



The Unit Graphs Mathematical Framework

Maxime Lefrançois

**RESEARCH
REPORT**

N° 8212

January 2013

Project-Team Wimmics

ISRN INRIA/RR--8212--FR+ENG

ISSN 0249-6399



The Unit Graphs Mathematical Framework

Maxime Lefrançois

Project-Team Wimmics

Research Report n° 8212 — January 2013 — 38 pages

Abstract: This research report introduces the fundamental concepts of the Unit Graphs mathematical framework that is designed to reconcile the Meaning-Text Model (MTM) and the Conceptual Graphs formalism (CGs), in order to represent, query, and reason formalized linguistic knowledge. Despite seeming similar, one cannot use GCs to natural to formalize the MTM. Indeed, a linguistic unit may be multi-typed (i.e., is an instance of concept types), and dually, links its actants in an utterance representation (i.e., is a n-ary relation). How shall we revise the basis of the CGs mathematical framework in order to take into account the duality of units ?

Key-words: Linguistic Knowledge Representation, Meaning-Text Theory, Conceptual Graphs, Unit Graphs

**RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE**

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Le formalisme mathématique des graphes d'unités

Résumé : Nous introduisons les concepts fondamentaux du formalisme des Graphes d'Unités, conçu pour concilier le modèle Sens-Texte (MST) et le formalisme des Graphes Conceptuels (GC) afin de représenter, interroger, et raisonner des connaissances linguistiques formalisées. Bien que ressemblants, il n'est pas naturel d'utiliser les GCs pour formaliser le MST. En effet, une unité linguistique peut être multi-typée (i.e., est une instance de types de concepts), et de manière duale, lie ses actants dans une représentation d'énoncé (i.e., est une relation n-aire). Comment doit-on revoir les bases du formalisme mathématique des GCs pour prendre en compte cette dualité des unités ?

Mots-clés : Représentation de Connaissances Linguistiques, Théorie Sens-Texte, Graphes Conceptuels, Graphes d'Unités

Contents

1	Introduction	5
2	Primitive Unit Types (PUTs)	8
2.1	Definition of PUTs	9
2.2	PUT Participant Slots (PSlots)	10
2.3	PUT Hidden Slots (HSlots)	12
2.4	PUT Actant Slots (ASlots)	14
2.5	PUT Signatures	16
3	Conjunctive Unit Types (CUTs)	17
3.1	Definition of CUTs	17
3.2	CUT Slots and Signatures	17
3.3	Pre-order over CUTs	19
3.4	CUTs Hierarchy	22
4	Characterizing CUTs	23
4.1	Necessary Conditions to Compare Two CUTs	23
4.2	Properties of CUT Slots and Signatures	24
4.3	CUT Equivalence Class Sets	28
4.4	Maximal CUTs	29
4.5	Concise CUTs	32
5	Unit Graphs (UGs)	33
5.1	Circumstantial Dependency Symbols Hierarchy	33
5.2	Definition of UGs	34
6	Conclusion and perspectives	36

Glossary

ASlot Actant Slot 8, 12–15, 23, 29, 31

CG Conceptual Graph 5, 6, 8

CUT Conjunctive Unit Type 7, 14–31

DepG Dependency Grammar 5

DSymbol Dependency Symbol 7, 31, 32

ECD Explanatory Combinatorial Dictionary 5, 32

FLN French Lexical Network 5

HSlot Hidden Slot 8, 11, 12, 14, 15, 23, 31

MTT Meaning-Text Theory 5, 6, 31, 32

NL-expressible expressible in a natural language utterance 10–12, 15, 31

PSlot Participant Slot 8–12, 14–17, 19, 21, 22, 24, 30, 31

PSymbol Participation Symbol 9–13, 31

PUT Primitive Unit Type 7–16, 19–21, 24–28, 31, 32

UG Unit Graph 5–9, 19, 29–32

1 Introduction

This research report introduces the fundamental concepts of the Unit Graphs mathematical framework that is designed to reconcile the highly linguistically precise **Meaning-Text Theory (MTT)** (c.f. for instance [Polguère, 1998](#)) and the **Conceptual Graph (CG)** formalism [Sowa \(1984\)](#); [Chein and Mugnier \(2008\)](#), in order to represent, query, and reason formalized linguistic knowledge such as the **Explanatory Combinatorial Dictionaries (ECDs)**, **Dependency Grammar (DepG)**, and linguistic representations *à la MTT* at different representation levels.

The **MTT** is against the mainstream with respect to other modern linguistic theories because it is mainly focused on the lexicon and not on the grammar. The **MTT** lexicon is called **ECD** (c.f. for instance [Mel'čuk, 1999](#); [Mel'čuk, 2006](#)). Only a few past or current projects consisted in implementing the **ECD**, the following list of projects is categorized according to the level of computerization:

- **Simple computerization.** Projects **NADIA-DEC** [Sérasset \(1997\)](#), **Dicoùèbe** [Polguère \(2000\)](#), **DicoInfo** and **DicoEnviro** [L'Homme \(2008\)](#), and **RLF** [Lux-Pogodalla and Polguère \(2011\)](#) for French, **DiCE** [Alonso Ramos \(2003\)](#) for Spanish. These projects are mainly focused towards lexicographic edition. As such, the **ECD** is only computerized, and not formalized thus one cannot reason over linguistic knowledge. These projects would benefit from such a formalization, as it would ease the lexicographers task. Indeed, it would be possible to check that a set of constraints is satisfied, or to suggest preliminary drafts for articles for instance.
- **Computerization and formalization.** **ETAP-3** [Apresian et al. \(2003\)](#); [Boguslavsky et al. \(2004\)](#) for Russian, English and Arabic, is a proprietary natural language processing system. Its applications are among others machine translation. Linguistic knowledge are asserted, and linguistic and grammatical rules are directly formalized in first order logic.

The **French Lexical Network (FLN)** starts to partially formalize definitions using the markup type that has been developed in the **Definiens** project [Barque and Polguère \(2008\)](#); [Barque et al. \(2010\)](#), but none of the above projects completed the computerization nor the formalization of lexical unit lexicographic definitions as the **MTT** wants them to be, which is as minimal semantic decomposition graphs.

Among knowledge representation formalisms, the **CGs** formalism¹ [Sowa \(1984\)](#); [Chein and Mugnier \(2008\)](#) has many similarities with the **MTT** (c.f., table 1). Actually, works that originally inspired these two models have a non-null intersection: [Tesnière \(1959\)](#). What's more, [Sowa \(1989\)](#) early suggested to introduce type definition of concepts and relations that do look similar to lexical units definitions in the **ECD**. Later on [Leclère \(1998\)](#) also worked on the possibility to reason with type and concept definitions. Two communities have been formed around these two models, one in the linguistic field, and the other in the knowledge representation field. A formalization of the **MTT** using the **CGs** would thus enable to formally represent, manipulate, query and reason linguistic knowledge such as **ECDs**, lexical and grammatical rules, and utterance representations at different representation levels.

Moreover, the **ULiS** project [Lefrançois and Gandon \(2011a,b\)](#) envisioned a **MTT**-compliant pivot-based multilingual knowledge base architecture, using the semantic web formalisms. As [Corby et al. \(2000\)](#); [Baget et al. \(2010\)](#) presented possible transformations between **CG** and **RDF/S**, one could adapt the architecture described in the **ULiS** project to **CGs**, and thus:

¹**CGs** in their basic version represent concept type instances interconnected by n -ary relations.

Through the [Universal Linguistic System (ULiS)], a user could interact with an interlingual knowledge base (IKB) in controlled natural language. Linguistic resources themselves [would be] part of a specific IKB: The Universal Lexical Knowledge base (ULK), so that actors may enhance their controlled natural language, through requests in controlled natural language.

Unfortunately, one cannot formalize the **MTT** with **CGs** in a natural manner, here are two reasons for that:

- A unit type in the **MTT** is formalizable *à priori* as a concept type as it is instantiated in linguistic situations. On the other hand, if this unit type is predicative (i.e., has actant slots) it may in a dual manner be formalized as a n -ary relation, such that one of its instances is linked to its actants. **CGs** thus don't allow to naturally represent this duality and hence the notion of linguistic predicate.
- In the theory of actants (c.f., Mel'čuk, 2004a,b) developed with the **MTT**, it is possible that the set of actant slots of a predicative unit type differ from that of another unit type from which it is derived.² Yet, the specification of concept types and relation types in the **CGs** formalism (i.e., the *support*) imposes that two relations with different arity are incomparable. One thus cannot use the natural inheritance mechanism in the context of a predicative unit type.

Meaning-Text Theory (MTT)	Conceptual Graphs (CGs)
Explanatory Combinatorial Dictionary (ECD)	Support
Lexicographic Definitions	Concept and relation types definitions
Linguistic and Grammatical rules	Rules
Utterance representations	Conceptual Graphs

Table 1: Manifest similarities between the **MTT** and **CGs**

This research report introduces the fundamental concepts of a novel mathematical framework called Unit Graphs³ to reconcile the **MTT** and the **CG** formalism. The research question here is thus:

*How shall we revise the basis of the **CGs** mathematical formalism
so that it complies with the **MTT**?*

This research question may be decomposed in several sub-questions that we address in this research report:

- What mathematical structure for a unit types hierarchy and their actant slots ?
- How to extend the hierarchy of unit types to multi-typing ?
- What about circumstantial dependencies ?

²For instance, the predicative meaning unit type (to rain) may be defined from (to fall) but its actant slots (what falls) and (from where it falls) are freezed to (water drops) and (sky/cloud).

³Unit here is a thus a generalization of *linguistic unit*, itself a generalization of different kinds of unit types and for different utterance representation levels (e.g., meaning unit, lexical unit, grammatical unit, word)

d) What would be a formal representation of utterance representations ?

As the **Unit Graphs (UGs)** mathematical framework is mainly conceived for linguistic applications, we will often use examples out of one of the linguistic representation levels from the **MTT**. We thus borrow and slightly extend the Meaning-Text notation system in this document:

S-	Surface	D-	Deep
-Sem-	Semantic	-Syn-	Syntactic
-Morph-	Morphologic	-Phon-	Phonologic
L-	Lexical	-G-	Grammatical
-U	Unit	-UT	Unit Type
$[_{\text{FALL}}]^L$	a specific SynLUT	$[_{\text{present}}]^G$	a specific SynGUT
$[_{\text{FALL}}]_{\text{present}} : *$	a specific SynLU	(ball)	a specific SemUT
(ball) : *	a specific SemU		

As a running example, let us consider the very simple sentence *The cat falls gently of a soft blanket*. In this example, $(\text{fall}) : *$ is a generic sense unit having type the semantic unit type (fall) . The following figure represents a Semantic representation of the utterance.

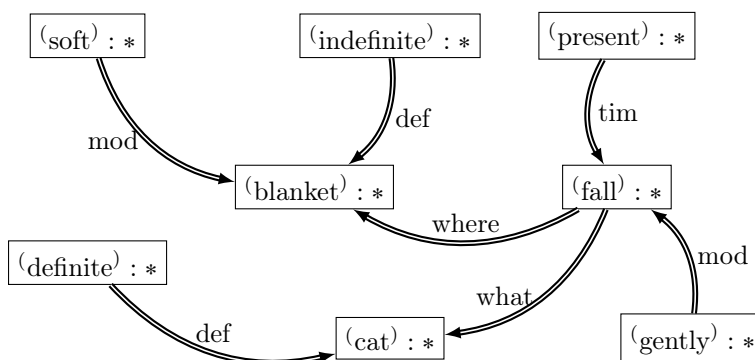


Figure 1: Semantic representations of utterance *The cat falls gently on a soft blanket*

The rest of this research report is organized as follows. We first introduce and study **Primitive Unit Types (PUTs)** (§2), then we will extend their definitions and results to **Conjunctive Unit Types (CUTs)**, and will introduce a hierarchy of **CUTs** (§3) and characterize it (§4). Section 5 will be devoted to the introduction of a hierarchy of **Circumstantial Dependency Symbols (CSymbols)**, and the definition of **UGs** *per se*.

2 Primitive Unit Types (PUTs)

In the mathematical formalism of **Unit Graphs (UGs)**, we establish a clear distinction which is crucial to be understood between unit types and units:

- Nodes of a **UG** are units, and they have one or more types.
- A unit is an instance of a unit type;
- In the formalized version of the **ECD**, we will describe unit types;

Unlike the **CGs** formalism, every type may be considered both as a concept (i.e., it has instances), and as a n -ary relation (i.e., it lies des instances). As described in this research report, unit types enable units categorization, and also the specification of how units must be linked in a **UG**.

This section first introduces atomic types which are called *Primitive Unit Types (PUTs)*, and then introduces notions used to characterize *glsplput* as n -ary relation: **Participant Slots (PSlots)** (§2.2), **Hidden Slots (HSlots)** (§2.3), **Actant Slots (ASlots)** (§2.4), and signatures (§2.5).

2.1 Definition of PUTs

The atoms of unit types are called *Primitive Unit Types (PUTs)* and are chosen among a finite set denoted \mathbf{T} and defined as follows.

Definition 2.1 (PUT Set). A *PUT Set* is a disjoint union denoted $\mathbf{T} \stackrel{\text{def}}{=} T_{\text{declared}} \cup \{\top\} \cup \{\perp\}$, where T_{declared} is a finite set of *declared PUTs*, \top is the *prime universal PUT*, and \perp is the *prime absurd PUT*.

As is classically done in knowledge representation for types and relations, we introduce a pre-order \lesssim over the set \mathbf{T} which models a specialization relation, e.g., '(to rain)' \lesssim '(to fall)' means that '(to rain)' is more specific than '(to fall)' (more specific semantically here). The pre-order over the set of **PUTs** is induced by a set $\mathbf{C}_{\mathbf{T}}$ of comparisons of **PUTs**.

Definition 2.2 (Pre-order over \mathbf{T}). The **PUT** set is pre-ordered by a relation \lesssim , which is induced by a set $\mathbf{C}_{\mathbf{T}} \stackrel{\text{def}}{=} C_{\text{asserted}} \cup C_{\top} \cup C_{\perp} \subseteq \mathbf{T}^2$ where $C_{\text{asserted}} \subseteq \mathbf{T}^2$ is the set of *asserted comparisons*, $C_{\top} \stackrel{\text{def}}{=} \{(\top, t)\}_{t \in \mathbf{T}}$, and $C_{\perp} \stackrel{\text{def}}{=} \{(t, \perp)\}_{t \in \mathbf{T}}$. $(\mathbf{T}, \mathbf{C}_{\mathbf{T}})$ is a directed graph on \mathbf{T} . Let $\mathbf{C}_{\mathbf{T}}^*$ be the reflexo-transitive closure of $\mathbf{C}_{\mathbf{T}}$, i.e., $(t, t') \in \mathbf{C}_{\mathbf{T}}^*$ iff t' is a descendant of t in $\mathbf{C}_{\mathbf{T}}$. The pre-order relation \lesssim is equal to $\mathbf{C}_{\mathbf{T}}^*$, i.e., $\forall t, t' \in \mathbf{T}, t' \lesssim t$ iff $(t, t') \in \mathbf{C}_{\mathbf{T}}^*$.

By construction, the set of **PUTs** is bounded by \top , a maximal element of \mathbf{T} , and \perp , a minimal element of \mathbf{T} :

Proposition 2.1. \top and \perp are respectively a maximal and a minimal element of \mathbf{T} .

Proof. Let us prove that 1) \top is a maximal element of \mathbf{T} , and 2) \perp is a minimal element of \mathbf{T} .

1) Let $t \in \mathbf{T}$. From definition 2.2, $(\top, t) \in C_{\top}$.

$C_{\top} \subseteq \mathbf{C}_{\mathbf{T}} \subseteq \mathbf{C}_{\mathbf{T}}^*$, so $(\top, t) \in \mathbf{C}_{\mathbf{T}}^*$. So $t \lesssim \top$ (def. 2.2) and \top is a maximal element of \mathbf{T} .

2) Conversely, let $t \in \mathbf{T}$. From definition 2.2, $(t, \perp) \in C_{\perp}$.

$C_{\perp} \subseteq \mathbf{C}_{\mathbf{T}} \subseteq \mathbf{C}_{\mathbf{T}}^*$, so $(t, \perp) \in \mathbf{C}_{\mathbf{T}}^*$, so $\perp \lesssim t$ (def. 2.2) and \perp is a maximal element of \mathbf{T} . \square

Let \simeq be the natural *equivalence relation* over **PUTs** defined by $t \simeq t' \Leftrightarrow t \lesssim t'$ and $t' \lesssim t$. The set of equivalence classes defines a partition of \mathbf{T} . Let $t \in \mathbf{T}$, we denote $[t]$ the equivalence class to which t belongs, i.e., $[t] \stackrel{\text{def}}{=} \{t' \in \mathbf{T} \mid t' \simeq t\}$.

Definition 2.3 (Equivalence **PUTs class set).** The *equivalence **PUTs** class set* \mathbf{T}^{\sim} is the quotient set of \mathbf{T} by \simeq , i.e., $\mathbf{T}^{\sim} \stackrel{\text{def}}{=} \mathbf{T}/\simeq = \{[t] \mid t \in \mathbf{T}\}$. We denote t^{\sim} a generic equivalence **PUTs** class set. We define a partial order \lesssim^{\sim} over \mathbf{T}^{\sim} with $t_1^{\sim} \lesssim^{\sim} t_2^{\sim}$ if and only if $\exists t_1 \in t_1^{\sim}, t_2 \in t_2^{\sim}; t_1 \lesssim t_2$.

For instance, figures 2, 3, and 4 visualize equivalence classes using circles. In a same circle, all the **PUTs** are equivalent. In the illustrated examples, **PUTs** '(drizzle)' and '(Scotch mist)' are equivalent.

\perp is the prime absurd **PUT** and should by definition have no unit that has it as a type. Thus, any **PUT** that is more specific than \perp should also be absurd. As \perp is also a minimal element of \mathbf{T} , any **PUT** which is more specific than \perp is actually equivalent to \perp . The equivalence class to which \perp belongs is thus called the set of absurd **PUTs** and denoted \perp^{\sim} . Any $t \in \perp^{\sim}$ is said to be absurd and no unit should have this **PUT** as a type.

Definition 2.4 (Absurd **PUT set).** The set of absurd **PUTs** is denoted \perp^{\sim} and is the set: $\perp^{\sim} \stackrel{\text{def}}{=} [\perp]$.

PUTs may both be considered as a concept (i.e., they have instances), and as a n -ary relation (i.e., they link instances). Next sections are devoted to **PUTs** seen as relations.

2.2 PUT Participant Slots (PSlots)

In the theory of semantic actants (c.f., Mel'čuk, 2004a), the **Actant Slots (ASlots)** of a semantic unit type $\langle L \rangle$ correspond roughly to participants of the linguistic situation $SIT(L)$ denoted by $\langle L \rangle$ that may be expressed in the sentence in a favoured manner by the lexical unit L . When L is specialized to L' , the **ASlots** may change, but participants of $SIT(L)$ are inherited by $SIT(L')$. This good property makes us introduce the notion of **PSlot**⁴ of a unit type. As we shall see, **PSlots** of a semantic unit type $\langle L \rangle$ is roughly : the union of $\langle L \rangle$'s **ASlots** and of every semantic unit type more generic than $\langle L \rangle$'s. Participant positions of $\langle L \rangle$ thus correspond to a subset of participants of $SIT(L)$.

The **UGs** mathematical framework formalizes the notion of **PSlot** as follows. Every **PUT** t has thus zero or more so-called **PSlot**, which represents slots that instances of t have and which may be filled by other units. Each **PSlots** of a **PUT** has a symbol that is chosen among a set of so-called **Participation Symbols (PSymbols)**.

Definition 2.5 (set of **PSymbols**, and **PSlots** of a **PUT**). A **Participation Symbol (PSymbol)** set is a finite set denoted $\mathcal{S}_{\mathcal{T}}$. Any **PUT** has a set of **PSlots** which is defined through a mapping ω from \mathbf{T} to $2^{\mathcal{S}_{\mathcal{T}}}$. For every $t \in \mathbf{T}$, $s \in \omega(t)$ means that t has a **PSlot** with **PSymbol** s .

For instance, in our example illustrated on figure 2 it is assumed that there is a **PSymbol** $what \in \mathcal{S}_{\mathcal{T}}$ to carry the participation meaning *what moves*. Moreover, it is assumed and illustrated that the **PUT** $\langle fall \rangle$ has a **PSlot** having symbol $what$, i.e., $what \in \omega(\langle fall \rangle)$.

Note. Let s be a **PSymbols**. As a shortcut, instead of "**PSlot** having symbol s ", we simply write: "**PSlot** s ".

We also introduce the notion the root of a **PSymbol**. Every **PSymbol** s has a so-called *root* in \mathbf{T} denoted $root(s)$ that introduces a **PSlot** with symbol s . We define the set of **PSlots** of a **PUT** $t \in \mathbf{T}$ as being the set of **PSlots** whose root is more general or equivalent to t :

Definition 2.6 (Root of a **PSlot**). The *root of PSlots* is defined through a mapping $root$ from $\mathcal{S}_{\mathcal{T}}$ to \mathbf{T} that links a **PSlot** s to the **PUT** that introduces s . The set of **PSlots** of a **PUT** $t \in \mathbf{T}$ is defined as the set of **PSlots** whose root is more general than t , i.e., $\forall t \in \mathbf{T}, \omega(t) \stackrel{def}{=} \{s \in \mathcal{S}_{\mathcal{T}} \mid t \lesssim root(s)\}$.

For instance, so that $\langle fall \rangle$ has a **PSlot** $what$, it must be more specific or equivalent than the root of $what$. In our example we chose $root(what) = \langle move \rangle$.

In the following, we use "more general" and "more specific" in their weak sense, i.e., respectively "more general or equivalent" and "more specific or equivalent", unless otherwise stated.

If we were to define a **PUT** more specific than $\langle fall \rangle$ such as $\langle tumble \rangle$, then it would inherit the **PSlot** $what$. Actually, every **PUT** more specific than $root(what)$, and only those **PUTs** more specific than $root(what)$ will have a **PSlot** with symbol $what$. The following property thus holds:

Proposition 2.2. Let $\downarrow root(s)$ be the smallest lower set of \mathbf{T} containing $root(s)$, i.e., $\downarrow root(s) \stackrel{def}{=} \{t \in \mathbf{T} \mid t \lesssim root(s)\}$. For any $s \in \mathcal{S}_{\mathcal{T}}$, $\{t \in \mathbf{T} \mid s \in \omega(t)\} = \downarrow root(s)$.

Proof. Let $s \in \mathcal{S}_{\mathcal{T}}$.

\subseteq : Let $t \in \{t \in \mathbf{T} \mid s \in \omega(t)\}$. $s \in \omega(t)$, by definition 2.6, $t \lesssim root(s)$, so $t \in \downarrow root(s)$.

\supseteq : Let $t \in \downarrow root(s)$. $t \lesssim root(s)$, and by definition 2.6, $s \in \omega(t)$, so $t \in \{t \in \mathbf{T} \mid s \in \omega(t)\}$. \square

In our example illustrated on figure 2, the root of **PSymbol** $what$ is illustrated by symbol \star , and the green zone represents the set of **PUTs** that have a **PSlot** $what$, i.e., the principal lower set of \mathbf{T} generated by $root(what)$.

⁴Despite the fact that the Moscow Semantic School likes to distinguish the notions of participants of a situation vs. semantic actants, we chose to use term *participant slot* instead of a neologism such that *actanciable slot* for instance.

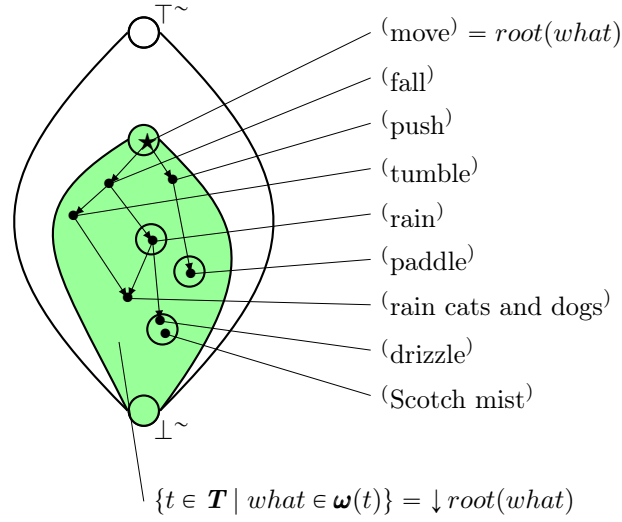


Figure 2: Illustration of the set of **PUTs** that have a **PSlot** *what*.

As a direct consequence, as units get more and more specific (i.e., as we go down the hierarchy of **PUTs**), the set of **PSlots** may only increase. We say that **PSlots** are inherited:

Proposition 2.3. *PSlots are inherited, i.e.,*

$$\forall t_x, t_y \in \mathbf{T} \text{ such that } t_x \lesssim t_y, \omega(t_y) \subseteq \omega(t_x) \quad (1)$$

Moreover, if $t_x \simeq t_y$, then $\omega(t_y) = \omega(t_x)$.

Proof. Let $t_x, t_y \in \mathbf{T}$ such that $t_x \lesssim t_y$, and $s \in \omega(t_y)$.

Definition 2.6 implies $t_y \lesssim \text{root}(s)$. So $t_x \lesssim t_y \lesssim \text{root}(s)$ and $s \in \omega(t_x)$.

The second result is obtained using twice this first result. □

Moreover, as *root* is a mapping, every **PSymbol** has a root. Any minimal element of \mathbf{T} will inherit all the **PSlots**:

Proposition 2.4. $\forall t \in \perp^\sim, \omega(t) = \mathcal{S}_{\mathcal{T}}$.

Proof. As \perp is a minimal element in \mathbf{T} , then $\forall s \in \mathcal{S}_{\mathcal{T}}, \perp \lesssim \text{root}(s)$. So $s \in \omega(\perp)$. Thus $\omega(\perp) = \mathcal{S}_{\mathcal{T}}$. Now, for all $t \in \perp^\sim$ we know that $t \lesssim \perp$, and using proposition 2.3 $\mathcal{S}_{\mathcal{T}} \subseteq \omega(t)$. So $\omega(t) = \mathcal{S}_{\mathcal{T}}$. □

2.3 PUT Hidden Slots (HSlots)

Although the set of **PSlots** increases as we go down the hierarchy of units due to the introduction of the new **PSlots**, the set of **ASlots** of a unit type remains in the range of one to approximately six according to the **MTT Mel'čuk (2004a,b)**. Others are still part of the linguistic situation, but somehow hidden and no-longer expressible in a favoured manner in the natural language utterance representation. We thus say that those positions among the **PSlots** that are not actantial are *hidden*, and we hence introduce the notion of **Hidden Slots (HSlots)** of a **PUT**.

For instance when it drizzles, what moves, i.e., water, is not expressible in a favoured manner by the lexical unit of type (type) anymore. *what* is thus a **HSlot** for (drizzle).

Like for the root, we consider that every **PSymbol** s has a set of so-called *hiders* in \mathbf{T} denoted $Hiders(s)$ that hide a **PSlot** s , and that the set of **HSlots** of a **PUT** $t \in \mathbf{T}$ is the set of **PSlots** that have a hider more general or equivalent to t :

Definition 2.7 (Hiders of a **PSymbol**, and **HSlots** of a **PUT**). The *hiders* of a **PSlot** is defined through a mapping $Hiders$ from $\mathbf{S}_{\mathcal{T}}$ to $2^{\downarrow root(s)} \setminus \emptyset$ that links a participation symbol s to the **PUTs** that have their **PSlot** s hidden. The set of **HSlots of PUTs** is defined through a mapping $\bar{\alpha}$ from \mathbf{T} to $2^{\mathbf{S}_{\mathcal{T}}}$ such that $\forall t \in \mathbf{T}, \bar{\alpha}(t) \stackrel{\text{def}}{=} \{s \in \mathbf{S}_{\mathcal{T}} \mid \exists t_h \in Hiders(s), t \lesssim t_h\}$

In our example, so that the **PSlot** *what* cannot be expressed for (drizzle), this **PUTs** must be more specific than one of the hiders of *what*. For instance, (rain) $\in Hiders(what)$. Furthermore, if we were to define another **PUT** more specific than (rain) such as (rain cats and dogs), then in every utterance in which a linguistic unit of type (rain cats and dogs) would be involved, the **PSlot** *what* is not actantial neither and is thus hidden. The hidden state of the **PSlot** *what* will actually be inherited by every **PUT** more specific than any **PUT** in $Hiders(what)$, and only those **PUTs** more specific than one **PUT** of $Hiders(what)$ will have **PSlot** *what* hidden⁵. The following property thus holds:

Proposition 2.5. Let $\downarrow Hiders(s)$ be the smallest lower set of \mathbf{T} that contains $Hiders(s)$, i.e., $\downarrow Hiders(s) \stackrel{\text{def}}{=} \{t \in \mathbf{T} \mid \exists t_h \in Hiders(s) : t \lesssim t_h\}$. For any $s \in \mathbf{S}_{\mathcal{T}}$, $\{t \in \mathbf{T} \mid s \in \bar{\alpha}(t)\} = \downarrow Hiders(s)$.

Proof. Let $s \in \mathbf{S}_{\mathcal{T}}$.

\subseteq : Let $t \in \{t \in \mathbf{T} \mid s \in \bar{\alpha}(t)\}$. $s \in \bar{\alpha}(t)$.

By definition 2.7, $\exists t_h \in Hiders(s), t \lesssim t_h$. Thus $t \in \downarrow Hiders(s)$.

\supseteq : Let $t \in \downarrow Hiders(s)$. $\exists t_h \in Hiders(s), t \lesssim t_h$

By definition 2.7, $s \in \bar{\alpha}(t)$. Thus $t \in \{t \in \mathbf{T} \mid s \in \bar{\alpha}(t)\}$. □

In our example illustrated on figure 3, the root of **PSymbol** *what* is illustrated by symbol \star , circles represent equivalence classes, and the hiders of *what* are illustrated by symbol \blacklozenge . The gray zone represents the set of **PUTs** that have a **HSlot** *what*, i.e., the lower set of \mathbf{T} generated by $Hiders(what)$.

As a direct consequence, as units get more and more specific (i.e., as we go down the hierarchy of **PUTs**), the set of **HSlots** may only increase. We say that **HSlots** are inherited:

Proposition 2.6. *Hidden slots are inherited, i.e.,*

$$\forall t_x, t_y \in \mathbf{T} \text{ such that } t_x \lesssim t_y, \bar{\alpha}(t_y) \subseteq \bar{\alpha}(t_x) \quad (2)$$

Moreover, if $t_x \simeq t_y$, then $\bar{\alpha}(t_y) = \bar{\alpha}(t_x)$.

⁵The good mathematical properties that this default behaviour brings do not limit the expressivity of our formalism. Indeed, we shall see later using unit type definitions that it is possible contravene this rule by inserting a new **PSlot** that corresponds to the **PSlot** hidden otherwise.

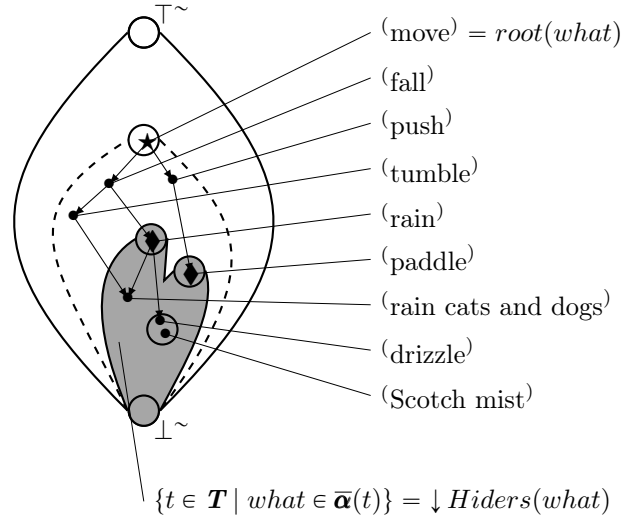


Figure 3: Illustration of the set of **PUTs** that have a **HSlot** *what*.

Proof. Let $t_x, t_y \in \mathbf{T}$ such that $t_x \lesssim t_y$, and $s \in \bar{\alpha}(t_y)$.

Definition 2.7 implies $\exists t_h \in Hiders(t_x), t_y \lesssim t_h$. So $t_x \lesssim t_y \lesssim t_h$ and $s \in \bar{\alpha}(t_x)$.

The second result is obtained using twice this first result. □

Second, as *Hiders* is taken in the lower set generated by $root(s)$, i.e., $2^{\downarrow root(s)} \setminus \emptyset$, any hiders is lower than $root(s)$. So the set of **HSlots** is always a subset of the set of **PSlots**:

Proposition 2.7. For any **PUT** $t \in \mathbf{T}$, $\bar{\alpha}(t)$ is a subset of $\omega(t)$.

Proof. Let $t \in \mathbf{T}$, and $s \in \bar{\alpha}(t)$.

From definition 2.7, $\exists t_h \in Hiders(s), t \lesssim t_h$.

Then as $Hiders(s) \in 2^{\downarrow root(s)} \setminus \emptyset$, we know that $t \lesssim t_h \lesssim root(s)$.

So by definition 2.6, $s \in \omega(t)$ and $\bar{\alpha}(t) \subseteq \omega(t)$. □

Finally, as *Hiders* is a mapping and cannot take value \emptyset , every **PSymbol** has at least one hider. Any minimal element of \mathbf{T} will thus inherit all of the **HSlots**:

Proposition 2.8. $\forall t \in \perp^{\sim}, \bar{\alpha}(t) = \mathbf{S}_{\mathcal{T}}$.

Proof. Let $s \in \mathbf{S}_{\mathcal{T}}$.

From definition 2.7 we know that $Hiders(s)$ is not empty and thus $\exists t_h \in Hiders(s)$.

As $\perp \lesssim t_h$, then $\perp \in \downarrow Hiders(s)$ and $s \in \bar{\alpha}(\perp)$.

Now, let $t \in \perp^{\sim}, t \lesssim \perp$ and using proposition 2.6, $\mathbf{S}_{\mathcal{T}} \subseteq \bar{\alpha}(t)$.

So for all $t \in \perp^{\sim}, \mathbf{S}_{\mathcal{T}} \subseteq \bar{\alpha}(t)$, and thus $\mathbf{S}_{\mathcal{T}} = \bar{\alpha}(t)$. □

2.4 PUT Actant Slots (ASlots)

The set of **Actant Slots (ASlots)** of a **PUT** is defined as the set of its participant slots that are not hidden:

Definition 2.8 (ASlots and valency of a PUT). The set of **ASlots of PUTs** is defined through a mapping α from \mathbf{T} to $2^{\mathcal{S}^\tau}$ such that $\forall t \in \mathbf{T}, \alpha(t) \stackrel{\text{def}}{=} \omega(t) - \bar{\alpha}(t)$. The number of **ASlots** of a **PUT** t is denoted the *valency* of t , i.e., $\text{valency}(t) \stackrel{\text{def}}{=} |\alpha(t)|$.

For instance, (paddle) inherits **PSlots** *who* and *what* from (push), but as **PSlots** *what* is hidden for (paddle), then it only has **ASlots** *who*.

The following proposition directly derives from definition 2.8:

Proposition 2.9. $\forall t \in \mathbf{T}, \omega(t) = \alpha(t) \cup \bar{\alpha}(t)$, i.e., $\omega(t)$ is the disjoint union of $\alpha(t)$ and $\bar{\alpha}(t)$.

Proof. From definition 2.8: $\forall t \in \mathbf{T}, \alpha(t) = \omega(t) - \bar{\alpha}(t) \Leftrightarrow \forall t \in \mathbf{T}, \omega(t) = \alpha(t) \cup \bar{\alpha}(t)$ □

Furthermore, due to propositions 2.2 and 2.5, we know that any **PUT** with **ASlot** *what* is in lower set $\downarrow \text{root}(\text{what})$ and is not in lower set $\downarrow \text{Hiders}(\text{what})$. So the following proposition holds:

Proposition 2.10. For any $s \in \mathcal{S}_\tau$,

$$\{t \in \mathbf{T} \mid s \in \alpha(t)\} = \downarrow \text{root}(s) - \downarrow \text{Hiders}(s) = \{t \in \mathbf{T} \mid t \lesssim \text{root}(s) \text{ and } \forall t_h \in \text{Hiders}(s), t \not\lesssim t_h\} \quad (3)$$

Proof. Let $s \in \mathcal{S}_\tau$.

\subseteq : Let $t \in \{t \in \mathbf{T} \mid s \in \alpha(t)\}$. $s \in \alpha(t)$.

By definition 2.8, $s \in \omega(t) - \bar{\alpha}(t)$.

By definition 2.6 and proposition 2.7, $s \lesssim \text{root}(s)$ and $\forall t_h \in \text{Hiders}(s), t \not\lesssim t_h$.

So $t \in \downarrow \text{root}(s) - \downarrow \text{Hiders}(s)$.

\supseteq : Let $t \in \downarrow \text{root}(s) - \downarrow \text{Hiders}(s)$.

$s \lesssim \text{root}(s)$ and $\forall t_h \in \text{Hiders}(s), t \not\lesssim t_h$.

By definition 2.6 and proposition 2.7, $s \in \omega(t)$ and $s \notin \bar{\alpha}(t)$.

So $s \in \alpha(t)$, and $t \in \{t \in \mathbf{T} \mid s \in \alpha(t)\}$. □

In our example illustrated on figure 4, the root of **PSymbol** *what* is illustrated by symbol \star , circles represent equivalence classes, and the hiders of *what* are illustrated by symbol \blacklozenge . The yellow zone represents the set of **PUTs** that have a **ASlot** *what*, i.e., the difference between the lower set of \mathbf{T} generated by $\text{root}(\text{what})$, and the lower set of \mathbf{T} generated by $\text{Hiders}(\text{what})$.

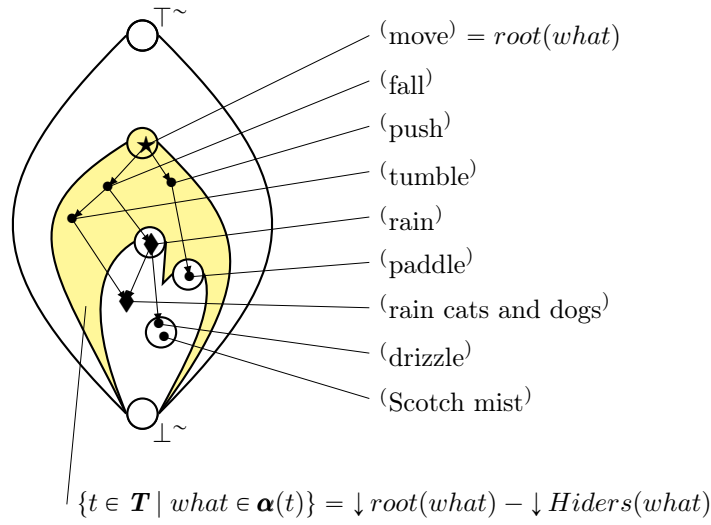


Figure 4: Illustration of the set of **PUT** that have **ASlots** *what*.

Thus in accordance with the **MTT**, **ASlots** are not always inherited by **PUTs**: they may be introduced, be inherited for a while, and disappear. Now from propositions 2.3 and 2.6, one concludes that two equivalent **PUTs** share the same set of **ASlots**:

Proposition 2.11. $\forall t, t' \in \mathbf{T}$ such that $t \simeq t'$, $\alpha(t) = \alpha(t')$.

Proof. From definition 2.8, $\alpha(t) = \omega(t) - \bar{\alpha}(t)$.

Moreover, from propositions 2.3 and 2.6, $\omega(t') = \omega(t)$ and $\bar{\alpha}(t') = \bar{\alpha}(t)$.

So $\omega(t') = \omega(t)$. □

Finally, from propositions 2.4 and 2.8, one concludes that any absurd **PUT** has no **ASlot**:

Proposition 2.12. $\forall t \in \perp\sim, \alpha(t) = \emptyset$.

Proof. Let $t \in \perp\sim$. From definition 2.8 and propositions 2.4 and 2.8, $\alpha(t) = \mathcal{S}_{\mathcal{T}} - \mathcal{S}_{\mathcal{T}} = \emptyset$. □

2.5 PUT Signatures

For any **PUT**, not any unit may fill one of its specific **ASlot**. For instance, any unit that fills **ASlot** *what* of a unit with type (move) should be of type (object), and absolutely not of type (idea) nor (eat). *Signatures* enable the assertion of what type fillers of **PSlots** are.

As **PUTs** get more and more specific, the signature of a given common **ASlot** may only become more and more specific. For instance, any unit that fills **ASlot** *what* of a unit with type (move) should be of type (object), but only units with type (weighing object) may fill **ASlots** *what* of a unit with type (fall).

Moreover, when a **PSlot** becomes hidden, its signature still is relevant and may be further restricted. For instance, although **PSlot** *what* is hidden for (rain), any unit that fills that **HSlot** should be of type (water drops). And for (drizzle) it becomes (small water drops).

Definition 2.9 (Signatures of **PUTs**). The set of signatures of **PUTs** $\{\mathfrak{s}_t\}_{t \in \mathbf{T}}$ is a set of functions from $\mathcal{S}_{\mathcal{T}}$ to $2^{\mathbf{T}}$. For every **PUT** t , \mathfrak{s}_t is a function with $\text{domain}(\mathfrak{s}_t) = \omega(t)$ that associates to each **PSlot** s of t a set of **PUT** $\mathfrak{s}_t(s)$ that characterize the type units that may fill this slot should be. The set of signatures $\{\mathfrak{s}_t\}_{t \in \mathbf{T}}$ must be such that for all $t_1, t_2 \in \mathbf{T}$, and $s \in \mathcal{S}_{\mathcal{T}}$ such that $t_1 \lesssim t_2$ and $s \in \omega(t_2)$, $\forall t'_2 \in \mathfrak{s}_{t_2}(s), \exists t'_1 \in \mathfrak{s}_{t_1}(s) : t'_1 \lesssim t'_2$.

The above definition is complex due to the fact that important notions will be introduced in next section. We will see that $2^{\mathbf{T}}$ is the set of so-called **CUTs**, and thus the signature \mathfrak{s}_t of t is a **CUT**. Moreover, when the pre-order relation $\hat{\lesssim}$ over the set of **CUTs** will be introduced, we will see that $\forall t'_2 \in \mathfrak{s}_{t_2}(s), \exists t'_1 \in \mathfrak{s}_{t_1}(s) : t'_1 \lesssim t'_2$ implies $\mathfrak{s}_{t_1}(s) \hat{\lesssim} \mathfrak{s}_{t_2}(s)$. Thus the set of signatures $\{\mathfrak{s}_t\}_{t \in \mathbf{T}}$ must be such that for all $t_1, t_2 \in \mathbf{T}$, and $s \in \mathcal{S}_{\mathcal{T}}$ such that $s \in \omega(t_2)$,

$$t_1 \lesssim t_2 \Rightarrow \mathfrak{s}_{t_1}(s) \hat{\lesssim} \mathfrak{s}_{t_2}(s) \quad (4)$$

3 Conjunctive Unit Types (CUTs)

In the **Meaning-Text Theory (MTT)**, a unit may actually have multiple types. For instance, a unit may have a lexical unit type and multiple grammatical unit types for instance, like $\{\lceil \text{def}^{\text{G}}, \lceil \text{USA}^{\text{L}} \rceil\}$ for "the USA".

In section 3.1 we hence introduce the set of **Conjunctive Unit Type (CUT)** as the set of all subsets of \mathbf{T} , and will define the set of asserted absurd CUT set, which is a set of combinations of **PUTs** that may not have instances.

Any **CUT** may also be seen both as a concept and as a n -ary relation, inheriting as such its constituting **PUTs Participant Slots (PSlots)**, and **Hidden Slots (HSlots)** (but not their **Actant Slots (ASlots)**!). Section 3.2 introduces slots and signatures for **CUTs**.

Then, section 3.3 introduces a specialization pre-order over **CUTs**, which is build from the pre-order over constituent **PUTs**, asserted absurd **CUTs** and absurd signatures. A natural definition of absurd **CUTs** is then given.

Finally, a first definition of the hierarchy of **CUTs** is introduced in 3.4.

3.1 Definition of CUTs

A unit has multiple types. For instance $\{\lceil \text{def}^{\text{G}}, \lceil \text{USA}^{\text{L}} \rceil\}$ for "the USA" is composed of a lexical unit type $\lceil \text{USA}^{\text{L}} \rceil$ and a grammatical unit type $\lceil \text{def}^{\text{G}} \rceil$. We hence introduce the set of possible **Conjunctive Unit Type (CUT)** which is the powerset of \mathbf{T} , i.e., the set of all subsets of \mathbf{T} :

Definition 3.1 (**CUT set over \mathbf{T}**). The **CUT set over \mathbf{T}** is the set of all subsets of \mathbf{T} , i.e., $\mathbf{T}^\cap \stackrel{\text{def}}{=} 2^{\mathbf{T}}$. Every element in \mathbf{T}^\cap is denoted both **CUT** and by the set itself. $\top^\cap = \{\top\}$ is denoted the *prime universal CUT*, and $\perp^\cap = \{\perp\}$ the *prime absurd CUT*. Any singleton **CUT** $\{t\}$ is said to be primitive, i.e., a primitive **CUT**.

Some **CUTs** such as $\{\lceil \text{def}^{\text{G}}, \lceil \text{indef}^{\text{G}} \rceil\}$ are said to be absurd. This means there is no unit that has both types $\lceil \text{def}^{\text{G}} \rceil$ and $\lceil \text{indef}^{\text{G}} \rceil$. Asserted absurd **CUT** set is defined as follows:

Definition 3.2 (**Asserted absurd CUT set**). The set of *asserted absurd CUTs* is a set of **CUTs** that is denoted $\perp_{\text{asserted}}^\cap$. And $\perp_{\text{asserted}}^\cap \subseteq \mathbf{T}^\cap$.

3.2 CUT Slots and Signatures

PSlots, HSlots, ASlots and signatures are naturally extended to **CUTs**.

For instance, consider a **CUT** $t^\cap = \{(\text{move}), (\text{quick})\}$. t^\cap contains the **PUT** (move) , and as $\text{what} \in \omega((\text{move}))$, what corresponds to a participant of $\text{SIT}(\text{move})$. As $\text{SIT}(\text{move}, \text{quick})$ is a specialization of $\text{SIT}(\text{move})$, we consider that what is also a **PSlot** of t^\cap :

Definition 3.3 (**Participant Slots of CUTs**). The set of **PSlots of CUTs** is defined through a mapping ω^\cap from \mathbf{T}^\cap to $2^{\mathcal{S}^\tau}$ Such that $\forall t^\cap \in \mathbf{T}^\cap, \omega^\cap(t^\cap) \stackrel{\text{def}}{=} \bigcup_{t \in t^\cap} \omega(t)$.

Now, consider **CUT** $t^\cap = \{(\text{move}), (\text{rain})\}$. **PSlot** what is hidden for (rain) , but is not for (move) . **PSlot** what is a **PSlot** for t^\cap and we consider that being hidden has priority over being an actant:

Definition 3.4 (**Hidden Slots of a CUTs**). The set of **HSlots of CUTs** is defined through a mapping $\bar{\alpha}^\cap$ from \mathbf{T}^\cap to $2^{\mathcal{S}^\tau}$ Such that $\forall t^\cap \in \mathbf{T}^\cap, \bar{\alpha}^\cap(t^\cap) \stackrel{\text{def}}{=} \bigcup_{t \in t^\cap} \bar{\alpha}(t)$.

Naturally, a **HSlot** is a hidden **PSlot**, so it is a **PSlot**. The following proposition underlines the fact that the previous definitions are coherent with this natural thought:

Proposition 3.1. For any **CUT** $t^\wedge \in \mathbf{T}^\wedge$, $\bar{\alpha}^\wedge(t^\wedge)$ is a subset of $\omega^\wedge(t^\wedge)$.

Proof. Let $t^\wedge \in \mathbf{T}^\wedge$ and $s \in \bar{\alpha}^\wedge(t^\wedge)$.

By definition 3.4, $\exists t \in t^\wedge, s \in \bar{\alpha}(t)$.

From proposition 2.7, $s \in \omega(t)$.

By definition 3.3, $s \in \omega^\wedge(t^\wedge)$. □

Finally, similarly as for **PUTs**, the **ASlots** of a **CUT** is defined as the difference between the set of **PSlots** and the set of **HSlots**:

Definition 3.5 (Actant Slots of a **CUTs**). The set of **ASlots** of **CUTs** is defined through a mapping α^\wedge from \mathbf{T}^\wedge to $2^{\mathcal{S}_T}$ Such that $\forall t^\wedge \in \mathbf{T}^\wedge, \alpha^\wedge(t^\wedge) \stackrel{\text{def}}{=} \omega^\wedge(t^\wedge) - \bar{\alpha}^\wedge(t^\wedge)$.

Just like for **PUTs**, any **PSlot** is either an actant, or hidden:

Proposition 3.2. $\forall t^\wedge \in \mathbf{T}^\wedge, \omega^\wedge(t^\wedge) = \alpha^\wedge(t^\wedge) \cup \bar{\alpha}^\wedge(t^\wedge)$, i.e., $\omega^\wedge(t^\wedge)$ is the disjoint union of $\alpha^\wedge(t^\wedge)$ and $\bar{\alpha}^\wedge(t^\wedge)$.

Proof. From proposition 4.6,

$\forall t^\wedge \in \mathbf{T}^\wedge, \alpha^\wedge(t^\wedge) = \omega^\wedge(t^\wedge) - \bar{\alpha}^\wedge(t^\wedge) \Leftrightarrow \forall t^\wedge \in \mathbf{T}^\wedge, \omega^\wedge(t^\wedge) = \alpha^\wedge(t^\wedge) \cup \bar{\alpha}^\wedge(t^\wedge)$. □

Signatures are also naturally extended to **CUTs**, but for the **PSlot** *what* for instance, we must take the union of signatures of **PUTs** constituents over a more complex set: the set of **PUTs** that have the **PSlot** *what*: $\{t \in t^\wedge \mid \text{what} \in \omega(t)\}$:

Definition 3.6 (Signature of a **CUT**). The set of signatures of **CUTs** $\{\varsigma_{t^\wedge}^\wedge\}_{t^\wedge \in \mathbf{T}^\wedge}$ is a set of functions from \mathcal{S}_T to \mathbf{T}^\wedge . For every **CUT** t^\wedge , $\varsigma_{t^\wedge}^\wedge$ is a function with $\text{domain}(\varsigma_{t^\wedge}^\wedge) = \omega^\wedge(t^\wedge)$ such that for all s in $\omega^\wedge(t^\wedge)$, $\varsigma_{t^\wedge}^\wedge(s) \stackrel{\text{def}}{=} \bigcup_{t \in t^\wedge \mid s \in \omega(t)} \varsigma_t(s)$.

Straightforwardly, the only **PSlots** of a singleton **CUT** are the **PSlots** of its single **PUT** element:

Proposition 3.3. For all $t \in \mathbf{T}$, all of the following is true:

- $\omega^\wedge(\{t\}) = \omega(t)$.
- $\bar{\alpha}^\wedge(\{t\}) = \bar{\alpha}(t)$.
- $\alpha^\wedge(\{t\}) = \alpha(t)$.
- for all $s \in \omega(t)$, $\varsigma_{\{t\}}^\wedge(s) = \varsigma_t(s)$

Proof. Let $t \in \mathbf{T}$.

1) $\omega^\wedge(\{t\}) = \bigcup_{t \in \{t\}} \omega(t) = \omega(t)$;

2) $\bar{\alpha}^\wedge(\{t\}) = \bigcup_{t \in \{t\}} \bar{\alpha}(t) = \bar{\alpha}(t)$;

3) $\alpha^\wedge(\{t\}) = \omega^\wedge(\{t\}) - \bar{\alpha}^\wedge(\{t\}) = \omega(t) - \bar{\alpha}(t) = \alpha(t)$.

4) let $s \in \omega(t)$. $t \in \{t \in t^\wedge \mid s \in \omega(t)\}$, so $\varsigma_{\{t\}}^\wedge(s) = \bigcup_{t \in \{t\} \mid s \in \omega(t)} \varsigma_t(s) = \varsigma_t(s)$. □

3.3 Pre-order over CUTs

In this section we define a Pre-order over \mathbf{T}^\cap which models a specialization relation, e.g., $\{(\text{happy}), (\text{cat})\} \lesssim \{(\text{hasMood}), (\text{animal})\}$ means that $\{(\text{happy}), (\text{cat})\}$ is more specific (semantically) than $\{(\text{hasMood}), (\text{animal})\}$.

Let us first introduce the definition of an iterative construction process of a comparisons set over \mathbf{T}^\cap , and we will describe the intuitive meaning of every element of this definition just after:

Definition 3.7 (Comparisons set over \mathbf{T}^\cap). Let $(\mathbf{C}_n^\cap)_{n \in \mathbb{N}}$ be a sequence of comparisons sets over \mathbf{T}^\cap , i.e., for all $n \in \mathbb{N}$, $\mathbf{C}_n^\cap \subseteq \mathbf{T}^{\cap 2}$, defined by

- for $i = 0$, $\mathbf{C}_0^\cap \stackrel{\text{def}}{=} \mathbf{C}_{\lesssim}^\cap \cup \mathbf{C}_\top^\cap \cup \mathbf{C}_\perp^\cap$, where:

$$\mathbf{C}_{\lesssim}^\cap \stackrel{\text{def}}{=} \{(t_y^\cap, t_x^\cap) \in \mathbf{T}^{\cap 2} \mid \forall t_y \in t_y^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t_y\} \quad (5)$$

$$\mathbf{C}_\top^\cap \stackrel{\text{def}}{=} \{(\top^\cap, \emptyset)\} \quad (6)$$

$$\mathbf{C}_\perp^\cap \stackrel{\text{def}}{=} \{(\perp^\cap, t^\cap) \in \mathbf{T}^{\cap 2} \mid t^\cap \in \perp_{\text{asserted}}^\cap\} \quad (7)$$

- for all $i > 0$, $\mathbf{C}_i^\cap \stackrel{\text{def}}{=} \mathbf{C}_{i-1}^\cap \cup \mathbf{C}_{\mathfrak{s}_i}^\cap \cup \mathbf{C}_{+i}^\cap$ where:

$$\mathbf{C}_{\mathfrak{s}_i}^\cap \stackrel{\text{def}}{=} \{(\perp^\cap, t^\cap) \in \mathbf{T}^{\cap 2} \mid \exists s \in \omega^\cap(t^\cap), (\perp^\cap, \mathfrak{s}_i^\cap(s)) \in \mathbf{C}_{i-1}^\cap\} \quad (8)$$

$$\mathbf{C}_{+i}^\cap \stackrel{\text{def}}{=} \{(t_z^\cap, t_x^\cap) \in \mathbf{T}^{\cap 2} \mid \exists t_y^\cap \in \mathbf{T}^\cap, (t_z^\cap, t_y^\cap) \in \mathbf{C}_{i-1}^\cap \text{ and } (t_y^\cap, t_x^\cap) \in \mathbf{C}_{i-1}^\cap\} \quad (9)$$

The sequence (\mathbf{C}_n^\cap) is a bounded monotonic increasing sequence, i.e., for all $n \in \mathbb{N}$, $\mathbf{C}_n^\cap \subseteq \mathbf{C}_{n+1}^\cap \subseteq \mathbf{T}^{\cap 2}$, so it is convergent. The least upper bound of the sequence $(\mathbf{C}_n^\cap)_{n \in \mathbb{N}}$ is denoted \mathbf{C}^\cap : the comparisons set over \mathbf{T}^\cap .

The intuitive meaning of the iterative construction process is the following:

- $\mathbf{C}_{\lesssim}^\cap$ represents the natural extension of a pre-order over a set to a pre-order over its powerset;
- \mathbf{C}_\top^\cap is introduced to flatten the top of the powerset so that \top^\cap is the most generic **CUT**;
- \mathbf{C}_\perp^\cap is introduced to flatten the bottom of the powerset so that any asserted absurd **CUT** is considered as a minimal **CUT** (a most specific **CUT**);
- $\mathbf{C}_{\mathfrak{s}_i}^\cap$ represents the fact that if a signature of a **CUT** for a given **PSlot** is absurd, then that **CUT** is absurd;
- \mathbf{C}_{+i}^\cap transitively closes the set of comparisons \mathbf{C}^\cap .

It is a well know result that $\mathbf{C}_{\lesssim}^\cap$ defines a pre-order over $2^{\mathbf{T}}$ (i.e., it is reflexive and transitive), let us recall how this result is obtained:

Proposition 3.4. $\mathbf{C}_{\lesssim}^\cap$ is reflexive and transitive.

Proof. Reflexivity: Let $t^\cap \in \mathbf{T}^\cap$.

$\forall t \in t^\cap, \exists t' (= t) \in t^\cap : t' \lesssim t$.

So $(t^\cap, t^\cap) \in \mathbf{C}_{\lesssim}^\cap$.

Transitivity: Let $t_z^\cap, t_y^\cap, t_x^\cap \in \mathbf{T}^\cap$ such that $(t_z^\cap, t_y^\cap) \in \mathbf{C}_{\lesssim}^\cap$, and $(t_y^\cap, t_x^\cap) \in \mathbf{C}_{\lesssim}^\cap$.

$\forall t_z \in t_z^\cap, \exists t_y \in t_y^\cap : t_y \lesssim t_z$, and $\forall t'_y \in t_y^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t'_y$.

Let $t'_y = t_y$, then $\forall t_z \in t_z^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t_z$,

and thus $(t_z^\cap, t_x^\cap) \in \mathbf{C}_{\lesssim}^\cap$. □

Now from that result and the construction process of \mathbf{C}^\cap , we may prove the reflexivity and the transitivity of \mathbf{C}^\cap :

Proposition 3.5. \mathbf{C}^\cap is reflexive and transitive.

Proof. As $\mathbf{C}_{\lesssim}^\cap \subseteq \mathbf{C}^\cap$ and $\mathbf{C}_{\lesssim}^\cap$ is reflexive, then \mathbf{C}^\cap is reflexive.

Transitivity is obtained from \mathbf{C}_+^\cap in the construction process of \mathbf{C}^\cap . \square

Now that we know \mathbf{C}^\cap is reflexive and transitive, it defines a pre-order relation over \mathbf{T}^\cap :

Definition 3.8 (Pre-order relation over \mathbf{T}^\cap). \mathbf{C}^\cap is a binary relation over \mathbf{T}^\cap which is transitive and reflexive. So it defines a pre-order relation $\hat{\lesssim}$ over \mathbf{T}^\cap , i.e., $t_x^\cap \hat{\lesssim} t_y^\cap$ if and only if $(t_y^\cap, t_x^\cap) \in \mathbf{C}^\cap$.

The following proposition underlines sufficient conditions so that a **CUT** is more specific than another **CUT**. Each of these conditions is the expression of one of the intuitive meaning of the elements in the construction process of the comparisons set over \mathbf{T}^\cap :

Proposition 3.6. The pre-order relation $\hat{\lesssim}$ is such that:

- **Extension of \lesssim** For all $t_x^\cap, t_y^\cap \in \mathbf{T}^\cap$, if $\forall t_y \in t_y^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t_y$, then $t_x^\cap \hat{\lesssim} t_y^\cap$;
- **Top CUT** $\emptyset \hat{\lesssim} \top^\cap$;
- **Asserted absurd types** For all $t^\cap \in \perp_{\text{asserted}}^\cap$, $t^\cap \hat{\lesssim} \perp^\cap$;
- **Absurd signatures** For all $t^\cap \in \mathbf{T}^\cap$, if there exists $s \in \omega^\cap(t^\cap)$ such that $\varsigma_{t^\cap}^\cap(s) \hat{\lesssim} \perp^\cap$, then $t^\cap \hat{\lesssim} \perp^\cap$;

Proof. **Extension of \lesssim :** For all $t_x^\cap, t_y^\cap \in \mathbf{T}^\cap$, if $\forall t_y \in t_y^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t_y$, then $(t_y^\cap, t_x^\cap) \in \mathbf{C}_{\lesssim}^\cap \subseteq \mathbf{C}^\cap$, so $t_x^\cap \hat{\lesssim} t_y^\cap$.

Top CUT: We know that $(\top^\cap, \emptyset) \in \mathbf{C}^\cap$, so $\emptyset \hat{\lesssim} \top^\cap$.

Asserted absurd types: For all $t^\cap \in \perp_{\text{asserted}}^\cap$, $(\perp^\cap, t^\cap) \in \mathbf{C}_\perp^\cap \subseteq \mathbf{C}^\cap$, so $t^\cap \hat{\lesssim} \perp^\cap$.

Absurd signatures: For all $t^\cap \in \mathbf{T}^\cap$, if there exists $s \in \omega^\cap(t^\cap)$ such that $\varsigma_{t^\cap}^\cap(s) \hat{\lesssim} \perp^\cap$, then $(\perp^\cap, \varsigma_{t^\cap}^\cap(s)) \in \mathbf{C}^\cap$. There exists $n \in \mathbb{N}$ such that $(\perp^\cap, \varsigma_{t^\cap}^\cap(s)) \in \mathbf{C}_n^\cap$, and thus for $n+1$, $(\perp^\cap, t^\cap) \in \mathbf{C}_{n+1}^\cap \subseteq \mathbf{C}^\cap$. So $t^\cap \hat{\lesssim} \perp^\cap$. \square

In the mathematical theory of categories, pre-ordered sets has category denoted **Ord** with monotonic functions as homomorphisms. In our case here, we have two pre-ordered sets: the powerset of \mathbf{T}^\cap with the inclusion relation $(\mathbf{T}^\cap, \subseteq)$ and the powerset of \mathbf{T}^\cap with the specialization relation $(\mathbf{T}^\cap, \hat{\lesssim})$. The morphism f that maps one onto the other and such that $f(t^\cap) = t^\cap$ is anti-monotonic:

Proposition 3.7. The function $f : (\mathbf{T}^\cap, \subseteq) \rightarrow (\mathbf{T}^\cap, \hat{\lesssim})$ such that $f(t^\cap) = t^\cap$ is an anti-monotonic pre-order homomorphism, i.e., for all $t_x^\cap, t_y^\cap \in \mathbf{T}^\cap$, $t_x^\cap \subseteq t_y^\cap \Rightarrow t_y^\cap \hat{\lesssim} t_x^\cap$.

Proof. Let $t_x^\cap, t_y^\cap \in \mathbf{T}^\cap$ such that $t_x^\cap \subseteq t_y^\cap$. Then $\forall t_x \in t_x^\cap, t_x \in t_y^\cap$.

Thus $\forall t_x \in t_x^\cap, \exists t_y (= t_x) \in t_y^\cap : t_y \lesssim t_x$, so by proposition 3.6, $t_y^\cap \hat{\lesssim} t_x^\cap$. \square

The set of comparisons \mathbf{C}^\cap is constructed such that the top (the less specific **CUTs**) and the bottom (the most specific **CUTs**) of the pre-ordered set $(\mathbf{T}^\cap, \hat{\lesssim})$ is flattened. The following proposition underlines the important maximal and minimal elements of $(\mathbf{T}^\cap, \hat{\lesssim})$.

Proposition 3.8. \top^\cap and \emptyset are maximal elements in \mathbf{T}^\cap , and \mathbf{T} , \perp^\cap , and \perp^\sim are minimal elements in $(\mathbf{T}^\cap, \hat{\lesssim})$.

Proof. Let $t^\circ \in \mathbf{T}^\circ$.

\emptyset : $\emptyset \subseteq t^\circ$, so using proposition 3.7: $t^\circ \lesssim \emptyset$.

\top° : Let $t^\circ \neq \emptyset$. We know that for all $t \in \mathbf{T}$, $t \lesssim \top$, so for \top in \top° , $\exists t \in t^\circ : t \lesssim \top$.

So using proposition 3.6, $t^\circ \lesssim \top^\circ$.

As we moreover know that $\emptyset \lesssim \top^\circ$, then for all $t^\circ \in \mathbf{T}^\circ$, $t^\circ \lesssim \top^\circ$.

\mathbf{T} : $t^\circ \subseteq \mathbf{T}$, so using proposition 3.7: $\mathbf{T} \lesssim t^\circ$.

\perp° : Let $t^\circ \neq \emptyset$. We know that for all $t \in \mathbf{T}$, $\perp \lesssim t$, so for all $t \in t^\circ$, $\perp \lesssim t$.

So using proposition 3.6, $\perp^\circ \lesssim t^\circ$.

As we moreover know that \emptyset is a maximal element, $\perp^\circ \lesssim \emptyset$, then for all $t^\circ \in \mathbf{T}^\circ$, $\perp^\circ \lesssim t^\circ$.

\perp^\sim : We know that $\perp \in \perp^\sim$, so item above also applies for \perp^\sim , i.e.,

for $t^\circ \neq \emptyset$, $\forall t \in t^\circ, \exists t' (= \perp) \in \perp^\sim : t' \lesssim t$, and $\perp^\sim \lesssim \emptyset$

So for all $t^\circ \in \mathbf{T}^\circ$, $\perp^\sim \lesssim t^\circ$. □

We introduce the natural equivalence relation \simeq defined by $t_x^\circ \simeq t_y^\circ \Leftrightarrow t_x^\circ \lesssim t_y^\circ$ and $t_y^\circ \lesssim t_x^\circ$. Elements in the flattened bottom of the pre-ordered set $(\mathbf{T}^\circ, \lesssim)$ are not only the most specific **CUTs**, but they all are considered absurd, and may not have instances.

Definition 3.9 (Absurd **CUT** set). The set of absurd **CUTs** is denoted \perp^\square and is the set: $\perp^\square \stackrel{\text{def}}{=} \{t^\circ \in \mathbf{T}^\circ \mid t^\circ \simeq \perp^\circ\}$.

Naturally, the prime absurd **CUT**, \perp° , is absurd.

The property of being absurd is hereditary: if a **CUT** t° is absurd, all conjunctive types lesser than t° are also absurd. For instance if $\{\text{def}^{\text{G}}, \text{indef}^{\text{G}}\}$ is asserted to be absurd and $\text{USA}^{\text{L}} \lesssim \text{def}^{\text{G}}$, then $\{\text{USA}^{\text{L}}, \text{indef}^{\text{G}}\}$ is absurd.

What is not automatic for **PUT**, is the fact that if a **PUT** has an absurd signature, then it is absurd. Consider for instance that one wants to define $\langle \text{rain-hail} \rangle$ as being a specialization of both $\langle \text{rain} \rangle$ and $\langle \text{hail} \rangle$. From definition 2.9, $\langle \text{rain-hail} \rangle$ has a **PSlot** *what*, whose signature is the union of those of $\langle \text{rain} \rangle$ and $\langle \text{hail} \rangle$:

$$\begin{aligned} \mathfrak{S}_{\langle \text{rain-hail} \rangle}(\text{what}) &= \mathfrak{S}_{\langle \text{rain} \rangle}(\text{what}) \cup \mathfrak{S}_{\langle \text{hail} \rangle}(\text{what}) \\ &= \{\langle \text{water} \rangle, \langle \text{liquid} \rangle\} \cup \{\langle \text{water} \rangle, \langle \text{solid} \rangle\} \\ &= \{\langle \text{water} \rangle, \langle \text{liquid} \rangle, \langle \text{solid} \rangle\} \end{aligned}$$

If one asserted that $\{\langle \text{liquid} \rangle, \langle \text{solid} \rangle\}$ is absurd and as being absurd is hereditary, the signature of **PUT** $\langle \text{rain-hail} \rangle$ is absurd. Yet, **PUT** is not automatically absurd. Equation 8 in the definition of the pre-order \lesssim enables **PUT** $\{\langle \text{rain-hail} \rangle\}$ to be absurd as it should be.

Finally, if one asserts that $\top^\circ \in \perp_{\text{asserted}}^\square$, then the whole hierarchy collapses and $\perp^\square = \mathbf{T}^\circ$. Same goes if $t^\circ \in \mathbf{T}^\circ$ such that $\top^\circ \lesssim t^\circ$, and $s \in \omega^\circ(t^\circ)$ such that $\mathfrak{S}_{t^\circ}^\circ(s) \lesssim \perp^\circ$. Then from proposition 3.6, $t^\circ \lesssim \perp^\circ$, and by the pre-order transitivity, $\perp^\square = \mathbf{T}^\circ$. Such situations must absolutely be avoided.

3.4 CUTs Hierarchy

We are now ready to introduce the **CUT** hierarchy, core of the **UGs** mathematical framework.

Definition 3.10 (CUT hierarchy). A **CUT hierarchy**

$\mathcal{T}^\cap = (T_{\text{declared}}, C_{\text{asserted}}, \perp_{\text{asserted}}^\cap, \mathcal{S}_{\mathcal{T}}, \text{root}, \text{Hiders}, \{\mathfrak{s}_t\}_{t \in \mathcal{T}})$, is composed of:

- T_{declared} a set of declared **PUTs**;
- C_{asserted} a set of asserted comparisons;
- $\perp_{\text{asserted}}^\cap$ a set of asserted absurd **CUTs**.
- $\mathcal{S}_{\mathcal{T}}$ a set of participation symbols;
- root a mapping that associates to each **PSlot** its root **PUT**
- Hiders a mapping that associates to each **PSlot** the set of its hider **PUTs**;
- $\{\mathfrak{s}_t\}_{t \in \mathcal{T}}$ the set of signatures of **PUTs**;

The **CUTs** hierarchy is the minimal set of mathematical objects that is necessary to form the consistent core of the **UG** formalism.

4 Characterizing CUTs

This section introduces remarkable aspects of the **CUTs**.

First, for two **CUTs** $t_x^\cap, t_y^\cap \in \mathbf{T}^\cap$ and according to the construction of the pre-order over **CUTs**, we introduce in §4.1 the necessary condition to have $t_x^\cap \lesssim t_y^\cap$.

Second, now that the pre-order \lesssim is introduced, we will see in §4.2 that most properties of **PUTs** slots and signatures are valid for **CUTs** slots and signatures, except for some deteriorated cases.

Next, §4.3 introduces a natural equivalence relation over the set of **CUTs**, and the partially ordered set of equivalent classes of **CUTs**.

Finally, §4.4 and 4.5 are devoted to the introduction of remarkable subsets of **CUTs** that will be important later on for the **UGs** mathematical framework.

4.1 Necessary Conditions to Compare Two CUTs

The pre-order \lesssim over **CUTs** is slightly more complex than the simple natural extension of a pre-order over a set to a pre-order over its powerset. The necessary condition to have two **CUTs** $t_x^\cap, t_y^\cap \in \mathbf{T}^\cap$ comparable may be stated as follows:

If $t_x^\cap \lesssim t_y^\cap$, then at least one of the following is true:

- t_x^\cap is absurd;
- t_x^\cap is the empty set (and thus t_y^\cap is a maximal element of \mathbf{T}^\cap);
- t_x^\cap is naturally more specific than t_y^\cap , i.e., according to the natural extension of a pre-order over a set to a pre-order over its powerset.

Or more formally:

Proposition 4.1. *Let $t_x^\cap, t_y^\cap \in \mathbf{T}^\cap$. if $t_x^\cap \lesssim t_y^\cap$, then at least one of the following is true:*

- $t_x^\cap \cong \perp^\cap$;
- $t_x^\cap = \emptyset$ (and thus $\top^\cap \cong t_y^\cap$);
- $\forall t_y \in t_y^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t_y$ (i.e., $(t_y^\cap, t_x^\cap) \in \mathbf{C}_{\lesssim}^\cap$).

Proof. Let $t_x^\cap, t_y^\cap \in \mathbf{T}^\cap$.

As \top^\cap (resp. \perp^\cap) is a maximal (resp. minimal) element of \mathbf{T}^\cap ,

$\top^\cap \lesssim t_y^\cap$ implies $\top^\cap \cong t_y^\cap$ (resp. $t_x^\cap \lesssim \perp^\cap$ implies $t_x^\cap \cong \perp^\cap$).

So it is sufficient to prove that one of the following is true:

i) $t_x^\cap \lesssim \perp^\cap$, ii) $t_x^\cap = \emptyset$ (and thus $\top^\cap \lesssim t_y^\cap$), or iii) $(t_y^\cap, t_x^\cap) \in \mathbf{C}_{\lesssim}^\cap$.

\mathbf{C}^\cap is the limit the sequence (\mathbf{C}_n^\cap) , so let $n \in \mathbb{N}$ be such that $(t_y^\cap, t_x^\cap) \in \mathbf{C}_n^\cap$ and $(t_y^\cap, t_x^\cap) \notin \mathbf{C}_{n-1}^\cap$.

We assume that $t_x^\cap \not\lesssim \perp^\cap$, $(t_y^\cap, t_x^\cap) \notin \mathbf{C}_{\lesssim}^\cap$, and that either $\top^\cap \not\lesssim t_y^\cap$ or $t_x^\cap \neq \emptyset$.

1) If $(t_y^\cap, t_x^\cap) \in \mathbf{C}_{\leq}^\cap$, then $t_x^\cap \lesssim \perp^\cap$.

This is impossible, so $(t_y^\cap, t_x^\cap) \in \mathbf{C}_{>}^\cap$ and there exists $t^\cap \in \mathbf{T}^\cap$ such that $(t_y^\cap, t^\cap) \in \mathbf{C}_{n-1}^\cap$ and $(t^\cap, t_x^\cap) \in \mathbf{C}_{n-1}^\cap$.

As $\mathbf{C}_{n-1}^\cap \subseteq \mathbf{C}^\cap$, $t_x^\cap \lesssim t^\cap$.

If $t^\cap \lesssim \perp^\cap$, then $t_x^\cap \lesssim \perp^\cap$ which is impossible.

So we know that $t^\cap \not\lesssim \perp^\cap$.

2) From proposition 3.4, we know that $\mathcal{C}_{\approx}^{\circ}$ is transitive. So either 2a) $(t^{\circ}, t_x^{\circ}) \notin \mathcal{C}_{\approx}^{\circ}$, or 2b) $(t_y^{\circ}, t^{\circ}) \notin \mathcal{C}_{\approx}^{\circ}$.

2a) In this case, consider $m \leq n - 1$ such that $(t^{\circ}, t_x^{\circ}) \in \mathcal{C}_m^{\circ}$ and $(t^{\circ}, t_x^{\circ}) \notin \mathcal{C}_{m-1}^{\circ}$.

We still have $t_x^{\circ} \not\lesssim \perp^{\circ}$ and we have now $(t^{\circ}, t_x^{\circ}) \notin \mathcal{C}_{\approx}^{\circ}$.

2b) In this case, consider $m \leq n - 1$ such that $(t_y^{\circ}, t^{\circ}) \in \mathcal{C}_m^{\circ}$ and $(t_y^{\circ}, t^{\circ}) \notin \mathcal{C}_{m-1}^{\circ}$.

We just showed in 1) that $t^{\circ} \not\lesssim \perp^{\circ}$ and we have now $(t_y^{\circ}, t^{\circ}) \notin \mathcal{C}_{\approx}^{\circ}$.

3) In both cases 2a) and 2b), the possibility of our hypothesis depend on the possibility of our hypothesis with different CUTs, and with a strictly lower natural number $m < n$.

Thus for each exploratory branch, at some point we will reach the case where (for instance for t_a° and t_b°): $(t_b^{\circ}, t_a^{\circ}) \in \mathcal{C}_0^{\circ}$, $t_a^{\circ} \not\lesssim \perp^{\circ}$ so $(\perp^{\circ}, t_a^{\circ}) \notin \mathcal{C}_{\perp}^{\circ}$, and $(t_b^{\circ}, t_a^{\circ}) \notin \mathcal{C}_{\approx}^{\circ}$. So $(t_b^{\circ}, t_a^{\circ}) \in \mathcal{C}_{\top}^{\circ}$.

Neither $\top^{\circ} \not\lesssim t_b^{\circ}$ nor $t_b^{\circ} \neq \emptyset$ is compatible with this last statement, so we highlighted a contradiction.

As a conclusion, if $t_x^{\circ} \approx t_y^{\circ}$, it is impossible that all of the following is false :

i) $t_x^{\circ} \approx \perp^{\circ}$, ii) $t_x^{\circ} = \emptyset$, iii) $(t_y^{\circ}, t_x^{\circ}) \in \mathcal{C}_{\approx}^{\circ}$. □

4.2 Properties of CUT Slots and Signatures

Now that the pre-order \approx is introduced, we will see how the properties of PUTs slots and signatures are valid for CUTs slots and signatures except for specific deteriorated cases: the void CUT and absurd CUTs.

Participant Slots First, remember proposition 2.2 and the fact that the PSlot *what* is inherited by every PUT more specific than $root(what)$, and only those PUTs more specific than $root(what)$ have a PSlot with symbol *what*. From the definition of PSlots of CUTs, this property is also valid for CUTs:

If $t_x^{\circ} = \{\text{fall}, \text{past}\}$ and $t_y^{\circ} = \{\text{move}\}$, then $t_x^{\circ} \approx t_y^{\circ}$, and in every linguistic situation in which a linguistic unit of type t_x° would be involved, there would still be something that moves. Apart for some deteriorated cases, the PSlot *what* is inherited by every CUT more specific than the CUT $\{root(what)\}$, and only those CUTs more specific than $\{root(what)\}$ will have a PSlot with symbol *what*. The following property thus holds:

Proposition 4.2. Let $\downarrow^{\circ} root(s)$ be the smallest lower set of \mathbf{T}° that contains $\{root(s)\}$:

$$\downarrow^{\circ} root(s) \stackrel{\text{def}}{=} \{t^{\circ} \in \mathbf{T}^{\circ} \mid t^{\circ} \approx \{root(s)\}\} \quad (10)$$

For any $s \in \mathbf{S}_{\mathcal{T}}$, the following holds:

$$\{t^{\circ} \in \mathbf{T}^{\circ} \mid s \in \omega^{\circ}(t^{\circ})\} \cup \perp^{\circ} \setminus \emptyset = \downarrow^{\circ} root(s) \setminus \emptyset \quad (11)$$

Proof. Let $s \in \mathbf{S}_{\mathcal{T}}$.

\subseteq_1 : Let $t^{\circ} \in \{t^{\circ} \in \mathbf{T}^{\circ} \mid s \in \omega^{\circ}(t^{\circ})\}$.

By definition 3.3, $\exists t \in t^{\circ}, s \in \omega(t)$. So $t^{\circ} \neq \emptyset$.

On the other hand, from definition 2.6, $t \lesssim root(s)$.

By proposition 3.6, $t^{\circ} \approx \{root(s)\}$ so $t^{\circ} \in \downarrow^{\circ} root(s)$.

So $t^{\circ} \in \downarrow^{\circ} root(s) \setminus \emptyset$, and $\{t^{\circ} \in \mathbf{T}^{\circ} \mid s \in \omega^{\circ}(t^{\circ})\} \subseteq \downarrow^{\circ} root(s) \setminus \emptyset$.

\subseteq_2 : Let $t^{\circ} \in \perp^{\circ} \setminus \emptyset$. $t^{\circ} \approx \perp^{\circ}$ which is a minimum element in \mathbf{T}° .

So $t^{\circ} \approx \{root(s)\}$ and $t^{\circ} \in \downarrow^{\circ} root(s)$ and $t^{\circ} \in \downarrow^{\circ} root(s) \setminus \emptyset$.

So $\perp^{\circ} \setminus \emptyset \subseteq \downarrow^{\circ} root(s) \setminus \emptyset$.

\supseteq : Let $t^\cap \in \downarrow^\cap \text{root}(s) \setminus \emptyset$. $t^\cap \lesssim \{\text{root}(s)\}$.

We use proposition 4.1, so at least one of the following is true:

- i) $t^\cap \lesssim \perp^\cap$: then $t^\cap \in \perp^\cap$. And as $t^\cap \neq \emptyset$, $t^\cap \in \perp^\cap \setminus \emptyset$.
- ii) $t^\cap = \emptyset$: impossible case.
- iii) $\exists t \in t^\cap : t \lesssim \text{root}(s)$: then using definition 2.6, $\exists t \in t^\cap : s \in \omega(t)$, and by definition 3.3, $s \in \omega^\cap(t^\cap)$. So $t^\cap \in \{t^\cap \in \mathbf{T}^\cap \mid s \in \omega^\cap(t^\cap)\}$. \square

One may have noticed that two kinds of **CUTs** are not considered here: the **CUT** \emptyset , and any absurd **CUT** $t^\cap \in \perp^\cap$:

- the **CUT** \emptyset : a problem would arise for instance with $s \in \mathcal{S}_\mathcal{T}$ such that $\text{root}(s) = \top$. By construction of the pre-order \lesssim , $\emptyset \lesssim \top^\cap$, yet \emptyset has no **PSlot** ($\omega^\cap(\emptyset) = \bigcup_{t \in \emptyset} \omega(t) = \emptyset$). So $s \notin \omega^\cap(\emptyset)$ and some **PSlots** may not be inherited from the **CUT** \emptyset ;
- any absurd **CUT** $t^\cap \in \perp^\cap$: problems arise for any $s \in \mathcal{S}_\mathcal{T}$ such that $s \notin \omega^\cap(t^\cap)$. By construction of the pre-order \lesssim , $t^\cap \lesssim \perp^\cap$, yet, from propositions 2.4 and 3.3, $\omega^\cap(\perp^\cap) = \mathcal{S}_\mathcal{T}$. So some **PSlots** are not inherited among absurd **CUTs**.

In fact, these two deteriorated cases are not considered in all of the following properties of this section.

As **CUTs** get more and more specific (i.e., as we go down the hierarchy of **CUTs**), the set of **PSlots** may only increase. **PSlots** are inherited also for **CUTs**, as long as they are not the empty conjunctive unit type, or absurd types.

Proposition 4.3. *PSlots are inherited as long as CUTs are not absurd and are not \emptyset , i.e., $\forall t_x^\cap \in \mathbf{T}^\cap \setminus \emptyset - \perp^\cap$, and $t_y^\cap \in \mathbf{T}^\cap$,*

$$t_x^\cap \lesssim t_y^\cap \Rightarrow \omega^\cap(t_y^\cap) \subseteq \omega^\cap(t_x^\cap) \quad (12)$$

Proof. Let $t_x^\cap \in \mathbf{T}^\cap \setminus \emptyset - \perp^\cap$ and $t_y^\cap \in \mathbf{T}^\cap$ such that $t_x^\cap \lesssim t_y^\cap$.

Let $s \in \omega^\cap(t_y^\cap)$. By definition 3.3, $\exists t \in t_y^\cap, s \in \omega(t)$.

We use proposition 4.1, but as $t_x^\cap \notin \perp^\cap$ and $t_x^\cap \neq \emptyset$, $t_x^\cap \lesssim t_y^\cap$ simply implies: $\forall t_y \in t_y^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t$. So $\exists t_x \in t_x^\cap : t_x \lesssim t_y$.

Using proposition 2.3, $s \in \omega(t_x)$, and using definition 3.3, $s \in \omega^\cap(t_x^\cap)$. \square

Hidden Slots Next, consider the **CUT** $\{\text{rain}\}$. **PSlot** *what* is hidden for $\langle \text{rain} \rangle$, so is it for $\{\langle \text{rain} \rangle\}$.

Consider any **CUT** more specific than $\{\langle \text{rain} \rangle\}$ such as $\{\langle \text{rain} \rangle, \langle \text{past} \rangle\}$. $\{\langle \text{rain} \rangle, \langle \text{past} \rangle\}$ does not have *what* as an actant slot. The hidden state of the **PSlot** *what* will thus be inherited by every **CUT** more specific than any **CUT** in the set of singletons $\{\{Hiders(\text{what})\}\}$, and only those **CUTs** more specific than one **CUT** of $\{\{Hiders(\text{what})\}\}$ will have **PSlot** *what* hidden. With the exception, of course, of the **CUT** \emptyset , and any absurd **CUT** $t^\cap \in \perp^\cap$. The following property thus holds:

Proposition 4.4. *Let $\downarrow^\cap Hiders(s)$ be the lower set of \mathbf{T}^\cap generated by $\{\{t_h\}\}_{t_h \in Hiders(s)}$, i.e.,*

$$\downarrow^\cap Hiders(s) \stackrel{\text{def}}{=} \{t^\cap \in \mathbf{T}^\cap \mid \exists t_h \in Hiders(s) : t^\cap \lesssim \{t_h\}\} \quad (13)$$

For any $s \in \mathcal{S}_\mathcal{T}$, the following holds:

$$\{t^\cap \in \mathbf{T}^\cap \mid s \in \bar{\alpha}^\cap(t^\cap)\} \cup \perp^\cap \setminus \emptyset = \downarrow^\cap Hiders(s) \setminus \emptyset \quad (14)$$

Proof. Let $s \in \mathbf{S}_{\mathcal{T}}$.

\subseteq_1 : Let $t^\circ \in \{t^\circ \in \mathbf{T}^\circ \mid s \in \bar{\alpha}^\circ(t^\circ)\}$.

a) By definition 3.4, $\exists t \in t^\circ, s \in \bar{\alpha}(t)$. So $t^\circ \neq \emptyset$.

b) On the other hand, from definition 2.7, $\exists t_h \in \text{Hiders}(s), t \lesssim t_h$. Thus using proposition 3.6 $t^\circ \lesssim \{t_h\}$, and $t^\circ \in \downarrow^\circ \text{Hiders}(s)$.

So $t^\circ \in \downarrow^\circ \text{Hiders}(s) \setminus \emptyset$, and $\{t^\circ \in \mathbf{T}^\circ \mid s \in \bar{\alpha}^\circ(t^\circ)\} \subseteq \downarrow^\circ \text{Hiders}(s) \setminus \emptyset$.

\subseteq_2 : Let $t^\circ \in \perp^\circ \setminus \emptyset$.

$t^\circ \lesssim \perp^\circ$ which is a minimum element in \mathbf{T}° .

So $\forall t_h \in \text{Hiders}(s), t^\circ \lesssim \{t_h\}$, i.e., $t^\circ \in \downarrow^\circ \text{Hiders}(s)$.

As we chose that t° can't be \emptyset , then $t^\circ \in \downarrow^\circ \text{Hiders}(s) \setminus \emptyset$.

So $\perp^\circ \setminus \emptyset \subseteq \downarrow^\circ \text{Hiders}(s) \setminus \emptyset$.

\supseteq : Let $t^\circ \in \downarrow^\circ \text{Hiders}(s) \setminus \emptyset$.

$\exists t_h \in \text{Hiders}(s), t^\circ \lesssim \{t_h\}$.

We use proposition 4.1, so at least one of the following is true:

i) $t^\circ \cong \perp^\circ$: then $t^\circ \in \perp^\circ$ and $t^\circ \in \perp^\circ \setminus \emptyset$.

ii) $t^\circ = \emptyset$: impossible case.

iii) $\exists t \in t^\circ : t \lesssim t_h$: then by definition 2.7 $\exists t \in t^\circ : s \in \bar{\alpha}(t)$, and by definition 3.4, $s \in \bar{\alpha}^\circ(t^\circ)$.

So $t^\circ \in \{t^\circ \in \mathbf{T}^\circ \mid s \in \bar{\alpha}^\circ(t^\circ)\}$

So $t^\circ \in \{t^\circ \in \mathbf{T}^\circ \mid s \in \bar{\alpha}^\circ(t^\circ)\} \cup \perp^\circ \setminus \emptyset$. \square

As a direct consequence, as **CUTs** get more and more specific (i.e., as we go down the hierarchy of **CUTs**), the set of **HSlots** may only increase. **HSlots** are thus inherited also for **CUTs**, as long as they are not the empty conjunctive unit type, or absurd types:

Proposition 4.5. *Hidden slots are inherited as long as **CUTs** are not absurd and are not \emptyset , i.e., $\forall t_x^\circ \in \mathbf{T}^\circ \setminus \emptyset - \perp^\circ$ and $t_y^\circ \in \mathbf{T}^\circ$,*

$$t_x^\circ \lesssim t_y^\circ \Rightarrow \bar{\alpha}^\circ(t_x^\circ) \subseteq \bar{\alpha}^\circ(t_y^\circ) \quad (15)$$

Proof. Let $t_x^\circ \in \mathbf{T}^\circ \setminus \emptyset - \perp^\circ$ and $t_y^\circ \in \mathbf{T}^\circ$ such that $t_x^\circ \lesssim t_y^\circ$.

Let $s \in \bar{\alpha}^\circ(t_y^\circ)$. By definition 3.4, $\exists t \in t_y^\circ, s \in \bar{\alpha}(t)$.

From proposition 4.1 and the facts that $t_x^\circ \notin \perp^\circ$ and $t_x^\circ \neq \emptyset$, we know that $t_x^\circ \lesssim t_y^\circ$ simply implies $\forall t_y \in t_y^\circ, \exists t_x \in t_x^\circ : t_x \lesssim t_y$. So $\exists t_x \in t_x^\circ : t_x \lesssim t$.

Using proposition 2.6, $s \in \bar{\alpha}(t_x)$, and using definition 3.4, $s \in \bar{\alpha}^\circ(t_x^\circ)$. \square

Actant Slots Furthermore, due to propositions 4.2 and 4.4, we know that apart from absurd **CUTs** and \emptyset , any **CUT** with **ASlot** *what* is in lower set $\downarrow^\circ \text{root}(\text{what})$ and is not in lower set $\downarrow^\circ \text{Hiders}(\text{what})$. So the following proposition holds:

Proposition 4.6. *For any $s \in \mathbf{S}_{\mathcal{T}}$,*

$$\begin{aligned} & \{t^\circ \in \mathbf{T}^\circ \mid s \in \bar{\alpha}^\circ(t^\circ)\} \cup \perp^\circ \setminus \emptyset \\ &= (\downarrow^\circ \text{root}(s) \setminus \emptyset - \downarrow^\circ \text{Hiders}(s) \setminus \emptyset) \cup \perp^\circ \setminus \emptyset \\ &= \left\{ t^\circ \in \mathbf{T}^\circ \mid t^\circ \lesssim \{\text{root}(s)\} \text{ and } \forall t_h \in \text{Hiders}(s), t^\circ \not\lesssim \{t_h\} \right\} \cup \perp^\circ \setminus \emptyset \end{aligned}$$

Proof. We use propositions 2.10, 4.2 and 4.4:

$$\begin{aligned}
& \{t^\cap \in \mathbf{T}^\cap \mid s \in \boldsymbol{\alpha}^\cap(t^\cap)\} \cup \perp^\cap \setminus \emptyset \\
&= \{t^\cap \in \mathbf{T}^\cap \mid s \in \boldsymbol{\omega}^\cap(t^\cap) - \bar{\boldsymbol{\alpha}}^\cap(t^\cap)\} \cup \perp^\cap \setminus \emptyset \\
&= (\{t^\cap \in \mathbf{T}^\cap \mid s \in \boldsymbol{\omega}^\cap(t^\cap)\} - \{t^\cap \in \mathbf{T}^\cap \mid s \in \bar{\boldsymbol{\alpha}}^\cap(t^\cap)\}) \cup \perp^\cap \setminus \emptyset \\
&= ((\{t^\cap \in \mathbf{T}^\cap \mid s \in \boldsymbol{\omega}^\cap(t^\cap)\} \cup \perp^\cap \setminus \emptyset) - (\{t^\cap \in \mathbf{T}^\cap \mid s \in \bar{\boldsymbol{\alpha}}^\cap(t^\cap)\} \cup \perp^\cap \setminus \emptyset)) \cup \perp^\cap \setminus \emptyset \\
&= (\downarrow^\cap \text{root}(s) \setminus \emptyset - \downarrow^\cap \text{Hiders}(s) \setminus \emptyset) \cup \perp^\cap \setminus \emptyset
\end{aligned}$$

□

Signatures Just like for PUTs, as CUTs get more and more specific, the signature of a given common PSlot may only become more and more specific:

Proposition 4.7. For all $t_x^\cap \in \mathbf{T}^\cap \setminus \emptyset - \perp^\cap$, $t_y^\cap \in \mathbf{T}^\cap$ and $s \in \mathbf{S}_\mathcal{T}$ such that $s \in \boldsymbol{\omega}^\cap(t_y^\cap)$,

$$t_x^\cap \overset{\circ}{\lesssim} t_y^\cap \Rightarrow \boldsymbol{\varsigma}_{t_x^\cap}^\cap(s) \overset{\circ}{\lesssim} \boldsymbol{\varsigma}_{t_y^\cap}^\cap(s) \quad (16)$$

Proof. Let $t_x^\cap \in \mathbf{T}^\cap \setminus \emptyset - \perp^\cap$ and $t_y^\cap \in \mathbf{T}^\cap$ and $s \in \mathbf{S}_\mathcal{T}$ such that $t_x^\cap \overset{\circ}{\lesssim} t_y^\cap$ and $s \in \boldsymbol{\omega}^\cap(t_y^\cap)$.

1) Let $t_a \in \boldsymbol{\varsigma}_{t_y^\cap}^\cap(s)$. Using definition 3.6, $\exists t_b \in t_y^\cap : s \in \boldsymbol{\omega}(t_b)$ and $t_a \in \boldsymbol{\varsigma}_{t_b}(s)$.

2) We use $t_x^\cap \overset{\circ}{\lesssim} t_y^\cap$ and proposition 4.1.

As $t_x^\cap \notin \perp^\cap$ and $t_x^\cap \neq \emptyset$, we know that $\forall t_y \in t_y^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t_y$.

We restrict those t_y to be such that $s \in \boldsymbol{\omega}(t_y)$.

We use definitions 2.9 and proposition 3.6: $\forall t_y \in t_y^\cap$ such that $s \in \boldsymbol{\omega}(t_y)$, there exists $t_x \in t_x^\cap$ such that $\forall t' \in \boldsymbol{\varsigma}_{t_y}(s), \exists t \in \boldsymbol{\varsigma}_{t_x}(s) : t \lesssim t'$

3) This is rewritten as: $\forall t_y \in t_y^\cap$ such that $s \in \boldsymbol{\omega}(t_y), \forall t' \in \boldsymbol{\varsigma}_{t_y}(s), \exists t_x, t$ such that $t_x \in t_x^\cap, t \in \boldsymbol{\varsigma}_{t_x}(s)$ and $t \lesssim t'$

4) combining 1) and 3) with $t_y = t_b$, and $t' = t_a$, one obtains:

$\exists t_b \in t_y^\cap$ such that $s \in \boldsymbol{\omega}(t_b)$ and $t_a \in \boldsymbol{\varsigma}_{t_b}(s), \exists t_x, t$ such that $t_x \in t_x^\cap, t \in \boldsymbol{\varsigma}_{t_x}(s)$ and $t \lesssim t_a$

5) So there exists t and t_x such that $t_x \in t_x^\cap$ and $t \in \boldsymbol{\varsigma}_{t_x}(s)$ and $t \lesssim t_a$,

6) using definition 3.6, $t \in \boldsymbol{\varsigma}_{t_x^\cap}^\cap(s)$. So $\forall t_a \in \boldsymbol{\varsigma}_{t_y^\cap}^\cap(s), \exists t \in \boldsymbol{\varsigma}_{t_x^\cap}^\cap(s) : t \lesssim t_a$, and using proposition 3.6,

$\boldsymbol{\varsigma}_{t_x^\cap}^\cap(s) \overset{\circ}{\lesssim} \boldsymbol{\varsigma}_{t_y^\cap}^\cap(s)$. □

4.3 CUT Equivalence Class Sets

Let \cong be the natural *equivalence* relation defined by $t_x^\circ \cong t_y^\circ \Leftrightarrow (t_x^\circ \lesssim t_y^\circ \wedge t_y^\circ \lesssim t_x^\circ)$. The set of equivalence classes defines a partition of \mathbf{T}° . Let $t^\circ \in \mathbf{T}^\circ$, we denote $[t^\circ] \stackrel{\text{def}}{=} \{t_x^\circ \in \mathbf{T}^\circ \mid t_x^\circ \cong t^\circ\}$ the equivalence class to which t° belongs. We will usually use the notation t^\square for an unknown equivalence class.

Definition 4.1 (CUTs equivalence class set). The *CUTs equivalence class set* \mathbf{T}^\square is the quotient set of \mathbf{T}° by \cong , i.e., $\mathbf{T}^\square \stackrel{\text{def}}{=} \mathbf{T}^\circ / \cong = \{[t^\circ] \mid t^\circ \in \mathbf{T}^\circ\}$. We define a partial order \leq over \mathbf{T}^\square with $[t_x^\circ] \leq [t_y^\circ]$ if and only if $t_x^\circ \lesssim t_y^\circ$.

We define $\top^\square \stackrel{\text{def}}{=} [\top^\circ]$ the top (less specific) CUTs equivalence class set, and $\perp^\square \stackrel{\text{def}}{=} [\perp^\circ]$ the absurd (most specific) CUTs equivalence class set.

Proposition 4.8. \top^\square and \perp^\square are respectively the greatest element and the least element of \mathbf{T}^\square .

Proof. \top^\square : From proposition 3.8, \top° is a maximal element for \lesssim .

So $\forall t^\circ \in \mathbf{T}^\circ, t^\circ \lesssim \top^\circ$. So $\forall t^\square \in \mathbf{T}^\square, t^\square \leq \top^\square$, and \top^\square is the greatest element of \mathbf{T}^\square .

\perp^\square : From proposition 3.8, \perp° is a minimal element for \lesssim . So $\forall t^\circ \in \mathbf{T}^\circ, \perp^\circ \lesssim t^\circ$.

So $\forall t^\square \in \mathbf{T}^\square, \perp^\square \leq t^\square$, so \perp^\square is the least element of \mathbf{T}^\square . □

A set of equivalence class CUTs is closed under the union operation:

Proposition 4.9. Let $t^\square \in \mathbf{T}^\square$, and $t_x^\circ, t_y^\circ \in t^\square$. Then $t_x^\circ \cup t_y^\circ \in t^\square$.

Proof. Let $t^\square \in \mathbf{T}^\square$, and $t_x^\circ, t_y^\circ \in t^\square$. We will prove that $t_x^\circ \cup t_y^\circ \cong t_x^\circ$.

\lesssim : We know from proposition 3.7 that $t_x^\circ \subseteq t_x^\circ \cup t_y^\circ$ implies $t_x^\circ \cup t_y^\circ \lesssim t_x^\circ$.

\gtrsim : We know that $t_x^\circ \gtrsim t_y^\circ$. From proposition 3.6, at least one of the followings holds:

i) $t_x^\circ \lesssim \perp^\circ$: then $t_x^\circ \cup t_y^\circ \cong t_x^\circ \cong \perp^\circ$;

ii) $t_x^\circ = \emptyset$: then $t_x^\circ \cup t_y^\circ = t_y^\circ \cong t_x^\circ$;

iii) $\forall t_y \in t_y^\circ, \exists t_x \in t_x^\circ : t_x \lesssim t_y$: So $\forall t \in t_x^\circ \cup t_y^\circ, \exists t_x \in t_x^\circ : t_x \lesssim t$, and $t_x^\circ \gtrsim t_x^\circ \cup t_y^\circ$. □

The following lemma is useful to prove upcoming propositions. It states that adding a PUT t to a CUT t° does not make it change its equivalence class, provided that t is greater than at least one PUT in t° :

Lemma 4.10. Let $t^\circ \in \mathbf{T}^\circ$ and $t \in \mathbf{T}$ such that $\exists t' \in t^\circ, t' \lesssim t$. Then $t^\circ \cup \{t\} \cong t^\circ$.

Proof. Let $t^\circ \in \mathbf{T}^\circ$ and $t \in \mathbf{T}$ such that $\forall t' \in t^\circ, t' \lesssim t$.

\lesssim : $t_y^\circ \subseteq t_y^\circ \cup \{t\}$ so $t^\circ \cup \{t\} \lesssim t^\circ$;

\gtrsim : $t^\circ \neq \emptyset$. Let $t' \in t^\circ \cup \{t\}$.

If $t' \in t^\circ$, then there exists $t'' (= t') \in t^\circ : t'' \lesssim t'$.

If $t' = t$, then there exists $t'' \in t^\circ : t'' \lesssim t'$.

So $\forall t' \in t^\circ \cup \{t\}, \exists t'' \in t^\circ : t'' \lesssim t'$, and $t^\circ \cup \{t\} \gtrsim t^\circ$. □

4.4 Maximal CUTs

CUTs enable a first inference mechanism, which is type inference. If a **CUT** t_x^\cap is lower than t_y^\cap , then any instance of t_x^\cap is also an instance of every **PUT** in t_y^\cap . In this section we consider a remarkable subset of **CUTs** that we call the set of maximal **CUTs**.

Let us first introduce a maximization operator \uparrow that associates to any **CUT** t^\cap the union of all **CUTs** in $[t^\cap]$:

Definition 4.2 (Maximization operator). The maximization operator \uparrow defined on \mathbf{T}^\cap associates to every **CUT** t^\cap the **CUT** $\uparrow t^\cap \stackrel{\text{def}}{=} \bigcup_{t^\cap \in [t^\cap]} t^\cap$.

The set of maximal **CUTs** \mathbf{T}_{max}^\cap is then denoted \mathbf{T}_{max}^\cap , and defined as follows:

$$\mathbf{T}_{max}^\cap = \{\uparrow t^\cap \mid t^\cap \in \mathbf{T}^\cap\} \quad (17)$$

In the rest of this section we will list useful facts about the maximization operator \uparrow and the set of maximal **CUTs** \mathbf{T}_{max}^\cap .

First, the maximization operator preserves the equivalence class. This means that a **CUT** t^\cap and its maximized **CUT** $\uparrow t^\cap$ are both in the same equivalence class set.

Proposition 4.11. *The restriction of the closure operator \uparrow to any **CUTs** equivalence class set is an endomorphism of this class, i.e., for all $t^\cap \in \mathbf{T}^\cap$, if $t^\cap \in t^\cap$, then $\uparrow t^\cap \in t^\cap$. Or equivalently, $\forall t^\cap \in \mathbf{T}^\cap, \uparrow t^\cap \simeq t^\cap$.*

Proof. Let $t^\cap \in \mathbf{T}^\cap$, and $t^\cap \in t^\cap$. By definition, $\uparrow t^\cap$ is the union of all **CUT** in t^\cap .

We know from proposition 4.9 that t^\cap is closed under the union operation, so $\uparrow t^\cap \in t^\cap$. \square

Next, all **CUTs** of a given equivalence class set have the same maximized **CUTs**.

Proposition 4.12. *The kernel of the closure operator \uparrow on \mathbf{T} is the equivalence relation \simeq , i.e., if $t_x^\cap \simeq t_y^\cap$, then $\uparrow t_x^\cap = \uparrow t_y^\cap$*

Proof. By definition, if $t_x^\cap \simeq t_y^\cap$, then $[t_x^\cap] = [t_y^\cap]$, and $\uparrow t_x^\cap = \uparrow t_y^\cap$ \square

Thus, there is a one-to-one correspondence between the set of a set of equivalence **CUTs** class sets \mathbf{T}^\cap and the set of maximal **CUTs** \mathbf{T}_{max}^\cap . Moreover, **PUTs** is a maximal **CUT** in the sense that among its equivalence class set, it is the unique **CUT** that has the maximal cardinality.

Let $t^\cap \in \mathbf{T}^\cap$, we denote $\uparrow t^\cap$ the upper-set of \mathbf{T} generated by **PUTs** in t^\cap , i.e.,

$$\uparrow t^\cap \stackrel{\text{def}}{=} \{t \in \mathbf{T} \mid \exists t' \in t^\cap, t' \lesssim t\} \quad (18)$$

Proposition 4.13. *The maximization operator \uparrow is such that all of the following is true:*

- if $t^\cap \in \mathbf{T}^\cap \setminus \emptyset - \perp^\cap$, then $\uparrow t^\cap = \uparrow t^\cap$;
- if $t^\cap \in \perp^\cap$, then $\uparrow t^\cap = \mathbf{T}$;
- $\uparrow \emptyset = \uparrow \top^\cap$;

Proof. Let $t^\cap \in \mathbf{T}^\cap$:

1) if $t^\cap \in \mathbf{T}^\cap \setminus \emptyset - \perp^\cap$:

\subseteq : Let $t \in \uparrow t^\cap$. By definition, $\exists t_x^\cap \in [t^\cap], t \in t_x^\cap$. So $t^\cap \simeq t_x^\cap$ and $t^\cap \lesssim t_x^\cap$.

From proposition 3.6, at least one of the followings holds:

- i) $t^\cap \lesssim \perp^\cap$ (impossible from our hypothesis);
- ii) $t^\cap = \emptyset$ (impossible from our hypothesis);

- iii) $\forall t_x \in t_x^\square, \exists t' \in t^\square : t' \lesssim t_x$. So for $t \in t_x^\square, \exists t' \in t^\square : t' \lesssim t$. And thus $t \in \uparrow t^\square$.
 \supseteq : Let $t \in \uparrow t^\square$. Then $\exists t' \in t^\square, t' \lesssim t$. From lemma 4.10, $t^\square \cup \{t\} \cong t^\square$, and thus $t^\square \cup \{t\} \in [t^\square]$, and thus $t \in \uparrow t^\square$.
 2) if $t^\square \in \perp^\square$, then $\mathbf{T} \in [t^\square]$ and $\uparrow t^\square = \mathbf{T}$;
 3) if $\emptyset \cong \top^\square$, then from proposition 4.12, $\uparrow \emptyset = \uparrow \top^\square$. \square

Figure 5 illustrates the most interesting case in proposition 4.13 which is the case where $t^\square \notin \perp^\square$ and $t^\square \neq \emptyset$: black dots represent PUTs of $t^\square \in \mathbf{T}^\square$, and the fact that a dot is lower than another dot roughly means that the former is more specific than the latter. Circles represent equivalence classes. If t^\square is a CUT consisting of all the black dots, then the gray zone represents the set of PUTs that belong to $\uparrow t^\square$.

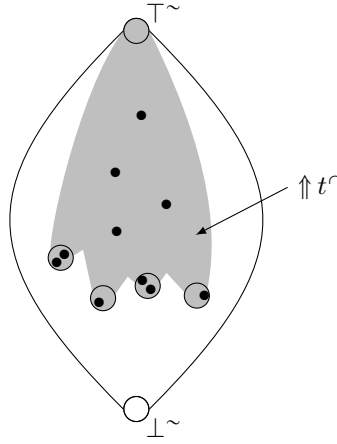


Figure 5: Illustration of a maximal CUT $\uparrow t^\square$ with $t^\square \in \mathbf{T}^\square \setminus \emptyset - \perp^\square$.

A CUT is thus maximal if and only if: either it is the whole set of PUTs, or it is a non-absurd non-empty upper set of PUTs:

Proposition 4.14. A CUT t^\square is maximal if and only if one of the following is true:

- $t^\square = \mathbf{T}$,
- $t^\square \in \mathbf{T}^\square \setminus \emptyset - \perp^\square$, and t^\square is a upper set of \mathbf{T} .

Proof. Let us prove both implications:

\Rightarrow : If t^\square is maximal, then there exists $t^{\square'} \in \mathbf{T}^\square, t^\square = \uparrow t^{\square'}$.

- i) if $t^{\square'} \in \perp^\square$, then $t^\square = \mathbf{T}$;
- ii) if $t^{\square'} = \emptyset$, then $t^\square = \uparrow \perp^\square$, and $\uparrow \perp^\square$ is either \mathbf{T} or a non-empty upper set of \mathbf{T} ;
- iii) if $t^{\square'} \in \mathbf{T}^\square \setminus \emptyset - \perp^\square$, then t^\square is an non-empty upper set of \mathbf{T} .

\Leftarrow :

- i) if $t^\square = \mathbf{T}$, then for all $t^{\square'} \in \perp^\square, t^\square = \uparrow t^{\square'}$ so t^\square is maximal;
- ii) if $t^\square \notin \perp^\square$, and t^\square is a non-empty upper set of \mathbf{T} , then using propositions 4.12 and 4.11, $\uparrow t^\square = t^\square$, so t^\square is maximal. \square

Remark. Anyways, as \mathbf{T} is an upper set of \mathbf{T} , every maximal CUT is an upper set of \mathbf{T} .

The last result of this section is the fact the operator \uparrow defines a closure operator on \mathbf{T} .

Proposition 4.15. \uparrow is a closure operator on \mathbf{T} , i.e., it satisfies the following conditions for all CUTs $t_x^\circ, t_y^\circ \in \mathbf{T}^\circ$:

- $t_x^\circ \subseteq \uparrow t_x^\circ$, (\uparrow is extensive);
- $t_x^\circ \subseteq t_y^\circ \implies \uparrow t_x^\circ \subseteq \uparrow t_y^\circ$, (\uparrow is increasing with respect to \subseteq);
- $\uparrow \uparrow t_x^\circ = \uparrow t_x^\circ$, (\uparrow is idempotent);

Proof. 1) *Extensive:* Let $t^\circ \in \mathbf{T}^\circ$. $t^\circ \in [t^\circ]$ so from definition 4.2, $t^\circ \subseteq \uparrow t^\circ$, and \uparrow is extensive.

2) *Increasing w.r.t. \subseteq :* Let $t_x^\circ, t_y^\circ \in \mathbf{T}^\circ$ such that $t_x^\circ \subseteq t_y^\circ$.

First, we know from proposition 3.7 that $t_y^\circ \lesssim t_x^\circ$.

i) If $t_y^\circ \in \perp^\circ$, then $\uparrow t_y^\circ = \mathbf{T}$ (prop. 4.13) and $\uparrow t_x^\circ \subseteq \uparrow t_y^\circ$.

ii) If $t_x^\circ \in \perp^\circ$, then so is t_y° , and $\uparrow t_x^\circ = \uparrow t_y^\circ = \mathbf{T}$ (prop. 4.13).

iii) If $t_y^\circ = \emptyset$, then $t_x^\circ = t_y^\circ = \emptyset$, and $\uparrow t_x^\circ = \uparrow t_y^\circ$.

iv) If $t_y^\circ \notin \perp^\circ$, $t_y^\circ \neq \emptyset$, $t_x^\circ \notin \perp^\circ$, and $t_x^\circ \neq \emptyset$: Let $t \in \uparrow t_x^\circ = \uparrow t_y^\circ$. We know that $\exists t' \in t_y^\circ, t' \lesssim t$. As $t_x^\circ \subseteq t_y^\circ$, then $\exists t' \in t_y^\circ, t' \lesssim t$, and $t' \in \uparrow t_y^\circ = \uparrow t_x^\circ$.

v) If $t_x^\circ = \emptyset$, then $\uparrow t_x^\circ = \uparrow \top^\circ$. \top is a maximal element of \mathbf{T} , so using lemma 4.10, $t_y^\circ \simeq t_y^\circ \cup \top^\circ$, and from definition 4.2, $\uparrow t_y^\circ = \uparrow t_y^\circ \cup \top^\circ$. Using items i) to iv), we know that $\uparrow t_x^\circ = \uparrow \top^\circ \subseteq \uparrow t_y^\circ \cup \top^\circ = \uparrow t_y^\circ$.

So $\uparrow t_x^\circ \subseteq \uparrow t_y^\circ$, and \uparrow is increasing.

3) *Idempotent:* Let $t^\circ \in \mathbf{T}^\circ$.

i) if $t^\circ \in \perp^\circ$, then $\uparrow t^\circ = \mathbf{T}$, and as we know that $\mathbf{T} \in \perp^\circ$, $\uparrow \uparrow t^\circ = \uparrow t^\circ = \mathbf{T}$.

ii) if $t^\circ \notin \perp^\circ$ and $t^\circ \neq \emptyset$, then:

\supseteq : We know from extensivity of \uparrow that $\uparrow t^\circ \subseteq \uparrow \uparrow t^\circ$.

\subseteq : Let $t \in \uparrow \uparrow t^\circ$. $\uparrow t^\circ \simeq t^\circ$ so (prop: 4.9 $\uparrow t^\circ \notin \perp^\circ$, and we also know that $t^\circ \subseteq \uparrow t^\circ$ so $\uparrow t^\circ \neq \emptyset$. Using upper-set properties: $\uparrow \uparrow t^\circ = \uparrow \uparrow t^\circ = \uparrow \uparrow t^\circ = \uparrow t^\circ = \uparrow t^\circ$.

iii) if $t^\circ = \emptyset$, then $\uparrow t^\circ = \uparrow \perp^\circ$, and using ii), $\uparrow t^\circ = \uparrow \perp^\circ = \uparrow \uparrow \perp^\circ = \uparrow \uparrow t^\circ$. \square

Thus a maximal CUTs t_\uparrow° is also maximal in the sense that any CUT of its equivalence class is contained in t_\uparrow° .

4.5 Concise CUTs

Concise **CUTs** will be of special interest in the following of this work. They are those (non-unique) **CUTs** that have a minimal cardinality in a given equivalence class:

Definition 4.3 (Concise **CUT**). Let t^\cap be a **CUT**. t^\cap is said to be *concise* if and only if:

- if t^\cap is absurd, then $t^\cap = \perp^\cap$.
- else, $\forall t \in t^\cap, \nexists t' \in t^\cap$ such that $t' < t$ and $\forall t, t' \in t^\cap, t \neq t'$.

Figure 6 below illustrates definition 4.3: black and white dots represent **PUTs**, and the fact that a dot is lower than another dot roughly means that the former is more specific than the latter. Circles represent equivalence classes. If t^\cap is a **CUT** consisting of all the black and white dots, then the set of black dots represents a concise **CUT** that is a subset of t^\cap . In the illustrated case, there are $2 \times 1 \times 2 \times 1$ such concise **CUT** that are subsets of t^\cap .

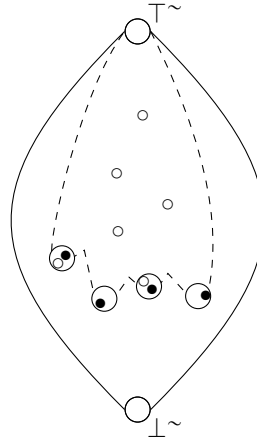


Figure 6: Let the union of black and white dots represent a **CUT** t^\cap . The set of black dots represents a concise **CUT** that is a subset of t^\cap .

5 Unit Graphs (UGs)

In previous sections we introduced the types that nodes of any **Unit Graph (UG)** have, and that specify through slots and signatures how these nodes must be linked to other nodes in the graph. Now is time to introduce the main objects of the Unit Graphs mathematical framework: the Unit Graphs, such as illustrated in figures 1 and 7.

Unit types specify how units must be linked to other nodes in the unit graph, but units may also be linked to other units through circumstantial dependencies. Circumstantial dependency symbols are first introduced in section 5.1.

Then section 5.2 introduce the **UGs**-support and the **UGs** themselves in section 5.2.

5.1 Circumstantial Dependency Symbols Hierarchy

Unit types specify how units must be linked to other nodes in the **Unit Graphs (UGs)**. As for any slot in a predicate, one **ASlot** of a unit may be filled by only one unit at a time. Now, in Dependency Grammars, units may be linked through binary relations of a very different nature: **Circumstantial Dependency Symbols (CSymbols)**. These binary relations are more of the kind instance-instance than predicate-argument. Such dependencies may be optional or multiple (i.e., a unit may govern zero or more other units through relations of a same type). Example of such relations are the **DSynRepresentation** relation **ATTR** as in figure 7.

We use symbols to distinguish the different kinds of circumstantial dependencies, and thus introduce a finite set of **CSymbols**.

Definition 5.1 (Circumstantial Dependency Symbols Set). A **CSymbols** set is a set of binary relation symbols denoted \mathcal{S}_C .

CSymbols are often classified in sets and subsets, we thus take into account this need classification and hierarchy, and introduce a partial order over the set of **CSymbols**.

Definition 5.2 (Pre-order over \mathcal{S}_C). The **CSymbols** set is pre-ordered by a relation \lesssim , which is induced by a set of *asserted comparisons* $\mathcal{C}_{\mathcal{S}_C} \subseteq \mathcal{T}^\cap$. $(\mathcal{S}_C, \mathcal{C}_{\mathcal{S}_C})$ is a directed graph on \mathcal{S}_C . The pre-order \lesssim is equal to $\mathcal{C}_{\mathcal{S}_C}^*$ the reflexo-transitive closure of $\mathcal{C}_{\mathcal{S}_C}$, i.e., $\forall s, s' \in \mathcal{S}_C, s' \lesssim s$ iff $(s, s') \in \mathcal{C}_{\mathcal{S}_C}^*$, i.e., s' is a descendant of s , and s is an ascendant of s' .

Every **CSymbol** is assigned a signature that specifies what kind of units may be linked through a relation having this symbol.

Definition 5.3 (Signature of **CSymbols**). The set of signatures of **CSymbols** $\{\sigma_s\}_{s \in \mathcal{S}_C}$ is a set of couples in $\mathcal{T}^{\cap 2}$. For every dependency relation symbol s , $\sigma_s \stackrel{\text{def}}{=} (\text{domain}(s), \text{range}(s))$. The set of signatures $\{\sigma_s\}_{s \in \mathcal{S}_C}$ must be such that for all $s_1, s_2 \in \mathcal{S}_C$ such that $s_1 \lesssim s_2$, $\text{domain}(s_1) \hat{\lesssim} \text{domain}(s_2)$ and $\text{range}(s_1) \hat{\lesssim} \text{range}(s_2)$.

We finally introduce the full hierarchy of **CSymbols**.

Definition 5.4 (**CSymbols** hierarchy). A *Circumstantial Dependency Symbols (CSymbols) hierarchy*

$\mathcal{C} \stackrel{\text{def}}{=} (\mathcal{S}_C, \mathcal{C}_{\mathcal{S}_C}, \mathcal{T}^\cap, \{\sigma_s\}_{s \in \mathcal{S}_C})$, is composed of:

- \mathcal{S}_C a set of **Circumstantial Dependency Symbols (CSymbols)**;
- $\mathcal{C}_{\mathcal{S}_C}$ a set of asserted comparisons;
- $\mathcal{T}^\cap = (T_{\text{declared}}, C_{\text{asserted}}, \perp_{\text{asserted}}, \mathcal{S}_{\mathcal{T}}, \text{root}, \text{Hiders}, \{\mathfrak{S}_t\}_{t \in \mathcal{T}})$ a **Conjunctive Unit Types (CUTs)** hierarchy;
- $\{\sigma_s\}_{s \in \mathcal{S}_C}$ a set of signatures of the **CSymbols**;

5.2 Definition of UGs

The **UGs** will enable the description of utterance representation at different representation levels. Parallel with the **CGs**, **UGs** are defined over a so-called support, which is composed of a **CUTs** hierarchy, a **CSymbols** hierarchy, and a set of individual markers.

Definition 5.5 (UG-Support). A *UG-Support* is a tuple $\mathcal{S} \stackrel{\text{def}}{=} (\mathcal{T}^\cap, \mathcal{C}, \mathbf{I})$ where:

- $\mathcal{T}^\cap = (T_{\text{declared}}, C_{\text{asserted}}, \perp_{\text{asserted}}^\square, \mathbf{S}_\mathcal{T}, \text{root}, \text{Hiders}, \{\zeta_t\}_{t \in \mathbf{T}})$ is the *CUTs hierarchy*;
- $\mathcal{C} = (\mathbf{S}_\mathcal{C}, \mathbf{C}_{\mathbf{S}_\mathcal{C}}, \mathcal{T}^\cap, \{\sigma_s\}_{s \in \mathbf{S}_\mathcal{C}})$ is the *CSymbols hierarchy*;
- \mathbf{I} is the set of *individual markers*, which is disjoint from \mathbf{T} , $\mathbf{S}_\mathcal{T}$ and $\mathbf{S}_\mathcal{C}$. $*$ denotes the generic marker, with $* \notin \mathbf{I}$. $\mathbf{I} \cup \{*\}$ denotes the set of markers and is partially ordered as follows: $*$ is greater than any element in \mathbf{I} , and elements in \mathbf{I} are pairwise incomparable.

The natural graph representation we will use is a finite oriented labelled multigraph, composed of unit nodes, participations, and circumstantial dependencies. A unit node is labelled by a couple composed of a **CUT** that specifies the nature of the unit that is represented, and a marker that enables to identify this unit. This marker may be an individual marker if the unit is known, or the generic marker $*$ if this unit is unknown; Every arc is labelled by a symbol in $\mathbf{S}_\mathcal{T} \cup \mathbf{S}_\mathcal{C}$ that specifies the nature of the link that exists between the unit nodes that this arc connects. Arcs and their symbols are defined in terms of so-called triples. A **UG** is thus a combination of interconnected unit nodes defined over a given **UG-Support**.

Definition 5.6 (UG). The set of *UGs* defined over a **UG-Support** $\mathcal{S} \stackrel{\text{def}}{=} (\mathcal{T}^\cap, \mathcal{C}, \mathbf{I})$ is denoted $\mathcal{G}(\mathcal{S})$ and is the set of tuples $G \stackrel{\text{def}}{=} (U, P, C)$ where:

- U is the set of *unit nodes* labelled by a couple $(t^\cap, m) \in \mathbf{T}^\cap \times \mathbf{I} \cup \{*\}$. For all $u \in U$, u is a unit that has type t^\cap and marker m . We denote $t^\cap = \text{type}(u)$ and $m = \text{marker}(u)$.
- P is the set of *participation triples* $(u, r, v) \in U \times \mathbf{S}_\mathcal{T} \times U$. For all $p = (u, r, v) \in P$, p links a unit node u to a unit node v that fills its **PSlot** r . We denote $u = \text{governor}(p)$, $r = \text{symbol}(p)$ and $v = \text{participant}(p)$. We also denote $\text{arc}(p) = (u, v)$.
- C is the set of *circumstantial dependency triples* $(u, r, v) \in U \times \mathbf{S}_\mathcal{C} \times U$. For all $c = (u, r, v) \in C$, c links a unit node u to a unit node v that u governs with respect to r . Conversely, v depends on u with respect to r . We denote $u = \text{governor}(c)$, $r = \text{symbol}(c)$ and $v = \text{circumstant}(c)$. We also denote $\text{arc}(c) = (u, v)$.
- The *underlying graph* of G denoted $\text{graph}(G)$ is a finite, oriented, labelled multigraph where:

for all $u \in U$, u is a node of $\text{graph}(G)$;

for all $p \in P$, $\text{arc}(p)$ is an arc of $\text{graph}(G)$ labelled by $\text{symbol}(p)$;

for all $c \in C$, $\text{arc}(c)$ is an arc of $\text{graph}(G)$ labelled by $\text{symbol}(c)$;

To distinguish participant and circumstantial dependency triples in a **UG**, we draw the former as a double arrow and the latter as a single arrow. For instance, figures below visualizes a **UG** that could represent the sentence *The cat falls gently of a soft blanket* at the DSyn representation level. In this example, $\boxed{\text{FALL}_{\text{present}} : *}$ is a generic syntactic unit having type the conjunction of the lexical unit type FALL^L , and the grammatical unit type present^G .

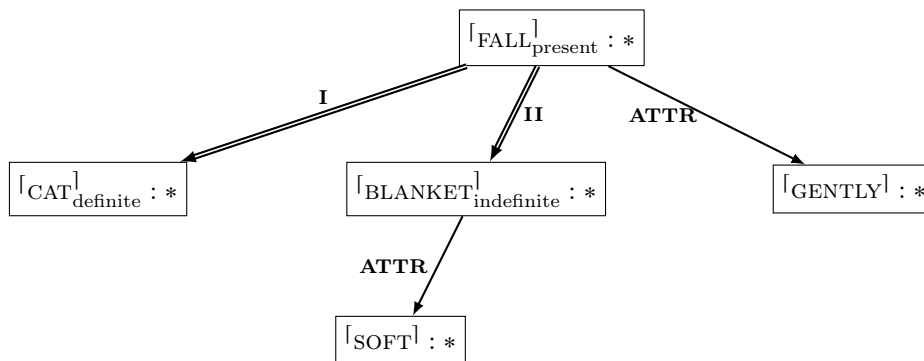


Figure 7: DSyn Representations of utterance *The cat falls gently on a soft blanket*

As we shall see, **UGs** so defined are base objects of the **UGs** mathematical framework, with which one may formalize among others:

- utterance representations at different representation levels;
- semantic decompositions of lexical units (for lexicographic definitions);
- premises and conclusions of linguistic and grammatical rules;

6 Conclusion and perspectives

This research report introduces basics of the **UGs** mathematical formalism, and we may now answer to questions that we asked in the introduction:

- a) What mathematical structure for a unit types hierarchy and their actant slots ? To take into account duality of units (concept type and relation), a unit of a **UG** has a marker that enables to identify this unit as an instance, and a type. Every type models both a concept type to which a set of units may belong to. Predicate-argument type relations are symbolized with **Participation Symbols (PSymbols)** and we assign to each **PSymbol** a root **PUT** $root(s)$ and a set of hidiers **PUTs** $Hiders(s)$. Thus, in the pre-ordered set of **PUTs**, a **PSlots** having s as a **PSymbol** is introduced by $root(s)$, and first defines a **ASlot** for any **PUT** more specific than $root(s)$ as long as that **PUT** is not more specific than one of the elements in $Hiders(s)$. If that happens, then the **PSlot** becomes non-actancial and defines a **HSlot**. Any **PUT** that possesses **ASlots** thus also models a relation type, that may link every instance of that type to the set of its actants in the utterance representation. Finally, to every **PUT** is assigned a signature that specifies the type of participants of its instances.
- b) How to extend the hierarchy of unit types to multi-typing ? **PUTs** are extended to their conjunctive version **CUTs** to model multi-typing. We introduced a set of declared absurd **CUT**, and extended definition of slots and signature to **CUTs**. We then constructed a new type of pre-order over **CUTs** so that an absurd signature of a **CUT** makes that **CUT** absurd. Finally, we showed that mathematical properties of **PUTs** extend to **CUTs**.
- c) What about circumstantial dependencies ? We introduced a **CSymbols** hierarchy which is a pre-ordered set of signed **CSymbols**.
- d) What would be a formal representation of utterance representations ? Finally, **UGs** are labelled, oriented, finite multi-graphs, whose nodes are units and arcs are participation or circumstantial dependency relations. The label of a unit is a couple (**CUTs**, marker), and the label of a arc is either a **PSymbol** or a **CSymbol**. **UGs** so defined are base objects of the **UGs** mathematical framework, with which one may formalize utterance representations at different representation levels.

This research report is currently being improved and a second version will answer the following points:

- **PUTs** and **CSymbols** hierarchies enable a first type of reasoning, but we need among other things lexical and grammatical rules. We will introduce a formalization of rules;
- We already foresee that the **PUTs** hierarchy contributes to the formal definition of a **PUT**, but one needs to formalize unit type definitions as precisely as for the **ECD** lexicographic definitions;
- We have not introduced lexical functions despite the fact that they are very important in the **MTT**. We think that formalizing lexical functions will lead to a natural classification.

What's more, we mainly illustrated our work with sense units. We will introduce a specialization of the **UGs** mathematical framework for the **MTT**, and more specifically the different representation levels and grammar modules.

Finally, We will specify what logical semantic may be associated with this formalism using first-order logic and models theory.

References

- Alonso Ramos, M. (2003). Hacia un diccionario de colocaciones del español y su codificación. *Lexicografía computacional y semántica*, page 11–34.
- Apresian, J., Boguslavsky, I., Iomdin, L., Lazursky, A., Sannikov, V., Sizov, V., and Tsinman, L. (2003). ETAP-3 linguistic processor: A full-fledged NLP implementation of the MTT. In *MTT 2003, First International Conference on Meaning-Text Theory*, page 279–288.
- Baget, J. F., Croitoru, M., Gutierrez, A., Leclere, M., and Mugnier, M. L. (2010). Translations between RDF (s) and conceptual graphs. *Conceptual Structures: From Information to Intelligence*, page 28–41.
- Barque, L., Nasr, A., and Polguere, A. (2010). From the definitions of the 'Trésor de la langue française' to a semantic database of the french language. In *Proceedings of the XIV Euralex International Congress*, Fryske Akademy, page 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. 22.
- Boguslavsky, I., Iomdin, L., and Sizov, V. (2004). Multilinguality in ETAP-3: reuse of lexical resources. In *Proceedings of the Workshop on Multilingual Linguistic Resources*, page 7–14.
- Chen, M. and Mugnier, M. L. (2008). *Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs*. Springer.
- Corby, O., Dieng, R., and Hébert, C. (2000). A conceptual graph model for W3C resource description framework. In Ganter, B. and Mineau, G. W., editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues*, number 1867 in Lecture Notes in Computer Science, pages 468–482. Springer Berlin Heidelberg.
- Leclère, M. (1998). Raisonner avec des définitions de types dans le modèle des graphes conceptuels. (12):243–278.
- Lefrançois, M. and Gandon, F. (2011a). ILexicOn: toward an ECD-Compliant interlingual lexical ontology described with semantic web formalisms. In *Proc. of the 5th International Conference on Meaning-Text Theory (MTT 2011)*, page 155–164, Barcelona, Spain. INALCO.
- Lefrançois, M. and Gandon, F. (2011b). ULiS: an expert system on linguistics to support multilingual management of interlingual knowledge bases. In Kageura, K. and Zweigenbaum, P., editors, *Proc. of the 9th International Conference on Terminology and Artificial Intelligence (TIA 2011)*, page 108–114, Paris, France. INALCO.
- L'Homme, M. C. (2008). Le DiCoInfo: méthodologie pour une nouvelle génération de dictionnaires spécialisés. *Traduire*, (217):78–103.
- Lux-Pogodalla, V. and Polguère, A. (2011). Construction of a french lexical network: Methodological issues. In *Proceedings of the International Workshop on Lexical Resources*, Ljubljana.
- Mel'čuk, I. A. (2004a). Actants in semantics and syntax I: Actants in semantics. *Linguistics*, 42(1):1–66.
- Mel'čuk, I. A. (2004b). Actants in semantics and syntax II: actants in syntax. *Linguistics*, 42(2):247–291.

- Mel'čuk, Igor, e. a. (1984–1999). *Dictionnaire explicatif et combinatoire du français contemporain. Recherches lexico-sémantiques*, volume I–IV. Les Presses de l'Université de Montréal, Montréal.
- Mel'čuk, I. A. (2006). Explanatory combinatorial dictionary. *Open problems in linguistics and lexicography*, page 225.
- Polguère, A. (2000). Une base de données lexicales du français et ses applications possibles en didactique. *Revue de Linguistique et de Didactique des Langues (LIDIL)*, 21:75–97.
- Polguère, A. (1998). La théorie sens-texte. *Dialangue*, 8-9:9–30.
- Sowa, J. F. (1984). *Conceptual structures: information processing in mind and machine*. System programming series. Addison-Wesley.
- Sowa, J. F. (1989). Using a lexicon of canonical graphs in a semantic interpreter. In *Relational models of the lexicon*, page 113–137.
- Sérasset, G. (1997). Le projet NADIA-DEC: vers un dictionnaire explicatif et combinatoire informatisé. *LTT*, 97:149–160.
- Tesnière, L. (1959). *Éléments de syntaxe structurale*. C. Klincksieck (Colombes, Impr. ITE).



**RESEARCH CENTRE
SOPHIA ANTIPOLIS – MÉDITERRANÉE**

2004 route des Lucioles - BP 93
06902 Sophia Antipolis Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399