The Unit Graphs Mathematical Framework

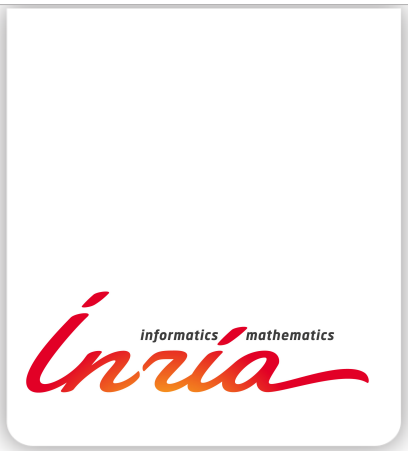
Maxime Lefrançois

The Unit Graphs Mathematical Framework

Maxime Lefrançois

RESEARCH REPORT
N° 8212
January 2013
Project-Team Wimmics

ISSN 0249-6399 ISRN INRIA/RR--8212--FR+ENG



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

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), Dicouèbe Polguere (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 formalism[1] 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 instanciated 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 Graphs [3] 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:

a) What mathematical structure for a unit types hierarchy and their actant slots ?

b) How to extend the hierarchy of unit types to multi-typing ?

c) What about circumstantial dependencies ?

[2] For 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⟩.

[3] Unit 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
$\lceil \text{FALL} \rceil^{L}$	a specific SynLUT	$\lceil \text{present} \rceil^{G}$	a specific SynGUT
$\lceil \text{FALL} \rceil_{\text{present}} : *$	a specific SynLU	(ball)	a specific SemUT
$(\text{ball}) : *$	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, $\boxed{(\text{fall}) : *}$ is a generic sense unit having type the semantic unit type (fall). The following figure represents a Semantic representation of the utterance.

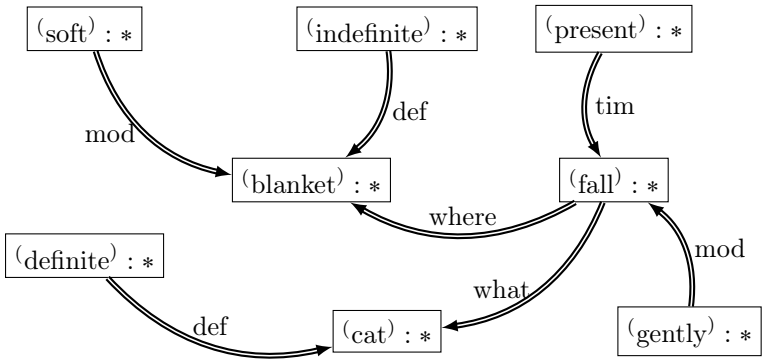


Figure 1: Semantic representations of utterance *The cat falls gently on a soft blanket*

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 Circumstancial 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 bot as a concept (i.e., it has instances), and as a n-ary relation (i.e., il lie 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 $\boldsymbol{T}$ and defined as follows.

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

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

By construction, the set of PUTs is bounded by $\top$, a maximal element of $\boldsymbol{T}$, and $\bot$, a minimal element of $\boldsymbol{T}$:

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

Proof. Let us prove that 1) $\top$ is a maximal element of $\boldsymbol{T}$, and 2) $\bot$ is a minimal element of $\boldsymbol{T}$.
1) Let $t \in \boldsymbol{T}$. From definition 2.2, $(\top, t) \in C_\top$.
$C_\top \subseteq \boldsymbol{C_T} \subseteq \boldsymbol{C_T^*}$, so $(\top, t) \in \boldsymbol{C_T^*}$. So $t \lesssim \top$ (def. 2.2) and $\top$ is a maximal element of $\boldsymbol{T}$.
2) Conversely, let $t \in \boldsymbol{T}$. From definition 2.2, $(t, \bot) \in C_\bot$.
$C_\bot \subseteq \boldsymbol{C_T} \subseteq \boldsymbol{C_T^*}$, so $(t, \bot) \in \boldsymbol{C_T^*}$, so $\bot \lesssim t$ (def. 2.2) and $\bot$ is a maximal element of $\boldsymbol{T}$. □

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 $\boldsymbol{T}$. Let $t \in \boldsymbol{T}$, we denote $[t]$ the equivalence class to which t belongs, i.e., $[t] \stackrel{\text{def}}{=} \{t' \in \boldsymbol{T} \mid t' \simeq t\}$.

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

$\bot$ 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 $\bot$ should also be absurd. As $\bot$ is also a minimal element of $\boldsymbol{T}$, any PUT which is more specific than $\bot$ is actually equivalent to $\bot$. The equivalence class to which $\bot$ belongs is thus called the set of absurd PUTs and denoted $\bot^\sim$. Any $t \in \bot^\sim$ 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 $\bot^\sim$ and is the set: $\bot^\sim \stackrel{\text{def}}{=} [\bot]$.

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 ⟨L⟩ correspond roughly to participants of the linguistic situation SIT(L) denoted by ⟨L⟩ 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 ⟨L⟩ is roughly : the union of ⟨L⟩'s ASlots and of every semantic unit type more generic than ⟨L⟩ 's. Participant positions of ⟨L⟩ 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 $\boldsymbol{S}_{\mathcal{T}}$. Any PUT has a set of *PSlots* which is defined through a mapping $\boldsymbol{\omega}$ from $\boldsymbol{T}$ to $2^{\boldsymbol{S}_{\mathcal{T}}}$. For every $t \in \boldsymbol{T}$, $s \in \boldsymbol{\omega}(t)$ 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* $\in \boldsymbol{S}_{\mathcal{T}}$ to carry the participation meaning *what moves*. Moreover, it is assumed and illustrated that the PUT ⟨fall⟩ has a PSlot having symbol *what*, i.e., *what* $\in \boldsymbol{\omega}(\langle\text{fall}\rangle)$.

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 $\boldsymbol{T}$ denoted $root(s)$ that introduces a PSlot with symbol s. We define the set of PSlots of a PUT $t \in \boldsymbol{T}$ 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 $\boldsymbol{S}_{\mathcal{T}}$ to $\boldsymbol{T}$ that links a PSlot s to the PUT that introduces s. The set of PSlots of a PUT $t \in \boldsymbol{T}$ is defined as the set of PSlots whose root is more general than t, i.e., $\forall t \in \boldsymbol{T}, \boldsymbol{\omega}(t) \stackrel{\text{def}}{=} \{s \in \boldsymbol{S}_{\mathcal{T}} \mid t \lesssim root(s)\}$.

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

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 ⟨fall⟩ such as ⟨tumble⟩, 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* $s \in \mathbf{S}_{\mathcal{T}}$*,* $\{t \in \mathbf{T} \mid s \in \boldsymbol{\omega}(t)\} = \downarrow root(s)$*.*

Proof. Let $s \in \boldsymbol{S}_{\mathcal{T}}$.
$\subseteq$: Let $t \in \{t \in \boldsymbol{T} \mid s \in \boldsymbol{\omega}(t)\}$. $s \in \boldsymbol{\omega}(t)$, 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, $s \in \boldsymbol{\omega}(t)$, so $t \in \{t \in \boldsymbol{T} \mid s \in \boldsymbol{\omega}(t)\}$. □

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

[4] Despite 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.

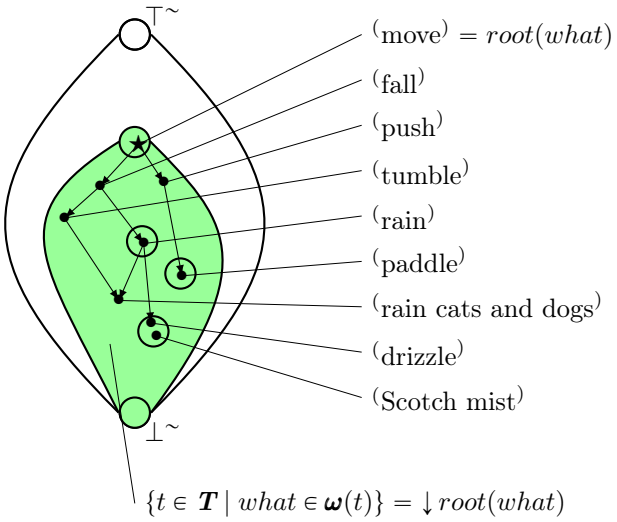


Figure 2: Illustration of the set of PUTs that have a PSlot *what*.

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.,*

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

Moreover, if $t_x \simeq t_y$, *then* $\boldsymbol{\omega}(t_y) = \boldsymbol{\omega}(t_x)$.

Proof. Let $t_x, t_y \in \boldsymbol{T}$ such that $t_x \lesssim t_y$, and $s \in \boldsymbol{\omega}(t_y)$.
Definition 2.6 implies $t_y \lesssim root(s)$. So $t_x \lesssim t_y \lesssim root(s)$ and $s \in \boldsymbol{\omega}(t_x)$.
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 $\boldsymbol{T}$will inherit all the PSlots:

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

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

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 $\boldsymbol{T}$ denoted $Hiders(s)$ that hide a PSlot s, and that the set of HSlots of a PUT $t \in \boldsymbol{T}$ 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 $\boldsymbol{S_\mathcal{T}}$ to $2^{\downarrow root(s)} \backslash \varnothing$ 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 $\overline{\boldsymbol{\alpha}}$ from $\boldsymbol{T}$ to $2^{\boldsymbol{S_\mathcal{T}}}$ such that $\forall t \in \boldsymbol{T}, \overline{\boldsymbol{\alpha}}(t) \stackrel{\text{def}}{=} \{s \in \boldsymbol{S_\mathcal{T}} \mid \exists t_h \in Hiders(s), t \lesssim t_h\}$

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* hidden [5]. 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.,* $\downarrow Hiders(s) \stackrel{def}{=} \{t \in \mathbf{T} \mid \exists t_h \in Hiders(s) : t \lesssim t_h\}$*. For any* $s \in \mathbf{S}_\mathcal{T}$*,* $\{t \in \mathbf{T} \mid s \in \overline{\boldsymbol{\alpha}}(t)\} = \downarrow Hiders(s)$*.*

Proof. Let $s \in \boldsymbol{S_\mathcal{T}}$.
$\subseteq$: Let $t \in \{t \in \boldsymbol{T} \mid s \in \overline{\boldsymbol{\alpha}}(t)\}$. $s \in \overline{\boldsymbol{\alpha}}(t)$.
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, $s \in \overline{\boldsymbol{\alpha}}(t)$. Thus $t \in \{t \in \boldsymbol{T} \mid s \in \overline{\boldsymbol{\alpha}}(t)\}$. □

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

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

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

[5] The 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.

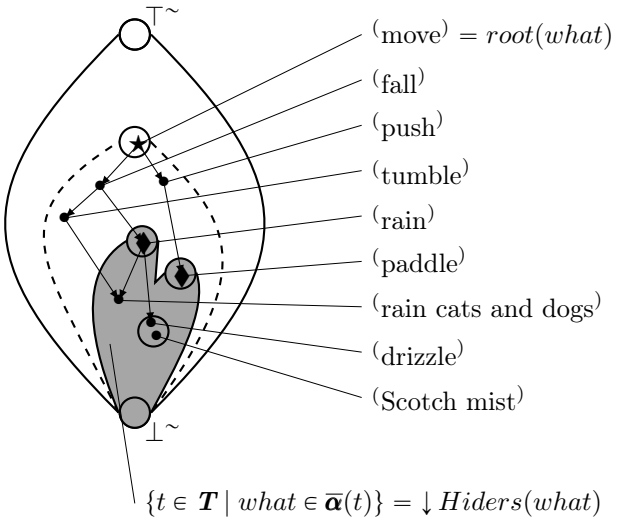


Figure 3: Illustration of the set of PUTs that have a HSlot *what*.

Proof. Let $t_x, t_y \in \boldsymbol{T}$ such that $t_x \lesssim t_y$, and $s \in \overline{\boldsymbol{\alpha}}(t_y)$.
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 $s \in \overline{\boldsymbol{\alpha}}(t_x)$.
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., $2^{\downarrow root(s)} \backslash \varnothing$, 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* $t \in \mathbf{T}$, $\overline{\boldsymbol{\alpha}}(t)$ *is a subset of* $\boldsymbol{\omega}(t)$.

Proof. Let $t \in \boldsymbol{T}$, and $s \in \overline{\boldsymbol{\alpha}}(t)$.
From definition 2.7, $\exists t_h \in Hiders(s), t \lesssim t_h$.
Then as $Hiders(s) \in 2^{\downarrow root(s)} \backslash \varnothing$, we know that $t \lesssim t_h \lesssim root(s)$.
So by definition 2.6, $s \in \boldsymbol{\omega}(t)$ and $\overline{\boldsymbol{\alpha}}(t) \subseteq \boldsymbol{\omega}(t)$. □

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

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

Proof. Let $s \in \boldsymbol{S}_{\mathcal{T}}$.
From definition 2.7 we know that $Hiders(s)$ is not empty and thus $\exists t_h \in Hiders(s)$.
As $\bot \lesssim t_h$, then $\bot \in \downarrow Hiders(s)$ and $s \in \overline{\boldsymbol{\alpha}}(\bot)$.
Now, let $t \in \bot^\sim$, $t \lesssim \bot$ and using proposition 2.6, $\boldsymbol{S}_{\mathcal{T}} \subseteq \overline{\boldsymbol{\alpha}}(t)$.
So for all $t \in \bot^\sim$, $\boldsymbol{S}_{\mathcal{T}} \subseteq \overline{\boldsymbol{\alpha}}(t)$, and thus $\boldsymbol{S}_{\mathcal{T}} = \overline{\boldsymbol{\alpha}}(t)$. □

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 $\boldsymbol{\alpha}$ from $\boldsymbol{T}$ to $2^{\boldsymbol{S}_\mathcal{T}}$ such that $\forall t \in \boldsymbol{T}, \boldsymbol{\alpha}(t) \stackrel{\text{def}}{=} \boldsymbol{\omega}(t) - \overline{\boldsymbol{\alpha}}(t)$. The number of ASlots of a PUT t is denoted the *valency* of t, i.e., $valency(t) \stackrel{\text{def}}{=} |\boldsymbol{\alpha}(t)|$.

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

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

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

Proposition 2.10. *For any* $s \in \mathbf{S}_\mathcal{T}$,

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

Proof. Let $s \in \boldsymbol{S}_\mathcal{T}$.
$\subseteq$: Let $t \in \{t \in \boldsymbol{T} \mid s \in \boldsymbol{\alpha}(t)\}$. $s \in \boldsymbol{\alpha}(t)$.
By definition 2.8, $s \in \boldsymbol{\omega}(t) - \overline{\boldsymbol{\alpha}}(t)$.
By definition 2.6 and proposition 2.7, $s \lesssim root(s)$ and $\forall t_h \in Hiders(s), t \not\lesssim t_h$.
So $t \in \downarrow root(s) - \downarrow Hiders(s)$.
$\supseteq$: Let $t \in \downarrow root(s) - \downarrow Hiders(s)$.
$s \lesssim root(s)$ and $\forall t_h \in Hiders(s), t \not\lesssim t_h$.
By definition 2.6 and proposition 2.7, $s \in \boldsymbol{\omega}(t)$ and $s \notin \overline{\boldsymbol{\alpha}}(t)$.
So $s \in \boldsymbol{\alpha}(t)$, and $t \in \{t \in \boldsymbol{T} \mid s \in \boldsymbol{\alpha}(t)\}$. □

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

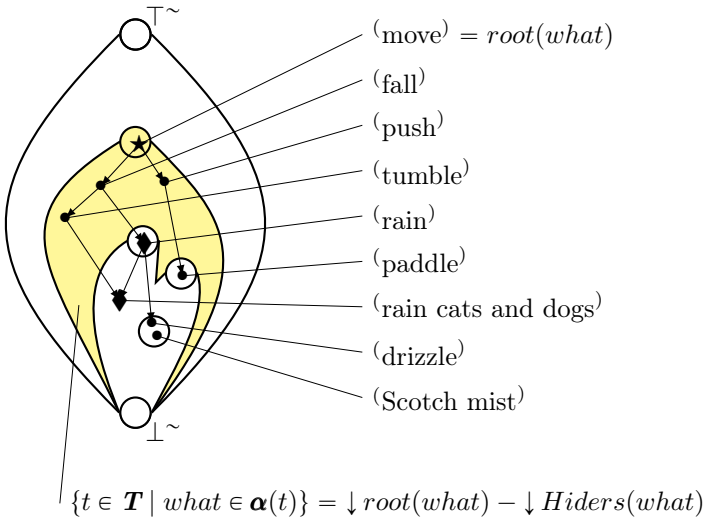


Figure 4: Illustration of the set of PUT that have ASlots *what*.

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* $t \simeq t'$, $\boldsymbol{\alpha}(t) = \boldsymbol{\alpha}(t')$.

Proof. From definition 2.8, $\boldsymbol{\alpha}(t) = \boldsymbol{\omega}(t) - \overline{\boldsymbol{\alpha}}(t)$.
Moreover, from propositions 2.3 and 2.6, $\boldsymbol{\omega}(t') = \boldsymbol{\omega}(t)$ and $\overline{\boldsymbol{\alpha}}(t') = \overline{\boldsymbol{\alpha}}(t)$.
So $\boldsymbol{\omega}(t') = \boldsymbol{\omega}(t)$. □

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

Proposition 2.12. $\forall t \in \perp^{\sim}, \boldsymbol{\alpha}(t) = \varnothing$.

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

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

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

$$t_1 \lesssim t_2 \Rightarrow \boldsymbol{\varsigma}_{t_1}(s) \overset{\frown}{\lesssim} \boldsymbol{\varsigma}_{t_2}(s) \tag{4}$$

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 have a lexical unit type and multiple grammatical unit types for instance, like $\{\ulcorner\text{def}\urcorner^G, \ulcorner\text{USA}\urcorner^L\}$ for *"the USA"*.

In section 3.1 we hence introduce the set of Conjunctive Unit Type (CUT) as the set of all subsets of $\boldsymbol{T}$, 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 $\{\ulcorner\text{def}\urcorner^G, \ulcorner\text{USA}\urcorner^L\}$ for *"the USA"* is composed of a lexical unit type $\ulcorner\text{USA}\urcorner^L$ and a grammatical unit type $\ulcorner\text{def}\urcorner^G$. We hence introduce the set of possible Conjunctive Unit Type (CUT) which is the powerset of $\boldsymbol{T}$, i.e., the set of all subsets of $\boldsymbol{T}$:

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

Some CUTs such as $\{\ulcorner\text{def}\urcorner^G, \ulcorner\text{indef}\urcorner^G\}$ are said to be absurd. This means there is no unit that has both types $\ulcorner\text{def}\urcorner^G$ and $\ulcorner\text{indef}\urcorner^G$. 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 $\bot^\sqcap_{asserted}$. And $\bot^\sqcap_{asserted} \subseteq \boldsymbol{T}^\cap$.

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 $what \in \boldsymbol{\omega}((\text{move}))$, *what* corresponds to a participant of SIT(move). As SIT(move,quick) is a specialization of SIT(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 $\boldsymbol{\omega}^\cap$ from $\boldsymbol{T}^\cap$ to $2^{\boldsymbol{S}_\mathcal{T}}$ Such that $\forall t^\cap \in \boldsymbol{T}^\cap, \boldsymbol{\omega}^\cap(t^\cap) \stackrel{\text{def}}{=} \bigcup_{t \in t^\cap} \boldsymbol{\omega}(t)$.

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 $\overline{\boldsymbol{\alpha}}^\cap$ from $\boldsymbol{T}^\cap$ to $2^{\boldsymbol{S}_\mathcal{T}}$ Such that $\forall t^\cap \in \boldsymbol{T}^\cap, \overline{\boldsymbol{\alpha}}^\cap(t^\cap) \stackrel{\text{def}}{=} \bigcup_{t \in t^\cap} \overline{\boldsymbol{\alpha}}(t)$.

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* $t^\cap \in \mathbf{T}^\cap$, $\overline{\boldsymbol{\alpha}}^\cap(t^\cap)$ *is a subset of* $\boldsymbol{\omega}^\cap(t^\cap)$.

Proof. Let $t^\cap \in \boldsymbol{T}^\cap$ and $s \in \overline{\boldsymbol{\alpha}}^\cap(t^\cap)$.
By definition 3.4, $\exists t \in t^\cap, s \in \overline{\boldsymbol{\alpha}}(t)$.
From proposition 2.7, $s \in \boldsymbol{\omega}(t)$.
By definition 3.3, $s \in \boldsymbol{\omega}^\cap(t^\cap)$. □

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 $\boldsymbol{\alpha}^\cap$ from $\boldsymbol{T}^\cap$ to $2^{\boldsymbol{S}_\mathcal{T}}$ Such that $\forall t^\cap \in \boldsymbol{T}^\cap, \boldsymbol{\alpha}^\cap(t^\cap) \stackrel{\text{def}}{=} \boldsymbol{\omega}^\cap(t^\cap) - \overline{\boldsymbol{\alpha}}^\cap(t^\cap)$.

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

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

Proof. From proposition 4.6,
$\forall t^\cap \in \boldsymbol{T}^\cap, \boldsymbol{\alpha}^\cap(t^\cap) = \boldsymbol{\omega}^\cap(t^\cap) - \overline{\boldsymbol{\alpha}}^\cap(t^\cap) \Leftrightarrow \forall t^\cap \in \boldsymbol{T}^\cap, \boldsymbol{\omega}^\cap(t^\cap) = \boldsymbol{\alpha}^\cap(t^\cap) \cup \overline{\boldsymbol{\alpha}}^\cap(t^\cap)$. □

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*: $\{t \in t^\cap \mid what \in \boldsymbol{\omega}(t)\}$:

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

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:*

- $\boldsymbol{\omega}^\cap(\{t\}) = \boldsymbol{\omega}(t)$.
- $\overline{\boldsymbol{\alpha}}^\cap(\{t\}) = \overline{\boldsymbol{\alpha}}(t)$.
- $\boldsymbol{\alpha}^\cap(\{t\}) = \boldsymbol{\alpha}(t)$.
- *for all* $s \in \boldsymbol{\omega}(t)$, $\boldsymbol{\varsigma}^\cap_{\{t\}}(s) = \boldsymbol{\varsigma}_t(s)$

Proof. Let $t \in \boldsymbol{T}$.
1) $\boldsymbol{\omega}^\cap(\{t\}) = \bigcup_{t \in \{t\}} \boldsymbol{\omega}(t) = \boldsymbol{\omega}(t)$;
2) $\overline{\boldsymbol{\alpha}}^\cap(\{t\}) = \bigcup_{t \in \{t\}} \overline{\boldsymbol{\alpha}}(t) = \overline{\boldsymbol{\alpha}}(t)$;
3) $\boldsymbol{\alpha}^\cap(\{t\}) = \boldsymbol{\omega}^\cap(\{t\}) - \overline{\boldsymbol{\alpha}}^\cap(\{t\}) = \boldsymbol{\omega}(t) - \overline{\boldsymbol{\alpha}}(t) = \boldsymbol{\alpha}(t)$.
4) let $s \in \boldsymbol{\omega}(t)$. $t \in \{t \in t^\cap \mid s \in \boldsymbol{\omega}(t)\}$, so $\boldsymbol{\varsigma}^\cap_{\{t\}}(s) = \bigcup_{t \in \{t\} \mid s \in \boldsymbol{\omega}(t)} \boldsymbol{\varsigma}_t(s) = \boldsymbol{\varsigma}_t(s)$. □

3.3 Pre-order over CUTs

In this section we define a Pre-order over $\boldsymbol{T}^{\frown}$ which models a specialization relation, e.g., $\{^{(}\text{happy}^{)}, ^{(}\text{cat}^{)}\} \overset{\frown}{\lesssim} \{^{(}\text{hasMood}^{)}, ^{(}\text{animal}^{)}\}$ 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 $\boldsymbol{T}^{\frown}$, and we will describe the intuitive meaning of every element of this definition just after:

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

- for $i = 0$, $\boldsymbol{C}_0^{\frown} \overset{\text{def}}{=} \boldsymbol{C}_{\lesssim}^{\frown} \cup \boldsymbol{C}_{\top}^{\frown} \cup \boldsymbol{C}_{\bot}^{\frown}$, where:

$$\boldsymbol{C}_{\lesssim}^{\frown} \overset{\text{def}}{=} \{(t_y^{\frown}, t_x^{\frown}) \in \boldsymbol{T}^{\frown 2} \mid \forall t_y \in t_y^{\frown}, \exists t_x \in t_x^{\frown} : t_x \lesssim t_y\} \tag{5}$$

$$\boldsymbol{C}_{\top}^{\frown} \overset{\text{def}}{=} \{(\top^{\frown}, \varnothing)\} \tag{6}$$

$$\boldsymbol{C}_{\bot}^{\frown} \overset{\text{def}}{=} \{(\bot^{\frown}, t^{\frown}) \in \boldsymbol{T}^{\frown 2} \mid t^{\frown} \in \bot_{asserted}^{\sqcap}\} \tag{7}$$

- for all $i > 0$, $\boldsymbol{C}_i^{\frown} \overset{\text{def}}{=} \boldsymbol{C}_{i-1}^{\frown} \cup \boldsymbol{C}_{\boldsymbol{\varsigma}\ i}^{\frown} \cup \boldsymbol{C}_{+i}^{\frown}$ where:

$$\boldsymbol{C}_{\boldsymbol{\varsigma}\ i}^{\frown} \overset{\text{def}}{=} \{(\bot^{\frown}, t^{\frown}) \in \boldsymbol{T}^{\frown 2} \mid \exists s \in \boldsymbol{\omega}^{\frown}(t^{\frown}), (\bot^{\frown}, \boldsymbol{\varsigma}_{t^{\frown}}^{\frown}(s)) \in \boldsymbol{C}_{i-1}^{\frown}\} \tag{8}$$

$$\boldsymbol{C}_{+i}^{\frown} \overset{\text{def}}{=} \{(t_z^{\frown}, t_x^{\frown}) \in \boldsymbol{T}^{\frown 2} \mid \exists t_y^{\frown} \in \boldsymbol{T}^{\frown}, (t_z^{\frown}, t_y^{\frown}) \in \boldsymbol{C}_{i-1}^{\frown} \text{ and } (t_y^{\frown}, t_x^{\frown}) \in \boldsymbol{C}_{i-1}^{\frown}\} \tag{9}$$

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

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

- $\boldsymbol{C}_{\lesssim}^{\frown}$ represents the natural extension of a pre-order over a set to a pre-order over its powerset;
- $\boldsymbol{C}_{\top}^{\frown}$ is introduced to flatten the top of the powerset so that $\top^{\frown}$ is the most generic CUT;
- $\boldsymbol{C}_{\bot}^{\frown}$ 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);
- $\boldsymbol{C}_{\boldsymbol{\varsigma}\ i}^{\frown}$ represents the fact that if a signature of a CUT for a given PSlot is absurd, then that CUT is absurd;
- $\boldsymbol{C}_{+i}^{\frown}$ transitively closes the set of comparisons $\boldsymbol{C}^{\frown}$.

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

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

Proof. Reflexivity: Let $t^{\frown} \in \boldsymbol{T}^{\frown}$.
$\forall t \in t^{\frown}, \exists t' (= t) \in t^{\frown} : t' \lesssim t$.
So $(t^{\frown}, t^{\frown}) \in \boldsymbol{C}_{\lesssim}^{\frown}$.
Transitivity: Let $t_x^{\frown}, t_y^{\frown}, t_z^{\frown} \in \boldsymbol{T}^{\frown}$ such that $(t_z^{\frown}, t_y^{\frown}) \in \boldsymbol{C}_{\lesssim}^{\frown}$, and $(t_y^{\frown}, t_x^{\frown}) \in \boldsymbol{C}_{\lesssim}^{\frown}$.
$\forall t_z \in t_z^{\frown}, \exists t_y \in t_y^{\frown} : t_y \lesssim t_z$, and $\forall t'_y \in t_y^{\frown}, \exists t_x \in t_x^{\frown} : t_x \lesssim t'_y$.
Let $t'_y = t_y$, then $\forall t_z \in t_z^{\frown}, \exists t_x \in t_x^{\frown} : t_x \lesssim t_z$,
and thus $(t_z^{\frown}, t_x^{\frown}) \in \boldsymbol{C}_{\lesssim}^{\frown}$. □

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

Proposition 3.5. $\mathbf{C}^\frown$ *is reflexive and transitive.*

Proof. As $\boldsymbol{C}^\frown_\lesssim \subseteq \boldsymbol{C}^\frown$ and $\boldsymbol{C}^\frown_\lesssim$ is reflexive, then $\boldsymbol{C}^\frown$ is reflexive.
Transitivity is obtained from $\boldsymbol{C}^\frown_+$ in the construction process of $\boldsymbol{C}^\frown$. □

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

Definition 3.8 (Pre-order relation over $\boldsymbol{T}^\frown$)**.** $\boldsymbol{C}^\frown$ is a binary relation over $\boldsymbol{T}^\frown$which is transitive and reflexive. So it defines a pre-order relation $\overset{\frown}{\lesssim}$ over $\boldsymbol{T}^\frown$, i.e., $t^\frown_x \overset{\frown}{\lesssim} t^\frown_y$ if and only if $(t^\frown_y, t^\frown_x) \in \boldsymbol{C}^\frown$.

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 $\boldsymbol{T}^\frown$:

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

- ***Extension of*** $\lesssim$ *For all* $t^\frown_x, t^\frown_y \in \mathbf{T}^\frown$*, if* $\forall t_y \in t^\frown_y, \exists t_x \in t^\frown_x : t_x \lesssim t_y$*, then* $t^\frown_x \overset{\frown}{\lesssim} t^\frown_y$*;*
- ***Top CUT*** $\varnothing \overset{\frown}{\lesssim} \top^\frown$*;*
- ***Asserted absurd types*** *For all* $t^\frown \in \bot^\sqcap_{asserted}$*,* $t^\frown \overset{\frown}{\lesssim} \bot^\frown$*;*
- ***Absurd signatures*** *For all* $t^\frown \in \mathbf{T}^\frown$*, if there exists* $s \in \boldsymbol{\omega}^\frown(t^\frown)$ *such that* $\boldsymbol{\varsigma}^\frown_{t^\frown}(s) \overset{\frown}{\lesssim} \bot^\frown$*, then* $t^\frown \overset{\frown}{\lesssim} \bot^\frown$*;*

Proof. Extension of $\lesssim$*:* For all $t^\frown_x, t^\frown_y \in \boldsymbol{T}^\frown$, if $\forall t_y \in t^\frown_y, \exists t_x \in t^\frown_x : t_x \lesssim t_y$, then $(t^\frown_y, t^\frown_x) \in \boldsymbol{C}^\frown_\lesssim \subseteq \boldsymbol{C}^\frown$, so $t^\frown_x \overset{\frown}{\lesssim} t^\frown_y$.
Top CUT: We know that $(\top^\frown, \varnothing) \in \boldsymbol{C}^\frown$, so $\varnothing \overset{\frown}{\lesssim} \top^\frown$.
Asserted absurd types: For all $t^\frown \in \bot^\sqcap_{asserted}$, $(\bot^\frown, t^\frown) \in \boldsymbol{C}^\frown_\bot \subseteq \boldsymbol{C}^\frown$, so $t^\frown \overset{\frown}{\lesssim} \bot^\frown$.
Absurd signatures: For all $t^\frown \in \boldsymbol{T}^\frown$, if there exists $s \in \boldsymbol{\omega}^\frown(t^\frown)$ such that $\boldsymbol{\varsigma}^\frown_{t^\frown}(s) \overset{\frown}{\lesssim} \bot^\frown$, then $(\bot^\frown, \boldsymbol{\varsigma}^\frown_{t^\frown}(s)) \in \boldsymbol{C}^\frown$. There exists $n \in \mathbb{N}$ such that $(\bot^\frown, \boldsymbol{\varsigma}^\frown_{t^\frown}(s)) \in \boldsymbol{C}^\frown_n$, and thus for $n+1$, $(\bot^\frown, t^\frown) \in \boldsymbol{C}^\frown_{\boldsymbol{\varsigma}\, n+1} \subseteq \boldsymbol{C}^\frown$. So $t^\frown \overset{\frown}{\lesssim} \bot^\frown$. □

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 $\boldsymbol{T}^\frown$ with the inclusion relation $(\boldsymbol{T}^\frown, \subseteq)$ and the powerset of $\boldsymbol{T}^\frown$ with the specialization relation $(\boldsymbol{T}^\frown, \overset{\frown}{\lesssim})$. The morphism f that maps one onto the other and such that $f(t^\frown) = t^\frown$ is anti-monotonic:

Proposition 3.7. *The function* $f : (\mathbf{T}^\frown, \subseteq) \to (\mathbf{T}^\frown, \overset{\frown}{\lesssim})$ *such that* $f(t^\frown) = t^\frown$ *is an anti-monotonic pre-order homomorphism, i.e., for all* $t^\frown_x, t^\frown_y \in \mathbf{T}^\frown$*,* $t^\frown_x \subseteq t^\frown_y \Rightarrow t^\frown_y \overset{\frown}{\lesssim} t^\frown_x$*.*

Proof. Let $t^\frown_x, t^\frown_y \in \boldsymbol{T}^\frown$ such that $t^\frown_x \subseteq t^\frown_y$. Then $\forall t_x \in t^\frown_x, t_x \in t^\frown_y$.
Thus $\forall t_x \in t^\frown_x, \exists t_y (= t_x) \in t^\frown_y : t_y \lesssim t_x$, so by proposition 3.6, $t^\frown_y \overset{\frown}{\lesssim} t^\frown_x$. □

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

Proposition 3.8. $\top^\frown$ *and* $\varnothing$ *are maximal elements in* $\mathbf{T}^\frown$*, and* $\mathbf{T}$*,* $\bot^\frown$*, and* $\bot^\sim$ *are minimal elements in* $(\mathbf{T}^\frown, \overset{\frown}{\lesssim})$*.*

Proof. Let $t^\cap \in \boldsymbol{T}^\cap$.
$\varnothing$: $\varnothing \subseteq t^\cap$, so using proposition 3.7: $t^\cap \mathrel{\widehat{\lesssim}} \varnothing$.
$\top^\cap$: Let $t^\cap \neq \varnothing$. We know that for all $t \in \boldsymbol{T}$, $t \lesssim \top$, so for $\top$ in $\top^\cap$, $\exists t \in t^\cap : t \lesssim \top$.
So using proposition 3.6, $t^\cap \mathrel{\widehat{\lesssim}} \top^\cap$.
As we moreover know that $\varnothing \mathrel{\widehat{\lesssim}} \top^\cap$, then for all $t^\cap \in \boldsymbol{T}^\cap$, $t^\cap \mathrel{\widehat{\lesssim}} \top^\cap$.
$\boldsymbol{T}$: $t^\cap \subseteq \boldsymbol{T}$, so using proposition 3.7: $\boldsymbol{T} \mathrel{\widehat{\lesssim}} t^\cap$.
$\bot^\cap$: Let $t^\cap \neq \varnothing$. We know that for all $t \in \boldsymbol{T}$, $\bot \lesssim t$, so for all $t \in t^\cap, \bot \lesssim t$.
So using proposition 3.6, $\bot^\cap \mathrel{\widehat{\lesssim}} t^\cap$.
As we moreover know that $\varnothing$ is a maximal element, $\bot^\cap \mathrel{\widehat{\lesssim}} \varnothing$, then for all $t^\cap \in \boldsymbol{T}^\cap$, $\bot^\cap \mathrel{\widehat{\lesssim}} t^\cap$.
$\bot^\sim$: We know that $\bot \in \bot^\sim$, so item above also applies for $\bot^\sim$, i.e.,
for $t^\cap \neq \varnothing$, $\forall t \in t^\cap, \exists t' (= \bot) \in \bot^\sim : t' \lesssim t$, and $\bot^\sim \mathrel{\widehat{\lesssim}} \varnothing$
So for all $t^\cap \in \boldsymbol{T}^\cap$, $\bot^\sim \mathrel{\widehat{\lesssim}} t^\cap$. □

We introduce the natural equivalence relation $\mathrel{\widehat{\simeq}}$ defined by $t_x^\cap \mathrel{\widehat{\simeq}} t_y^\cap \Leftrightarrow t_x^\cap \mathrel{\widehat{\lesssim}} t_y^\cap$ and $t_y^\cap \mathrel{\widehat{\lesssim}} t_x^\cap$. Elements in the flattened bottom of the pre-ordered set $(\boldsymbol{T}^\cap, \mathrel{\widehat{\lesssim}})$ 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 $\bot^\sqcap$ and is the set: $\bot^\sqcap \stackrel{\text{def}}{=} \{t^\cap \in \boldsymbol{T}^\cap \mid t^\cap \mathrel{\widehat{\simeq}} \bot^\cap\}$.

Naturally, the prime absurd CUT, $\bot^\cap$, is absurd.

The property of being absurd is hereditary: if a CUT $t^\cap$ is absurd, all conjunctive types lesser than $t^\cap$ are also absurd. For instance if $\{\lceil \text{def} \rfloor^G, \lceil \text{indef} \rfloor^G\}$ is asserted to be absurd and $\lceil \text{USA} \rfloor^L \lesssim \lceil \text{def} \rfloor^G$, then $\{\lceil \text{USA} \rfloor^L, \lceil \text{indef} \rfloor^G\}$ 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 (rain-hail) as being a specialization of both (rain) and (hail). From definition 2.9, (rain-hail) has a PSlot *what*, whose signature is the union of those of (rain) and (hail):

$$\begin{aligned}\varsigma_{(\text{rain-hail})}(\mathit{what}) &= \varsigma_{(\text{rain})}(\mathit{what}) \cup \varsigma_{(\text{hail})}(\mathit{what}) \\ &= \{(\text{water}), (\text{liquid})\} \cup \{(\text{water}), (\text{solid})\} \\ &= \{(\text{water}), (\text{liquid}), (\text{solid})\}\end{aligned}$$

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

Finally, if one asserts that $\top^\cap \in \bot^\sqcap_{asserted}$, then the whole hierarchy collapses and $\bot^\sqcap = \boldsymbol{T}^\cap$. Same goes if $t^\cap \in \boldsymbol{T}^\cap$ such that $\top^\cap \mathrel{\widehat{\lesssim}} t^\cap$, and $s \in \boldsymbol{\omega}^\cap(t^\cap)$ such that $\varsigma^\cap_{t^\cap}(s) \mathrel{\widehat{\lesssim}} \bot^\cap$. Then from proposition 3.6, $t^\cap \mathrel{\widehat{\lesssim}} \bot^\cap$, and by the pre-order transitivity, $\bot^\sqcap = \boldsymbol{T}^\cap$. 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* $\mathcal{T}^{\cap} = (T_{declared}, C_{asserted}, \bot^{\sqcap}_{asserted}, \boldsymbol{S}_{\mathcal{T}}, root, Hiders, \{\boldsymbol{\varsigma}_t\}_{t \in \boldsymbol{T}})$, is composed of:

- $T_{declared}$ a set of declared PUTs;
- $C_{asserted}$ a set of asserted comparisons;
- $\bot^{\sqcap}_{asserted}$ a set of asserted absurd CUTs.
- $\boldsymbol{S}_{\mathcal{T}}$ 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;
- $\{\boldsymbol{\varsigma}_t\}_{t \in \boldsymbol{T}}$ 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 $t_x^\cap, t_y^\cap \in \boldsymbol{T}^\cap$ and according to the construction of the pre-order over CUTs, we introduce in §4.1 the necessary condition to have $t_x^\cap \mathrel{\hat{\lesssim}} t_y^\cap$.

Second, now that the pre-order $\mathrel{\hat{\lesssim}}$ 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 $\mathrel{\hat{\lesssim}}$ 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 $t_x^\cap, t_y^\cap \in \boldsymbol{T}^\cap$ comparable may be stated as follows:

If $t_x^\cap \mathrel{\hat{\lesssim}} t_y^\cap$, 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 $\boldsymbol{T}^\cap$);
- $t_x^\cap$ 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 $t_x^\cap \mathrel{\hat{\lesssim}} t_y^\cap$, then at least one of the following is true:*

- $t_x^\cap \mathrel{\hat{\simeq}} \bot^\cap$;
- $t_x^\cap = \varnothing$ *(and thus* $\top^\cap \mathrel{\hat{\simeq}} t_y^\cap$*)*;
- $\forall t_y \in t_y^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t_y$ *(i.e.,* $(t_y^\cap, t_x^\cap) \in \mathbf{C}_{\hat{\lesssim}}^\cap$*)*.

Proof. Let $t_x^\cap, t_y^\cap \in \boldsymbol{T}^\cap$.
As $\top^\cap$ (resp. $\bot^\cap$) is a maximal (resp. minimal) element of $\boldsymbol{T}^\cap$,
$\top^\cap \mathrel{\hat{\lesssim}} t_y^\cap$ implies $\top^\cap \mathrel{\hat{\simeq}} t_y^\cap$ (resp. $t_x^\cap \mathrel{\hat{\lesssim}} \bot^\cap$ implies $t_x^\cap \mathrel{\hat{\simeq}} \bot^\cap$).
So it is sufficient to prove that one of the following is true:
i) $t_x^\cap \mathrel{\hat{\lesssim}} \bot^\cap$, ii) $t_x^\cap = \varnothing$ (and thus $\top^\cap \mathrel{\hat{\lesssim}} t_y^\cap$), or iii) $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_{\hat{\lesssim}}^\cap$.

$\boldsymbol{C}^\cap$ is the limit the sequence $(\boldsymbol{C}_n^\cap)$, so let $n \in \mathbb{N}$ be such that $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_n^\cap$ and $(t_y^\cap, t_x^\cap) \notin \boldsymbol{C}_{n-1}^\cap$.
We assume that $t_x^\cap \mathrel{\hat{\not\lesssim}} \bot^\cap$, $(t_y^\cap, t_x^\cap) \notin \boldsymbol{C}_{\hat{\lesssim}}^\cap$, and that either $\top^\cap \mathrel{\hat{\not\lesssim}} t_y^\cap$ or $t_x^\cap \neq \varnothing$.

1) If $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_{\boldsymbol{\varsigma}\,n}^\cap$, then $t_x^\cap \mathrel{\hat{\lesssim}} \bot^\cap$.
This is impossible, so $(t_y^\cap, t_x^\cap) \in \boldsymbol{C}_{+n}^\cap$ and there exists $t^\cap \in \boldsymbol{T}^\cap$ such that $(t_y^\cap, t^\cap) \in \boldsymbol{C}_{n-1}^\cap$ and $(t^\cap, t_x^\cap) \in \boldsymbol{C}_{n-1}^\cap$.
As $\boldsymbol{C}_{n-1}^\cap \subseteq \boldsymbol{C}^\cap$, $t_x^\cap \mathrel{\hat{\lesssim}} t^\cap$.
If $t^\cap \mathrel{\hat{\lesssim}} \bot^\cap$, then $t_x^\cap \mathrel{\hat{\lesssim}} \bot^\cap$ which is impossible.
So we know that $t^\cap \mathrel{\hat{\not\simeq}} \bot^\cap$.

2) From proposition 3.4, we know that $\boldsymbol{C}^{\frown}_{\lesssim}$ is transitive. So either 2a) $(t^{\frown}, t^{\frown}_x) \notin \boldsymbol{C}^{\frown}_{\lesssim}$, or 2b) $(t^{\frown}_y, t^{\frown}) \notin \boldsymbol{C}^{\frown}_{\lesssim}$.

2a) In this case, consider $m \leqslant n-1$ such that $(t^{\frown}, t^{\frown}_x) \in \boldsymbol{C}^{\frown}_{m}$ and $(t^{\frown}, t^{\frown}_x) \notin \boldsymbol{C}^{\frown}_{m-1}$.
We still have $t^{\frown}_x \mathrel{\hat{\nlesssim}} \bot^{\frown}$ and we have now $(t^{\frown}, t^{\frown}_x) \notin \boldsymbol{C}^{\frown}_{\lesssim}$.

2b) In this case, consider $m \leqslant n-1$ such that $(t^{\frown}_y, t^{\frown}) \in \boldsymbol{C}^{\frown}_{m}$ and $(t^{\frown}_y, t^{\frown}) \notin \boldsymbol{C}^{\frown}_{m-1}$.
We just showed in 1) that $t^{\frown} \mathrel{\hat{\nlesssim}} \bot^{\frown}$ and we have now $(t^{\frown}_y, t^{\frown}) \notin \boldsymbol{C}^{\frown}_{\lesssim}$.

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 $t^{\frown}_a$ and $t^{\frown}_b$): $(t^{\frown}_b, t^{\frown}_a) \in \boldsymbol{C}^{\frown}_{0}$, $t^{\frown}_a \mathrel{\hat{\nlesssim}} \bot^{\frown}$ so $(\bot^{\frown}, t^{\frown}_a) \notin \boldsymbol{C}^{\frown}_{\bot}$, and $(t^{\frown}_b, t^{\frown}_a) \notin \boldsymbol{C}^{\frown}_{\lesssim}$. So $(t^{\frown}_b, t^{\frown}_a) \in \boldsymbol{C}^{\frown}_{\top}$.
Neither $\top^{\frown} \mathrel{\hat{\nlesssim}} t^{\frown}_b$ nor $t^{\frown}_b \neq \varnothing$ is compatible with this last statement, so we highlighted a contradiction.

As a conclusion, if $t^{\frown}_x \mathrel{\hat{\lesssim}} t^{\frown}_y$, it is impossible that all of the following is false :
i) $t^{\frown}_x \mathrel{\hat{\lesssim}} \bot^{\frown}$, ii) $t^{\frown}_x = \varnothing$, iii) $(t^{\frown}_y, t^{\frown}_x) \in \boldsymbol{C}^{\frown}_{\lesssim}$. □

4.2 Properties of CUT Slots and Signatures

Now that the pre-order $\mathrel{\hat{\lesssim}}$ 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 $t^{\frown}_x = \{(\text{fall}), (\text{past})\}$ and $t^{\frown}_y = \{(\text{move})\}$, then $t^{\frown}_x \mathrel{\hat{\lesssim}} t^{\frown}_y$, and in every linguistic situation in which a linguistic unit of type $t^{\frown}_x$ 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 $\downarrow^{\frown} root(s)$ be the smallest lower set of $\mathbf{T}^{\frown}$ that contains $\{root(s)\}$:*

$$\downarrow^{\frown} root(s) \stackrel{def}{=} \{t^{\frown} \in \mathbf{T}^{\frown} \mid t^{\frown} \mathrel{\hat{\lesssim}} \{root(s)\}\} \tag{10}$$

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

$$\{t^{\frown} \in \mathbf{T}^{\frown} \mid s \in \boldsymbol{\omega}^{\frown}(t^{\frown})\} \cup \bot^{\sqcap} \backslash \varnothing = \downarrow^{\frown} root(s) \backslash \varnothing \tag{11}$$

Proof. Let $s \in \boldsymbol{S}_{\mathcal{T}}$.
$\subseteq_1$: Let $t^{\frown} \in \{t^{\frown} \in \boldsymbol{T}^{\frown} \mid s \in \boldsymbol{\omega}^{\frown}(t^{\frown})\}$.
By definition 3.3, $\exists t \in t^{\frown}, s \in \boldsymbol{\omega}(t)$. So $t^{\frown} \neq \varnothing$.
On the other hand, from definition 2.6, $t \lesssim root(s)$.
By proposition 3.6, $t^{\frown} \mathrel{\hat{\lesssim}} \{root(s)\}$ so $t^{\frown} \in \downarrow^{\frown} root(s)$.
So $t^{\frown} \in \downarrow^{\frown} root(s) \backslash \varnothing$, and $\{t^{\frown} \in \boldsymbol{T}^{\frown} \mid s \in \boldsymbol{\omega}^{\frown}(t^{\frown})\} \subseteq \downarrow^{\frown} root(s) \backslash \varnothing$.
$\subseteq_2$: Let $t^{\frown} \in \bot^{\sqcap} \backslash \varnothing$. $t^{\frown} \mathrel{\hat{\lesssim}} \bot^{\frown}$ which is a minimum element in $\boldsymbol{T}^{\frown}$.
So $t^{\frown} \mathrel{\hat{\lesssim}} \{root(s)\}$ and $t^{\frown} \in \downarrow^{\frown} root(s)$ and $t^{\frown} \in \downarrow^{\frown} root(s) \backslash \varnothing$.
So $\bot^{\sqcap} \backslash \varnothing \subseteq \downarrow^{\frown} root(s) \backslash \varnothing$.

$\supseteq$: Let $t^\frown \in \downarrow^\frown root(s)\backslash\varnothing$. $t^\frown \mathrel{\widehat{\lesssim}} \{root(s)\}$.
We use proposition 4.1, so at least one of the following is true:
i) $t^\frown \mathrel{\widehat{\lesssim}} \bot^\frown$: then $t^\frown \in \bot^\sqcap$. And as $t^\frown \neq \varnothing$, $t^\frown \in \bot^\sqcap\backslash\varnothing$.
ii) $t^\frown = \varnothing$: impossible case.
iii) $\exists t \in t^\frown : t \lesssim root(s)$: then using definition 2.6, $\exists t \in t^\frown : s \in \boldsymbol{\omega}(t)$, and by definition 3.3, $s \in \boldsymbol{\omega}^\frown(t^\frown)$. So $t^\frown \in \{t^\frown \in \boldsymbol{T}^\frown \mid s \in \boldsymbol{\omega}^\frown(t^\frown)\}$. □

One may have noticed that two kinds of CUTs are not considered here: the CUT $\varnothing$, and any absurd CUT $t^\frown \in \bot^\sqcap$:

- the CUT $\varnothing$: a problem would arise for instance with $s \in \boldsymbol{S}_\mathcal{T}$ such that $root(s) = \top$. By construction of the pre-order $\mathrel{\widehat{\lesssim}}$, $\varnothing \mathrel{\widehat{\lesssim}} \top^\frown$, yet $\varnothing$ has no PSlot ($\boldsymbol{\omega}^\frown(\varnothing) = \bigcup_{t\in\varnothing} \boldsymbol{\omega}(t) = \varnothing$). So $s \notin \boldsymbol{\omega}^\frown(\varnothing)$ and some PSlots may not be inherited from the CUT $\varnothing$;
- any absurd CUT $t^\frown \in \bot^\sqcap$: problems arise for any $s \in \boldsymbol{S}_\mathcal{T}$ such that $s \notin \boldsymbol{\omega}^\frown t^\frown$. By construction of the pre-order $\mathrel{\widehat{\lesssim}}$, $t^\frown \mathrel{\widehat{\lesssim}} \bot^\frown$, yet, from propositions 2.4 and 3.3, $\boldsymbol{\omega}^\frown(\bot^\frown) = \boldsymbol{S}_\mathcal{T}$. 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 $\varnothing$, i.e., $\forall t_x^\frown \in \mathbf{T}^\frown\backslash\varnothing - \bot^\sqcap$, and $t_y^\frown \in \mathbf{T}^\frown$,*

$$t_x^\frown \mathrel{\widehat{\lesssim}} t_y^\frown \Rightarrow \boldsymbol{\omega}^\frown(t_y^\frown) \subseteq \boldsymbol{\omega}^\frown(t_x^\frown) \tag{12}$$

Proof. Let $t_x^\frown \in \boldsymbol{T}^\frown\backslash\varnothing - \bot^\sqcap$ and $t_y^\frown \in \boldsymbol{T}^\frown$ such that $t_x^\frown \mathrel{\widehat{\lesssim}} t_y^\frown$.
Let $s \in \boldsymbol{\omega}^\frown(t_y^\frown)$. By definition 3.3, $\exists t \in t_y^\frown, s \in \boldsymbol{\omega}(t)$.
We use proposition 4.1, but as $t_x^\frown \notin \bot^\sqcap$ and $t_x^\frown \neq \varnothing$, $t_x^\frown \mathrel{\widehat{\lesssim}} t_y^\frown$ simply implies: $\forall t_y \in t_y^\frown, \exists t_x \in t_x^\frown : t_x \lesssim t$. So $\exists t_x \in t_x^\frown : t_x \lesssim t_y$.
Using proposition 2.3, $s \in \boldsymbol{\omega}(t_x)$, and using definition 3.3, $s \in \boldsymbol{\omega}^\frown(t_x^\frown)$. □

Hidden Slots Next, consider the CUT $\{\langle\text{rain}\rangle\}$. 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 $\{\{Hiders(what)\}\}$, and only those CUTs more specific than one CUT of $\{\{Hiders(what)\}\}$ will have PSlot *what* hidden. With the exception, of course, of the CUT $\varnothing$, and any absurd CUT $t^\frown \in \bot^\sqcap$. The following property thus holds:

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

$$\downarrow^\frown Hiders(s) \stackrel{def}{=} \{t^\frown \in \mathbf{T}^\frown \mid \exists t_h \in Hiders(s) : t^\frown \mathrel{\widehat{\lesssim}} \{t_h\}\} \tag{13}$$

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

$$\{t^\frown \in \mathbf{T}^\frown \mid s \in \overline{\boldsymbol{\alpha}}^\frown(t^\frown)\} \cup \bot^\sqcap\backslash\varnothing = \downarrow^\frown Hiders(s)\backslash\varnothing \tag{14}$$

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

$\subseteq_1$: Let $t^\cap \in \{t^\cap \in \boldsymbol{T}^\cap \mid s \in \overline{\boldsymbol{\alpha}}^\cap(t^\cap)\}$.
a) By definition 3.4, $\exists t \in t^\cap, s \in \overline{\boldsymbol{\alpha}}(t)$. So $t^\cap \neq \varnothing$.
b) On the other hand, from definition 2.7, $\exists t_h \in \mathit{Hiders}(s), t \lesssim t_h$. Thus using proposition 3.6 $t^\cap \mathrel{\hat{\lesssim}} \{t_h\}$, and $t^\cap \in \downarrow^\cap \mathit{Hiders}(s)$.
So $t^\cap \in \downarrow^\cap \mathit{Hiders}(s)\backslash\varnothing$, and $\{t^\cap \in \boldsymbol{T}^\cap \mid s \in \overline{\boldsymbol{\alpha}}^\cap(t^\cap)\} \subseteq \downarrow^\cap \mathit{Hiders}(s)\backslash\varnothing$.

$\subseteq_2$: Let $t^\cap \in \bot^\sqcap\backslash\varnothing$.
$t^\cap \mathrel{\hat{\lesssim}} \bot^\cap$ which is a minimum element in $\boldsymbol{T}^\cap$.
So $\forall t_h \in \mathit{Hiders}(s), t^\cap \mathrel{\hat{\lesssim}} \{t_h\}$, i.e., $t^\cap \in \downarrow^\cap \mathit{Hiders}(s)$.
As we chose that $t^\cap$ can't be $\varnothing$, then $t^\cap \in \downarrow^\cap \mathit{Hiders}(s)\backslash\varnothing$.
So $\bot^\sqcap\backslash\varnothing \subseteq \downarrow^\cap \mathit{Hiders}(s)\backslash\varnothing$.

$\supseteq$: Let $t^\cap \in \downarrow^\cap \mathit{Hiders}(s)\backslash\varnothing$.
$\exists t_h \in \mathit{Hiders}(s), t^\cap \mathrel{\hat{\lesssim}} \{t_h\}$.
We use proposition 4.1, so at least one of the following is true:
i) $t^\cap \mathrel{\hat{\simeq}} \bot^\cap$: then $t^\cap \in \bot^\sqcap$ and $t^\cap \in \bot^\sqcap\backslash\varnothing$.
ii) $t^\cap = \varnothing$: impossible case.
iii) $\exists t \in t^\cap : t \lesssim t_h$: then by definition 2.7 $\exists t \in t^\cap : s \in \overline{\boldsymbol{\alpha}}(t)$, and by definition 3.4, $s \in \overline{\boldsymbol{\alpha}}^\cap(t^\cap)$.
So $t^\cap \in \{t^\cap \in \boldsymbol{T}^\cap \mid s \in \overline{\boldsymbol{\alpha}}^\cap(t^\cap)\}$
So $t^\cap \in \{t^\cap \in \boldsymbol{T}^\cap \mid s \in \overline{\boldsymbol{\alpha}}^\cap(t^\cap)\} \cup \bot^\sqcap\backslash\varnothing$. □

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 $\varnothing$, i.e., $\forall t_x^\cap \in \mathbf{T}^\cap\backslash\varnothing - \bot^\sqcap$ and $t_y^\cap \in \mathbf{T}^\cap$,*

$$t_x^\cap \mathrel{\hat{\lesssim}} t_y^\cap \Rightarrow \overline{\boldsymbol{\alpha}}^\cap(t_x^\cap) \subseteq \overline{\boldsymbol{\alpha}}^\cap(t_y^\cap) \tag{15}$$

Proof. Let $t_x^\cap \in \boldsymbol{T}^\cap\backslash\varnothing - \bot^\sqcap$ and $t_y^\cap \in \boldsymbol{T}^\cap$ such that $t_x^\cap \mathrel{\hat{\lesssim}} t_y^\cap$.
Let $s \in \overline{\boldsymbol{\alpha}}^\cap(t_y^\cap)$. By definition 3.4, $\exists t \in t_y^\cap, s \in \overline{\boldsymbol{\alpha}}(t)$.
From proposition 4.1 and the facts that $t_x^\cap \notin \bot^\sqcap$ and $t_x^\cap \neq \varnothing$, we know that $t_x^\cap \mathrel{\hat{\lesssim}} t_y^\cap$ simply implies $\forall t_y \in t_y^\cap, \exists t_x \in t_x^\cap : t_x \lesssim t_y$. So $\exists t_x \in t_x^\cap : t_x \lesssim t$.
Using proposition 2.6, $s \in \overline{\boldsymbol{\alpha}}(t_x)$, and using definition 3.4, $s \in \overline{\boldsymbol{\alpha}}^\cap(t_x^\cap)$. □

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

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

$$\begin{aligned}
&\{t^\cap \in \mathbf{T}^\cap \mid s \in \boldsymbol{\alpha}^\cap(t^\cap)\} \cup \bot^\sqcap\backslash\varnothing \\
&= (\downarrow^\cap \mathit{root}(s)\backslash\varnothing - \downarrow^\cap \mathit{Hiders}(s)\backslash\varnothing) \cup \bot^\sqcap\backslash\varnothing \\
&= \left\{t^\cap \in \mathbf{T}^\cap \mid t^\cap \mathrel{\hat{\lesssim}} \{\mathit{root}(s)\} \text{ and } \forall t_h \in \mathit{Hiders}(s), t^\cap \mathrel{\hat{\nlesssim}} \{t_h\}\right\} \cup \bot^\sqcap\backslash\varnothing
\end{aligned}$$

Proof. We use propositions 2.10, 4.2 and 4.4:

$$\begin{aligned}
&\{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\alpha}^\cap(t^\cap)\} \cup \bot^\sqcap \backslash \varnothing \\
&= \{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\omega}^\cap(t^\cap) - \overline{\boldsymbol{\alpha}}^\cap(t^\cap)\} \cup \bot^\sqcap \backslash \varnothing \\
&= (\{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\omega}^\cap(t^\cap)\} - \{t^\cap \in \boldsymbol{T}^\cap \mid s \in \overline{\boldsymbol{\alpha}}^\cap(t^\cap)\}) \cup \bot^\sqcap \backslash \varnothing \\
&= ((\{t^\cap \in \boldsymbol{T}^\cap \mid s \in \boldsymbol{\omega}^\cap(t^\cap)\} \cup \bot^\sqcap \backslash \varnothing) - (\{t^\cap \in \boldsymbol{T}^\cap \mid s \in \overline{\boldsymbol{\alpha}}^\cap(t^\cap)\} \cup \bot^\sqcap \backslash \varnothing)) \cup \bot^\sqcap \backslash \varnothing \\
&= (\downarrow^\cap root(s) \backslash \varnothing - \downarrow^\cap Hiders(s) \backslash \varnothing) \cup \bot^\sqcap \backslash \varnothing
\end{aligned}$$

□

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* $t_x^\cap \in \mathbf{T}^\cap \backslash \varnothing - \bot^\sqcap$, $t_y^\cap \in \mathbf{T}^\cap$ *and* $s \in \mathbf{S}_\mathcal{T}$ *such that* $s \in \boldsymbol{\omega}^\cap(t_y^\cap)$,

$$t_x^\cap \hat{\lesssim} t_y^\cap \Rightarrow \boldsymbol{\varsigma}_{t_x^\cap}^\cap(s) \hat{\lesssim} \boldsymbol{\varsigma}_{t_y^\cap}^\cap(s) \tag{16}$$

Proof. Let $t_x^\cap \in \boldsymbol{T}^\cap \backslash \varnothing - \bot^\sqcap$ and $t_y^\cap \in \boldsymbol{T}^\cap$ and $s \in \boldsymbol{S}_\mathcal{T}$ such that $t_x^\cap \hat{\lesssim} t_y^\cap$ 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 $t_x^\cap \hat{\lesssim} t_y^\cap$ and proposition 4.1.
As $t_x^\cap \notin \bot^\sqcap$ and $t_x^\cap \neq \varnothing$, 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, $\boldsymbol{\varsigma}_{t_x^\cap}^\cap(s) \hat{\lesssim} \boldsymbol{\varsigma}_{t_y^\cap}^\cap(s)$. □

4.3 CUT Equivalence Class Sets

Let $\hat{\simeq}$ be the natural *equivalence* relation defined by $t^\frown_x \hat{\simeq} t^\frown_y \Leftrightarrow \left(t^\frown_x \hat{\lesssim} t^\frown_y \wedge t^\frown_y \hat{\lesssim} t^\frown_x\right)$. The set of equivalence classes defines a partition of $\boldsymbol{T}^\frown$. Let $t^\frown \in \boldsymbol{T}^\frown$, we denote $[t^\frown] \stackrel{\text{def}}{=} \{t^\frown_x \in \boldsymbol{T}^\frown \mid t^\frown_x \hat{\simeq} t^\frown\}$ the equivalence class to which $t^\frown$ belongs. We will usually use the notation $t^\sqcap$ for an unknown equivalence class.

Definition 4.1 (CUTs equivalence class set)**.** The *CUTs equivalence class set* $\boldsymbol{T}^\sqcap$ is the quotient set of $\boldsymbol{T}^\frown$ by $\hat{\simeq}$, i.e., $\boldsymbol{T}^\sqcap \stackrel{\text{def}}{=} \boldsymbol{T}^\frown/\hat{\simeq} = \{[t^\frown] \mid t^\frown \in \boldsymbol{T}^\frown\}$. We define a partial order $\sqsubseteq$ over $\boldsymbol{T}^\sqcap$ with $[t^\frown_x] \sqsubseteq [t^\frown_y]$ if and only if $t^\frown_x \hat{\lesssim} t^\frown_y$.

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

Proposition 4.8. *$\top^\sqcap$ and $\bot^\sqcap$ are respectively the greatest element and the least element of $\mathbf{T}^\sqcap$.*

Proof. $\top^\sqcap$: From proposition 3.8, $\top^\frown$ is a maximal element for $\hat{\lesssim}$.
So $\forall t^\frown \in \boldsymbol{T}^\frown, t^\frown \hat{\lesssim} \top^\frown$. So $\forall t^\sqcap \in \boldsymbol{T}^\sqcap, t^\sqcap \sqsubseteq \top^\sqcap$, and $\top^\sqcap$ is the greatest element of $\boldsymbol{T}^\sqcap$.
$\bot^\sqcap$: From proposition 3.8, $\bot^\frown$ is a minimal element for $\hat{\lesssim}$. So $\forall t^\frown \in \boldsymbol{T}^\frown, \bot^\frown \hat{\lesssim} t^\frown$.
So $\forall t^\sqcap \in \boldsymbol{T}^\sqcap, \bot^\sqcap \sqsubseteq t^\sqcap$, so $\bot^\sqcap$ is the least element of $\boldsymbol{T}^\sqcap$. □

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

Proposition 4.9. *Let $t^\sqcap \in \mathbf{T}^\sqcap$, and $t^\frown_x, t^\frown_y \in t^\sqcap$. Then $t^\frown_x \cup t^\frown_y \in t^\sqcap$.*

Proof. Let $t^\sqcap \in \boldsymbol{T}^\sqcap$, and $t^\frown_x, t^\frown_y \in t^\sqcap$. We will prove that $t^\frown_x \cup t^\frown_y \hat{\simeq} t^\frown_x$.
$\hat{\lesssim}$: We know from proposition 3.7 that $t^\frown_x \subseteq t^\frown_x \cup t^\frown_y$ implies $t^\frown_x \cup t^\frown_y \hat{\lesssim} t^\frown_x$.
$\hat{\gtrsim}$: We know that $t^\frown_x \hat{\lesssim} t^\frown_y$. From proposition 3.6, at least one of the followings holds:
i) $t^\frown_x \hat{\lesssim} \bot^\frown$: then $t^\frown_x \cup t^\frown_y \hat{\simeq} t^\frown_x \hat{\simeq} \bot^\frown$;
ii) $t^\frown_x = \varnothing$: then $t^\frown_x \cup t^\frown_y = t^\frown_y \hat{\simeq} t^\frown_x$;
iii) $\forall t_y \in t^\frown_y, \exists t_x \in t^\frown_x : t_x \lesssim t_y$: So $\forall t \in t^\frown_x \cup t^\frown_y, \exists t_x \in t^\frown_x : t_x \lesssim t$, and $t^\frown_x \hat{\lesssim} t^\frown_x \cup t^\frown_y$. □

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

Lemma 4.10. *Let $t^\frown \in \mathbf{T}^\frown$ and $t \in \mathbf{T}$ such that $\exists t' \in t^\frown, t' \lesssim t$. Then $t^\frown \cup \{t\} \hat{\simeq} t^\frown$.*

Proof. Let $t^\frown \in \boldsymbol{T}^\frown$ and $t \in \boldsymbol{T}$ such that $\forall t' \in t^\frown, t' \lesssim t$.
$\hat{\lesssim}$: $t^\frown_y \subseteq t^\frown_y \cup \{t\}$ so $t^\frown \cup \{t\} \hat{\lesssim} t^\frown$;
$\hat{\gtrsim}$: $t^\frown \neq \varnothing$. Let $t' \in t^\frown \cup \{t\}$.
If $t' \in t^\frown$, then there exists $t''(= t') \in t^\frown : t'' \lesssim t'$.
If $t' = t$, then there exists $t'' \in t^\frown t'' \lesssim t'$.
So $\forall t' \in t^\frown \cup \{t\}, \exists t'' \in t^\frown : t'' \lesssim t'$, and $t^\frown \cup \{t\} \hat{\gtrsim} t^\frown$. □

4.4 Maximal CUTs

CUTs enable a first inference mechanism, which is type inference. If a CUT $t^\cap_x$ is lower than $t^\cap_y$, then any instance of $t^\cap_x$ is also an instance of every PUT in $t^\cap_y$. 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 $\boldsymbol{T}^\cap$ 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 $\boldsymbol{T}^\cap_{max}$ is then denoted $\boldsymbol{T}^\cap_{max}$, and defined as follows:

$$\boldsymbol{T}^\cap_{max} = \{\Uparrow t^\cap \mid t^\cap \in \boldsymbol{T}^\cap\} \tag{17}$$

In the rest of this section we will list useful facts about the maximization operator $\Uparrow$ and the set of maximal CUTs $\boldsymbol{T}^\cap_{max}$.

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 $t^\sqcap \in \mathbf{T}^\sqcap$, if $t^\cap \in t^\sqcap$, then $\Uparrow t^\cap \in t^\sqcap$. Or equivalently, $\forall t^\cap \in \mathbf{T}^\cap, \Uparrow t^\cap \mathrel{\hat{\simeq}} t^\cap$.*

Proof. Let $t^\sqcap \in \boldsymbol{T}^\sqcap$, and $t^\cap \in t^\sqcap$. By definition, $\Uparrow t^\cap$ is the union of all CUT in $t^\sqcap$.
We know from proposition 4.9 that $t^\sqcap$ is closed under the union operation, so $\Uparrow t^\cap \in t^\sqcap$. □

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 $\hat{\simeq}$, i.e., if $t^\cap_x \mathrel{\hat{\simeq}} t^\cap_y$, then $\Uparrow t^\cap_x = \Uparrow t^\cap_y$*

Proof. By definition, if $t^\cap_x \mathrel{\hat{\simeq}} t^\cap_y$, then $[t^\cap_x] = [t^\cap_y]$, and $\Uparrow t^\cap_x = \Uparrow t^\cap_y$ □

Thus, there is a one-to-one correspondence between the set of a set of equivalence CUTs class sets $\boldsymbol{T}^\sqcap$ and the set of maximal CUTs $\boldsymbol{T}^\cap_{max}$. 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 $t^\cap \in \boldsymbol{T}^\cap$, we denote $\uparrow t^\cap$ the upper-set of $\boldsymbol{T}$ generated by PUTs in $t^\cap$, i.e.,

$$\uparrow t^\cap \stackrel{\text{def}}{=} \{t \in \boldsymbol{T} \mid \exists t' \in t^\cap, t' \lesssim t\} \tag{18}$$

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

- *if $t^\cap \in \mathbf{T}^\cap \backslash \varnothing - \bot^\sqcap$, then $\Uparrow t^\cap = \uparrow t^\cap$;*
- *if $t^\cap \in \bot^\sqcap$, then $\Uparrow t^\cap = \mathbf{T}$;*
- *$\Uparrow \varnothing = \Uparrow \top^\cap$;*

Proof. Let $t^\cap \in \boldsymbol{T}^\cap$:
1) if $t^\cap \in \boldsymbol{T}^\cap \backslash \varnothing - \bot^\sqcap$:
$\subseteq$: Let $t \in \Uparrow t^\cap$. By definition, $\exists t^\cap_x \in [t^\cap], t \in t^\cap_x$. So $t^\cap \mathrel{\hat{\simeq}} t^\cap_x$ and $t^\cap \mathrel{\hat{\lesssim}} t^\cap_x$.
From proposition 3.6, at least one of the followings holds:
i) $t^\cap \mathrel{\hat{\lesssim}} \bot^\cap$ (impossible from our hypothesis);
ii) $t^\cap = \varnothing$ (impossible from our hypothesis);

iii) $\forall t_x \in t^\cap_x, \exists t' \in t^\cap : t' \lesssim t_x$. So for $t \in t^\cap_x$, $\exists t' \in t^\cap : t' \lesssim t$. And thus $t \in \uparrow t^\cap$.
$\supseteq$: Let $t \in \uparrow t^\cap$. Then $\exists t' \in t^\cap, t' \lesssim t$. From lemma 4.10, $t^\cap \cup \{t\} \mathrel{\hat{\simeq}} t^\cap$, and thus $t^\cap \cup \{t\} \in [t^\cap]$, and thus $t \in \Uparrow t^\cap$.
2) if $t^\cap \in \bot^\sqcap$, then $\boldsymbol{T} \in [t^\cap]$ and $\Uparrow t^\cap = \boldsymbol{T}$;
3) if $\varnothing \mathrel{\hat{\simeq}} \top^\cap$, then from proposition 4.12, $\Uparrow \varnothing = \Uparrow \top^\cap$. □

Figure 5 illustrates the most interesting case in proposition 4.13 which is the case where $t^\cap \notin \bot^\sqcap$ and $t^\cap \neq \varnothing$: black dots represent PUTs of $t^\cap \in \boldsymbol{T}^\cap$, 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 dots, then the gray zone represents the set of PUTs that belong to $\Uparrow t^\cap$.

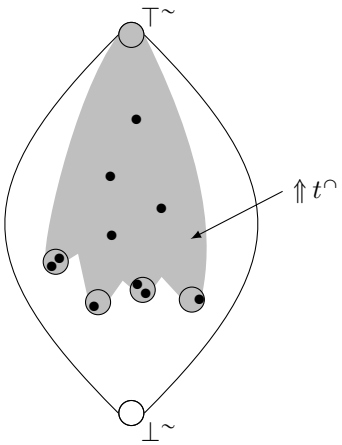


Figure 5: Illustration of a maximal CUT $\Uparrow t^\cap$ with $t^\cap \in \boldsymbol{T}^\cap \backslash \varnothing - \bot^\sqcap$.

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 $t^\cap$ is maximal if and only if one of the following is true:*

- $t^\cap = \mathbf{T}$,
- $t^\cap \in \mathbf{T}^\cap \backslash \varnothing - \bot^\sqcap$, *and $t^\cap$ is a upper set of* $\mathbf{T}$.

Proof. Let us prove both implications:
$\Rightarrow$: If $t^\cap$ is maximal, then there exists $t^{\cap\prime} \in \boldsymbol{T}^\cap, t^\cap = \Uparrow t^{\cap\prime}$.
i) if $t^{\cap\prime} \in \bot^\sqcap$, then $t^\cap = \boldsymbol{T}$;
ii) if $t^{\cap\prime} = \varnothing$, then $t^\cap = \Uparrow \bot^\cap$, and $\Uparrow \bot^\cap$ is either $\boldsymbol{T}$ or a non-empty upper set of $\boldsymbol{T}$;
iii) if $t^{\cap\prime} \in \boldsymbol{T}^\cap \backslash \varnothing - \bot^\sqcap$, then $t^\cap$ is an non-empty upper set of $\boldsymbol{T}$.
$\Leftarrow$:
i) if $t^\cap = \boldsymbol{T}$, then for all $t^{\cap\prime} \in \bot^\sqcap$, $t^\cap = \Uparrow t^{\cap\prime}$ so $t^\cap$ is maximal;
ii) if $t^\cap \notin \bot^\sqcap$, and $t^\cap$ is a non-empty upper set of $\boldsymbol{T}$, then using propositions 4.12 and 4.11, $\Uparrow t^\cap = t^\cap$, so $t^\cap$ is maximal. □

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

The last result of this section is the fact the operator $\Uparrow$ defines a closure operator on $\boldsymbol{T}$.

Proposition 4.15. *$\Uparrow$ is a closure operator on $\mathbf{T}$, i.e., it satisfies the following conditions for all CUTs $t^\cap_x, t^\cap_y \in \mathbf{T}^\cap$:*

- $t^\cap_x \subseteq \Uparrow t^\cap_x$, *($\Uparrow$ is extensive);*
- $t^\cap_x \subseteq t^\cap_y \implies \Uparrow t^\cap_x \subseteq \Uparrow t^\cap_y$, *($\Uparrow$ is increasing with respect to $\subseteq$);*
- $\Uparrow \Uparrow t^\cap_x = \Uparrow t^\cap_x$, *($\Uparrow$ is idempotent);*

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

2) *Increasing w.r.t. $\subseteq$:* Let $t^\cap_x, t^\cap_y \in \boldsymbol{T}^\cap$ such that $t^\cap_x \subseteq t^\cap_y$.
First, we know from proposition 3.7 that $t^\cap_y \overset{\cap}{\lesssim} t^\cap_x$.
i) If $t^\cap_y \in \bot^\sqcap$, then $\Uparrow t^\cap_y = \boldsymbol{T}$ (prop. 4.13) and $\Uparrow t^\cap_x \subseteq \Uparrow t^\cap_y$.
ii) If $t^\cap_x \in \bot^\sqcap$, then so is $t^\cap_y$, and $\Uparrow t^\cap_x = \Uparrow t^\cap_y = \boldsymbol{T}$ (prop. 4.13).
iii) If $t^\cap_y = \varnothing$, then $t^\cap_x = t^\cap_y = \varnothing$, and $\Uparrow t^\cap_x = \Uparrow t^\cap_y$.
iv) If $t^\cap_y \notin \bot^\sqcap$, $t^\cap_y \neq \varnothing$, $t^\cap_x \notin \bot^\sqcap$, and $t^\cap_x \neq \varnothing$: Let $t \in \Uparrow t^\cap_x = \uparrow t^\cap_x$. We know that $\exists t' \in t^\cap_x, t' \lesssim t$. As $t^\cap_x \subseteq t^\cap_y$, then $\exists t' \in t^\cap_y, t' \lesssim t$, and $t' \in \uparrow t^\cap_y = \Uparrow t^\cap_y$.
v) If $t^\cap_x = \varnothing$, then $\Uparrow t^\cap_x = \Uparrow \top^\cap$. $\top$ is a maximal element of $\boldsymbol{T}$, so using lemma 4.10, $t^\cap_y \overset{\cap}{\simeq} t^\cap_y \cup \top^\cap$, and from definition 4.2, $\Uparrow t^\cap_y = \Uparrow t^\cap_y \cup \top^\cap$. Using items i) to iv), we know that $\Uparrow t^\cap_x = \Uparrow \top^\cap \subseteq \Uparrow t^\cap_y \cup \top^\cap = \Uparrow t^\cap_y$.
So $\Uparrow t^\cap_x \subseteq \Uparrow t^\cap_y$, and $\Uparrow$ is increasing.

3) *Idempotent:* Let $t^\cap \in \boldsymbol{T}^\cap$.
i) if $t^\cap \in \bot^\sqcap$, then $\Uparrow t^\cap = \boldsymbol{T}$, and as we know that $\boldsymbol{T} \in \bot^\sqcap$, $\Uparrow \Uparrow t^\cap = \Uparrow t^\cap = \boldsymbol{T}$.
ii) if $t^\cap \notin \bot^\sqcap$ and $t^\cap \neq \varnothing$, then:
$\supseteq$*:* We know from extensivity of $\Uparrow$ that $\Uparrow t^\cap \subseteq \Uparrow \Uparrow t^\cap$.
$\subseteq$*:* Let $t \in \Uparrow \Uparrow t^\cap$. $\Uparrow t^\cap \overset{\cap}{\simeq} t^\cap$ so (prop: 4.9 $\Uparrow t^\cap \notin \bot^\sqcap$, and we also know that $t^\cap \subseteq \Uparrow t^\cap$ so $\Uparrow t^\cap \neq \varnothing$. Using upper-set properties: $\Uparrow \Uparrow t^\cap = \uparrow \Uparrow t^\cap = \uparrow \uparrow t^\cap = \uparrow t^\cap = \Uparrow t^\cap$.
iii) if $t^\cap = \varnothing$, then $\Uparrow t^\cap = \Uparrow \bot^\cap$, and using ii), $\Uparrow t^\cap = \Uparrow \bot^\cap = \Uparrow \Uparrow \bot^\cap = \Uparrow \Uparrow t^\cap$. □

Thus a maximal CUTs $t^\cap_\Uparrow$ is also maximal in the sense that any CUT of its equivalence class is contained in $t^\cap_\Uparrow$.

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 $t^\cap = \bot^\cap$.
- else, $\forall t \in t^\cap, \nexists t' \in t^\cap$ such that $t' < t$ and $\forall t, t' \in t^\cap$, $t \not\asymp t'$.

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 $t^\cap$.

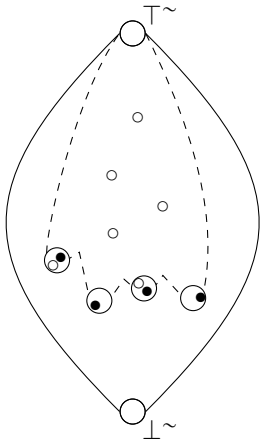


Figure 6: Let the union of black and white dots represent a CUT $t^\cap$. The set of black dots represents a concise CUT that is a subset of $t^\cap$.

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 Circumstancial 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: Circumstancial 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 $\boldsymbol{S_C}$.

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 $\boldsymbol{S_C}$)**.** The CSymbols set is pre-ordered by a relation $\overset{\mathcal{C}}{\lesssim}$, which is induced by a set of *asserted comparisons* $\boldsymbol{C_{S_C}} \subseteq \boldsymbol{T}^{\cap}$. $(\boldsymbol{S_C}, \boldsymbol{C_{S_C}})$ is a directed graph on $\boldsymbol{S_C}$. The pre-order $\overset{\mathcal{C}}{\lesssim}$ is equal to $\boldsymbol{C^*_{S_C}}$ the reflexo-transitive closure of $\boldsymbol{C_{S_C}}$, i.e., $\forall s, s' \in \boldsymbol{S_C}, s' \overset{\mathcal{C}}{\lesssim} s$ iif $(s, s') \in \boldsymbol{C^*_{S_C}}$, 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 trough a relation having this symbol.

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

We finally introduce the full hierarchy of CSymbols.

Definition 5.4 (CSymbols hierarchy)**.** A *Circumstancial Dependency Symbols (CSymbols) hierarchy*
$\mathcal{C} \overset{\text{def}}{=} (\boldsymbol{S_C}, \boldsymbol{C_{S_C}}, \mathcal{T}^{\cap}, \{\boldsymbol{\sigma}_s\}_{s\in\boldsymbol{S_C}})$, is composed of:

- $\boldsymbol{S_C}$ a set of Circumstancial Dependency Symbols (CSymbols);
- $\boldsymbol{C_{S_C}}$ a set of asserted comparisons;
- $\mathcal{T}^{\cap} = (T_{declared}, C_{asserted}, \bot^{\sqcap}_{asserted}, \boldsymbol{S_{\mathcal{T}}}, root, Hiders, \{\boldsymbol{\varsigma}_t\}_{t\in\boldsymbol{T}})$ a Conjunctive Unit Types (CUTs) hierarchy;
- $\{\boldsymbol{\sigma}_s\}_{s\in\boldsymbol{S_C}}$ 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 $\mathcal{S} \stackrel{\text{def}}{=} (\mathcal{T}^\frown, \mathcal{C}, \boldsymbol{I})$ where:

- $\mathcal{T}^\frown = (T_{declared}, C_{asserted}, \bot^\sqcap_{asserted}, \boldsymbol{S}_\mathcal{T}, root, Hiders, \{\varsigma_t\}_{t\in\boldsymbol{T}})$ is the *CUTs hierarchy*;
- $\mathcal{C} = (\boldsymbol{S}_\mathcal{C}, \boldsymbol{C}_{\boldsymbol{S}_\mathcal{C}}, \mathcal{T}^\frown, \{\boldsymbol{\sigma}_s\}_{s\in\boldsymbol{S}_\mathcal{C}})$ is the *CSymbols hierarchy*;
- $\boldsymbol{I}$ is the set of *individual markers*, which is disjoint from $\boldsymbol{T}$, $\boldsymbol{S}_\mathcal{T}$ and $\boldsymbol{S}_\mathcal{C}$. $*$ denotes the generic marker, with $* \notin \boldsymbol{I}$. $\boldsymbol{I} \cup \{*\}$ denotes the set of markers and is partially ordered as follows: $*$ is greater than any element in $\boldsymbol{I}$, and elements in $\boldsymbol{I}$ 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 $\boldsymbol{S}_\mathcal{T} \cup \boldsymbol{S}_\mathcal{C}$ 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 $\mathcal{S} \stackrel{\text{def}}{=} (\mathcal{T}^\frown, \mathcal{C}, \boldsymbol{I})$ 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 $(t^\frown, m) \in \boldsymbol{T}^\frown \times \boldsymbol{I} \cup \{*\}$. For all $u \in U$, u is a unit that has type $t^\frown$ and marker m. We denote $t^\frown = type(u)$ and $m = marker(u)$.
- P is the set of *participation triples* $(u, r, v) \in U \times \boldsymbol{S}_\mathcal{T} \times U$. 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 $u = governor(p)$, $r = symbol(p)$ and $v = participant(p)$. We also denote $arc(p) = (u, v)$.
- C is the set of *circumstantial dependency triples* $(u, r, v) \in U \times \boldsymbol{S}_\mathcal{C} \times U$. 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 $u = governor(c)$, $r = symbol(c)$ and $v = circumstant(c)$. We also denote $arc(c) = (u, v)$.
- The *underlying graph of* G denoted $graph(G)$ is a finite, oriented, labelled multigraph where:

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

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

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

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, $\boxed{\ulcorner \text{FALL} \urcorner_{\text{present}} : *}$ is a generic syntactic unit having type the conjunction of the lexical unit type $\ulcorner \text{FALL} \urcorner^L$, and the grammatical unit type $\ulcorner \text{present} \urcorner^G$.

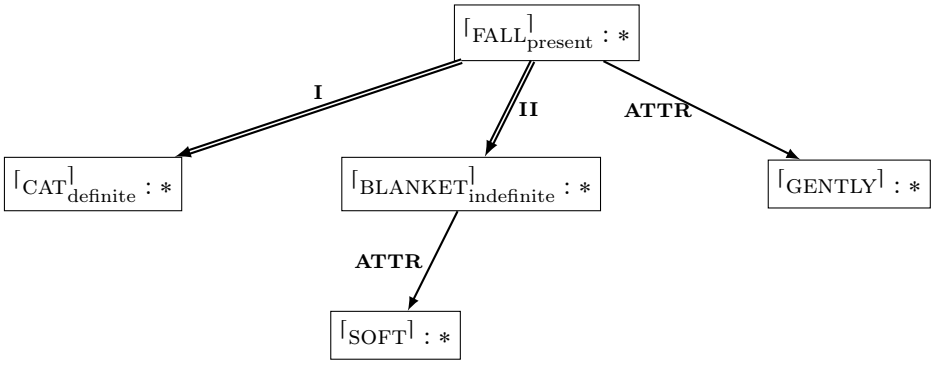


Figure 7: DSyn Representations of utterance *The cat falls gently on a soft blanket*

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 hiders 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 $rrot(s)$ 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. Finaly, 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 Ressources*, page 7–14.

Chein, 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.

Polguere, 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.

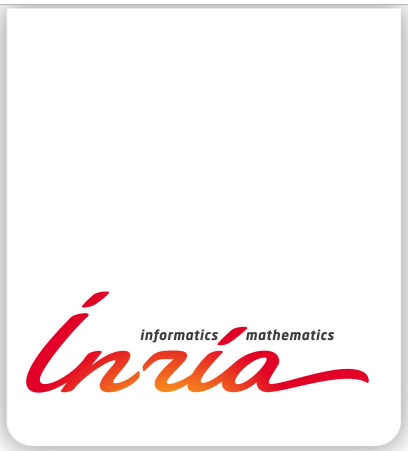
Polguère, A. (1998). La théorie sens-texte. *Dialangue*, 8-9:9–30.

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