A Formalization of Generalized Parameters in Situated Information

Roussanka Loukanova
Department of Mathematics, Stockholm University, Stockholm, Sweden

Keywords: Formal Language, Situation Theory, Parametric Information, Generalized Parameters, Partiality, Restrictions, Memory Variables, Recursion, Situations, Space-time Locations.

Abstract: The paper introduces a higher-order, type-theoretical formal language L_{GP}^{ST} of information content that is partial, parametric, underspecified, dependent on situations, and recursive. The terms of the formal language represent situation-theoretic objects. The language has specialized terms for constrained computations by mutual recursion. It introduces terms representing nets of parameters that are simultaneously constrained to satisfy restrictions.

1 INTRODUCTION

The formal language L_{GP}^{ST} is a generalization of the formal language introduced in (Loukanova, 2015) and serves as a language for algorithmic semantics by using semantic structures that are models of information by Situation Theory introduced in (Seligman and Moss, 2011) and (Loukanova, 2014).

The semantic type system of Situation Theory, as model-theory of information, is important in the informational structures of the theory, because, on the one hand, its types classify the distribution of objects in the situation-theoretic domains, and, on the other hand, the types participate in the informational components of the situation-theoretic objects themselves. We re-introduce that type system, formally, into the components of the formal language L_{GP}^{ST}, presented in this paper. By this, the formal types serve the interface between the formal syntax of L_{GP}^{ST} and the situational, semantic frames. We are designing the syntax of L_{GP}^{ST} to be as close as possible to the multi-dimensional geometry of abstract, semantic objects.

The types play important role of the algorithmic semantics in the syntax-semantics interfaces of L_{GP}^{ST}, where L_{GP}^{ST}-terms of specialized forms determine the algorithms for computing their denotations. We consider that such features of L_{GP}^{ST} contribute to adequateness of computational semantics of human language, targeting advanced applications to computational processing of human language. This is a central motivation for our work on extending and developing both Situation Theory, as a fine-grained model-theory of information, as well as a specialized formal language for it, e.g., L_{GP}^{ST} in this paper.

We consider that developing L_{GP}^{ST}, introduced here, facilitates important aspects of syntax-semantics interfaces of human language. Developing computational semantics of human language by a direct mappings between its syntax and situation-theoretic, i.e., semantic set-theoretic, domains, would be difficult, hard work, even for simple applications. Instead, such tasks would be facilitated, by having a suitable formal language, e.g., as the one introduced in (Loukanova, 2015), and extending it by L_{GP}^{ST}, introduced in this paper. It would be advantageous to use such a formal language that represents the fine-granularity of information content in situation-theoretical objects.

The formal language L_{GP}^{ST} has expressions for functions, properties, relations, and types, with explicitly expressed argument roles, not by linearly ordered argument slots typically used in traditional formal languages and semantic model-theories. The argument roles of functions, relations, and types are associated with appropriateness constraints. Only objects satisfying the corresponding constraints can fill up the roles.

We define terms of generalized recursion over situation-theoretic objects that can have functional and relational components with constraints. These terms have separate, but connected components, respectively, for recursive computations with assignments and propositional restrictions. The recursion component of a term represents instantiations of parametric information in recursion variables, i.e., in

Loukanova, R.
A Formalization of Generalized Parameters in Situated Information.
DOI: 10.5220/0005850303430353
In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART 2016) - Volume 1*, pages 343-353
ISBN: 978-989-758-172-4

memory cells. The cells, and other parameters that occur in the objects saved in the cells, can be restricted to satisfy constraints, which also can be linked by mutual recursion.

The recursion terms represent algorithms for computation of values that are saved in memory slots, i.e., in memory variables. The memory variables that occur in L_{GP}^{ST} terms, represent the structures in memory sections of a computational entity, which are engaged in algorithmic computations by using informational content saved in the memory slots. From the perspective of neuroscience, L_{GP}^{ST} terms represent neural nets of recursively linked memory cells for processing and saving information. Our inspiration for this is from the work (Kandel et al., 2000) and (Squire and Kandel, 2009). Information is saved in the memory cells of a neural network via mutual recursion. In addition, the memory cells are restricted to satisfy higher-order, typed constraints. The components of recursive assignments and of recursive constraints represent neural cells for memory and exchange of partial, parametric, information that depends on situations.

We introduce specialized terms that generalize the situation theoretical notion of a complex, restricted parameter to a generalized, parametric net. The parametric net consists of memory components, which are restricted to be simultaneously of a given complex type, and can involve recursive computations.

In this paper, we focus on introducing the formal syntax of L_{GP}^{ST} and its motivation. We support this work with examples. Full presentation of denotational and algorithmic semantics of the formal language L_{GP}^{ST}, interpreted in the situation-theoretic models, requires rather technical means, formal reduction calculus and inference system, which are outside the scope of this paper.

We hope that this suffices for using L_{GP}^{ST} for formal streamlining developments of various applications. In particular, L_{GP}^{ST} can contribute to applications, where semantic information is important, while it carries partiality, ambiguity, underspecification, context-dependency. Fine-grained, detailed and structured semantic information can be included in such applications by using terms of L_{GP}^{ST}. Among potential applications, we would like to point, at first place, the potentials for using L_{GP}^{ST} and its specialized variants, for computational syntax-semantics interfaces in large-scale grammars of human language, such as HPSG, LFG, GF, and grammars using Logic Programming. Other important applications include database, in relational, object-oriented, and hybrid approaches; formal representation and storing of semantic information in ontology systems; semantic information in text processing; information retrieval; etc.

By the examples, which we include in this paper, we address how space-time information can be included formally, along all components of semantic information. Specialized variables, representing semantic parameters for space-time locations, which can be linked to specific, abstract, real, or virtual situations that carry partial information.

2 BACKGROUNDS

Originally, Situation Theory was introduced by (Barwise, 1981) as a general theory of information, by mathematical structures of information. The ideas of Situation Theory ensued a broad program with wide spectrum of theoretical research and applications. The central concepts developed around representing relational and partial information, and its dependence on situations. The ideas were presented in great details by (Barwise and Perry, 1983) and (Barwise, 1989). For an informal introduction, see (Devlin, 2008). A substantial mathematical presentation of Situation Theory is given by (Seligman and Moss, 2011). For Situation Theory that we take as a primary semantic structure of the formal language in this paper, see (Loukanova, 2014). The non-well-founded Aczel set theory (Aczel, 1988), with anti-foundation axiom, is the set-theoretic foundation of Situation Theory in its full strength that supports circular information.

Our paper is on a largely open topic of formalization of Situation Theory, with computational syntax and semantic models of finely-grained information, initiated in (Loukanova, 2014; Loukanova, 2015). Higher-order, typed Situation Theory of information and type-theoretic formal languages for its versions are opening new theoretical investigations and practical applications. Computational semantics and computational neuroscience of language are among the primary applications of Situation Theory by using formal languages for it. This paper is based on our work on several new directions, in particular: (1) functional type-theory of recursion (a functional approach); (2) relational type-theory of situated, partial, and parametric information (a relational approach); (3) applications of these theories to computational syntax-semantics interfaces in natural and formal languages.

Typical syntax of formal and natural languages is formulated by rules that express syntactic dependencies between sub-expressions and syntactic categories. Such syntactic dependencies can be visualized as appropriately labeled graphs, often as parse trees, in 2-dimensional plane. In contrast, by the formal language L_{GP}^{ST}, we express fine-grained, semantic infor-

mation about semantic objects, properties of objects, relations between them, operations, processes, activities, which usually take place in 3-dimensional space, and in time. Situation Theory is abstract model-theory of information concerning real or virtual situations, represented by abstract set-theoretical objects. The formal constructs of L_{GP}^{ST} express, i.e., designate situation-theoretic objects. (In a similar manner, a specialized formal language of first-order logic is employed in axiomatic set theories.)

In this paper, we introduce a higher-order, typed formal language of Situation Theory with memory variables. Memory variables represent memory slots where underspecified objects can be saved. Such underspecified objects can be simple semantic parameters, or more complex objects with parametric components. Informally, while the memory "slots" serve saving, i.e., memorizing information, they are dynamic so that information stored in them can be updated. The memory variables can be thought of being labels, or addresses of memory slots. The values saved in the memory variables are subject to two kinds of "specializations":

- Memory variables, which correspond to semantic parameters, are constrained to satisfy situation-theoretic restrictions.
- Values can be assigned to memory variables, respectively to parameters, by recursive computations.

The L_{GP}^{ST} has terms for functions, types, relations, and recursive computations with restrictions. Typically, by traditions, formal languages use symbols and other more complex expressions for functions, properties, and relations, with linearly ordered argument slots that represent argument roles, according to conventional agreements. The formal language L_{GP}^{ST} diverges from this tradition, by explicit association of the argument roles to functions, properties, relations, and types. The argument roles are associated with appropriateness constraints. Only objects satisfying the corresponding constraints can fill up the roles.

These features of L_{GP}^{ST} are more direct representations of operations and relations between objects in nature and especially, in bio-structures. The type system of L_{GP}^{ST} models structures of basic and complex objects, and information in nature.

The terms of the formal language L_{GP}^{ST} are demonstrated with examples. We give intuitions about the denotational and algorithmic semantics of L_{GP}^{ST}. In particular, we provide examples of computational patterns in The work (Loukanova, 2015) introduces a similar formal language L_{GP}^{ST} for Situation Theory. The language L_{GP}^{ST} introduced in this paper is a proper extension of L_{GP}^{ST}. The restricted recursion L_{GP}^{ST}-terms constrain memory locations individually, by constraints of the form $(p : T)$. The language L_{GP}^{ST} covers restrictions over sets of objects, e.g., by propositional constraints $(\{p_1, \ldots, p_m\} : T)$, where T is a term for a type with m argument roles. By such a generalized constraint, the memory variables $p_1, \ldots, p_m$ represent objects that are restricted to be simultaneously of the type T. E.g., $p_1, \ldots, p_m$ can be memory nets for memorizing such objects.

Thus, the language L_{GP}^{ST} has terms for designating a new kind of semantic objects — complex units of informational parameters, that consist of sets of parameters that are simultaneously constrained to satisfy recursive conditions, instead of singleton parameters with recursive restrictions. Such terms, in effect, extend Situation Theory with complex parametric objects.

3 TYPES AND VOCABULARY OF L_{GP}^{ST}

As in Situation Theory, the class Types, which can be a proper class, i.e., a non-well-founded set, is defined at recursive stages, by starting with a set of *primitive (basic) types*. The choice of the basic types depends on technical choices and applications of L_{GP}^{ST}. Note that some of the expressions of the formal language L_{GP}^{ST}, defined later, will be its complex types.

Primitive Types. We take a set, BTypes, of *primitive (basic) types*:

$$\begin{aligned}\mathsf{BTypes} = \{ &\textsc{ind}, \textsc{rel}, \textsc{fun}, \textsc{argr}, \textsc{loc}, \\ &\textsc{pol}, \textsc{par}, \textsc{infon}, \textsc{sit}, \\ &\textsc{prop}, \textsc{set}, \textsc{type}, \models \}\end{aligned} \tag{1}$$

where:

- IND is the type for individuals and expressions denoting individuals;
- REL, for relations, primitive and complex, and expressions denoting such;
- FUN, for functions, primitive and complex, and function expressions;
- ARGR, for abstract argument roles of properties, relations, functions, and types basic and complex (Typically, in formal languages and theories, argument roles are represented by argument slots, according to some fixed, conventional linear order.);
- LOC, for space-time locations;
- POL, for two polarities denoted by 1 and 0 (which are markers for a property, relation, activity, action, or process, taking place, not truth values.);

- PAR, for basic and complex parameters, as semantic underspecified objects, and for expressions denoting semantic parameters;
- INFON, for situation-theoretic objects that are basic or complex information units, and for expressions denoting such;
- SIT, for situations;
- PROP, for abstract objects that are propositions, and for expressions denoting such;
- TYPE, for primitive and complex types and type expressions;
- SET, for sets and set expressions;
- $\models$ is a designated type called "supports", which is used to state and express that informational content holds in, or is supported by, a certain situation s (more details later).

Typed Vocabulary. The vocabulary of $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ consists of typed constants and two kinds of variables in all types, as follows.

Constants: $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ has a countable set of constants, $K_\tau = \{\mathsf{c}^\tau_0, \mathsf{c}^\tau_1, \dots\}$, for all $\tau \in \mathsf{Types}$, and K is the set of all constants:

$$K = \bigcup_{\tau \in \mathsf{Types}} K_\tau \tag{2}$$

Some of the sets of constants can be finite, and some empty (e.g., for complex types). Here we assume non-empty sets of constants: $K_{\mathrm{IND}} \neq \emptyset$ — for *primitive individuals*, $K_{\mathrm{LOC}} \neq \emptyset$ — for *space-time locations*, $K_{\mathrm{REL}} \neq \emptyset$ — for *primitive relations*, $K_{\mathrm{FUN}} \neq \emptyset$ — for *primitive functions*, $K_{\mathrm{POL}} = \{0,1\}$ – for *polarity values*[1].

Note that in Situation Theory as semantic model-theory, there is a non-empty domain Rel_n of primitive relations with n argument roles, for each natural number $n \in \mathbb{N}$. A given primitive relation r is represented by some unique set-theoretic object, e.g., an atomic element (urelement), not by a set of the ordered n-tuples of objects assumed to stand in that relation. The reason is that, in Situation Theory, n objects can stand in the relation r, only in some situation s. However, we can define the extension of the relation r in a given situation s, as the set of the ordered n-tuples of objects, which stand in the relation n, in the situation s.

Another note is in due that, in Situation theory, the domain $\mathcal{A}_{\mathrm{IND}}$ of the individuals is a collection that, depending on specific applications, can be either a set or a proper class, i.e., non-well-founded set. See (Aczel, 1988) about Aczel non-well-founded set theory. Furthermore, we shell assume that $\mathcal{A}_{\mathrm{IND}}$ includes numbers, e.g., all natural numbers, $\mathbb{N} = \{0,1,\dots\}$, integers, real numbers, etc. One may choose also that $\mathcal{A}_{\mathrm{IND}}$ is a proper class that includes the universe of all sets (which is not a set), or a specific sub-collection of sets, e.g., the hereditarily countable sets. A more fine-grained Situation Theory may have such objects like sets, in a domain of type SET, which can either a sub-type of the type IND, or a separate type. We leave these choices open here.

Pure Variables: The language $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ has a set of typed *pure variables*, $\mathsf{PV}_\tau = \{\mathsf{v}^\tau_0, \mathsf{v}^\tau_1, \dots\}$, for all types $\tau \in \mathsf{Types}$, and PV is the set of the pure variables:

$$\mathsf{PV} = \bigcup_{\tau \in \mathsf{Types}} \mathsf{PV}_\tau \tag{3}$$

Restricted Memory Variables: The formal language $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ has a distinctive set MV_τ of typed variables $\mathsf{MV}_\tau = \{\mathsf{p}^\tau_0, \mathsf{p}^\tau_1, \dots\}$, for all types $\tau \in \mathsf{Types}$. The set of all restricted memory variables is:

$$\mathsf{MV} = \bigcup_{\tau \in \mathsf{Types}} \mathsf{MV}_\tau, \tag{4}$$

The variables of MV are also simply called *memory variables* or *recursion variables*. Thus, the $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ variables are typed, and the sets of all variables are classified as follows:

$$\mathsf{Vars}_\tau = \mathsf{PV}_\tau \cup \mathsf{MV}_\tau, \tag{5a}$$

$$\mathsf{Vars} = \mathsf{PV} \cup \mathsf{MV} \tag{5b}$$

For each of the basic types, we take a set of basic, restricted memory variables, which can be used for assigning, i.e., "saving", information and objects in them, by respecting the corresponding types. The typing of the memory variables, serves among other computational tasks, also reserving specific shape or construction structure of the memory slots, as undetermined blue-prints. E.g., $\mathrm{PAR}_{\mathrm{IND}}$, $\mathrm{PAR}_{\mathrm{LOC}}$, $\mathrm{PAR}_{\mathrm{REL}}$, $\mathrm{PAR}_{\mathrm{FUN}}$, $\mathrm{PAR}_{\mathrm{INFON}}$, $\mathrm{PAR}_{\mathrm{SIT}}$, $\mathrm{PAR}_{\mathrm{PROP}}$, etc.

We also interprete the memory variables as memory slots carrying information about partly known objects, where the information available in a memory slot is subject to update. Alternatively, memory variables are for objects under developmental changes.

For semantic reasons, we distinguish between restricted variables, which are syntactic objects, and semantic parameters as semantic objects, while such details are not in the subject of this paper. Sometimes, when the context makes it clear, the restricted, memory variables can be called parameters. Typically, unless otherwise specified either explicitly or by the syntax of the expressions, we shall use letters x, y, z, with or without subscripts, to vary over pure variables of any types, and letters p, q, r, with or without subscripts, to vary over memory variables of any type.

[1] To distinguish the polarities from propositional truth values, one may take two other distinctive symbols, e.g., $K_{\mathrm{POL}} = \{-,+\}$.

Definition 1 (Argument roles)**. Argument roles.**

1. We take a set of expressions (basic or complex, by relatively simple rules of construction), called *basic argument roles*, which can be associated with some of the expressions, such as the constants for basic relations, functions, and types:

$$\text{BRoles} = \{\rho_1, \ldots, \rho_m, \ldots\}. \tag{6}$$

2. ARoles is the class of *basic and complex argument roles*, which includes the set of the basic argument roles, BRoles $\subset$ ARoles, and other argument roles, denoted by $[\xi]$, for $\xi \in \mathsf{PV}$, that are assigned to complex relations, functions, and types via rules of term formations (given later)

$$\text{ARoles} = \text{BRoles} \cup \{\, [\xi] \mid \xi \in \mathsf{PV} \,\} \tag{7}$$

Note that the argument roles $[\xi]$ are argument roles generated by using only pure variables from the collection PV, which are typed. The argument roles $[\xi]$ of relations, functions, and types, and terms denoting such, get generated via the term formation rules given later. Note that the elements of the set ARoles, esp. for the primitives, are specialized expressions, which can vary depending on specific applications of $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$.

Example 3.1. As usually in relational approaches, we take relation constants such as smile, read, read-to, and give. We can associate sets of argument roles with them, in a simple way, without including constraints over the expressions filling the argument roles, and by using conventional linking and understanding between the actual roles representing "who-does-what-to-whom", which is used in many applications involving functions and relations:

$$\text{ARGR}(\mathsf{read}) = \{\, \mathsf{arg}_1, \mathsf{arg}_2 \,\} \tag{8a}$$

$$\text{ARGR}(\mathsf{read\text{-}to}) = \{\, \mathsf{arg}_1, \mathsf{arg}_2, \mathsf{arg}_3 \,\} \tag{8b}$$

$$\text{ARGR}(\mathsf{give}) = \{\, \mathsf{arg}_1, \mathsf{arg}_2, \mathsf{arg}_3 \,\} \tag{8c}$$

Rules for syntax-semantics interface can provide the necessary links between $\{\, \mathsf{arg}_1, \mathsf{arg}_2, \mathsf{arg}_3 \,\}$, and the actual roles "who-does-what-to-whom".

Typically, the basic relations and types are associated with argument roles that have to satisfy constraints for their appropriate filling. We stress that there is no assumption that the argument roles of the relations, functions, and types are ordered.

Argument Roles with Appropriateness Constraints.

1. Every relation and type constant and pure or memory variable γ, is associated with a set $\text{ARGR}(\gamma)$ of typed expressions arg_i, $i = 1, \ldots, n$, for argument roles:

$$\text{ARGR}(\gamma) = \{\, T_1 : \mathsf{arg}_1, \ldots, T_n : \mathsf{arg}_n \,\} \quad \text{for all } \gamma \in K_{\text{REL}} \cup K_{\text{TYPE}} \cup \mathsf{Vars}_{\text{REL}} \cup \mathsf{Vars}_{\text{TYPE}} \tag{9}$$

where $n \geq 0$, $\mathsf{arg}_i \in$ ARoles, and T_i are sets of basic types. The expressions $\mathsf{arg}_1, \ldots, \mathsf{arg}_n$ are called the *argument roles* (or the *argument slots*) of γ. Each T_i is specific for the corresponding argument role arg_i of γ, $i = 1, \ldots, n$. T_i is called the *appropriateness constraint of* arg_i, $i = 1, \ldots, n$.

2. Every function constant and function variable γ, i.e., $\gamma \in \mathcal{A}_{\text{FUN}} \cup \mathsf{Vars}_{\text{FUN}}$, is associated with two sets of sub-typed expressions:

$$\text{ARGR}(\gamma) = \{\, T_1 : \mathsf{arg}_1, \ldots, T_n : \mathsf{arg}_n \,\} \tag{10a}$$

$$\mathit{Value}(\gamma) = \{\, T_{n+1} : \mathsf{arg}_{n+1} \,\} \tag{10b}$$

where $n \geq 0$., $\mathsf{arg}_i \in$ ARoles, and T_i are sets of basic types. The expressions $\mathsf{arg}_1, \ldots, \mathsf{arg}_n$ are called the *argument roles* (or the *argument slots*) of γ. The expression arg_{n+1} is called the *value role* of γ. Each T_i is specific for the corresponding argument role arg_i of γ, $i = 1, \ldots, n+1$. T_i is called the *appropriateness constraint of* arg_i, $i = 1, \ldots, n+1$.

More sensible approaches, in compare to Example (3.1), use more explicit, informative naming of the specific argument roles for the primitive relations. E.g., semantic roles of the verbs denoting actions, processes, etc., have been subject of syntactic representations in Transformational Grammar (TG) and Government and Binding Theory (GB, GBT), by the so called Theta Theory and Θ-roles in (Chomsky, 1993) and (Dowty, 1979). For a more recent work on the topic, see (Jaworski and Przepórkowski, 2014). Situation-theoretic approaches to semantics of human language, e.g., in HPSG grammars, have been using a traditional convention, by which the semantic argument roles of a relation denoted by a verb are named by using English word-formation rules, e.g., adding the suffixes "er", "ed", etc. (sometimes by misspelling), respectively for the agent and patient. See Example (11a)–(11d).

Example 3.2.

$$\text{ARGR}(\text{IND} : \mathsf{smile}) = \{\, \text{IND} : \mathsf{smiler} \,\} \tag{11a}$$

$$\text{ARGR}(\mathsf{read}) = \{\, \text{IND} : \mathsf{reader}, \ \text{IND} : \mathsf{readed} \,\}, \tag{11b}$$

$$\text{ARGR}(\mathsf{read\text{-}to}) = \{\, \text{IND} : \mathsf{reader}, \ \text{IND} : \mathsf{readed}, \ \text{IND} : \mathsf{listener} \,\}$$

$$\text{ARGR}(\mathsf{give}) = \{\, \text{IND} : \mathsf{giver}, \ \text{IND} : \mathsf{recipient}, \ \text{IND} : \mathsf{what} \,\} \tag{11c}$$

$$\text{ARGR}(\gamma) = \{\, T_1 : \mathsf{arg}_1, \ldots, T_n : \mathsf{arg}_n \,\} \quad (\text{for } \gamma \in K_{\text{REL}}, n \in \mathbb{N}) \tag{11d}$$

Note 1. Note that complex types can be associated with relation, type, and function terms, including constants and variables, via more complex terms, i.e., via restriction terms, introduced later.

Theorem 1. *(1)* BR*oles can be countable, given that the sets of basic relations, functions, and types are countable, and each have a countable set of argument roles.*

(2) BR*oles can be (equinumerously) generated from a finite base of symbols, by assuming Situation Theory with basic relations, functions, and types that have finite number of argument roles, and that the sets of the basic relations, functions and types are countable.*

(3) BR*oles can be restricted to a finite set given that the sets of primitive relations, functions and types are finite.*

Proof. By using Cantor Theorem for countable sets. □

4 TERMS OF $\mathrm{L}^{\mathbf{ST}}_{\mathbf{GP}}$

The classes of Terms_τ, for τ : TYPE, are defined recursively, at stages with respect to the recursive levels of type constructions.

Constants (as terms)**.** If $\mathsf{c} \in K_\tau$, then $\mathsf{c} \in \mathsf{Terms}_\tau$, denoted as $\mathsf{c} : \tau$. There are no free and no bound variables in c. **Variables** (as terms). If $x \in \mathsf{Vars}_\tau$, then, $x \in \mathsf{Terms}_\tau$. denoted $x : \tau$. The only occurrence of the variable x in the term x is free; there are no bound occurrences of variables in x.

Infon Terms. For every relation term (basic or complex) $\rho \in \mathsf{Terms}_{\mathrm{REL}}$, associated with argument roles

$$\mathrm{ARGR}(\rho) = \{ T_1 : \mathsf{arg}_1, \ldots, T_n : \mathsf{arg}_n \} \tag{12}$$

and every sequence of terms $\xi_1, \ldots, \xi_n$ such that

$$\xi_1 \in \mathsf{Terms}_{T_1}, \ldots, \xi_n \in \mathsf{Terms}_{T_n} \tag{13}$$

(i.e., terms $\xi_1, \ldots, \xi_n$ that satisfy the corresponding appropriateness constraints of the argument roles of ρ, denoted also as $T_1 : \xi_1, \ldots, T_n : \xi_n$), and every space-time term $\tau \in \mathsf{Terms}_{\mathrm{LOC}}$, i.e., LOC : τ, every polarity term $t \in \mathsf{Terms}_{POL}$, i.e., $POL : t$, i.e., $t \in \{0,1\} \cup \mathrm{PAR}_{POL}$, the expression (14a) is an *infon term*, i.e., an element of $\mathsf{Terms}_{\mathrm{INFON}}$, alternatively denoted with type assignment in (14b).

$$\begin{aligned} \ll \rho, & T_1 : \mathsf{arg}_1 : \xi_1, \ldots, \\ & T_n : \mathsf{arg}_n : \xi_n, \\ & \mathrm{LOC} : Loc : \tau, \\ & \mathrm{POL} : Pol : t \gg \in \mathsf{Terms}_{\mathrm{INFON}} \end{aligned} \tag{14a}$$

$$\begin{aligned} \ll \rho, & T_1 : \mathsf{arg}_1 : \xi_1, \ldots, T_n : \mathsf{arg}_n : \xi_n, \\ & \mathrm{LOC} : Loc : \tau, \mathrm{POL} : Pol : t \gg : \mathrm{INFON} \end{aligned} \tag{14b}$$

All free (bound) occurrences of variables in ρ, $\xi_1, \ldots, \xi_n$, τ are also free (bound) in the infon term (14a).

The notation (14b) is used when the type labeling is relevant. We use the notation $\ll \rho, \mathsf{arg}_1 : \xi_1, \ldots, \mathsf{arg}_n : \xi_n, Loc : \tau; t \gg$ when the type constraints are understood; or $\ll \rho, \xi_1, \ldots, \xi_n, \tau; t \gg$ when also there is an understood order of the arguments.

Proposition Terms. For every type term (basic or complex) $\gamma \in \mathsf{Terms}_{\mathrm{TYPE}}$, associated with argument roles $\mathrm{ARGR}(\gamma) = \{ T_1 : \mathsf{arg}_1, \ldots, T_n : \mathsf{arg}_n \}$, and $\xi_1 \in \mathsf{Terms}_{T_1}, \ldots, \xi_n \in \mathsf{Terms}_{T_n}$, i.e., terms $\xi_1, \ldots, \xi_n$ that satisfy the corresponding appropriateness constraints of the argument roles of γ, denoted $T_1 : \xi_1, \ldots, T_n : \xi_n$, the expression in (15a) is a *proposition term*, in a *postfix notation*, i.e., an element of $\mathsf{Terms}_{\mathrm{PROP}}$, alternatively denoted in (15b) with the type assignment that it is a term of type PROP.

$$\begin{aligned} (\{ & T_1 : \mathsf{arg}_1 : \xi_1, \ldots, \\ & T_n : \mathsf{arg}_n : \xi_n \} \ : \ \gamma) \ \in \ \mathsf{Terms}_{\mathrm{PROP}} \end{aligned} \tag{15a}$$

$$\begin{aligned} (\{ & T_1 : \mathsf{arg}_1 : \xi_1, \ldots, \\ & T_n : \mathsf{arg}_n : \xi_n \} \ : \ \gamma) \ : \ \mathrm{PROP} \end{aligned} \tag{15b}$$

All free (bound) occurrences of variables in γ, ξ_1, $\ldots$, ξ_n, are also free (bound) in the proposition term in (15a) and (15b).

Often, we shall skip the braces around the argument role assignments in (15a) and (15b). As customary in formal languages, depending on circumstances, postfix notation of the proposition terms (as above in (15a) and (15b)) can be alternated with prefix, or infix notations.

Notation 1. In some cases, the full prefix notations (16a)–(16b) are more convenient.

$$(\gamma, \{ T_1 : \mathsf{arg}_1 : \xi_1, \ldots, T_n : \mathsf{arg}_n : \xi_n \}) : \mathrm{PROP} \tag{16a}$$

$$(\gamma, T_1 : \mathsf{arg}_1 : \xi_1, \ldots, T_n : \mathsf{arg}_n : \xi_n) : \mathrm{PROP} \tag{16b}$$

Complex Proposition Terms. Terms for proposition are formed by using the usual logic connectives, $\neg$, $\wedge$, $\vee$, etc.

Notation 2. Negated propositions as in (17a) are sometimes denoted by (17b).

$$\neg(\{ T_1 : \mathsf{arg}_1 : \xi_1, \ldots, T_n : \mathsf{arg}_n : \xi_n \}\gamma) \tag{17a}$$

$$(\{ T_1 : \mathsf{arg}_1 : \xi_1, \ldots, T_n : \mathsf{arg}_n : \xi_n \} :\!\!\!/ \gamma) \tag{17b}$$

Notation 3. The prefex notation (18a) is used when the type constraints over the argument roles are understood. The notation (18b) is used when also there is an understood order of the argument roles.

$$(\gamma, \mathsf{arg}_1 : \xi_1, \ldots, \mathsf{arg}_n : \xi_n) \tag{18a}$$
$$(\gamma, \xi_1, \ldots, \xi_n) \tag{18b}$$

Notation 4. A postfix notation (19a) can be used when the type constraints over the argument roles are understood; and (19b) when the proposition is negated. The notations (19c) and (19d) are used when also there is an understood order of the arguments, since they conform with a traditional notation of type association to terms in the case of $n = 1$. A precaution is due with the later notation since it is an abbreviated notation of a proposition term of the formal language $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ not a type assignment to a $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ term.

$$(\{\mathsf{arg}_1 : \xi_1, \ldots, \mathsf{arg}_n : \xi_n\} : \gamma) \tag{19a}$$
$$(\{\mathsf{arg}_1 : \xi_1, \ldots, \mathsf{arg}_n : \xi_n\} \not: \gamma) \tag{19b}$$
$$(\xi_1, \ldots, \xi_n : \gamma) \tag{19c}$$
$$(\xi_1, \ldots, \xi_n \not: \gamma) \tag{19d}$$

A precaution is due for proposition terms with $n > 1$. (19c), and respectively, (19d), does not mean that each one of the objects ξ_i is (is not) of type γ, since the type γ has n arguments. The proposition terms (19c) and (19d), i.e., $(\xi_1, \ldots, \xi_n : \gamma)$ and $(\xi_1, \ldots, \xi_n \not: \gamma)$ denote statements that the objects $\{\xi_1, \ldots, \xi_n\}$, together as a set (or a group), respectively are not, of type γ, by filling the corresponding argument roles. I.e., in these notations, each ξ_i fills the corresponding argument role arg_i, and linearity is only visual notation.

Application Terms. For every function term (basic or complex) $\gamma \in \mathsf{Terms}_{\mathrm{FUN}}$, associated with argument roles $\textsc{Argr}(\gamma) = \{T_1 : \mathsf{arg}_1, \ldots, T_n : \mathsf{arg}_n\}$ and a value role $\mathit{Value}(\gamma) = \{T_{n+1} : \mathsf{val}\}$, and every $\xi_1 \in \mathsf{Terms}_{T_1}$, ..., ξ_n, $\xi_{n+1} \in \mathsf{Terms}_{T_{n+1}}$, the expression in (20a), respectively (20b), is an *application term*:

$$\gamma\{T_1 : \mathsf{arg}_1 : \xi_1, \ldots, T_n : \mathsf{arg}_n : \xi_n\} \in \mathsf{Terms}_{T_{n+1}} \tag{20a}$$
$$\gamma\{T_1 : \mathsf{arg}_1 : \xi_1, \ldots, T_n : \mathsf{arg}_n : \xi_n\} : T_{n+1} \tag{20b}$$

The notation (20b), which includes the type association, is used when the type labeling is relevant. In addition, the term (21a), respectively in (21b), is a proposition term.

$$\big(\gamma\{T_1 : \mathsf{arg}_1 : \xi_1, \ldots, T_n : \mathsf{arg}_n : \xi_n\} = \xi_{n+1}\big) \in \mathsf{Terms}_{\mathrm{PROP}} \tag{21a}$$
$$\big(\gamma\{T_1 : \mathsf{arg}_1 : \xi_1, \ldots, T_n : \mathsf{arg}_n : \xi_n\} = \xi_{n+1}\big) : \mathrm{PROP} \tag{21b}$$

All free (bound) occurrences of variables in γ, ξ_1, ..., ξ_{n+1} are also free (bound) in the application terms.

Note that the application expression in (20a) does not by default designate the value, i.e., the result of a process of valuation of the function application. Furthermore, we allow partial functions, i.e., the value of a function application term $\gamma\{T_1 : \mathsf{arg}_1 : \xi_1, \ldots, T_n : \mathsf{arg}_n : \xi_n\}$ may not exist. In such a case, the denotation of the corresponding proposition term (21a) might be undefined, or a special object that designates error. When there is an understood order of the function arguments, we may use the traditional terms of functional application: $\gamma(\xi_1, \ldots, \xi_n) : T_{n+1}$.

λ-Terms. For every term (basic or complex) $\Phi \in \mathsf{Terms}$ and any pure variables $\xi_1, \ldots, \xi_n \in \mathsf{PV}$, the expression $\lambda\{\xi_1, \ldots, \xi_n\}\Phi$ is a λ-*abstraction term*, i.e.:

$$\lambda\{\xi_1, \ldots, \xi_n\}\Phi \in \mathsf{Terms} \tag{22a}$$

$[\xi_1], \ldots, [\xi_n]$ are expressions for the argument roles of $\lambda\{\xi_1, \ldots, \xi_n\}\Phi$, and are associated with corresponding *appropriateness constraints* as follows:

$$\textsc{Argr}(\lambda\{\xi_1, \ldots, \xi_n\}\Phi) = \{T_1 : [\xi_1], \ldots, T_n : [\xi_n]\} \tag{23}$$

where, for each $i \in \{1, \ldots, n\}$, T_i is the union of all sets (of types) that are the appropriateness constraints of all the argument roles that occur in Φ, and such that ξ_i fills up them, without being bound. (Note that ξ_i may fill more than one argument role in Φ.)

All free occurrences of $\xi_1, \ldots, \xi_n$ in Φ are bound in the term $\lambda\{\xi_1, \ldots, \xi_n\}\Phi$. All other free (bound) occurrences of variables in Φ are free (bound) in the term $\lambda\{\xi_1, \ldots, \xi_n\}\Phi$.

Case 1: Complex Relations with Complex Arguments. In the case when $\Phi \in \mathsf{Terms}_{\mathrm{INFON}}$ the expression $\lambda\{\xi_1, \ldots, \xi_n\}\Phi$ is a *complex-relation term*, i.e.:

$$\lambda\{\xi_1, \ldots, \xi_n\}\Phi \in \mathsf{Terms}_{\mathrm{REL}} \tag{24a}$$
$$\lambda\{\xi_1, \ldots, \xi_n\}\Phi : \mathrm{REL} \tag{24b}$$

Case 2: Complex Types with Complex Arguments. In the case when $\Phi \in \mathsf{Terms}_{\mathrm{PROP}}$, the expression $\lambda\{\xi_1, \ldots, \xi_n\}\Phi$ is a *complex-type term*, i.e.:

$$\lambda\{\xi_1, \ldots, \xi_n\}\Phi \in \mathsf{Terms}_{\mathrm{TYPE}} \tag{25a}$$
$$\lambda\{\xi_1, \ldots, \xi_n\}\Phi : \mathrm{TYPE} \tag{25b}$$

The expressions for types, as result of *abstraction formation* over pure variables in proposition terms, are significantly distinct from other λ-terms, and we use the following expressions for abstraction types:

$$[T_1 : \xi_1, \ldots, T_n : \xi_n \mid \Phi] \in \mathsf{Terms}_{\mathrm{TYPE}} \tag{26a}$$
$$[T_1 : \xi_1, \ldots, T_n : \xi_n \mid \Phi] : \mathrm{TYPE} \tag{26b}$$

When the type assignment is understood, we use the notation (27).

Notation 5.

$$[\xi_1,\ldots,\xi_n \mid \Phi] \tag{27}$$

Case 3: Complex Function Terms (Also Operation Terms) with Complex Arguments. For every $\varphi \in \mathsf{Terms}_\tau$ where $\tau \in \mathsf{Types}$, $\tau \not\equiv$ INFON, $\tau \not\equiv$ PROP, and for any pure variables $\xi_1,\ldots,\xi_n \in \mathsf{PV}$, the expression $\lambda\{\xi_1,\ldots,\xi_n\}\varphi$ is a *complex-function term*:

$$\lambda\{\xi_1,\ldots,\xi_n\}\varphi \in \mathsf{Terms}_{\text{FUN}} \tag{28a}$$

$$\lambda\{\xi_1,\ldots,\xi_n\}\varphi : \text{FUN} \tag{28b}$$

Every function term $\lambda\{\xi_1,\ldots,\xi_n\}\varphi$ has a role for the function value, $\mathit{Value}(\lambda\{\xi_1,\ldots,\xi_n\}\varphi) = \{\tau : \mathsf{val}\}$, and argument roles $[\xi_1],\ldots,[\xi_n]$, which are associated with corresponding *appropriateness constraints* as follows:

$$\begin{aligned}&\text{ARGR}(\lambda\{\xi_1,\ldots,\xi_n\}\varphi) = \\ &\qquad \{T_1 : [\xi_1],\ldots,T_n : [\xi_n]\}\end{aligned} \tag{29a}$$

$$\mathit{Value}(\lambda\{\xi_1,\ldots,\xi_n\}\varphi) = \{\tau : \mathsf{val}\} \tag{29b}$$

where, for each $i \in \{1,\ldots,n\}$, T_i is the union of all sets (of types) that are the appropriateness constraints of all the argument roles that occur in φ, and such that ξ_i fills up them, without being bound. (Note that any of the pure variables ξ_i may fill more than one argument role in φ.)

Constrained Recursion Terms. For any type terms C_k : TYPE, $k = 1,\ldots,m$ ($m \geq 0$), with argument roles $\text{ARGR}(\mathsf{C}_k)$:

$$\begin{aligned}&\text{ARGR}(\mathsf{C}_k) = \{T_{k,1} : \mathit{arg}_{k,l_1},\ldots,T_{k,l_k} : \mathit{arg}_{k,l_k}\} \\ &(l_k \geq 1)\end{aligned} \tag{30}$$

memory (recursion) variables $\mathsf{q}_{k,j} \in \mathsf{MV}_{T_{k,j}}$ (i.e., $\mathsf{q}_{k,j} : T_{k,j}$), $j = 1,\ldots,l_k$, terms $\mathsf{A}_i : \sigma_i$, $i = 0,\ldots,n$ ($n \geq 0$), and pairwise different memory variables $\mathsf{p}_i \in \mathsf{MV}_{\sigma_i}$ (i.e., $\mathsf{p}_i : \sigma_i$), $i = 1,\ldots,n$, such that the two sequences (31a) and (31b)

$$\begin{aligned}&\{(\mathsf{q}_{1,1},\ldots,\mathsf{q}_{1,l_1} : \mathsf{C}_1),\ldots, \\ &\ (\mathsf{q}_{m,1},\ldots,\mathsf{q}_{m,l_m} : \mathsf{C}_m)\}\end{aligned} \tag{31a}$$

$$\{\mathsf{p}_1 := \mathsf{A}_1,\ldots,\mathsf{p}_n := \mathsf{A}_n\} \tag{31b}$$

(jointly) satisfy the Acyclicity Constraint 1, the expression in (32a), respectively (32b), is a *restricted recursion term of type* σ_0:

$$\begin{aligned}&\mathsf{A}_0 \text{ such that } \{(\mathsf{q}_{1,1},\ldots,\mathsf{q}_{1,l_1} : \mathsf{C}_1),\ldots, \\ &\qquad (\mathsf{q}_{m,1},\ldots,\mathsf{q}_{m,l_m} : \mathsf{C}_m)\} \\ &\quad \text{where } \{\mathsf{p}_1 := \mathsf{A}_1,\ldots, \\ &\qquad \mathsf{p}_n := \mathsf{A}_n\} \in \mathsf{Terms}_{\sigma_0}\end{aligned} \tag{32a}$$

$$\begin{aligned}&[\mathsf{A}_0 \text{ such that } \{(\mathsf{q}_{1,1},\ldots,\mathsf{q}_{1,l_1} : \mathsf{C}_1),\ldots, \\ &\qquad (\mathsf{q}_{m,1},\ldots,\mathsf{q}_{m,l_m} : \mathsf{C}_m)\} \\ &\quad \text{where } \{\mathsf{p}_1 := \mathsf{A}_1,\ldots,\mathsf{p}_n := \mathsf{A}_n\}] : \sigma_0\end{aligned} \tag{32b}$$

Acyclicity Constraint 1. The sequences of type constraints (31a) and assignments (31b) are (jointly) *acyclic* iff there is a ranking function

$$\mathsf{rank} : \{\bigcup_{k=1}^{m}\{\mathsf{q}_{k,j}\}_{j=1}^{l_k} \bigcup \{\mathsf{p}_i\}_{i=1}^{n}\} \longrightarrow \mathbb{N} \tag{33}$$

such that:

1. for all $\mathsf{q}_{k,j}, \mathsf{q}_{k',j'} \in \bigcup_{k=1}^{m}\{\mathsf{q}_{k,j}\}_{j=1}^{l_k}$, if $\mathsf{q}_{k',j'}$ occurs freely in C_k, then $\mathsf{rank}(\mathsf{q}_{k',j'}) < \mathsf{rank}(\mathsf{q}_{k,j})$
2. for all $\mathsf{p}_i, \mathsf{p}_j \in \{\mathsf{p}_i\}_{i=1}^{n}$, if p_j occurs freely in A_i, then $\mathsf{rank}(\mathsf{p}_j) < \mathsf{rank}(\mathsf{p}_i)$.

Note that the acyclicity constraint is a proper part of the recursive definition of the $\text{L}_{\text{GP}}^{\text{ST}}$-terms.

All free occurrences of $\mathsf{p}_1,\ldots,\mathsf{p}_n$ in A_0, …, A_n, C_1, …, C_m are bound in the term (32b). All other free (bound) occurrences of variables in A_0, …, A_n, C_1, …, C_m are free (bound) in (32b). Sometimes we enclose the recursion terms in extra brackets to separate them from the surrounding text, for example as in (32b).

Generalized, Restricted Variables. For any given

1. type term C : TYPE, such that

$$\begin{aligned}&\text{ARGR}(\mathsf{C}) = \{T_1 : \mathit{arg}_1,\ldots,T_k : \mathit{arg}_k\} \\ &(k \geq 1)\end{aligned} \tag{34}$$

2. memory (recursion) variables $\mathsf{q}_j \in \mathsf{MV}_{T_j}$ (that is, $\mathsf{q}_j : T_j$), for $j = 1,\ldots,k$ ($k > 1$), which are not necessarily pairwise different and such that
   (a) the proposition term

$$(\{T_1 : \mathsf{arg}_1 : q_1,\ldots,T_k : \mathsf{arg}_k : q_k\} \ : \ C) \tag{35}$$

   abbreviated as $(\mathsf{q}_1,\ldots,\mathsf{q}_k : \mathsf{C})$, is acyclic, i.e., no q_j ($j = 1,\ldots,k$) occurs freely in C, and
   (b) $\{\mathsf{q}_{k_1},\ldots,\mathsf{q}_{k_l}\}$ is the set of the pairwise different variables, such that: $\{\mathsf{q}_{k_1},\ldots,\mathsf{q}_{k_l}\} = \{\mathsf{q}_1,\ldots,\mathsf{q}_k\}$

the expression in (36a), abbreviated as (36b) and (36c), is a term of type PAR_{SET}. We call it a *memory network* (or a *restricted memory net*).

$$\begin{aligned}&\{\mathsf{q}_{k_1},\ldots,\mathsf{q}_{k_l}\} \text{ such that } \{(\mathsf{q}_1,\ldots,\mathsf{q}_k : \mathsf{C}), \\ &\qquad (\mathsf{q}_{1,1},\ldots,\mathsf{q}_{1,l_1} : \mathsf{C}_1),\ldots, \\ &\qquad (\mathsf{q}_{m,1},\ldots,\mathsf{q}_{m,l_m} : \mathsf{C}_m)\} \\ &\quad \text{where } \{\mathsf{p}_1 := \mathsf{A}_1,\ldots, \\ &\qquad \mathsf{p}_n := \mathsf{A}_n\}\end{aligned} \tag{36a}$$

$$\begin{aligned}&\{\mathsf{q}_{k_1},\ldots,\mathsf{q}_{k_l}\} \text{ such that } \{(\mathsf{q}_1,\ldots,\mathsf{q}_k : \mathsf{C}), \\ &\qquad (\overrightarrow{\mathsf{q}} : \overrightarrow{\mathsf{C}})\} \\ &\quad \text{where } \{\overrightarrow{\mathsf{p}} := \overrightarrow{\mathsf{A}}\}\end{aligned} \tag{36b}$$

$$\{\mathsf{q}_{k_1},\ldots,\mathsf{q}_{k_l}\}\ \mathsf{s.th.}\ \{(\mathsf{q}_1,\ldots,\mathsf{q}_k:\mathsf{C}),\ (\overrightarrow{\mathsf{q}}:\overrightarrow{\mathsf{C}})\}\ \{\overrightarrow{\mathsf{p}}:=\overrightarrow{\mathsf{A}}\} \tag{36c}$$

We call the type term C the *major constraint* of the generalized memory network in (36a). The term (36a) denotes a complex situation-theoretic object, which is a generalized *constrained parameter*.

For abbreviation, sometimes, the two constants operators where and such that can be collapsed by using just one of them. In the special case when $k=1$, the type C has just one argument role, $\textsc{Arg}\mathrm{R}(\mathsf{C}) = \{T : arg\}$. In such a case, we can identify the singleton set $\{\mathsf{q}\}$ with the parameter $q:T$, and the terms in (37a) and (37b):

$$\{\mathsf{q}\}\ \mathsf{s.th.}\ \{(q:C),(\overrightarrow{\mathsf{q}}:\overrightarrow{\mathsf{C}})\}\{\overrightarrow{\mathsf{p}}:=\overrightarrow{\mathsf{A}}\} \tag{37a}$$

$$\equiv \mathsf{q}\ \mathsf{s.th.}\ \{(q:C),(\overrightarrow{\mathsf{q}}:\overrightarrow{\mathsf{C}})\}\{\overrightarrow{\mathsf{p}}:=\overrightarrow{\mathsf{A}}\} \tag{37b}$$

In the special case when $n,m=1$, we have the generalized memory variable (38a), which can be abbreviated by the expression (38b).

$$\{\mathsf{q}_{k_1},\ldots,\mathsf{q}_{k_l}\}\ \mathsf{s.th.}\ \{(\mathsf{q}_1,\ldots,\mathsf{q}_k:\mathsf{C})\} \tag{38a}$$

$$\{\mathsf{q}_{k_1},\ldots,\mathsf{q}_{k_l}\}^{(\mathsf{q}_1,\ldots,\mathsf{q}_k:\mathsf{C})}:\mathrm{PAR}_{\mathrm{SET}} \tag{38b}$$

The term $\{\mathsf{q}_{k_1},\ldots,\mathsf{q}_{k_l}\}^{(\mathsf{q}_1,\ldots,\mathsf{q}_k:\mathsf{C})}:\mathrm{PAR}$ provides formalization and generalization of the more simple, situation theoretic restricted parameters q^T that are singletons, i.e., $l=1$. For restricted parameters, as singletons, see, e.g., (Barwise and Perry, 1983) and (Loukanova, 2014). A generalized parameter designated by a $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ term of the form (36a) is a complex, parametric object that corresponds to a finite set of parameters. The term (36a) represents a partly known complex of entities, or a complex of entities that are under-development, and which are simultaneously restricted and "bundled" together by the constraint $(\mathsf{q}_1,\ldots,\mathsf{q}_k:\mathsf{C})$.

5 RELATIONS BETWEEN SITUATIONS WITH UNDERSPECIFICATION

As a typical example of a complex term that includes most of the formal concepts introduced in the first part of the paper, we use the semantic representation of the verb "stop" in two of its typical usages. We have chosen this verb since it is a typical representative of a class of verbs that exhibit several specific semantic phenomena. Firstly, the verb changes its semantics depending whether it co-occurs with a present participle clause as its complement, or with infinitival clause. Secondly, in both usages, the semantics of sentences having such a verb as their head verb, involves shared arguments and underspecification, which we can express with type constraints and recursion (memory) locations that represent semantic parameters. A grammatical analysis that includes syntax-semantics interface of phrases with such verbs is not in the space of this paper. We do not purport any full semantic analysis of such verbs that take clausal complements. The section offers an example for a possible semantic representation, with the purpose of the demonstration of the formal language $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ and situational concepts.

We follow a tradition to represent lexical items that have the same orthography, but differ either in their lexical syntax or in semantics, as alternative lexical items with subscripts. Here we render the verb "stop" to different relation constants distinguished by their subscripts, i.e., stop_1 and stop_2. We give $T_{\mathsf{namedJohn}}$ as an example of a complex term for a type, which includes, in the scope of the λ-abstraction, an assignment (39d) that satisfies the constraint in (39c), as a condition for well-formedness. I.e., we assume that in a relevant application system, the date verifies the proposition $(s \models \ll \mathsf{name},\mathsf{John},l,1 \gg)$. Note that while we use linear notation in order to safe typing, and as a tradition from the predominant writing systems, there is no specific order over the λ-abstractions and the argument roles associated with relations, types, and functions in Situation Theory and in the terms of its language $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$.

$$T_{\mathsf{namedJohn}} \equiv \tag{39a}$$

$$\lambda\{\mathsf{s},\mathsf{m},\mathsf{l}\}\big[(\mathsf{s}\models \ll \mathsf{named},\mathsf{m},\mathsf{n},\mathsf{l},1\gg) \tag{39b}$$

$$\mathsf{s.th.}\ \{(\mathsf{n}:\lambda\{N\}(s\models \ll \mathsf{name},\mathsf{N},l,1\gg))\} \tag{39c}$$

$$\mathsf{where}\ \{\mathsf{n}:=\mathsf{John}\}\big] \tag{39d}$$

Let T_{per} and $T_{\mathsf{namedJohn}}$ be the following types:

$$T_{\textsc{animate}} \equiv \lambda\{\mathsf{s},\mathsf{l},\mathsf{m}\}(\mathsf{s}\models \ll \mathsf{animate},\mathsf{m},\mathsf{l},1\gg) \tag{40}$$

$$T_{\mathsf{per}} \equiv \lambda\{\mathsf{s},\mathsf{m},\mathsf{l}\}(\mathsf{s}\models \ll \mathsf{person},\mathsf{m},\mathsf{l},1\gg) \tag{41}$$

Let $\mathbb{A}$ be the type of activities by animate actors i, abstracted from specific activities, actors, space-time locations, and polarity, i.e., abstraction over polarity denoted by a pure variable p:

$$\mathbb{A} \equiv \lambda\{p_1,i,l,p\}(s\models \ll activity, \tag{42a}$$

$$\textsc{rel}:arg_1:p_1, \tag{42b}$$

$$T_{\textsc{animate}}(s_1,l):actor:i, \tag{42c}$$

$$\textsc{loc}:Loc:l,\ \textsc{pol}:Pol:p\gg) \tag{42d}$$

and d be the relation between individuals i, locations l, and polarities p, such that i is or is not (expressed by p) drinking tea at l:

$$d \equiv \lambda\{i,l,p\} \ll \mathsf{drink},\ \mathsf{drinker} : i, \tag{43a}$$
$$\mathsf{liquid} : \mathsf{tea}, \tag{43b}$$
$$\mathsf{Loc} : l,\ \mathsf{Pol} : p \gg \tag{43c}$$

Now, we can represent the interpretation of an utterance of the sentence "John stopped drinking tea." by rendering it into the $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$-term T_1, as in (44a).

$$\text{John stopped drinking tea.} \xrightarrow{\mathsf{render}} T_1 \tag{44a}$$
$$\text{John stopped to drink tea.} \xrightarrow{\mathsf{render}} T_2 \tag{44b}$$

where T_1, T_2 are the terms in (45a)–(45h) and (46a)–(46k), respectively:

$$T_1 \equiv (s_d \models \ll \mathsf{stop}_1,\ A_0 : \mathsf{arg}_0 : j, \tag{45a}$$
$$A_1 : \mathsf{arg}_1 : a_1, \tag{45a}$$
$$\mathsf{Loc} : l_d,\ \mathsf{Pol} : 1 \gg) \tag{45a}$$
$$\mathsf{s.th.}\ (s_d, j, l_d : T_{\mathsf{per}}), \tag{45b}$$
$$(s_d, j, l_d : T_{\mathsf{namedJohn}}), \tag{45c}$$
$$(a_1, j, l_b, 1 : \mathbb{A}), \tag{45d}$$
$$(s_d \models \ll a_1,\ [i] : j,\ [l] : l_b,\ [p] : 1 \gg), \tag{45e}$$
$$(s_d \models \ll a_1,\ [i] : j,\ [l] : l_a,\ [p] : 0 \gg), \tag{45f}$$
$$(s_d \models l_b \prec l_d), (s_d \models l_d \prec l_a)\} \tag{45g}$$
$$\mathsf{where}\ \{a_1 := d\} \tag{45h}$$

$$T_2 \equiv \tag{46a}$$
$$(s_d \models \ll \mathsf{stop}_2,\ A_0 : \mathsf{arg}_0 : j, \tag{46b}$$
$$A_1 : \mathsf{arg}_1 : a_1,\ A_2 : \mathsf{arg}_2 : a_2, \tag{46c}$$
$$\mathsf{Loc} : l_d,\ \mathsf{Pol} : 1 \gg) \tag{46d}$$
$$\mathsf{s.th.}\ (s_d, j, l_d : T_{\mathsf{per}}), (s_d, j, l_d : T_{\mathsf{namedJohn}}), \tag{46e}$$
$$(a_1, j, l_b, 1 : \mathbb{A}), (a_2, j, l_a, 1 : \mathbb{A}), \tag{46f}$$
$$(s_d \models \ll a_1,\ [i] : j,\ [l] : l_b,\ [p] : 1 \gg), \tag{46g}$$
$$(s_d \models \ll a_1,\ [i] : j,\ [l] : l_a,\ [p] : 0 \gg), \tag{46h}$$
$$(s_d \models \ll \mathsf{intends}, j, P, l_d, 1 \gg), \tag{46i}$$
$$(s_d \models l_b \prec l_d), (s_d \models l_d \prec l_a)\} \tag{46j}$$
$$\mathsf{where} \tag{46k}$$
$$\{a_2 := d, \tag{46l}$$
$$P := (a_2 : \lambda\{A\}(s_f \models \ll A,\ [i] : j,\ [l] : l_a,\ [p] : 1 \gg))\} \tag{46m}$$

6 FUTURE WORK

Our target is development of Situation Theory and its formal languages, like $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ introduced in this paper, for enhanced theoretical developments and applications. By (36a), we introduced a new kind of $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ terms, which extend the formal language in (Loukanova, 2015). Informally said, by the denotational and algorithmic semantics (the formal apparatus of which is outside the subject of this paper) of the $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ terms, we also extend the informational coverage introduced in the original Situation Theory, e.g., (Barwise and Perry, 1983), (Devlin, 2008), (Seligman and Moss, 2011), and (Loukanova, 2014). Denotationally, the terms (36a) designate complex sets of parameters, which, via the binding operators where and such that, are memory networks of constrained parametric objects. Information carried by terms such as (32a), respectively (32b), and (36a) is computed according to an algorithm, by mutual recursion, represented by the recursion system of assignments, via the scope of the operator where. The scope of the operator such that introduces constraints over the parameters and the algorithm.

Generalized recursion terms, introduced here, provide a formal introduction of more complex restricted parameters than the ones in classic Situation Theory. The assignment sequences in such terms formalize, and generalize, the situation theoretic notion of parameters and their anchoring to more specific objects. The Acyclicity Constraint 1 provides a formal technique for computational representation of information with acyclic algorithmic steps.

This paper demonstrates the informational richness and fine-granularity of Situation Theory and its potentials for applications via formal languages. E.g., the formal language $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ provides formal techniques for (1) investigation of informational richness of mathematical model-theory of Situation Theory, and (2) useful applications that usually rely on formal language, via syntax-semantic interface between formal syntax and informational structures (Loukanova, 2010). Subjects such as formal reduction calculi, theory of $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$, distinctions between denotational and algorithmic semantics of $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ in Situation Theoretical models are beyond the scope of this paper. Such topics will be presented in future work. Furthermore, we investigate in details the differences in expressiveness between the formal language of Situation Theory introduced in (Loukanova, 2015) and the language $\mathrm{L}^{\mathrm{ST}}_{\mathrm{GP}}$ introduced in this paper, which is richer. E.g., the restricted recursion terms in (32b) generalize the recursion terms introduced in (Loukanova, 2015). The language in (Loukanova, 2015) provides a basis for

proving proper inclusion in it of the functional, formal language introduced in (Moschovakis, 2006), which is different, future work.

REFERENCES

Aczel, P. (1988). *Non-well-founded Sets*. Number 14 in CSLI Lecture Notes. CSLI Publications, Stanford, California.

Barwise, J. (1981). Scenes and other situations. *The Journal of Philosophy*, 78:369–397.

Barwise, J. (1989). *The Situation in Logic*. Number 17 in CSLI Lecture Notes. CSLI Publications, Stanford, California.

Barwise, J. and Perry, J. (1983). *Situations and Attitudes*. Cambridge, MA:MIT press. Republished as (Barwise and Perry, 1999).

Barwise, J. and Perry, J. (1999). *Situations and Attitudes*. The Hume Series. CSLI Publications, Stanford, California.

Chomsky, N. (1993). *Lectures on government and binding: The Pisa lectures*. Number 9 in Studies in generative grammar. Walter de Gruyter.

Devlin, K. (2008). Situation theory and situation semantics. In Gabbay, D. and Woods, J., editors, *Handbook of the History of Logic*, volume 7, pages 601–664. Elsevier.

Dowty, D. (1979). *Word Meaning and Montague Grammar*. D. Reidel, Dordrecht, Holland.

Jaworski, W. and Przepórkowski, A. (2014). Semantic roles in grammar engineering. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM 2014)*, pages 81–86, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Kandel, E., Schwartz, J., and Jessell, T. (2000). *Principles of neural science*. McGraw-Hill, Health Professions Division.

Loukanova, R. (2010). Computational Syntax-Semantics Interface. In Bel-Enguix, G. and Jiménez-López, M. D., editors, *Language as a Complex System: Interdisciplinary Approaches*, pages 111–150. Cambridge Scholars Publishing.

Loukanova, R. (2014). Situation Theory, Situated Information, and Situated Agents. In Nguyen, N. T., Kowalczyk, R., Fred, A., and Joaquim, F., editors, *Transactions on Computational Collective Intelligence XVII*, volume 8790 of *Lecture Notes in Computer Science*, pages 145–170. Springer Berlin Heidelberg.

Loukanova, R. (2015). Underspecified Relations with a Formal Language of Situation Theory. In Loiseau, S., Filipe, J., Duval, B., and van den Herik, J., editors, *Proceedings of the 7th International Conference on Agents and Artificial Intelligence*, volume 1, pages 298–309. SCITEPRESS — Science and Technology Publications, Lda.

Moschovakis, Y. N. (2006). A logical calculus of meaning and synonymy. *Linguistics and Philosophy*, 29:27–89.

Seligman, J. and Moss, L. S. (2011). Situation Theory. In van Benthem, J. and ter Meulen, A., editors, *Handbook of Logic and Language*, pages 253–329. Elsevier, Amsterdam.

Squire, L. and Kandel, E. (2009). *Memory: From Mind to Molecules*. Roberts & Co.