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 Polgùere, 2001) proposed a formalization of Lexical Functions, and the Definiens project (Barque and Polgùere, 2008; Barque et al., 2010) aims at formalizing lexicographic definitions with genus and specific differences for the TLFi¹. 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 (Lefrancçois and Gandon, 2011; Boguslavsky, 2011), CGs at the conceptual level in (Bohnet and Wanner, 2010)). Yet Lefrancçois, 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 (Lefrancçois, 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-

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:

$$\mathcal{T} \overset{\text{def}}{=} (T_D, S_\mathcal{T}, \gamma, \gamma_1, \gamma_0, C_A, \bot_A; \{\varsigma_t\}_{t \in \mathcal{T}})$$

This structure has been thoroughly described in (Lefrancçois and Gandon, 2013a; Lefrancçois, 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). $S_\mathcal{T}$ is a set of Actant Symbols (ASymbols). $\gamma$ (resp1. $\gamma_1$, resp2. $\gamma_0$) assigns to every $s \in S_\mathcal{T}$ its radix\(^2\) (resp1. obligat\(^3\), resp2. prohibit\(^4\)) unit type $\gamma(s)$ (resp1. $\gamma_1(s)$, resp2. $\gamma_0(s)$) that introduces (resp1. makes obligatory, resp2. makes prohibited) an Actant Slot (ASlot) of symbol $s$. The set of PUTs is denoted $\mathcal{T}$ and defined as the disjoint union of $T_D$, the ranges of $\gamma$, $\gamma_1$ and $\gamma_0$, plus the prime universal PUT $\top$ and the prime absurd PUT $\bot$ (eq. 1).

$$\mathcal{T} \overset{\text{def}}{=} T_D \cup \gamma(S_\mathcal{T}) \cup \gamma_1(S_\mathcal{T}) \cup \gamma_0(S_\mathcal{T}) \cup \{\bot, \top\}$$

(1)

$\mathcal{T}$ is then pre-ordered by a relation $\preceq$ which is computed from the set $C_A \subseteq \mathcal{T}^2$ of asserted PUTs comparisons. $t_1 \preceq t_2$ 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 $S_\mathcal{T}$. 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 $\alpha(t) \subseteq S_\mathcal{T}$ (resp1. $\alpha_1(t)$, resp2. $\alpha_0(t)$, resp3. $\alpha_\gamma(t)$).

The set of ASlots (resp1. obligatory ASlots, resp2. prohibited ASlots) of a PUT $t \in \mathcal{T}$ is defined as the set of ASymbol whose radix (resp1. obligat, resp2. prohibit) 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. $\{\varsigma_t\}_{t \in \mathcal{T}}$, the set of signatures of PUTs, is a set of functions. For all PUT $t$, $\varsigma_t$ 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 $t_1 \preceq t_2$ 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 $t_1 \preceq t_2$ implies that the actancial structure of $t_1$ is more specific than the actancial structure of $t_2$.

Now a unit type may consist of several conjoint PUTs. We introduce the set $\mathcal{T}^\uparrow$ of possible Conjunctive Unit Types (CUTs) over $\mathcal{T}$ as the power-
set of CUTs. The set $\bigcup t^s \alpha$ is the set of declared absurd CUTs that can not be instantiated. The definition of the actantial structure of PUTs is naturally extended to CUTs as follows:

\[ \alpha(\mathfrak{t}') \triangleq \bigcup \{ t \in \mathcal{T} \mid \alpha(t) \} \quad (2) \]
\[ \alpha_1(\mathfrak{t}') \triangleq \bigcup \{ t \in \mathcal{T} \mid \alpha_1(t) \} \quad (3) \]
\[ \alpha_2(\mathfrak{t}') \triangleq \bigcup \{ t \in \mathcal{T} \mid \alpha_2(t) \} \quad (4) \]
\[ \gamma(\mathfrak{t}') \triangleq \bigcup \{ t \in \mathcal{T} \mid \gamma(t) \} \quad (5) \]
\[ \varsigma(\mathfrak{t}') \triangleq \bigcup \{ t \in \mathcal{T} \mid \varsigma(t) \} \quad (6) \]

Finally the pre-order $\lesssim$ over $\mathcal{T}$ is extended to a pre-order $\lesssim$ over $\mathcal{T}$ as defined by Lefrancçois and Gandon, (2013a). Lefrancçois and Gandon, 2013b proved that in the hierarchy of unit types, if $t_1 \lesssim t_2$ then the actantial structure of $t_2^s$ is more specific than that of $t_1^s$, 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 $\{ \gamma(s) \}$ and first defines an optional ASlot for any unit type $t$ more specific than $\{ \gamma(s) \}$, as long as $t$ is not more specific than the obligat $\{ \gamma_1(s) \}$ (resp. the prohibit $\{ \gamma_0(s) \}$) 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 $\text{ATTR}$, $\text{COORD}$, $\text{APPEND}$ 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) $\mathcal{S}_C$ which is a set of binary relation symbols. In order to classify $\mathcal{S}_C$ in sets and subsets, we introduce a partial order $\subseteq$ over $\mathcal{S}_C$. $\subseteq$ is the reflexo-transitive closure of a set of asserted comparisons $\mathcal{C}_{S_C} \subseteq \mathcal{T}^2$.

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

\[ s_1 \lesssim s_2 \Rightarrow \mathfrak{s}(s_1) \lesssim \mathfrak{s}(s_2) \quad (7) \]

We may hence introduce the hierarchy of CSymbols:

**Definition 2.2.** The hierarchy of CSymbols, denoted $\mathcal{C} \triangleq (\mathcal{S}_C, \mathcal{C}_{S_C}, \mathcal{T}, \{ \mathfrak{s}_s \}_{s \in \mathcal{S}_C})$, is composed of a finite set of CSymbols $\mathcal{S}_C$, a set of declared comparisons of CSymbol $\mathcal{C}_{S_C}$, a hierarchy of CUTs $\mathcal{T}$, and a set of signatures of the CSymbols $\{ \mathfrak{s}_s \}_{s \in \mathcal{S}_C}$.

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 $\mathcal{S} \triangleq (T, C, M)$ and is composed of a hierarchy of unit types $T$, a hierarchy of circumstantial symbols $C$, and a set of unit identifiers $M$. Every element of $M$ identifies a specific unit, but multiple elements of $M$ 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 $G \triangleq (U, I, A, C, Eq)$ where $U$ is the set of unit nodes, $I$ 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 $\mathcal{M}^u$ and is the powerset of $\mathcal{M}$. If a unit node is marked by $\emptyset$, 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. $l$ is thus a labelling mapping over $U$ that assigns to each unit node $u \in U$ a couple $l(u) = (t^\alpha, m^\alpha) \in T^\alpha \times M^\alpha$ of a CUT and a unit node marker. We denote $t^\alpha = \text{type}(u)$ and $m^\alpha = \text{marker}(u)$.

$A$ is the set of actantial triples $(u, s, v) \in U \times S_T \times U$. 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 $u = \text{governor}(a)$, $s = \text{symbol}(a)$ and $v = \text{actant}(a)$. We also denote $\text{arc}(a) = (u, v)$.

$C$ is the set of circumstantial triples $(u, s, v) \in U \times S_C \times U$. 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 $u = \text{governor}(c)$, $s = \text{symbol}(c)$ and $v = \text{circumstantial}(c)$. We also denote $\text{arc}(c) = (u, v)$.

$\text{Eq} \subseteq U^2$ is the set of so-called asserted unit node equivalences. For all $(u_1, u_2) \in U^2$, $(u_1, u_2) \in \text{Eq}$ means that $u_1$ and $u_2$ represent the same unit. The $\text{Eq}$ 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 $S = (T, C, M)$ be a support. A model of $S$ 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. $\delta$ is denoted the interpretation function and must be such that:

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

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

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

\[
\forall t \in T, \delta(\{t\}) \subseteq D^{1 + \text{valency}(t)} \text{ with attributes } \{0\} \cup \alpha(t)
\]  

(8)

Every tuple $r$ of $\delta(\{t\})$ can be identified to a mapping, still denoted $r$, from the attribute set $\{0\} \cup \alpha(t)$ 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 $s \in \alpha(t)$, $r(s)$ 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:

\[
\forall t \in T, \forall r_1, r_2 \in \delta(\{t\}), r_1(0) = r_2(0) \Rightarrow r_1 = r_2
\]  

(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.

\[
\pi_0 \delta(\{\top\}) = D \setminus \bullet
\]  

(10)

\[
\delta(\{\bot\}) = \emptyset
\]  

(11)

The ASlot $s$ of the obligat $\gamma_1(s)$ must be filled by some unit, but no unit may fill ASlot $s$ of the prohibit $\gamma_0(s)$. As for every $s \in \alpha(t)$, 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.
\[ \forall s \in S_T, \bullet \notin \pi_s(\{\gamma_1(s)\}) ; \]  
\[ \forall s \in S_T, \pi_s(\{\gamma_0(s)\}) = \{\bullet\} ; \]  
\[ \forall t_1 \preceq t_2, \pi_{(0)\cup a(t_2)}(\{t_1\}) \subseteq \delta(\{t_2\}) \]

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 \( \delta(\{t_2\}) \):

\[ \forall t_1 \preceq t_2, \pi_{(0)\cup a(t_2)}(\{t_1\}) \subseteq \delta(\{t_2\}) \]  

The interpretation of a CUT is the join of the interpretation of its constituting PUTs, except for \( \emptyset \) which has the same interpretation as \( \{\top\} \), and asserted absurd CUTs \( t^\gamma \in \bot_A \) that contain no unit.

\[ \forall t^\gamma \in T^\gamma \setminus \emptyset \subseteq \bot_A, \delta(t^\gamma) = \sqcup_{t \in T^\gamma} \delta(t) \]  
\[ \delta(\emptyset) = \delta(\{\top\}) \]  
\[ \forall t^\gamma \in \bot_A, \delta(t^\gamma) = \emptyset \]

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 \( \zeta_t(s) \):

\[ \forall t \in T, \forall s \in a(t), \pi_s(\{t\}) \bullet \subseteq \pi_0(\zeta_t(s)) \]

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

**Definition 3.2.** Let be a unit types hierarchy \( T = (T_D, S_T, \gamma, \gamma_1, \gamma_0, C_A, \bot_A, \{\zeta_t\}_{t \in T}) \). A model of \( T \) is a couple \( M = (D, \delta) \) such that the interpretation function \( \delta \) 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 \( \delta \) must be extended and further restricted as follows.

The interpretation function \( \delta \) associates with every CSymbol \( s \in S_C \) a binary relation \( \delta(s) \) with two attributes: \( \text{gov} \) which stands for governor, and \( \text{circ} \) which stands for circumstantial.
Then, the assignment of any unit node $u$ must belong to the set of units that have type $\text{type}(u)$.

$$\forall u \in U, \beta(u) \in \pi_0 \delta(\text{type}(u))$$  \hspace{1cm} (24)

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

$$\forall (u, s, v) \in A, \pi_{0,s} \delta(\{\gamma(s)\}) = \{(\beta(u), \beta(v))\}$$  \hspace{1cm} (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 $s$.

$$\forall (u, s, v) \in C, (\beta(u), \beta(v)) \in \delta(s)$$  \hspace{1cm} (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.

$$\forall (u_1, u_2) \in E_{eq}, \beta(u_1) = \beta(u_2)$$  \hspace{1cm} (27)

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

**Definition 3.5** (Model satisfying a UG). Let $G = (U, I, A, C, E_{eq})$ be a UG defined over a support $S$, and $(D, \delta, \beta)$ be a model of $G$. $(D, \delta, \beta)$ is a model satisfying $G$, noted $(D, \delta, \beta) \models_m G$, if $\beta$ 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 $S$.

- $G$ entails $H$, or $H$ is a semantic consequence of $G$, noted $G \models_m H$, if and only if for any model $(D, \delta)$ of $S$ and for any assignment $\beta_G$ such that $(D, \delta, \beta_G) \models_m G$, then there exists an assignment $\beta_H$ of the unit nodes in $H$ such that $(D, \delta, \beta_H) \models_m H$.
- $H$ and $G$ are model-equivalent, noted $H \equiv_m G$, if and only if $H \models_m G$ and $G \models_m H$.

## 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 • 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


