The Unit Graphs Framework: Foundational Concepts and Semantic Consequence

Maxime Lefrançois and **Fabien Gandon**
Wimmics, Inria, I3S, CNRS, UNSA
2004 rte des Lucioles, BP. 93, 06902 Sophia Antipolis, France
{maxime.lefrancois,fabien.gandon}@inria.fr

Abstract

We are interested in a graph-based Knowledge Representation formalism that would allow for the representation, manipulation, query, and reasoning over dependency structures, and linguistic knowledge of the Explanatory and Combinatorial Dictionary in the Meaning-Text Theory framework. Neither the semantic web formalisms nor the conceptual graphs appear to be suitable for this task, and this led to the introduction of the new Unit Graphs framework. This paper first introduces the foundational concepts of this framework: Unit Graphs are defined over a support that contains: i) a hierarchy of unit types which is strongly driven by their actantial structure, ii) a hierarchy of circumstantial symbols, and iii) a set of unit identifiers. Then, this paper provides all of these objects with a model semantics that enables to define the notion of semantic consequence between Unit Graphs.

1 Introduction

We are interested in the ability to reason over dependency structures and linguistic knowledge of the Explanatory and Combinatorial Dictionary (ECD), which is the lexicon at the core of the Meaning-Text Theory (MTT) (Mel'čuk, 2006).

Some formalisation works have been led on the ECD. For instance Kahane and Polguère, 2001) proposed a formalization of Lexical Functions, and the Definiens project (Barque and Polguère, 2008; Barque et al., 2010) aims at formalizing lexicographic definitions with genus and specific differences for the TLFi[1]. Adding to these formalization works, the goal of the Unit Graphs formalism is to propose a formalization from a knowledge engineering perspective, compatible with standard Knowledge Representation (KR) formalisms. The term *formalization* here means not only *make non-ambiguous*, but also *make operational*, i.e., *such that it supports logical operations* (e.g., knowledge manipulation, query, reasoning). We thus adopt a knowledge engineering approach applied to the domain of the MTT.

At first sight, two existing KR formalisms seemed interesting for representing dependency structures: semantic web formalisms (RDF/S, OWL, SPARQL), and Conceptual Graphs (CGs) (Sowa, 1984; Chein and Mugnier, 2008). Both formalisms are based on directed labelled graph structures, and some research has been done towards using them to represent dependency structures and knowledge of the lexicon (OWL in (Lefrançois and Gandon, 2011; Boguslavsky, 2011), CGs at the conceptual level in (Bohnet and Wanner, 2010)). Yet Lefrançois, 2013) showed that neither of these KR formalisms can represent linguistic predicates. As the CG formalism is the closest to the semantic networks, the following choice has been made (Lefrançois, 2013): *Modify the CGs formalism basis, and define transformations to the RDF syntax for sharing, and querying knowledge.* As we are to represents linguistic units of different nature (e.g., semantic units, lexical units, grammatical units, words), term *unit* has been chosen to be used in a generic manner, and the result of this adaptation is thus *the Unit Graphs (UGs) framework*. The valency-based predicates are represented by unit types, and are described in a structure called the unit types hierarchy. Unit types specify through actant slots and signatures how their instances (i.e., units) may be linked to other units in a UG. Unit Graphs are then defined over a support that contains: i) a hierarchy of unit types which is strongly driven by their actantial structure, ii) a hierarchy of circumstantial sym-

[1]Trésor de la Langue Française informatisé, http://atilf.atilf.fr

bols, and iii) a set of unit identifiers.

Apart from giving an overview foundational concepts of the UGs framework, the main goal of this paper is to answer the following research question: *What semantics can be attributed to UGs, and how can we define the entailment problem for UGs ?*

The rest of this paper is organized as follows. Section 2 overviews the UGs framework: the hierarchy of unit types (§2.1), the hierarchy of circumstantial symbols (§2.2), and the Unit Graphs (§2.3). Then, section 3 provides all of these mathematical objects with a model, and finally the notion of semantic consequence between UGs is introduced (§3.4).

2 Background: overview of the Unit Graphs Framework

For a specific Lexical Unit L, (Mel'čuk, 2004, p.5) distinguishes considering L in language (i.e., in the lexicon), or in speech (i.e., in an utterance). KR formalisms and the UGs formalism also make this distinction using types. In this paper and in the UGs formalism, there is thus a clear distinction between *units* (e.g., semantic unit, lexical unit), which will be represented in the UGs, and their *types* (e.g., semantic unit type, lexical unit type), which are roughly classes of units for which specific features are shared. It is those types that specify through actant slots and signatures how their instances (i.e., units) are to be linked to other units in a UG.

2.1 Hierarchy of Unit Types

Unit types and their actantial structure are described in a structure called *hierarchy*, that specifies how units may, must, or must not be interlinked in a UG.

Definition 2.1. A hierarchy of unit types is denoted $\mathcal{T}$ and is defined by a tuple:

$$\mathcal{T} \overset{\text{def}}{=} (T_D, \boldsymbol{S_\mathcal{T}}, \boldsymbol{\gamma}, \boldsymbol{\gamma_1}, \boldsymbol{\gamma_0}, C_A, \perp^{\sqcap}_{A}, \{\varsigma_t\}_{t\in\boldsymbol{T}})$$

This structure has been thoroughly described in (Lefrançois and Gandon, 2013a; Lefrançois, 2013). Let us overview its components.

T_D is a set of *declared Primitive Unit Types (PUTs)*. This set is partitioned into linguistic PUTs of different nature (e.g., deep semantic, semantic, lexical). $\boldsymbol{S_\mathcal{T}}$ is a set of Actant Symbols (ASymbols). $\boldsymbol{\gamma}$ (resp1. $\boldsymbol{\gamma_1}$, resp2. $\boldsymbol{\gamma_0}$) assigns to every $s \in \boldsymbol{S_\mathcal{T}}$ its radix[2] (resp1. obligat[3], resp2. prohibet[4]) unit type $\boldsymbol{\gamma}(s)$ (resp1. $\boldsymbol{\gamma_1}(s)$, resp2. $\boldsymbol{\gamma_0}(s)$) that introduces (resp1. makes obligatory, resp2. makes prohibited) an Actant Slot (ASlot) of symbol s. The set of PUTs is denoted $\boldsymbol{T}$ and defined as the disjoint union of T_D, the ranges of $\boldsymbol{\gamma}$, $\boldsymbol{\gamma_1}$ and $\boldsymbol{\gamma_0}$, plus the *prime universal PUT* $\top$ and the *prime absurd PUT* $\perp$ (eq. 1).

$$\boldsymbol{T} \overset{\text{def}}{=} T_D \uplus \boldsymbol{\gamma}(\boldsymbol{S_\mathcal{T}}) \uplus \boldsymbol{\gamma_1}(\boldsymbol{S_\mathcal{T}}) \uplus \boldsymbol{\gamma_0}(\boldsymbol{S_\mathcal{T}}) \uplus \{\perp, \top\} \tag{1}$$

$\boldsymbol{T}$ is then pre-ordered by a relation $\lesssim$ which is computed from the set $C_A \subseteq \boldsymbol{T}^2$ of asserted PUTs comparisons. $t_1 \lesssim t_2$ models the fact that the PUT t_1 is more specific than the PUT t_2. Then a unit type has a set (that may be empty) of ASlots, whose symbols are chosen in the set $\boldsymbol{S_\mathcal{T}}$. Moreover, ASlots may be obligatory, prohibited, or optional. The set of ASlots (resp1. obligatory ASlots, resp2. prohibited ASlots, resp3. optional ASlots) of a PUT is thus defined as the set of their symbols $\boldsymbol{\alpha}(t) \subseteq \boldsymbol{S_\mathcal{T}}$ (resp1. $\boldsymbol{\alpha_1}(t)$, resp2. $\boldsymbol{\alpha_0}(t)$, resp3. $\boldsymbol{\alpha_?}(t)$).

The set of ASlots (resp1. obligatory ASlots, resp2. prohibited ASlots) of a PUT $t \in \boldsymbol{T}$ is defined as the set of ASymbol whose radix (resp1. obligat, resp2. prohibet) is more general or equivalent to t, and the set of optional ASlots of a PUT t is the set of ASlots that are neither obligatory nor prohibited. The number of ASlots of a PUT is denoted its *valency*. $\{\varsigma_t\}_{t\in\boldsymbol{T}}$, the set of signatures of PUTs, is a set of functions. For all PUT t, ς_t is a function that associates to every ASlot s of t a set of PUT $\varsigma_t(s)$ that characterises the type of the unit that fills this slot. Signatures participate in the specialization of the actantial structure of PUTs, which means that if $t_1 \lesssim t_2$ and s is a common ASlot of t_1 and t_2, the signature of t_1 for s must be more specific or equivalent than that of t_2. Hence $t_1 \lesssim t_2$ implies that the actancial structure of t_1 is more specific than the actantial structure of t_2.

Now a unit type may consist of several conjoint PUTs. We introduce the set $\boldsymbol{T}^{\sqcap}$ of possible *Conjunctive Unit Types (CUTs)* over $\boldsymbol{T}$ as the power-

[2]radix is a latin word that means ⌜root⌝.

[3]obligat is the conjugated form of the latin verb obligo, 3p sing. indic., ⌜it makes mandatory⌝.

[4]prohibet is the conjugated form of the latin verb prohibeo, 3p sing. indic., ⌜it prohibits⌝.

set[5] of $\boldsymbol{T}$. The set $\bot_A^\cap$ is the set of declared absurd CUTs that can not be instantiated. The definition of the actancial structure of PUTs is naturally extended to CUTs as follows:

$$\boldsymbol{\alpha}^\cap(t^\cap) \overset{\text{def}}{=} \bigcup_{t\in t^\cap}\boldsymbol{\alpha}(t) \quad (2)$$

$$\boldsymbol{\alpha_1}^\cap(t^\cap) \overset{\text{def}}{=} \bigcup_{t\in t^\cap}\boldsymbol{\alpha_1}(t) \quad (3)$$

$$\boldsymbol{\alpha_0}^\cap(t^\cap) \overset{\text{def}}{=} \bigcup_{t\in t^\cap}\boldsymbol{\alpha_0}(t) \quad (4)$$

$$\boldsymbol{\alpha_?}^\cap(t^\cap) \overset{\text{def}}{=} \boldsymbol{\alpha}^\cap(t^\cap) - \boldsymbol{\alpha_1}^\cap(t^\cap) - \boldsymbol{\alpha_0}^\cap(t^\cap) \quad (5)$$

$$\boldsymbol{\varsigma}_{t^\cap}^\cap(s) \overset{\text{def}}{=} \bigcup_{t\in t^\cap | s\in\boldsymbol{\alpha}(t)}\boldsymbol{\varsigma}_t(s) \quad (6)$$

Finally the pre-order $\lesssim$ over $\boldsymbol{T}$ is extended to a pre-order $\overset{\cap}{\lesssim}$ over $\boldsymbol{T}^\cap$ as defined by Lefrançois and Gandon, 2013a). Lefrançois and Gandon, 2013b) proved that in the hierarchy of unit types, if $t_1^\cap \overset{\cap}{\lesssim} t_2^\cap$ then the actantial structure of $t_1^\cap$ is more specific than that of $t_2^\cap$, except for some degenerated cases. Thus as one goes down the hierarchy of unit types, an ASlot with symbol s is introduced by the radix $\{\boldsymbol{\gamma}(s)\}$ and first defines an optional ASlot for any unit type $t^\cap$ more specific than $\{\boldsymbol{\gamma}(s)\}$, as long as $t^\cap$ is not more specific than the obligat $\{\boldsymbol{\gamma_1}(s)\}$ (resp. the prohibet $\{\boldsymbol{\gamma_0}(s)\}$) of s. If that happens, the ASlot becomes obligatory (resp. prohibited). Moreover, the signature of an ASlot may only become more specific.

2.2 Hierarchy of Circumstantial Symbols

Unit types specify how unit nodes are linked to other unit nodes in the UGs. As for any slot in a predicate, one ASlot of a unit may be filled by only one unit at a time. Now, one may also encounter dependencies of another type in some dependency structures: circumstantial dependencies (Mel'čuk, 2004). Circumstantial relations are considered of type instance-instance contrary to actantial relations. Example of such relations are the deep syntactic representation relations **ATTR**, **COORD**, **APPEND** of the MTT, but we may also define other such relations to represent the link between a lexical unit and its sense for instance.

We thus introduce a finite set of so-called Circumstantial Symbols (CSymbols) $\boldsymbol{S_\mathcal{C}}$ which is a set of binary relation symbols. In order to classify $\boldsymbol{S_\mathcal{C}}$ in sets and subsets, we introduce a partial order $\overset{\mathcal{C}}{\lesssim}$ over $\boldsymbol{S_\mathcal{C}}$. $\overset{\mathcal{C}}{\lesssim}$ is the reflexo-transitive closure of a set of *asserted comparisons* $\boldsymbol{C_{S_\mathcal{C}}} \subseteq \boldsymbol{T}^2$.

[5]The powerset of X is the set of all subsets of X: 2^X

Finally, to each CSymbol is assigned a signature that specifies the type of units that are linked through a relation having this symbol. The set of signatures of CSymbol $\{\boldsymbol{\sigma}_s\}_{s\in\boldsymbol{S_\mathcal{C}}}$ is a set of couples of CUTs: $\{(domain(s), range(s))\}_{s\in\boldsymbol{S_\mathcal{C}}}$. As one goes down the hierarchy of PUTs, we impose that the signature of a CSymbol may only become more specific (eq. 7).

$$s_1 \lesssim s_2 \Rightarrow \boldsymbol{\sigma}(s_1) \overset{\cap}{\lesssim} \boldsymbol{\sigma}(s_2) \quad (7)$$

We may hence introduce the hierarchy of CSymbols:

Definition 2.2. The hierarchy of CSymbols, denoted $\mathcal{C} \overset{\text{def}}{=} (\boldsymbol{S_\mathcal{C}}, \boldsymbol{C_{S_\mathcal{C}}}, \mathcal{T}, \{\boldsymbol{\sigma}_s\}_{s\in\boldsymbol{S_\mathcal{C}}})$, is composed of a finite set of CSymbols $\boldsymbol{S_\mathcal{C}}$, a set of declared comparisons of CSymbol $\boldsymbol{C_{S_\mathcal{C}}}$, a hierarchy of CUTs $\mathcal{T}$, and a set of signatures of the CSymbols $\{\boldsymbol{\sigma}_s\}_{s\in\boldsymbol{S_\mathcal{C}}}$.

2.3 Definition of Unit Graphs (UGs)

The UGs represent different types of dependency structures. Parallel with the Conceptual Graphs, UGs are defined over a so-called *support*.

Definition 2.3. A UGs *support* is denoted $\mathcal{S} \overset{\text{def}}{=} (\mathcal{T}, \mathcal{C}, \boldsymbol{M})$ and is composed of a hierarchy of unit types $\mathcal{T}$, a hierarchy of circumstantial symbols $\mathcal{C}$, and a set of unit identifiers $\boldsymbol{M}$. Every element of $\boldsymbol{M}$ identifies a specific unit, but multiple elements of $\boldsymbol{M}$ may identify the same unit.

In a UG, unit nodes that are typed and marked are interlinked by dependency relations that are either actantial or circumstantial.

Definition 2.4. A UG G defined over a UG-support $\mathcal{S}$ is a tuple denoted $G \overset{\text{def}}{=} (U, \boldsymbol{l}, A, C, Eq)$ where U is the set of unit nodes, $\boldsymbol{l}$ is a labelling mapping over U, A and C are respectively actantial and circumstantial triples, and Eq is a set of asserted unit node equivalences.

Let us detail the components of G.

U is the set of *unit nodes*. Every unit node represents a specific unit, but multiple unit nodes may represent the same unit. Unit nodes are typed and marked so as to respectively specify what CUT they have and what unit they represent. The marker of a unit node is a set of unit identifiers for mathematical reasons. The set of *unit node markers* is denoted $\boldsymbol{M}^\cap$ and is the powerset[5] of $\boldsymbol{M}$. If a unit node is marked by $\varnothing$, it is said to be *generic*, and the represented unit is unknown. On the other hand, if a unit node is marked $\{m_1, m_2\}$, then the unit identifiers m_1 and m_2 actually identify the same unit. $\boldsymbol{l}$ is thus a labelling mapping over U that assigns to each unit node $u \in U$ a couple $\boldsymbol{l}(u) = (t^\cap, m^\cap) \in \boldsymbol{T}^\cap \times \boldsymbol{M}^\cap$ of a CUT and a unit node marker. We denote $t^\cap = type(u)$ and $m^\cap = marker(u)$.

A is the set of *actantial triples* $(u, s, v) \in U \times \boldsymbol{S}_\mathcal{T} \times U$. For all $a = (u, s, v) \in A$, the unit represented by v fills the ASlot s of the unit represented by u. We denote $u = governor(a)$, $s = symbol(a)$ and $v = actant(a)$. We also denote $arc(a) = (u, v)$.

C is the set of *circumstantial triples* $(u, s, v) \in U \times \boldsymbol{S}_\mathcal{C} \times U$. For all $c = (u, s, v) \in C$, the unit represented by u governs the unit represented by v with respect to s. We denote $u = governor(c)$, $s = symbol(c)$ and $v = circumstantial(c)$. We also denote $arc(c) = (u, v)$.

$Eq \subseteq U^2$ is the set of so-called *asserted unit node equivalences*. For all $(u_1, u_2) \in U^2$, $(u_1, u_2) \in Eq$ means that u_1 and u_2 represent the same unit. The Eq relation is not an equivalence relation over unit nodes[6]. We thus distinguish explicit and implicit knowledge.

UGs so defined are the core dependency structures of the UGs mathematical framework. On top of these basic structures, one may define for instance rules and lexicographic definitions. Due to space limitation we will not introduce such advanced aspects of the UGs formalism, and we will provide a model to UGs defined over a support that does not contain definitions of PUTs.

3 Model Semantic for UGs

3.1 Model of a Support

In this section we will provide the UGs framework with a model semantic based on a relational algebra. Let us first introduce the definition of the model of a support.

Definition 3.1 (Model of a support)**.** Let $\mathcal{S} = (\mathcal{T}, \mathcal{C}, \boldsymbol{M})$ be a support. A model of $\mathcal{S}$ is a couple $M = (D, \delta)$. D is a set called the domain of M that contains a special element denoted $\bullet$ that represents *nothing*, plus at least one other element. δ is denoted the *interpretation function* and must be such that:

- M is a model of $\mathcal{T}$;
- M is a model of $\mathcal{C}$;
- $\forall m \in \boldsymbol{M}, \delta(m) \in D \setminus \bullet$;

[6]An equivalent relation is a reflexive, symmetric, and transitive relation.

This definition requires the notion of model of a unit types hierarchy, and model of a CSymbols hierarchy. We will sequentially introduce these notions in the following sections.

3.2 Model of a Hierarchy of Unit Types

The interpretation function δ associates with any PUT $t \in \boldsymbol{T}$ a relation $\delta(\{t\})$ of arity $1 + valency(t)$ with the following set of attributes (eq. 8):

- a primary attribute denoted 0 ($0 \notin \boldsymbol{S}_\mathcal{T}$) that provides $\{t\}$ with the semantics of a class;
- an attribute for each of its ASlot in $\boldsymbol{\alpha}(t)$ that provides $\{t\}$ with the dual semantics of a relation.

$$\begin{aligned}\forall t \in \boldsymbol{T}, \delta(\{t\}) \subseteq D^{1+valency(t)} \\ \text{with attributes } \{0\} \cup \boldsymbol{\alpha}(t)\end{aligned} \tag{8}$$

Every tuple r of $\delta(\{t\})$ can be identified to a mapping, still denoted r, from the attribute set $\{0\} \cup \boldsymbol{\alpha}(t)$ to the universe D. r describes how a unit of type $\{t\}$ is linked to its actants. $r(0)$ is the unit itself, and for all $s \in \boldsymbol{\alpha}(t)$, $r(s)$ is the unit that fills ASlot s of $r(0)$. If $r(s) = \bullet$, then *there is no unit* that fills ASlot s of $r(0)$. A given unit may be described at most once in $\delta(\{t\})$, so 0 is a unique key in the interpretation of every PUT:

$$\begin{aligned}\forall t \in \boldsymbol{T}, \forall r_1, r_2 \in \delta(\{t\}), \\ r_1(0) = r_2(0) \Rightarrow r_1 = r_2\end{aligned} \tag{9}$$

$\top$ must be the type of every unit, except for the special *nothing* element $\bullet$, and $\bot$ must be the type of no unit. As the projection $\pi_0\delta(\{t\})$ on the main attribute 0 represents the set of units having type $\{t\}$, equations 10 and 11 model these restrictions.

$$\pi_0\delta(\{\top\}) = D \setminus \bullet; \tag{10}$$

$$\delta(\{\bot\}) = \varnothing \tag{11}$$

The ASlot s of the obligat $\boldsymbol{\gamma_1}(s)$ must be filled by some unit, but no unit may fill ASlot s of the prohibet $\boldsymbol{\gamma_0}(s)$. As for every $s \in \boldsymbol{\alpha}(t)$, the projection $\pi_s\delta(\{t\})$ represents the set of units that fill the ASlot s of some unit that has type t, equations 12 and 13 model these restrictions.

$$\forall s \in \boldsymbol{S}_{\mathcal{T}}, \bullet \notin \pi_s\delta(\{\boldsymbol{\gamma_1}(s)\}); \tag{12}$$

$$\forall s \in \boldsymbol{S}_{\mathcal{T}}, \pi_s\delta(\{\boldsymbol{\gamma_0}(s)\}) = \{\bullet\}; \tag{13}$$

Now if a unit $i \in D$ is of type $\{t_1\}$ and t_1 is more specific than t_2, then the unit is also of type $\{t_2\}$, and the description of i in $\delta(\{t_2\})$ must correspond to the description of i in $\delta(\{t_1\})$. Equivalently, the projection of $\delta(\{t_1\})$ on the attributes of $\delta(\{t_2\})$ must be a sub-relation of $\delta(\{t_2\})$:

$$\begin{aligned}&\forall t_1 \lesssim t_2,\\ &\quad \pi_{\{0\}\cup\boldsymbol{\alpha}(t_2)}\delta(\{t_1\}) \subseteq \delta(\{t_2\})\end{aligned} \tag{14}$$

The interpretation of a CUT is the join of the interpretation of its constituting PUTs, except for $\varnothing$ which has the same interpretation as $\{\top\}$, and asserted absurd CUTs $t^\cap \in \bot^\sqcap_A$ that contain no unit.

$$\begin{aligned}&\forall t^\cap \in \boldsymbol{T}^\cap \setminus \varnothing - \bot^\sqcap_A,\\ &\quad \delta(t^\cap) = \bowtie_{t\in t^\cap} \delta(\{t\})\end{aligned} \tag{15}$$

$$\delta(\varnothing) = \delta(\{\top\}) \tag{16}$$

$$\forall t^\cap \in \bot^\sqcap_A, \delta(t^\cap) = \varnothing \tag{17}$$

Finally, for every unit of type $\{t\}$ and for every ASlot of t, the unit that fills ASlot s must be either *nothing*, or a unit of type $\boldsymbol{\varsigma}_t(s)$:

$$\begin{aligned}&\forall t \in \boldsymbol{T}, \forall s \in \boldsymbol{\alpha}(t),\\ &\quad \pi_s\delta(\{t\}) \setminus \bullet \subseteq \pi_0\delta(\boldsymbol{\varsigma}_t(s))\end{aligned} \tag{18}$$

We may now define the model of a unit type hierarchy.

Definition 3.2. Let be a unit types hierarchy $\mathcal{T} = (T_D, \boldsymbol{S}_{\mathcal{T}}, \boldsymbol{\gamma}, \boldsymbol{\gamma_1}, \boldsymbol{\gamma_0}, C_A, \bot^\sqcap_A, \{\boldsymbol{\varsigma}_t\}_{t\in\boldsymbol{T}})$. A model of $\mathcal{T}$ is a couple $M = (D, \delta)$ such that the interpretation function δ satisfies equations 8 to 18.

3.3 Model of a Hierarchy of Circumstantial Symbols

So as to be also a model of a CSymbols hierarchy, the interpretation function δ must be extended and further restricted as follows.

The interpretation function δ associates with every CSymbol $s \in \boldsymbol{S}_{\mathcal{C}}$ a binary relation $\delta(s)$ with two attributes : gov which stands for governor, and $circ$ which stands for circumstantial.

$$\begin{aligned}&\forall s \in \boldsymbol{S}_{\mathcal{C}}, \delta(s) \subseteq (D \setminus \bullet)^2,\\ &\quad \text{a relation with attributes } \{gov, circ\};\end{aligned} \tag{19}$$

Parallel with binary relations in the semantic model of the CGs formalism, if a CSymbol s_1 is more specific than another CSymbol s_2, then the interpretation of s_1 must be included in the interpretation of s_2.

$$\forall s_1, s_2 \in \boldsymbol{S}_{\mathcal{C}}, s_1 \overset{\mathcal{C}}{\lesssim} s_2 \Rightarrow \delta(s_1) \subseteq \delta(s_2) \tag{20}$$

Finally, the type of the units that are linked through a CSymbol s must correspond to the signature of s.

$$\forall s \in \boldsymbol{S}_{\mathcal{C}}, \pi_{gov}\delta(s) \subseteq \pi_0\delta(domain(s)); \tag{21}$$

$$\forall s \in \boldsymbol{S}_{\mathcal{C}}, \pi_{circ}\delta(s) \subseteq \pi_0\delta(range(s)); \tag{22}$$

We may thus define the model of a CSymbols hierarchy.

Definition 3.3 (Model of a Circumstantial Dependency Symbols Hierarchy)**.** Let be a CSymbols hierarchy $\mathcal{C} = (\boldsymbol{S}_{\mathcal{C}}, \boldsymbol{C}_{\boldsymbol{S}_{\mathcal{C}}}, \mathcal{T}, \{\boldsymbol{\sigma}_s\}_{s\in\boldsymbol{S}_{\mathcal{C}}})$. A model of $\mathcal{C}$ is a model $M = (D, \delta)$ of $\mathcal{T}$ such that the interpretation function δ satisfies equations 19 to 22.

3.4 Model Satisfying a UG and Semantic Consequence

Now that the model of a support is fully defined, we may define the model of a UG. A model of a UG is a model of the support on which it is defined, augmented with an *assignment* mapping over unit nodes that assigns to every unit node an element of D.

Definition 3.4 (Model of a UG)**.** Let $G = (U, \boldsymbol{l}, A, C, Eq)$ be a UG defined over a support $\mathcal{S}$. A model of G is a triple (D, δ, β) where:

- (D, δ) is a model of $\mathcal{S}$;
- β, called an *assignment*, is a mapping from U to D.

So as to satisfy the UG, the assignment β must satisfy a set of requirements. First, if a unit node $u \in U$ has a marker $m \in marker(u)$, then the assignment of u must correspond to the interpretation of m.

$$\forall u \in U, \forall m \in marker(u), \beta(u) = \delta(m) \tag{23}$$

Then, the assignment of any unit node u must belong to the set of units that have type $type(u)$.

$$\forall u \in U, \beta(u) \in \pi_0 \delta(type(u)) \tag{24}$$

For every actantial triple $(u, s, v) \in A$, and as $\{\boldsymbol{\gamma}(s)\}$ is the CUT that introduces a ASlot s, the interpretation $\delta(\{\boldsymbol{\gamma}(s)\})$ must reflect the fact that the unit represented by v fills the actant slot s of the unit represented by u.

$$\begin{aligned} &\forall (u, s, v) \in A, \\ &\quad \pi_{0,s}\delta(\{\boldsymbol{\gamma}(s)\}) = \{(\beta(u), \beta(v))\} \end{aligned} \tag{25}$$

Similarly, for every circumstantial triple $(u, s, v) \in C$, the interpretation of s must reveal the fact that the unit represented by v depends on the unit represented by u with respect to s.

$$\forall (u, s, v) \in C, (\beta(u), \beta(v)) \in \delta(s) \tag{26}$$

Finally, if two unit nodes are asserted to be equivalent, then the unit they represent are the same and their assignment must be the same.

$$\forall (u_1, u_2) \in Eq, \beta(u_1) = \beta(u_2) \tag{27}$$

We may now define the notion of satisfaction of a UG by a model.

Definition 3.5 (Model satisfying a UG)**.** Let $G = (U, \boldsymbol{l}, A, C, Eq)$ be a UG defined over a support $\mathcal{S}$, and (D, δ, β) be a model of G. (D, δ, β) is a *model satisfying* G, noted $(D, \delta, \beta) \models_{\mathrm{m}} G$, if β is an assignment that satisfies equations 23 to 27.

Using the notion of a support model and a UG model it is possible to define an entailment relation between UGs as follows.

Definition 3.6 (Entailment and equivalence)**.** Let H and G be two UGs defined over a support $\mathcal{S}$.

- *G entails H*, or H is a *semantic consequence* of G, noted $G \models_{\mathrm{m}} H$, if and only if for any model (D, δ) of $\mathcal{S}$ and for any assignment β_G such that $(D, \delta, \beta_G) \models_{\mathrm{m}} G$, then there exists an assignment β_H of the unit nodes in H such that $(D, \delta, \beta_H) \models_{\mathrm{m}} H$.
- H and G are *model-equivalent*, noted $H \equiv_{\mathrm{m}} G$, if and only if $H \models_{\mathrm{m}} G$ and $G \models_{\mathrm{m}} H$.

4 Conclusion

We thus studied how to formalize, in a knowledge engineering perspective, the dependency structures and the valency-based predicates. We gave an overview of the foundational concepts of the new graph-based Unit Graphs KR formalism. The valency-based predicates are represented by unit types, and are described in a unit types hierarchy. Circumstantial relations are another kind of dependency relation that are described in a hierarchy, and along with a set of unit identifiers these two structures form a UGs support on which UGs may be defined.

We then provided these foundational structures with a model, in the logical sense, using a relational algebra. We dealt with the problem of prohibited and optional actant slots by adding a special *nothing* element $\bullet$ in the domain of the model, and listed the different equations that the interpretation function must satisfy so that a model satisfies a UG. We finally introduced the notion of semantic consequence, which is a first step towards reasoning with dependency structure in the UGs framework.

We identify three future directions of research.

- We did not introduce the definition of PUTs that are to model lexicographic definitions in the ECD and shall be included to the support. The definition of the model semantics of the UGs shall be completed so as to take these into account.
- A UG represents explicit knowledge that only partially define the interpretations of unit types, CSymbols, and unit identifiers. One need to define algorithms to complete the model, so as to check the entailment of a UG by another.
- We know from ongoing works that such an algorithm may lead to an infinite domain. A condition over the unit types hierarchy must be found so as to ensure that the model is decidable for a finite UG.

References

Barque, L., Nasr, A., and Polguère, A. (2010). From the Definitions of the 'Trésor de la Langue Française' To a Semantic Database of the French Language. In Fryske Akademy, editor, *Proceedings of the XIV Euralex International Congress*, Fryske Akademy, pages 245–252, Leeuwarden, Pays-Bas. Anne Dykstra et Tanneke Schoonheim, dir.

Barque, L. and Polguère, A. (2008). Enrichissement formel des définitions du Trésor de la Langue Française informatisé (TLFi) dans une perspective lexicographique. *Lexique*, 22.

Boguslavsky, I. (2011). Semantic Analysis Based on Linguistic and Ontological Resources. In Boguslavsky, I. and Wanner, L., editors, *Proceedings of the 5th International Conference on Meaning-Text Theory (MTT'2011)*, pages 25–36, Barcelona, Spain. INALCO.

Bohnet, B. and Wanner, L. (2010). Open source graph transducer interpreter and grammar development environment. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 19–21, Valletta, Malta. European Language Resources Association (ELRA).

Chein, M. and Mugnier, M.-L. (2008). *Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs*. Springer-Verlag New York Incorporated.

Kahane, S. and Polguère, A. (2001). Formal foundation of lexical functions. In *Proceedings of ACL/EACL 2001 Workshop on Collocation*, pages 8–15.

Lefrançois, M. (2013). Représentation des connaissances du DEC: Concepts fondamentaux du formalisme des Graphes d'Unités. In *Actes de la 15ème Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL'2013)*, pages 164–177, Les Sables d'Olonne, France.

Lefrançois, M. and Gandon, F. (2011). ILexicOn: Toward an ECD-Compliant Interlingual Lexical Ontology Described with Semantic Web Formalisms. In Boguslavsky, I. and Wanner, L., editors, *Proceedings of the 5th International Conference on Meaning-Text Theory (MTT'2011)*, pages 155–164, Barcelona, Spain. INALCO.

Lefrançois, M. and Gandon, F. (2013a). The Unit Graphs Framework: A graph-based Knowledge Representation Formalism designed for the Meaning-Text Theory. In *Proceedings of the 6th International Conference on Meaning-Text Theory (MTT'2013)*, Prague, Czech Republic.

Lefrançois, M. and Gandon, F. (2013b). The Unit Graphs Mathematical Framework. Research Report RR-8212, Inria.

Mel'čuk, I. (2004). Actants in Semantics and Syntax I: Actants in Semantics. *Linguistics*, 42(1):247–291.

Mel'čuk, I. (2006). Explanatory combinatorial dictionary. *Open Problems in Linguistics and Lexicography*, pages 225–355.

Sowa, J. F. (1984). *Conceptual structures: information processing in mind and machine*. System programming series. Addison-Wesley Pub., Reading, MA.