Studies in History and Philosophy of Science*, 37(1): 1–4.

[19] P. Lincoln. Deciding provability of linear logic formulas. In *Proc. Workshop on Advances in Linear Logic*. Cambridge University Press, 1995.

[20] E.D. Mares. *Relevant Logic: A Philosophical Interpretation*. Cambridge University Press, 2004.

[21] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, and L. Schneider (2002). The wonderweb library of foundational ontologies. Deliverable 17, WonderWeb European Project, 2002.

[22] W3C OWL Working Group (2012). OWL 2 Web Ontology Language. Document Overview - W3C. Recommendation 11 december 2012, 2012. http://www.w3.org/TR/2012/REC-owl2-overview-20121211/

[23] B. Smith. Against fantology. In Johann C. Marek and Maria E. Reicher, editors, *Experience and Analysis*, pages 153–170. HPT&ÖBV, 2005.

[24] B. Smith et al. (2012). Basic formal ontology 2.0 - draft specification and user's guide. http://purl.obolibrary.org/obo/bfo/2012-07-20/

[25] A. S. Troelstra. *Lectures on Linear Logic*. CSLI Publications, 1992.

[26] A. Urquhart. Semantics for relevant logics. *J. Symb. Log.*, 37(1):159–169, 1972.