Adding Communicative Structure to the MTT into ACG Encoding

Marie Cousin

Adding Communicative Structure to the MTT into ACG Encoding

Marie Cousin

1 Introduction

This chapter takes place in a context of encoding the meaning-text theory (MTT) into abstract categorial grammars (ACGs), and extends previous work presented in (Cousin 2023b; Cousin 2023a), whose ACG architecture we modify, and complement current work presented in (Cousin 2025b; Cousin 2025a), to provide an encoding of the theme-rheme opposition as described in MTT (Mel'čuk et al. 2012; Mel'cuk 2001; Polguère 1990) similarly to (Steedman 2024) works with CCGs on a theme-rheme opposition. This encoding concerns the semantic-syntax interface of MTT, and more specifically the semantic and deep-syntactic levels of MTT.

1.1 Context

This work falls within a text generation context with formal methods. The aims are to have strong control over generated text in order to ensure that it indeed conveys the message to be expressed, and to implement MTT's linguistic structures through which this control operates. In order to do so, we rely on a formal model, ACGs. Similar motivations are to be found in Grammatical Framework (Ranta 2004).
MTT is a linguistic theory aiming at describing the correspondence between the meaning of an utterance and its surface form. We use ACGs, a grammatical formalism based on λ-calculus, to implement a version of the syntax-semantic interface of MTT (see section 2). This implementation hinges upon abstract categorial grammars composition in order to encode level transitions with transduction operations.
(Cousin 2023b; Cousin 2023a) offer an encoding of MTT into ACGs, but only takes the predicative structures into account. However, MTT heavily relies on the communicative structures that come along the predicative ones. Theme and rheme play a crucial role in MTT as they determine which deep-syntactic graph will be obtained from the semantic one, e.g., producing a verbal phrase or a noun phrase, such as illustrated in section 2.3. This chapter extends Cousin's encoding with communicative structure information.

1.2 MTT

MTT (Mel'čuk et al. 2012; Mel'čuk et al. 2013; Mel'čuk et al. 2015, see section 2) is a linguistic theory describing the link between the meaning and the text of an utterance. MTT can be used for generation as well as for analysis purposes. It is composed of seven representation levels

(among which semantic and deep-syntactic levels) and six transition modules between these levels. MTT heavily relies on the key concepts of lexical functions (LF) and paraphrase.
To each level corresponds a representation. Each representation is composed of at least one predicative structure and one communicative structure. The predicative structure describes predicative dominance between governors and dependants at each levels. It can be a graph (semantic representation), a tree (syntactic representations) or strings (morphological and phonological representations). Communicative structures adds communicative oppositions to predicative structures, such as the theme-rheme opposition, also called thematicity. Theme-rheme opposition (Mel'cuk 2001; Polguère 1990), included in communicative structures, plays a crucial role in determining the deep-syntactic tree corresponding to the semantic graph of an utterance. Indeed, depending on the communicative structure, for two semantic representations sharing the same predicative structure, the obtained expression differ. It is the theme-rheme opposition, that, for instance, makes a copula appear (or not) in deep-syntax when an adjective modify a noun, and therefore decides if the adjective will be expressed as an attributive (" *[the **calm** Charlie]*$_{Rheme}$ ") or as a subject complement (" *[Charlie]*$_{Theme}$ *[is **calm***$]_{Rheme}$ "). Two expressions obtained from the same semantic predicative structure are different if they have different communicative structures. In this case, the two expressions are usually not paraphrases.

1.3 ACGs

ACGs (de Groote 2001, see section 3) are a grammatical framework based on λ-calculus. They enable the representation of other grammatical formalisms (Groote and Pogodalla 2004) like an implementation of TAGs into ACGs (Pogodalla 2017). A specific class of ACGs, ACGs of order 2 (see section 3), can be used in generation as well as in analysis (Kanazawa 2007): they are reversible. Their reversibility is a property we take avantage of.
ACGs represent trees, strings, but also other structures, such as logical formulas for example, as λ-terms. It offers an unified way to process them. If the written grammar changes along the structure, the algorithms used to process them for all operations (see section 3.2) remain the same. They can be used for different processes as well: for analysis, with the parsing of string structures, or for generation, like it is the case here where logical formulas and tree structures are parsed.
The encoding presented here uses a specific class of ACGs that is reversible: it can be used for generation purposes as well as for analysis ones. On top of that, some similarities can be found with other grammatical theories. That enables a comparison with these formalisms (or theories), and enables to draw from these formalisms for specific parts of our system. It is the case of CCGs (Steedman 2024) for the thematic opposition, or the case of (adverbial or adjectival modification) detailed in (Cousin 2025b; Cousin 2025a) inspired by Pogodalla's TAG encoding.

1.4 Related Work

Theme and Rheme Theme and rheme are recurring notions in linguistics, that are also called topic and comment depending on the used linguistic theory. They can be linked to intonation structures or to semantic structures of an utterance (Kruijff-Korbayova and Steedman 2003). The definitions we use here are the ones of MTT (see section 2.3), that, to our knowledge,

slightly differ with all other definitions. Additionally, some similarities can be found between theme-rheme opposition and information structure, but we won't go into detail on this subject.

Contextual Categorial Grammars (CCGs) Our encoding of theme-rheme opposition in ACG according to MTT is similar to Steedman's encoding of theme and rheme into CCGs (Steedman 2024). As stated earlier, the definitions we use slightly differ from the ones he uses. Hence, the theme-rheme marking of our predicates and constants differ accordingly. However, we did encode the theme-rheme opposition similarly, *i.e.*, in the predicates' types, with indexes (that stand for theme or rheme) indicating which dependants of each predicate is labelled as theme or rheme.

Other Systems Encoding MTT Other systems, also used in a generation context, encode MTT. For instance, MARQUIS (Wanner et al. 2010) generates weather forecast, and is based on MATE (Bohnet, Langjahr, and Wanner 2000), a development environment for MTT-based sentence generator. GenDR (Lareau et al. 2018) is another system that enables sentence generation with MTT-based lexicon and dictionaries. All these systems are based on graph transducers, encode MTT communicative structure, and can be used for generation purposes, but their generation process is not reversible.

1.5 Former and Current Work

This chapter presents the part of our MTT into ACG encoding that focuses on the theme-rheme opposition. Hence, it is part of the larger project encoding the whole MTT semantic to surface syntax interface into ACGs.
The ACG architecture of previous work is extended here in order to add the theme-rheme encoding to the project (see section 4.2). Cousin (2023b) presents the syntax-semantic interface that we extend in this chapter, with a focus on the semantic and deep-syntactic encodings, as well as the deep-syntactic paraphrasing. Cousin (2023a) presents the way LFs are handled and realized, and focuses on the deep-syntactic and surface-syntactic encodings.
In order to take adverbial and adjectival modification into account, and encode the project with ACGs of order 2 (the ones that are reversible), and not higher-order ACGs, we rewrote both deep-syntactic and surface-syntactic levels encodings. Cousin (2025b) details the surface-syntax encoding, while Cousin (2025a) details the deep-syntactic one. Both articles present how modification operates.

1.6 Contributions

This article present the encoding of MTT's thematic communicative structure into ACGs, with a focus on the semantic structures. It reuses original aspects of MTT, such as paraphrase and LFs (that enable the representation and handling of idiomatic expressions for instance). We are interested in the way linguistic structures, especially MTT's ones, can be expressed in ACGs. Our contributions are the following:

- an encoding of the theme-rheme opposition within ACGs;

- an encoding that naturally falls within the ACG architecture of Cousin (2023a), whose transduction between semantic and syntactic structures is more faithful to MTT thanks to the thematicity encoding.
- an ACG encoding in the ACGtk software, with files available here[1].

Sections 2 and 3 respectively present MTT and ACGs. Section 4 details the used ACG architecture and the thematicity encoding. Section 5 presents obtained results and section 6 our perspectives before the conclusion in section 7.

2 Meaning-Text Theory

In MTT (Mel'čuk et al. 2012; Mel'čuk et al. 2013; Mel'čuk et al. 2015), the meaning is "a linguistic content to be communicated" and the text is "any fragment of speech" (Jasmina Milićević 2006).

2.1 Overview

MTT uses a meaning-text model (MTM, see figure 1), composed of seven representation levels and six transition modules between each of them. Each representation level corresponds to an eponym utterance representation, namely the semantic representation (SemR), deep-syntactic representation (DSyntR), surface-syntactic representation (SSyntR), deep-morphologic representation and surface-morphologic representation (resp. DMorphR and SMorphR), deep-phonologic representation and surface-phonologic representation (resp. DPhonR and SPhonR). Between each pair of adjacent levels operates a transition module, named after the level (of the

$$\overrightarrow{\hspace{8em}\textit{Generation}\hspace{8em}}$$
$$\text{SemR} \longleftrightarrow \text{DSyntR} \longleftrightarrow \text{SSyntR} \longleftrightarrow \text{DMorphR} \longleftrightarrow \text{SMorphR} \longleftrightarrow \text{DPhonR} \longleftrightarrow \text{SPhonR}$$
$$\overleftarrow{\hspace{8em}\textit{Analyse}\hspace{8em}}$$

Figure 1: Hierarchy of MTT's model (Mel'čuk et al. 2012)

pair) nearest to the semantic level. For instance, the semantic transition module operates the transition between SemR and DSyntR.
Each representation level is made of several substructures, among which a predicative structure and a communicative structure. The latter bears (among other communicative oppositions) the theme-rheme opposition (Polguère 1990), that decorates and complements the predicative structure (see section 2.3).
A transition module carry out all necessary operations between two levels to transform the previous representation into the next one. It handles the structure change, such as the semantic module that transforms semantic graphs into deep-syntactic trees, but can also perform other operations such as paraphrase steps (see section 2.2.3).
MTT uses key concepts of LFs (Mel'Čuk and Polguère 2021) and paraphrase (Iordanskaja, Kittredge, and Polguère 1991; J. Milićević 2007). LFs aim at describing linguistic phenomena that exist in all languages. They are functions describing relations between lexical units

[1]The link is not given here for anonymity issues but will be in the final version.

(LU) and associate with that LU the set of all other alternative choices of LUs consistent with the relation they describe. For example, `Anti` and `Copul` describe antonyms and copula respectively: `anti`(CALM) = {UPSET, RESTLESS} and `copul`(CALM) = {BE}. LFs hinge on semantics, syntax and morphology. They are part of the lexicon of the language, as well as part of its grammar. Paraphrases are two expressions that share the same meaning. In the generation process of MTT, paraphrases can be produced during level transition (one previous structure produces several new ones), or when a structure can be be transformed to produce a different expression at the same level. The deep-syntactic paraphrase is of the latter kind for instance.
This work focuses on the semantic and deep-syntactic representation layers, and the semantic module that bridges them.

2.2 Semantic Representation & Deep-Syntactic Representation

2.2.1 Semantic Representation (SemR)

A SemR (Mel'čuk et al. 2012) consists of four substructures (see figure 2), namely: the semantic structure (SemR), the semantic communicative structure (Sem-CommS), the rhetorical structure (RhetS) and the referential structure (RefS). In this work, we only consider SemS and Sem-CommS.

$$\text{SemR} = \langle\ \text{SemS, Sem-CommS, RhetS, RefS}\ \rangle$$

Figure 2: Composition of a SemR

SemS are semantic predicative graphs, whose nodes are semantemes and edges are semantic relations (see examples on figure 3). A semanteme ‘word’ is "the signified of a full LU WORD. A semanteme ‘word’ stands in a one-to-one correspondance with the lexical unit (LU) it constitutes: ‘word’ ⇔ WORD" (Mel'čuk et al. 2012, p. 40). Semantic relations number the dependants of the governor in the order given by its semantic actantial structure. For instance, **1** corresponds to the first semantic actant and **2** to the second.

(a) SemS (b) SemS with thematic opposition

Figure 3: Illustration of a SemS without a Sem-CommS (a) and with a Sem-CommS (b), associated with the expression " *Charlie is calm* ".

Sem-CommS is represented by markers that decorate the SemS with communicative oppositions. For instance, theme and rheme markers are represented as boxes that encapsulate subgraphs (see figure 3b). The subgraph in the theme-box (respectively rheme-box) is labeled as theme (resp. rheme) and therefore represents the theme (resp. rheme). Figure 3a does not posess a thematic opposition. Therefore, any expression sharing this predicative structure can be associated to it, regardless of the thematic opposition of the expression: figure 3a can be associated to both expressions " *the calm Charlie* " and " *Charlie is calm* ". Contrarily, figure 3b posesses a thematic opposition. Therefore, it can only be associated with the latter expression.

2.2.2 Deep-Syntactic Representation (DSyntR)

A DSyntR (Mel'čuk et al. 2013) consists of four substructures (see figure 4), namely: the deep-syntactic structure (DSyntS), the deep-syntactic communicative structure (DSynt-CommS), the deep-syntactic prosodic structure (DSynt-ProsS) and the deep-syntactic anaphoric structure (DSynt-AnaphS). In this work, we only consider DSyntS and DSynt-CommS.

$$\text{DSyntR} = \langle\ \text{DSyntS, DSynt-CommS, DSynt-ProsS, DSynt-AnaphS}\ \rangle$$

Figure 4: Composition of a DSyntR

DSyntS are dependency tree structures, whose nodes are deep LU and edges are deep-syntactic relations (see figure 5). A deep LU "can be a lexeme (= a word taken in one specific sense), a (full) phraseme (= an idiom) [...], or [...] a lexical function." (Jasmina Milićević 2006). Phraseme and idioms can be seen as multi-word expressions. Deep-syntactic relations are mainly actantial (*actant* **I**, *actant* **II**, etc.), attributive (**ATTR**), or coordinative (**COORD**) relations.

(a) a DSyntS

(b) DSyntS with thematic opposition

Figure 5: Illustration of a DSyntS without a DSynt-CommS (a) and with a DSynt-CommS (b), associated with the expression " *Charlie is calm* ".

DSynt-CommS is represented by markers that decorate the DSyntS with communicative oppositions. For instance, theme and rheme markers are represented as boxes that encapsulates subtrees (see figure 5). The subtree in the theme-box (respectively rheme-box) is labeled as theme (resp. rheme) and therefore represents the theme (resp. rheme).

2.2.3 Semantic Transition Module

The semantic module performs the transition between SemR and DSyntR. For instance, in the case of the expression " *Charlie is calm* ", it will perform the transition between figure 3b and figure 5b. Hence, the semantic module performs structural operations to transform SemS into DSyntS. While doing so, some LF might appear, such as the ones expressing support verbs (like $\texttt{Oper}_i$ or `Copul`), that are semantically empty and therefore no semanteme represent them in SemS (see figure 5).
On top of such structural operations, the semantic module also performs paraphrase steps, such as semantic paraphrasing and deep-syntactic paraphrasing. Semantic paraphrasing can bee seen as unfolding the semantic graph with semanteme definitions. Deep-syntactic paraphrasing relies heavily on LFs. Both paraphrase steps are described in (J. Milićević 2007).

2.3 Theme and Rheme

Theme and rheme (Polguère 1990) are the markers of the thematic opposition in MTT and play a crucial role in the generation process of MTT.

Definition 1 *In a sentence, the **theme** is what we are talking about. It fills the blank in "Speaking of ...", when the **rheme** is what we are saying about the theme and answers the question "What is said about theme ?".*

(a) "*Charlie is Taylor's son.*" (b) "*Charlie, the son of Taylor.*"

Figure 6: Illustration of different theme and rheme areas on the same SemS

The realization of a semantic representation (SemR) as a verbal or nominal expression depends on the SemR's theme-rheme partition. For instance, " *Charlie is Taylor's son* " talks about Charlie (the theme), and says that he is Taylor's son (the rheme), while " *Charlie, the son of Taylor* " does not have a communicative opposition. The same can be observed for figure 3, where figure 3a may be associated to several expressions while the theme-rheme opposition in figure 3b enforces the associated expression, " *Charlie is calm* ".

(a) "*Charlie invites Taylor.*" (b) "*Taylor is invited by Charlie.*"

Figure 7: Illustration of different theme and rheme areas on the same SemS

Another case where thematicity plays a decisive role is when the speaker talks about one specific argument of a semantic predicate (see figure 7). For instance, the meaning of " *Charlie invites Taylor* " is different than the one of " *Taylor is invited by Charlie* ". The former is talking about Charlie (the theme) who invites Taylor (the rheme) (see figure 7a). The latter is talking about Taylor (the theme), who is invited by Charlie (the rheme) (see figure 7b).

3 Abstract Categorial Grammars (ACGs)

ACGs (de Groote 2001, whose definitions are reminded here) are a grammatical framework based on λ-calculus. They enable the implementation of other grammatical formalisms and can be used in generation or analysis (Kanazawa 2007), what has proven useful to model MTT (Cousin 2023a).

3.1 Definitions

3.1.1 Types, Signatures, and Lexicon

Definition 2 *Let A be a set of atomic types. $\mathscr{T}(A)$ is the set of* ***implicative types****, and $\mathscr{T}^0(A) \subset \mathscr{T}(A)$ is the set of* ***linear implicative types****, both obtained inductively over A:*

- *if $a \in A$ then $a \in \mathscr{T}^0(A)$ and $a \in \mathscr{T}(A)$;*
- *if $\alpha, \beta \in \mathscr{T}^0(A)$ then $(\alpha \to \beta) \in \mathscr{T}^0(A)$ and $(\alpha \to \beta) \in \mathscr{T}(A)$;*
- *if $\alpha, \beta \in \mathscr{T}(A)$ then $(\alpha \Rightarrow \beta) \in \mathscr{T}(A)$.*

Definition 3 *A* ***higher order signature*** *Σ is a tuple $\Sigma = \langle A, C, \tau \rangle$, where:*

- *A is a set of atomic types;*
- *C is a set of constants;*
- *$\tau : C \longrightarrow \mathscr{T}(A)$ is a function associating a type to a constant.*

Σ is a linear higher order signature if $\tau : C \longrightarrow \mathscr{T}^0(A)$.

Figure 8 below shows terms that have atomic type (like G in $\Sigma_{\textit{deep-syntactic}}$). It also shows complex terms, like $\textit{invite}^{ds}$, that enable the construction of a complex structure.

$$\Sigma_{\textbf{semantic}}$$

$$\begin{array}{rcl} n, t & : & \textit{type} \\ \exists & : & (n \Rightarrow t) \to t \\ \wedge & : & t \to t \to t \\ \mathbf{1}, \mathbf{2} & : & n \to n \to t \\ \textit{charlie}^s, \textit{taylor}^s & : & n \Rightarrow t \\ \textit{invite}^s, \textit{son}^s & : & n \Rightarrow t \end{array}$$

$$\Sigma_{\textbf{deep-syntactic}}$$

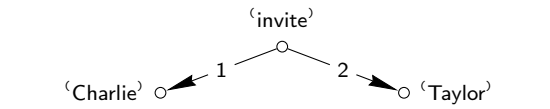
$$\begin{array}{rcl} G, G' & : & \textit{type} \\ \textit{charlie}^{ds} & : & G \\ \textit{taylor}^{ds} & : & G \\ \textit{invite}^{ds} & : & G' \to G \to G \\ \textit{son}^{ds} & : & G' \to G \to G \end{array}$$

Figure 8: Extract of two higher-order signatures. *(Type G' is for optionally expressible semantic arguments. The difference with type G is not relevant for the purpose of this section, and it can be ignored.)*

Definition 4 *Let X be a countably infinite set of λ-variables. The set $\Lambda(\Sigma)$ of* ***λ-terms*** *built upon a higher-order signature $\Sigma = \langle A, C, \tau \rangle$ is inductively defined:*

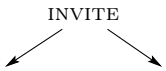
- *if $c \in C$ then $c \in \Lambda(\Sigma)$;*
- *if $x \in X$ then $x \in \Lambda(\Sigma)$;*
- *if $x \in X$ and $t \in \Lambda(\Sigma)$ and x occurs free in t exactly once, then $\lambda^0 x.\, t \in \Lambda(\Sigma)$;*
- *if $x \in X$ and $t \in \Lambda(\Sigma)$ then $\lambda x.\, t \in \Lambda(\Sigma)$;*
- *if $t, u \in \Lambda(\Sigma)$ then $(tu) \in \Lambda(\Sigma)$;*

The linear abstraction is denoted λ^0. Notation $\vdash_\Sigma t : s$ expresses that the type of the λ-term t is s in the signature Σ. If there is no ambiguity on the used signature, we can use the notation $t : s$. We use $\beta\eta$-equivalence as equality between λ-terms.



$\gamma^{s}_{cit} = \lambda^{0}\, e0.\ \exists e_x e_y.\ (invite^{s}\ e_0) \wedge (R_1\ e_0\ e_x) \wedge (R_2\ e_0\ e_y) \wedge (charlie^{s}\ e_x) \wedge (taylor^{s}\ e_y)$

(a) λ-term built upon $\Lambda(\Sigma_{semantic})$



$\gamma^{ds}_{cit} = invite^{ds}\ charlie^{ds}\ taylor^{ds}$

(b) λ-term built upon $\Lambda(\Sigma_{deep-syntacic})$

Figure 9: Example of two λ-terms built upon the signatures $\Sigma_{semantic}$ (a) and $\Sigma_{deep\text{-}syntactic}$ (b), and associated graphic representations.

Figure 9 shows two examples of λ-terms built upon the signatures from figure 8. Both terms represent the same expression " *Charlie is calm* ". The term of figure 9a is built upon $\Sigma_{semantic}$ and encode a semantic graph while the term of figure 9b is built upon $\Sigma_{deep\text{-}syntactic}$ and encode a syntactic structure.

Definition 5 *Let Σ_1 and Σ_2 be two signatures. A **lexicon** $\mathscr{L}_{12}$ from Σ_1 to Σ_2 is a pair of morphisms $\langle F, G\rangle$ such that:*

- $F\ :\ \tau(A_1) \longrightarrow \tau(A_2)$;
- $G\ :\ \Lambda(\Sigma_1) \longrightarrow \Lambda(\Sigma_2)$.

We write $\mathscr{L}_{12}(t) = \gamma$ to express that γ is the interpretation of t by $\mathscr{L}_{12}$. If there is no ambiguity on the used lexicon, we can write $t := \gamma$. Both notations are used when t and γ are types and the used morphism is F, but also when t and γ are terms and it is G that is used.

Figure 10 shows an example of a lexicon, with simple interpretations, like the one of $charlie^{ds}$ but also more complex ones, that detail complex graph construction, like the one of $invite^{ds}$. It also details the calculus of the interpretation of $\gamma^{d}_{cit}s$ (figure) by $\mathscr{L}_{sem}$: $\mathscr{L}_{sem}(\gamma^{ds}_{cit}) = \gamma^{s}_{cit}$.

$\mathcal{L}_{sem}$

$$
\begin{aligned}
G, G' &:= n \Rightarrow t \\
charlie^{ds} &:= \lambda\ e0.\ (charlie^{s}\ e0) \\
taylor^{ds} &:= \lambda\ e0.\ (taylor^{s}\ e0) \\
invite^{ds} &:= \lambda^{0}\ x\ y.\ \lambda\ e_0.\ (\exists\ e_x.\exists\ e_y.\ (invite^{s}\ e_0) \wedge (R_1\ e_0\ e_x) \wedge (R_2\ e_0\ e_y) \wedge (x\ e_x) \wedge (y\ e_y)) \\
son^{ds} &:= \lambda^{0}\ x\ y.\ \lambda\ e_0.\ (\exists\ e_x.\ \exists\ e_y.\ (son^{s}\ e_0) \wedge (R_1\ e_0\ e_x) \wedge (R_2\ e_0\ e_y) \wedge (x\ e_x) \wedge (y\ e_y))
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{L}_{sem}(\gamma^{ds}_{cit}) &= \mathcal{L}_{sem}(invite^{ds}\ charlie^{ds}\ taylor^{ds}) \\
&= \mathcal{L}_{sem}(invite^{ds})(\mathcal{L}_{sem}(charlie^{ds}))(\mathcal{L}_{sem}(taylor^{ds})) \\
&= (\lambda^{0}\ x\ y.\ \lambda\ e_0.\ \exists\ e_x.\exists\ e_y.\ (invite^{s}\ e_0) \wedge (R_1\ e_0\ e_x) \wedge (R_2\ e_0\ e_y) \wedge (x\ e_x) \wedge (y\ e_y)) \\
&\quad (\lambda\ e0'.\ (charlie^{s}\ e0'))(\lambda\ e0''.\ (taylor^{s}\ e0'')) \\
&= (\lambda\ e_0.\ \exists\ e_x.\exists\ e_y.\ (invite^{s}\ e_0) \wedge (R_1\ e_0\ e_x) \wedge (R_2\ e_0\ e_y) \wedge ((\lambda\ e0'.\ (charlie^{s}\ e0'))e_x) \wedge ((\lambda\ e0''.\ (taylor^{s}\ e0''))\ e_y)) \\
&= \lambda\ e_0.\ \exists\ e_x.\exists\ e_y.\ (invite^{s}\ e_0) \wedge (R_1\ e_0\ e_x) \wedge (R_2\ e_0\ e_y) \wedge ((charlie^{s}\ e_x)) \wedge ((taylor^{s}\ e_y)) \\
&= \gamma^{s}_{cit}
\end{aligned}
$$

Figure 10: Extract of $\mathscr{L}_{sem}$ (from $\Sigma_{deep\text{-}syntactic}$ to $\Sigma_{semantic}$) and interpretation of γ^{ds}_{cit} by $\mathscr{L}_{sem}$

3.1.2 ACG

Definition 6 *An **abstract categorial grammar** is a tuple $\mathscr{G} = \langle \Sigma_1, \Sigma_2, \mathscr{L}_{12}, s\rangle$ where:*

- $\Sigma_1 = \langle A_1, C_1, \tau_1\rangle$ *and* $\Sigma_2 = \langle A_2, C_2, \tau_2\rangle$ *are two higher-order signatures;*
- $\mathscr{L}_{12} = \Sigma_1 \longrightarrow \Sigma_2$ *is a lexicon;*

- $s \in \mathscr{T}(A_1)$ *is the distinguished type of the grammar.*

Figure 11 gives a representation of the ACG made of $\Sigma_{semantic}$, $\Sigma_{deep\text{-}syntactic}$ and $\mathscr{L}_{sem}$. The distinguished type of this ACG is G.

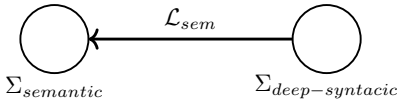


Figure 11: Illustration of an ACG

Definition 7 *The **abstract language** $\mathscr{A}$ and the **object language** $\mathscr{O}$ of an ACG $\mathscr{G} = \langle \Sigma_1, \Sigma_2, \mathscr{L}_{12}, s \rangle$ are:*

- $\mathscr{A} = \{t \in \Lambda(\Sigma_1) | \vdash_{\Sigma_1} t : s \text{ is derivable}\}$;
- $\mathscr{O} = \{t \in \Lambda(\Sigma_2) | \exists u \in \mathscr{A}(\mathscr{G}) \text{ such that } t = \mathscr{L}_{12}(u)\}$.

Both abstract and object languages of an ACG are sets of λ-terms obtained by induction over the abstract and object signatures respectively. The abstract language is the set of abstract grammatical structures, such as analysis trees, whereas the object language is the set of the surface representations generated by the abstract language, such as strings or logical representations in the form of a graph. For example, the object language of the ACG made of $\Sigma_{deep\text{-}syntactic\text{-}tr}$ (its abstract signature), $\Sigma_{semantic\text{-}tr}$ (its object signature), and $\mathscr{L}_{semRel}$ (illustrated in figure 12) is the set of semantic graphs.

3.2 Operations

In figure 12, $\langle\ \Sigma_{deep\text{-}syntacic}, \Sigma_{semantic}, \mathscr{L}_{sem}, G\ \rangle$ and $\langle\ \Sigma_{deep\text{-}syntacic}, \Sigma_{dsynt\text{-}tree}, \mathscr{L}_{dsyntRel}, G\ \rangle$ are two ACGs that share the same abstract signature, $\Sigma_{deep\text{-}syntacic}$. Figure 13 gives extacts of $\Sigma_{dsynt\text{-}tree}$ and $\mathscr{L}_{dsyntRel}$. Three operations are defined over ACGs (see figure 12).

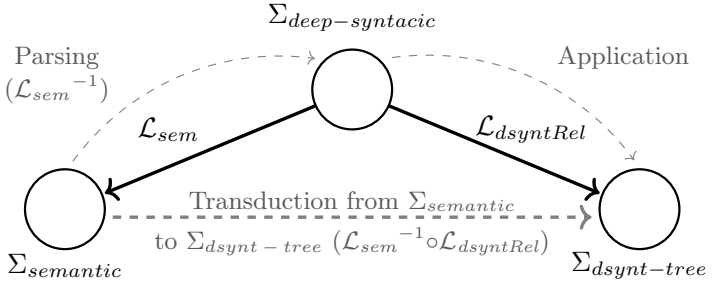


Figure 12: Example of application, parsing and transduction operations

Application is the operation of applying the lexicon of an ACG from its abstract signature to its object signature. For instance $\mathscr{L}_{sem}(\gamma^{ds}_{cit}) = \gamma^{s}_{cit}$ (see figure 10). Provided the grammar is reversible so is the lexicon (the ACG needs to belong to a specific class of ACG, ACGs of order 2, see below). The operation of reversing the lexicon from the object signature of an ACG to its abstract signature is called parsing. For instance, $\mathscr{L}^{-1}_{sem}(\gamma^{s}) = \gamma^{ds}$.

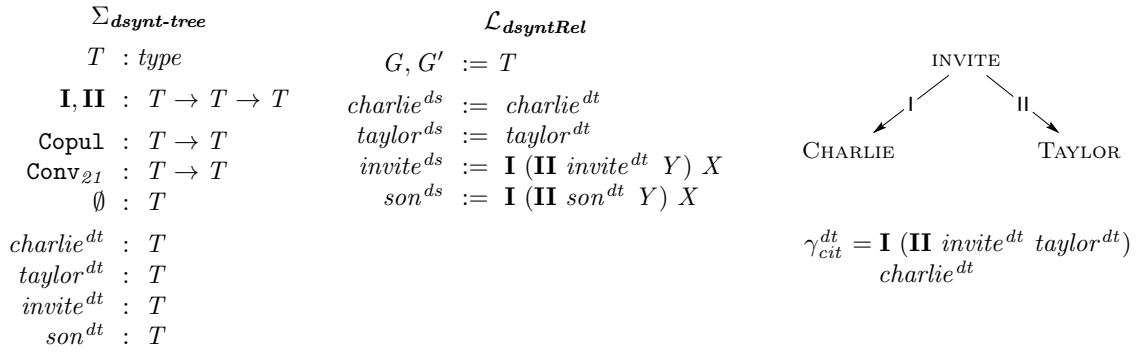


Figure 13: Extracts of $\Sigma_{dsynt\text{-}tree}$ and $\mathscr{L}_{dsyntRel}$, and illustration of a λ-term build upon $\Sigma_{dsynt\text{-}tree}$

Both operations (application and parsing) can be composed to form a transduction operation: $\mathscr{L}_{dsyntRel}(\mathscr{L}^{-1}_{sem}(\gamma^s)) = \gamma^{dt}$ (figure 13 gives the expression of γ^{dt}_{cit}).
An ACG is characterized by its order and its complexity. The complexity of an ACG is its parsing complexity.

Definition 8 *The order of a type $\tau \in \mathscr{T}(A)$ is inductively defined:*

- $order(\tau) = 1$ *if* $\tau \in A$,
- $order(\alpha \to \beta) = order(\alpha \Rightarrow \beta) = max(1 + order(\alpha), order(\beta))$ *otherwise.*

The order of an abstract constant is the order of its type, and the ***order of an ACG*** *is the maximum of the order of its abstract constants.*
The ***complexity*** *of an ACG is the maximum of the orders of its atomic type realizations. An ACG of order γ and of complexity η is written $ACG_{(\gamma,\eta)}$.*

This encoding aims to use a specific class of ACGs, ACGs said of order two, since their parsing complexity is decidable and polynomial (Salvati 2005; Kanazawa 2017).

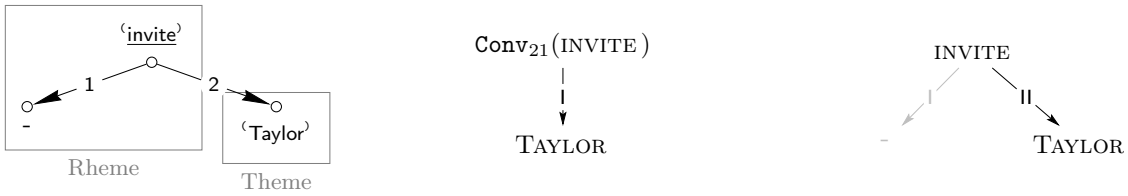
4 ACG Encoding of MTT

We focus on Cousin (2023a)'s transduction between semantic and deep-syntax (see figure 12) and illustrate how we extend it (see figure 16). In MTT, two representations from two adjacent levels stand in a transition relation with each other. We encode these transitions with a transduction relation. For example, the transduction relation $\mathscr{R}$ in figure 12 between a SemR x and a DSyntR y is such that : $\mathscr{R} = \{(x,y) | \exists u. \mathscr{L}_{sem}(u) = x \text{ and } \mathscr{L}_{dsyntRel}(u) = y\}$. Then, the set of DSyntR that corresponds to a SemR x is $\{y | (x,y) \in \mathscr{R}, y = \mathscr{L}_{dsyntRel}(\mathscr{L}^{-1}_{sem}(x))\}$. Since each level interface is a relation in MTT, we encode each transition relation from MTT with a transduction relation. Additionally, several steps require an additional transduction relation: a rewriting of the terms to perform deep-syntactic paraphrasing, the deep-syntactic paraphrasing step, and lexical function realization (see figure 15).

4.1 Problem

Figure 12 shows the ACG architecture of (Cousin 2023a) for the encoding of MTT's semantic to deep-syntax interface. $\Sigma_{semantic}$ contains constants to represent the predicative semantic graphs

(SemS), i.e., mostly semantemes and semantic relations (see figure 8). Constants of $\Sigma_{dsynt\text{-}tree}$ are the ones to represent MTT's DSyntS, i.e., mostly lexemes, deep-syntactic relations and lexical functions (see figure 13). $\Sigma_{deep\text{-}syntactic}$ is the abstract signature used to implement the transduction between $\Sigma_{semantic}$ and $\Sigma_{dsynt\text{-}tree}$ (see figure 8). Figure 10 gives the interpretation of some constants of $\Sigma_{deep\text{-}syntacic}$ by $\mathscr{L}_{sem}$, and figure 13 gives the interpretation of some constants of $\Sigma_{deep\text{-}syntacic}$ by $\mathscr{L}_{dsynt\text{-}tree}$.
Hence, by parsing of $\mathscr{L}_{sem}$ and application of $\mathscr{L}_{dsyntRel}$, all DSyntS associated with a SemS are obtained. This requires that the SemS has an antecedent by $\mathscr{L}_{sem}$ (see section 4.3.3).
Cousin (2023a)'s encoding do not enable thematicity representation. For instance, γ^{s}_{cit}, and γ^{dt}_{cit} (see figures 9 and 13), that are in transduction relation (see section 3.2), can be associated with both expressions " *Charlie invites Taylor* " and " *Taylor is invited by Charlie* ", whereas these expressions differ. In the former, the theme is " *Charlie* ", and in the latter, the theme is " *Taylor* ". In MTT, the passive and active form of an expression do not share the same SemR. We need a more fine-grained encoding to tackle this issue.
The fact that thematicity is not accounted is especially problematic in several cases as described in section 2.3. For example, when a choice between the use of a copula or a noun phrase is needed (see figure 6 with ‘son’), both expressions are clearly not paraphrases (" *Charlie is Taylor's son* " and " *Charlie, the son of Taylor* "), and should not be associated to the same semantic encoding. In this encoding however, they share the same semantic encoding. Another example is the case of unexpressed optional semantic arguments, like in " *Taylor is invited* ". The first actant of ‘invite’ is not expressed (see figure 14). Its corresponding DSyntR in MTT (see figure 14b) only has one syntactic actant, and is different from the one obtained by Cousin (2023a) (see figure 14c).



(a) SemS and Sem-CommS associated with "*Taylor is invited.*"

(b) DSyntS that should be associated with "*Taylor is invited.*"

(c) DSyntS (incorrectly) associated with "*Taylor is invited.*"

Figure 14: SemS and DSyntS of " *Taylor is invited* " *(Conv$_{21}$: see (Mel'Čuk and Polguère 2021))*

To account for the theme-rheme opposition, we propose to extend Cousin (2023a)'s encoding in the following sections by adding type information corresponding to the communicative structure. Figure 15 shows the new hierarchy of ACGs with contants being typed according to their thematicity in Σ_{dst}, and figure 16 focuses on the semantic to deep-syntax interface. We use composition capabilities of ACGs to have the theme-rheme opposition information percolate from Σ_{dst} to Σ_{s} and Σ_{dt}. Σ_{s} corresponds to $\Sigma_{semantic}$, and Σ_{dt} to $\Sigma_{dsynt\text{-}tree}$, whose encoding barely change.

4.2 General Overview

Figure 15 shows the extended architecture and composition of (Cousin 2023a) that we use here. It is divided into five areas, this chapter focuses on areas 1 and 2:

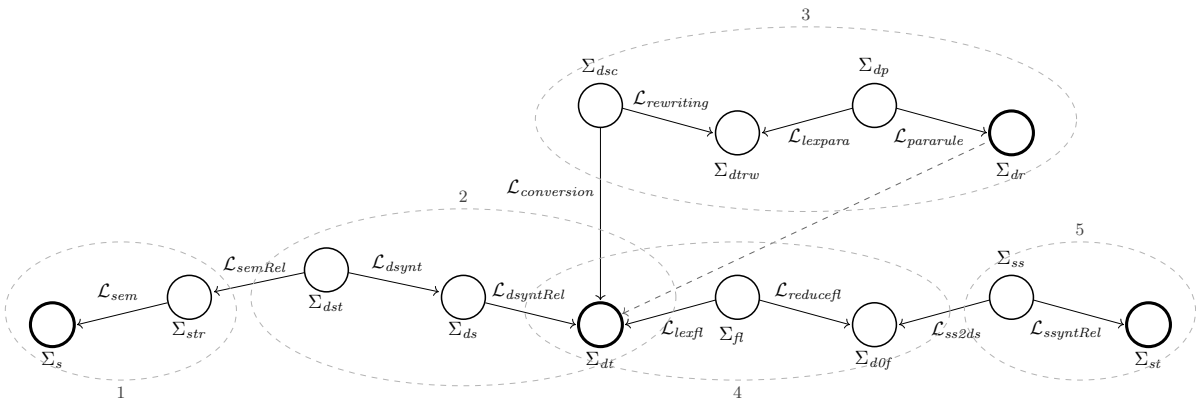


Figure 15: ACG composition of MTT encoding

- Areas 1,2: transduction between Σ_{str} (semantic with thematicity) and Σ_{dt} (deep-syntactic trees) allows transitions between SemR and DSyntR. Area 1 corresponds to semantic and area 2 to deep-syntax.
- Area 3: transduction between Σ_{dtrw} (deep-syntactic trees after rewriting) and Σ_{dr} performs deep-syntactic paraphrase steps, and transduction between Σ_{dt} and Σ_{dtrw} performs a rewriting of all terms, more suited to the encoding of the deep-syntactic paraphrasing;
- Area 4: transduction between Σ_{dt} and Σ_{d0f} (deep-syntactic trees without LF) realizes LFs eventually present in DSyntR;
- Area 5: transduction between Σ_{d0f} and Σ_{st} (surface-syntacic trees) transforms DSyntR (without LFs) into SSyntS. Area 5 corresponds to surface-syntax.

A constant representing a lexeme LEX in the signature Σ_S is written lex^S. For instance, $charlie^{dst}$ is the constant representing the lexeme CHARLIE in Σ_{dst}. A term representing an expression E in $\Lambda(\Sigma_S)$ is written γ_E^S. For instance, the term encoding the SemR associated to the expression (cit): " *Charlie invites Taylor* " in Σ_{str} is written γ_{cit}^{str}.

4.3 Semantic to Deep-Syntax Interface

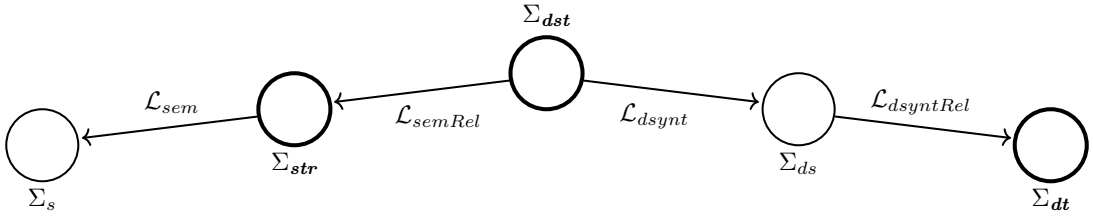


Figure 16: Hierarchy of the syntax-semantic interface of the implementation of MTT within ACGs that takes theme and rheme into account

Σ_{str} enables the representation of SemR, Σ_s enables the representation of SemS, and Σ_{dt} enables the representation of DSyntS. Σ_{dst} is the abstract signature used to encode the transduction relation between Σ_{str} and Σ_{dt}. When two ACG are functionally composed, we can consider it as one single ACG. It is the case between Σ_{dst} and Σ_{dt}, where Σ_{ds} is both the object signature of the

ACG $\langle \Sigma_{dst}, \Sigma_{ds}, \mathscr{L}_{dsynt}, G \rangle$ and the abstract signature of another ACG: $\langle \Sigma_{ds}, \Sigma_{dt}, \mathscr{L}_{dsyntRel}, G \rangle$. Σ_{ds} is only there for calculation simplification purposes, and erases theme and rheme marking. Therefore, we can consider $G' = \langle \Sigma_{dst}, \Sigma_{dt}, \mathscr{L}' = \mathscr{L}_{dsynt} \circ \mathscr{L}_{dsyntRel}, G \rangle$ as an ACG.
This semantic to deep-syntax interface focuses on the reduced version of figure 16, made of Σ_{str}, $\mathscr{L}_{semRel}$, Σ_{dst}, $\mathscr{L}'$ and Σ_{dt}.

4.3.1 Adding Thematicity at the Semantic Level

In the extract of $\Sigma_{semantic}$, in figure 8, each semanteme is encoded in this implementation as a constant of type $n \Rightarrow t$, that takes a node (of type n) as argument and returns a truth value (type t). Each semanteme is encoded as a label, i.e., a predicate applied to nodes. A semanteme is then of type t. A semantic relation starts at one lexeme and points toward another. It is represented by a constant that takes two arguments of type n (*i.e.*, nodes, being resp. the governor and the dependent), and that has a resulting type t.
Σ_{str} enables the representation of semantic graphs with thematicity annotations. It is similar to $\Sigma_{semantic}$, except that nodes can be annotated as theme, or rheme, or neither. Hence, they can have type n, n_t or n_r. Indices $-_t$ and $-_r$ are used to indicate a theme annotation and a rheme annotation respectively.
Then, a semanteme is encoded by a constant of type $n \Rightarrow t$ if no thematic annotation occur, $n_t \Rightarrow t$ if its node is annotated as part of a theme area, and $n_r \Rightarrow t$ if its node is annotated as part of a rheme area. Since one semanteme may be annotated either as theme, as rheme, or not annotated, up to three constantes may exist for the same semanteme. (1) shows the example of both theme annotated and rheme annotated constants encoding the ‹charlie› semanteme. For reading purposes, the index of these constants reflects the thematic annotation of their typing.

$$\begin{aligned} invite_t^{str} &: n_t \Rightarrow t \\ invite_r^{str} &: n_r \Rightarrow t \end{aligned} \tag{1}$$

Then relation types need to enable branching between two nodes that may have thematic annotation. (2) shows some examples of constants encoding variations of the semantic **1** relation. Again, these constants are differentiated by they indexes, that reflect (for reading purposes) the thematic annotation of the typing of the relation.

$$\begin{aligned} \mathbf{1}_{\emptyset t} &: n \to n_t \to t \\ \mathbf{1}_{rt} &: n_r \to n_t \to t \\ \mathbf{1}_{tr} &: n_t \to n_r \to t \end{aligned} \tag{2}$$

Variations of the existential quantifer compared to $\Sigma_{semantic}$ (see figure 11) are encoded similarly (see figure 17). Remaining parts of Σ_{str} are very similar to $\Sigma_{semantic}$. The terms encoding the SemR corresponding to the expressions " *Charlie is calm* " (see figure 3b), " *Charlie invites Taylor* " (see figure 7a), and " *Taylor is invited by Charlie* " (see figure 7b) are given in (3).

$$\begin{aligned} \gamma_{cic}^{str} &= \lambda e_0.\exists_t e_x.(calm_r^{str}\ e_0) \wedge (\mathbf{1}\ e_0\ e_x) \wedge (charlie_t^{str}\ e_x) \\ \gamma_{cit}^{str} &= \lambda e_0.\exists_t e_x.\exists_r e_y.(invite_r^{str}\ e_0) \wedge (\mathbf{1}_{rt}\ e_0\ e_x) \wedge (\mathbf{2}_{rr}\ e_0\ e_y) \wedge (charlie_t^{str}\ e_x) \wedge (taylor_r^{str}\ e_y) \\ \gamma_{tiic}^{str} &= \lambda e_0.\exists_r e_x.\exists_t e_y.(invite_r^{str}\ e_0) \wedge (\mathbf{1}_{rr}\ e_0\ e_x) \wedge (\mathbf{2}_{rt}\ e_0\ e_y) \wedge (charlie_r^{str}\ e_x) \wedge (taylor_t^{str}\ e_y) \end{aligned} \tag{3}$$

Figure 17 shows an extract of Σ_{str} as well as an extract of $\mathscr{L}_{sem}$. Each constant of Σ_{str} is interpreted as the corresponding (unmarked) constant of Σ_s by $\mathscr{L}_{sem}$. $\mathscr{L}_{sem}$ is straightforward

and erases all thematicity markings and duplicates that may results from erasing thematicity markings from constants of Σ_{str}.

$$
\begin{array}{rl}
\multicolumn{2}{c}{\Sigma_{\boldsymbol{str}}} \\
n, n_t, n_r & : type \\
t & : type \\
\exists & : (n \Rightarrow t) \rightarrow t \\
\exists_t & : (n_t \Rightarrow t) \rightarrow t \\
\exists_r & : (n_r \Rightarrow t) \rightarrow t \\
\wedge & : t \rightarrow t \rightarrow t \\
\mathbf{1}_{rt}, \mathbf{2}_{rt} & : n_r \rightarrow n_t \rightarrow t \\
\mathbf{1}_{rr}, \mathbf{2}_{rr} & : n_r \rightarrow n_r \rightarrow t \\
calm_r^{str}, charlie_r^{str}, taylor_r^{str} & : n_r \Rightarrow t \\
calm_t^{str}, charlie_t^{str}, taylor_t^{str} & : n_t \Rightarrow t \\
invite_r^{str}, son_r^{str} & : n_r \Rightarrow t
\end{array}
\qquad
\begin{array}{rl}
\multicolumn{2}{c}{\mathcal{L}_{\boldsymbol{sem}}} \\
n, n_t, n_r & := n \\
t & := t \\
\exists, \exists_t, \exists_r & := \exists \\
\wedge & := \wedge \\
\mathbf{1}_{rt}, \mathbf{1}_{rr} & := \mathbf{1} \\
\mathbf{2}_{rt}, \mathbf{2}_{rr} & := \mathbf{2} \\
calm_r^{str}, calm_t^{str} & := calm^s \\
charlie_r^{str}, charlie_t^{str} & := charlie^s \\
taylor_r^{str}, taylor_t^{str} & := taylor^s \\
invite_r^{str} & := invite^s \\
son_r^{str} & := son^s
\end{array}
$$

Figure 17: Extracts of Σ_{str}, and $\mathscr{L}_{sem}$ (lexicon from Σ_s to Σ_{str})

Σ_s is the same signature as $\Sigma_{semantic}$ presented earlier. (4) gives the expression describing the semantic graph associated to the expressions " *Charlie is calm* " (γ^s_{cic}) and " *Charlie invites taylor* " (γ^s_{cit}).

$$
\begin{array}{l}
\gamma^s_{cic} = \lambda e_0.\exists e_x.(calm^s\ e_0) \wedge (\mathbf{1}\ e_0\ e_x) \wedge (charlie^s\ e_x) \\
\gamma^s_{cit} = \lambda e_0.\exists e_x e_y.(invite^s\ e_0) \wedge (\mathbf{1}\ e_0\ e_x) \wedge (\mathbf{2}\ e_0\ e_y) \wedge (charlie^s\ e_x) \wedge (taylor^s\ e_y)
\end{array}
\tag{4}
$$

Then, we have the following relations:

$$
\begin{array}{l}
\mathscr{L}_{sem}(\gamma^{str}_{cic}) = \gamma^s_{cic} \\
\mathscr{L}_{sem}(\gamma^{str}_{cit}) = \gamma^s_{cit}
\end{array}
\tag{5}
$$

This encoding of theme-rheme opposition at the semantic level enables for two SemR sharing the same predicative structure (SemS) to be different. Therefore, the two SemR corresponding to the expressions " *Charlie invites Taylor* " (figure 7a) and " *Taylor is invited by Charlie* "(figure 7b) are represented by respectively the two different terms γ^{str}_{cit} and γ^{str}_{tiic} (see (3)).

4.3.2 Deep-Syntactic Structure

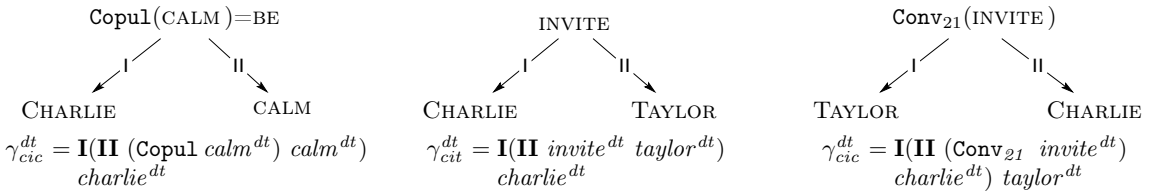
Similiarly to Σ_s that is identical to $\Sigma_{semantic}$, Σ_{dt} is identical to $\Sigma_{dsynt\text{-}tree}$, and contains all constants necessary to represent DSyntS, *i.e.*, deep-syntactic predicative structures (that are dependency trees).
Each deep lexeme is encoded in this implementation by a constant of type *T* (see (6)). In MTT, categories are distinguished at the deep-syntactic level. However, for simplification purposes, we chose to represent them only at the surface-syntactic level of the present encoding and not at its deep-syntactic level. Therefore, grammatically incorrect structures can be produced. However, they will be filtered out by lexicons (such structures don't have antecedents, see section 4.3.3) and won't be generated at the surface syntactic level. Hence, no distinction is made between grammatical categories in Σ_{dt}.

$$
charlie^{dt}, invite^{dt} : T \tag{6}
$$

Since a relation starts at one lexeme and points toward another, it is represented by a constant that takes two arguments of type T (being resp. the governor and the dependent), and that has a resulting type T. Hence, each deep-syntactic relation (such as the ones in (7)) is of type $T \to T \to T$.

$$\mathbf{I}, \mathbf{ATTR} : \; T \to T \to T \tag{7}$$

Since the resulting type of a relation is T, a dependency (*i.e.* a governor, the relation, and its dependent) is represented by a term of type T as well. This term can in turn be used as a governor (to add another dependency to its governor) or as a dependent (to make it depend from another lexeme). Hence, type T is used for constants and terms encoding trees, sub-trees, and single nodes.
Figure 18 shows some examples of DSyntS along with their associated terms from $\Lambda(\Sigma_{dt})$.



$\gamma^{dt}_{cic} = \mathbf{I}(\mathbf{II}\ (\texttt{Copul}\ calm^{dt})\ calm^{dt})\ charlie^{dt}$

(a) "*Charlie is calm*"

$\gamma^{dt}_{cit} = \mathbf{I}(\mathbf{II}\ invite^{dt}\ taylor^{dt})\ charlie^{dt}$

(b) "*Charlie invites Taylor.*"

$\gamma^{dt}_{cic} = \mathbf{I}(\mathbf{II}\ (\texttt{Conv}_{21}\ invite^{dt})\ charlie^{dt})\ taylor^{dt}$

(c) "*Taylor is invited by Charlie*"

Figure 18: Representation of several DSyntS and their associated object terms (from $\Lambda(\Sigma_{dt})$).

The communicative structure is absent from Σ_{dt} (and Σ_{ds}, see below), but is present in Σ_{dst}. The choice was made for now to focus on encoding thematicty in semantic but not to keep it in deep-syntax, since the role it plays there is less important than in the syntax-semantic interface. However, figures 18b and 18c are different. Both corresponding expressions, that (incorrectly) shared the same DSyntS in section 4.1, have now different DSyntS, and are not encoded as paraphrases anymore. The following sections focuses on the encoding of the relation between Σ_{str} (SemR) and Σ_{dt} (DSyntS). To do so, we introduce an abstract signature Σ_{dst} in order to perform a transduction opration from Σ_{str} to Σ_{dt}.

4.3.3 Actancial Structure (in Σ_{dst} & Σ_{ds})

Since all lexemes of Σ_{dt} are encoded by constants of type T, incorrect structures can be build upon $\Lambda(\Sigma_{dt})$, like the one corresponding to (8). In this example, two major issues occur. First, in a DSyntR, according to well-formedness rules of MTT, each node (or lexeme) has at most one actancial relation of each kind; $charlie^{dt}$ can't be the governor of two **I** relations. Second, $invite^{dt}$ is a lexeme expecting at least one dependent (" *Y is invited* "), and usually two, since its predicative structure contains two arguments: *X invites Y*.

$$\begin{aligned} \gamma^{dt}_{illicit} = \; & \mathbf{I}\ (\mathbf{I}\ charlie^{dt}\ taylor^{dt})\ invite^{dt} \\ & : \; T \end{aligned} \tag{8}$$

Thus, too many structures can be built upon Σ_{dt}. To control these structures, we use the (abstract) signature Σ_{dst}. Then, the actancial structure is encoded in the abstract constants' type, and the lexicon enforces the well-formedness (correct actants and correct number of them) of

this actancial structure. We rely on the abstract signature to do so, and use them to *control obtained structures by encoding the constraints in the constants' types.*
Hence, the actancial structure is encoded in the abstract constants' types. In both Σ_{dst} and Σ_{ds}, lexemes are encoded with constants of type G-like. These types stand for graph, subgraph or node and follows the same use principle as type T. Variants can be found, such as G_t or G_r, that respectively express a theme or a rheme marking. The theme-rheme marking of G-like types reflect the thematic markings of their semantic actants' typing. Consequenlty, we encode the number of actants, or dependants, (and their nature), *in the abstract constants' types* (see (9)).
For instance, $invite^{dst}_{rtr}$ has type $G_t \rightarrow G_r \rightarrow G_{f,r}$: it expects its first actant (labelled as theme), its second actant (labelled as rheme) to form a graph (labelled as rheme). The index $-_f$ indicates that the resulting graph contains a thematic opposition, i.e. both theme and rheme areas. $invite^{dst}_{rtr}$ is the constant representing INVITE that is used when its first argument is labelled as theme: " *Charlie invites Taylor* ", and $invite^{dst}_{rrt}$ is the constant representing INVITE that will be used when the second semantic argument of INVITE is labelled as theme: " *Taylor is invited (by Charlie)* ". In the case of $\gamma^{dt}_{illicit}$, such a term can not be built, since the actancial structure of $taylor^{dt}$ does not have any dependents.

$$\begin{aligned} taylor^{dst}_{r} &: G_r \\ invite^{dst}_{rtr} &: G_t \rightarrow G_r \rightarrow G_{f,r} \\ invite^{dst}_{rrt} &: G_r \rightarrow G_t \rightarrow G_{f,r} \end{aligned} \tag{9}$$

To denote an optionally expressible argument, we use a prime symbol (like G' or G'_t, see (Cousin 2023b) for further detail). Each semantic predicate has a predicative structure that expects mandatory dependents. Among these dependents, some are necessarily expressible, and other may be omitted. Let's take the example of INVITE. Its first semantic actant is mandatory: it always exists in its semantic predicative structure. But, it can be expressed (see figure 7) to form an expression like " *Charlie invites Taylor* " or " *Taylor is invited by Charlie* ", or it can be ommitted (see figure 14) to form an expression like " *Taylor is invited* ". This first argument is then optionally expressible. When the optionally expressible argument is expressed, a constant EXP (for "explicit") is used in the λ-term describing the DSyntR of the expression, and when it is ommited, a constant IMP is used (for "implicit").
(10) shows the typing of constants of (9) updated to account for optionally expressible arguments, where $invite^{dst}_{rrt}$, has one.

$$\begin{aligned} taylor^{dst}_{r} &: G_r \\ invite^{dst}_{rtr} &: G_t \rightarrow G_r \rightarrow G_{f,r} \\ invite^{dst}_{rrt} &: G'_r \rightarrow G_t \rightarrow G_{f,r} \end{aligned} \tag{10}$$

Then, the well-formedness of a DSyntR is enforced in $\mathscr{L}_{dsyntRel}$ (see Figure 19) that associates abstract constants from Σ_{ds} to their object terms in $\Lambda(\Sigma_{dt})$, *i.e.*, that really build all DSyntS in $\Lambda(\Sigma_{dt})$. (11) shows the interpretations by $\mathscr{L}_{dsyntRel}$ of a predicate expecting two dependents, respectively an actant **I** and an actant **II**, $invite^{ds}_{rtr}$, and the interpretation of a noun which predicative structure does not except any dependents, $taylor^{ds}_{r}$.

$$\begin{aligned} &\mathscr{L}_{dsyntRel}(taylor^{ds}) := taylor^{dt} \\ &\mathscr{L}_{dsyntRel}(invite^{ds}_{rtr}) := \lambda\ XY.\ \mathbf{I}(\mathbf{II}\ invite^{dt}\ Y)\ X \end{aligned} \tag{11}$$

The encoding of interpretations by the lexicon solves the issue of having only well-formed structures in $\Lambda(\Sigma_{dt})$. Since only correct interpretations are encoded, only correct structures will be generated. In the case of $\gamma^{dt}_{illicit}$, even if it could have been built upon $\Lambda(\Sigma_{dt})$ (provided the numer and type of dependents were correct for all lexemes), such a term can not be obtained by application of $\mathscr{L}_{dsyntRel}$, since no interpretation by $\mathscr{L}_{dsyntRel}$ allows two **I** relations. Similarily, if an incorrect structure is build upon Σ_{dt}, no antecedent will be found by parsing of $\mathscr{L}_{dsyntRel}$, and hence no SemR can be obtained (when continuing the analysis). This principle holds for any lexicon of this encoding. Extracts of Σ_{dst}, and $\mathscr{L}_{dsyntRel}$ are given below in figure 19.

4.3.4 Modification

Modification is not detailled here, but is enabled by this encoding. Further details can be found in (Cousin 2025a; Cousin 2025b). However, some examples used here bear some traces of modification. Modification can be made by an adjective or an adverb. The former modify a noun, while the latter modify a verb, an adjective, or an adverb. For simplification purposes, modifiers are kept out of examples here, but types like MOD and MODj can be encountered. The former corresponds to an adverbial modification, and the second to an adjectival modification. (12) gives the updated types of constants from (10), that takes modification into account. Theme-rheme markings of MOD-like types reflect the thematic markings of their semantic actants' typing.

$$\begin{aligned} taylor^{dst}_{r} &: \ \text{MOD}j_{rr} \rightarrow G_r \\ invite^{dst}_{rtr} &: \ \text{MOD}_{tr} \rightarrow G_t \rightarrow G_r \rightarrow G_{f,r} \\ invite^{dst}_{rrt} &: \ \text{MOD}'_{rr} \rightarrow G'_r \rightarrow G_t \rightarrow G_{f,r} \end{aligned} \tag{12}$$

Constants I_{MOD} and $I_{\text{MOD}j}$ (with potential thematicity markings) may be used to fill these spots. They are present in both Σ_{dst} and Σ_{ds}, and interpreted by $\mathscr{L}_{dsyntRel}$ in Σ_{dt} as the identity (see figure 19). Variables A and M encode modifiers (A for adjectives and M for adverbs) in the constants interpretations by lexicons. (In other words, the reader may ignore MOD-like types, I_{MOD}-like constants, and A and M variables.)

4.3.5 From Σ_{str} to Σ_{dst}

Since Σ_{dst} is used to encode the relation between Σ_{str} and Σ_{dt}, there will be lexicon interpretation toward both signatures. The previous section mainly covers the transition towards Σ_{dt}. Σ_{dst} contains theme-rheme annotations, while Σ_{dt} doesn't. An intermediary signature, Σ_{ds}, is used to erase these markings. Its only use it to erase thematicity from Σ_{dst}, and eliminate resulting duplicated. Then, lexicon $\mathscr{L}_{dsynt}$ from Σ_{dst} to Σ_{ds} is straightforward. Figure 20 gives extracts from $\mathscr{L}_{dsynt}$, and Σ_{ds}.
Therefore, expressions of terms from $\Lambda(\Sigma_{dst})$ and $\Lambda(\Sigma_{ds})$ are really similar. (13) gives the terms encoding the expressions " *Charlie invites Taylor* " and " *Taylor is invited by Charlie* " in both $\Lambda(\Sigma_{dst})$ and Σ_{dt}.

$$\begin{aligned} \gamma^{dst}_{cit} &= invite^{dst}_{rtr}\ I^{dst}_{\text{MOD},tr}\ (charlie^{dst}_{t}\ I^{dst}_{\text{MOD}j,tt})(taylor^{dst}_{r}\ I^{dst}_{\text{MOD}j,rr}) \\ \gamma^{ds}_{cit} &= invite^{ds}_{rtr}\ I^{ds}_{\text{MOD}}\ (charlie^{ds}\ I^{ds}_{\text{MOD}j})(taylor^{ds}\ I^{ds}_{\text{MOD}j}) \\ \gamma^{dst}_{tiic} &= invite^{dst}_{rrt}\ I^{dst}_{\text{MOD},rr'}\ (\text{EXP}_r\ ((charlie^{dst}_{r}\ I^{dst}_{\text{MOD}j,rr})))(taylor^{dst}_{t}\ I^{dst}_{\text{MOD}j,tt}) \\ \gamma^{ds}_{tiic} &= invite^{ds}_{rrt}\ I^{ds}_{\text{MOD}'}\ (\text{EXP}\ ((charlie^{ds}\ I^{ds}_{\text{MOD}j})))(taylor^{ds}\ I^{ds}_{\text{MOD}j}) \end{aligned} \tag{13}$$

Σ_{dst}

$$
\begin{array}{rl}
G_r,\ G_t,\ G'_r & : type \\
\mathrm{MOD}_{rr}, \mathrm{MOD}_{tr} & : type \\
\mathrm{MOD}'_{rr} & : type \\
\mathrm{MOD}j_{rr}, \mathrm{MOD}j_{tt} & : type \\
\\
\mathrm{EXP}_r & : G_r \to G'_r \\
\mathrm{IMP}_r & : G_r \to G'_r \\
I^{dst}_{\mathrm{MOD},rr} & : \mathrm{MOD}_{rr} \\
I^{dst}_{\mathrm{MOD},tr} & : \mathrm{MOD}_{tr} \\
I^{dst}_{\mathrm{MOD}',rr} & : \mathrm{MOD}'_{rr} \\
I^{dst}_{\mathrm{MOD}j,tt} & : \mathrm{MOD}j_{tt} \\
I^{dst}_{\mathrm{MOD}j,rr} & : \mathrm{MOD}j_{rr} \\
charlie^{dst}_r & : \mathrm{MOD}j_{rr} \to G_r \\
charlie^{dst}_t & : \mathrm{MOD}j_{tt} \to G_t \\
invite^{dst}_{rtr} & : \mathrm{MOD}_{tr} \to G_t \to G_r \to G_{f,r} \\
invite^{dst}_{rrt} & : \mathrm{MOD}'_{rr} \to G'_r \to G_t \to G_{f,r} \\
son^{dst}_{rtr} & : \mathrm{MOD}_{tr} \to \mathrm{MOD}j_{rr} \to G_t \to G_r \to G_{f,r} \\
son^{dst}_{rrr} & : \mathrm{MOD}j_{rr} \to G'_r \to G_t \to G_{f,r} \\
taylor^{dst}_r & : \mathrm{MOD}j_{rr} \to G_r \\
taylor^{dst}_t & : \mathrm{MOD}j_{tt} \to G_t
\end{array}
$$

$\mathcal{L}_{dsyntRel}$

$$
\begin{array}{rl}
G,\ G'_r & := T \\
\mathrm{MOD}, \mathrm{MOD}' & := (T \to T) \to (T \to T) \\
\mathrm{MOD}j & := T \to T \\
\\
\mathrm{EXP} & := \lambda l.\ l \\
\mathrm{IMP} & := \emptyset \\
I^{ds}_{\mathrm{MOD}}, I^{ds}_{\mathrm{MOD}'} & := \lambda v.\ v \\
I^{ds}_{\mathrm{MOD}j} & := \lambda n.\ n \\
calm^{ds} & := \lambda M\ A\ P.\ (\mathbf{ATTR}\ (A\ P) \\
 & \quad (M\ (\lambda m.\ m)\ calm^{dt})) \\
calm^{ds}_{rt} & := \lambda M\ A\ P.\ \mathbf{I}(\mathbf{II}\ (\texttt{Copul}\ calm^{dt}) \\
 & \quad (M\ (\lambda x.\ x)\ calm^{dt}))(A\ X) \\
charlie^{ds} & := \lambda A.\ A\ charlie^{dt} \\
invite^{ds}_{rtr} & := \lambda M\ X\ Y.\ M\ (\lambda x.\ \mathbf{I}\ (\mathbf{II}\ invite^{dt}\ Y)\ x)\ X \\
invite^{ds}_{rrt} & := \lambda M\ X\ Y.\ M\ (\lambda x. \\
 & \quad \mathbf{I}\ (\mathbf{II}\ (Conv_{21}\ invite^{dt})\ x)\ Y)X \\
taylor^{ds} & := \lambda A.\ A\ taylor^{dt} \\
son^{ds}_{rrr} & := \lambda M\ X\ Y.M\ (\lambda x.\ \mathbf{I}\ (\mathbf{II}\ son^{dt}\ Y)\ x)\ X \\
son^{ds}_{rtr} & := \lambda M\ A\ X\ Y.\ M\ (\lambda x.\ \mathbf{I}\ (\mathbf{II}\ (\texttt{Copul}\ son^{dt}) \\
 & \quad (A\ (\mathbf{I}\ (\mathbf{II}\ son^{dt}\ Y)\ \emptyset^{dt})))\ x)\ X
\end{array}
$$

Figure 19: Extract of Σ_{dst} and $\mathscr{L}_{dsyntRel}$ from Σ_{ds} to $\Sigma_{dsynt\text{-}tree}$

Σ_{ds}

$$
\begin{array}{rl}
G,\ G' & : type \\
\mathrm{MOD} & : type \\
\mathrm{MOD}' & : type \\
\mathrm{MOD}j & : type \\
\\
\mathrm{EXP} & : G \to G' \\
\mathrm{IMP} & : G \to G' \\
I^{ds}_{\mathrm{MOD}} & : \mathrm{MOD} \\
I^{ds}_{\mathrm{MOD}'} & : \mathrm{MOD}' \\
I^{dst}_{\mathrm{MOD}j} & : \mathrm{MOD}j \\
charlie^{ds}, taylor^{ds} & : \mathrm{MOD}j \to G \\
invite^{ds}_{rtr} & : \mathrm{MOD} \to G \to G \to G \\
invite^{ds}_{rrt} & : \mathrm{MOD}' \to G' \to G \to G \\
son^{ds}_{rtr} & : \mathrm{MOD} \to \mathrm{MOD}j \to G \to G \to G \\
son^{ds}_{rrr} & : \mathrm{MOD}j \to G' \to G \to G
\end{array}
$$

$\mathcal{L}_{dsynt}$

$$
\begin{array}{rl}
G_r, G_t & := G \\
G'_r & := G' \\
\mathrm{MOD}_{rr}, \mathrm{MOD}_{tr} & := \mathrm{MOD} \\
\mathrm{MOD}'rr & := \mathrm{MOD}' \\
\mathrm{MOD}j_{rr}, \mathrm{MOD}j_{tt} & := \mathrm{MOD}j \\
\\
\mathrm{EXP}_r & := \mathrm{EXP} \\
\mathrm{IMP}_r & := \mathrm{IMP} \\
I^{dst}_{\mathrm{MOD},rr}, I^{dst}_{\mathrm{MOD},tr} & := I^{ds}_{\mathrm{MOD}} \\
I^{dst}_{\mathrm{MOD}',rr} & := I^{ds}_{\mathrm{MOD}'} \\
I^{dst}_{\mathrm{MOD}j,rr}, I^{dst}_{\mathrm{MOD}j,tt} & := I^{ds}_{\mathrm{MOD}j} \\
charlie^{dst}_r, charlie^{dst}_t & := charlie^{ds} \\
invite^{dst}_{rtr} & := invite^{ds}_{rtr} \\
invite^{dst}_{rrt} & := invite^{ds}_{rrt} \\
son^{dst}_{rtr} & := son^{ds}_{rtr} \\
son^{dst}_{rrr} & := son^{ds}_{rrr} \\
taylor^{dst}_r, taylor^{dst}_t & := taylor^{ds}
\end{array}
$$

Figure 20: Extracts of Σ_{ds} and of $\mathscr{L}_{dsynt}$ (from Σ_{dst} to Σ_{ds})

Then, the following equations holds:

$$\begin{aligned}\mathscr{L}'(\gamma_{cit}^{dst}) &= \mathscr{L}_{dsyntRel}(\mathscr{L}_{dsynt}(\gamma_{cit}^{dst})) = \mathscr{L}_{dsyntRel}(\gamma_{cit}^{ds}) = \gamma_{cit}^{dt}\\ \mathscr{L}'(\gamma_{tiic}^{dst}) &= \mathscr{L}_{dsyntRel}(\mathscr{L}_{dsynt}(\gamma_{tiic}^{dst})) = \mathscr{L}_{dsyntRel}(\gamma_{tiic}^{ds}) = \gamma_{tiic}^{dt}\end{aligned} \tag{14}$$

On the other hand, the transition towards Σ_{str} is enforced by the lexicon $\mathscr{L}_{semRel}$, that builds SemR in $\Lambda(\Sigma_{str})$ from $\Lambda(\Sigma_{dst})$. Figure 21 gives an extract of $\mathscr{L}_{semRel}$.

$$\mathcal{L}_{\boldsymbol{semRel}}$$

$$\begin{aligned}
G_t &:= n_t \Rightarrow t\\
G_r, G'_r &:= n_r \Rightarrow t\\
\mathrm{MOD}_{rr}, \mathrm{MOD}'_{rr} &:= ((n_r \Rightarrow t) \rightarrow (n_r \Rightarrow t)) \rightarrow ((n_r \Rightarrow t) \rightarrow (n_r \Rightarrow t))\\
\mathrm{MOD}_{tr} &:= ((n_t \Rightarrow t) \rightarrow (n_r \Rightarrow t)) \rightarrow ((n_t \Rightarrow t) \rightarrow (n_r \Rightarrow t))\\
\mathrm{MOD}j_{rr} &:= (n_r \Rightarrow t) \rightarrow (n_r \Rightarrow t)\\
\mathrm{MOD}j_{tt} &:= (n_t \Rightarrow t) \rightarrow (n_t \Rightarrow t)\\
\mathrm{EXP}_r &:= \lambda^0 x.\ \lambda e_0.\ x\ e_0\\
\mathrm{IMP}_r &:= \lambda^0 x.\ \lambda e_0.\ x\ e_0\\
I^{dst}_{\mathrm{MOD},rr}, I^{dst}_{\mathrm{MOD},tr}, I^{dst}_{\mathrm{MOD}',rr}, I^{dst}_{\mathrm{MOD}j,rr}, I^{dst}_{\mathrm{MOD}j,tt} &:= \lambda^0\ e.\ e\\
charlie^{dst}_r, charlie^{dst}_t &:= \lambda^0 A.\ A\ (\lambda^0\ e0.\ (charlie^{str}\ e0))\\
invite^{dst}_{rtr} &:= \lambda^0\ M\ x\ y.\ M(\lambda^0 s.\ \lambda\ e_0.\ (\exists_t\ e_x.\exists_r\ e_y.\ (invite^{str}_r\ e_0)\\
&\quad \wedge(\mathbf{1}_{rt}\ e_0\ e_x) \wedge (\mathbf{2}_{rr}\ e_0\ e_y) \wedge(s\ e_x) \wedge (y\ e_y)))x\\
invite^{dst}_{rrt} &:= \lambda^0\ M\ x\ y.\ M(\lambda^0 s.\ \lambda\ e_0.\ (\exists_r\ e_x.\exists_t\ e_y.\ (invite^{str}_r\ e_0)\\
&\quad \wedge(\mathbf{1}_{rt}\ e_0\ e_x) \wedge (\mathbf{2}_{rr}\ e_0\ e_y) \wedge(s\ e_x) \wedge (y\ e_y)))x\\
son^{dst}_{rtr} &:= \lambda^0\ M\ A\ x\ y.\ M(\lambda^0 s.\ A\ (\lambda\ e_0.\ (\exists_t\ e_x.\exists_r\ e_y.\ (son^{str}_r\ e_0)\\
&\quad \wedge(\mathbf{1}_{rt}\ e_0\ e_x) \wedge (\mathbf{2}_{rr}\ e_0\ e_y) \wedge(s\ e_x) \wedge (y\ e_y))))x\\
son^{dst}_{rrr} &:= \lambda^0\ A\ x\ y.\ A\ (\lambda\ e_0.\ (\exists_r\ e_x\ e_y.\ (son^{str}_r\ e_0)\\
&\quad \wedge(\mathbf{1}_{rr}\ e_0\ e_x) \wedge (\mathbf{2}_{rr}\ e_0\ e_y) \wedge(x\ e_x) \wedge (y\ e_y)))\\
taylor^{dst}_r, taylor^{dst}_t &:= \lambda^0 A.\ A\ (\lambda^0\ e0.\ (taylor^{str}\ e0))
\end{aligned}$$

Figure 21: Extract of $\mathscr{L}_{semRel}$ (from Σ_{dst} to Σ_{str})

Then, when considering both expressions " *Charlie invites Taylor* " and " *Taylor is invited by Charlie* ", we now have the equations given in (15) that describe the transduction from Σ_{str} to Σ_{dt}. Terms used in (15) can be found in (3), (12) and (13).

$$\begin{aligned}
\mathscr{L}'(\mathscr{L}^{-1}_{semRel}(\gamma_{cit}^{str})) &= \mathscr{L}_{dsyntRel}(\mathscr{L}_{dsynt}(\mathscr{L}^{-1}_{semRel}(\gamma_{cit}^{str}))) = \mathscr{L}_{dsyntRel}(\mathscr{L}_{dsynt}(\gamma_{cit}^{dst}))\\
&= \mathscr{L}_{dsyntRel}(\gamma_{cit}^{ds}) = \gamma_{cit}^{dt}\\
\mathscr{L}'(\mathscr{L}^{-1}_{semRel}(\gamma_{tiic}^{str})) &= \mathscr{L}_{dsyntRel}(\mathscr{L}_{dsynt}(\mathscr{L}^{-1}_{semRel}(\gamma_{tiic}^{str}))) = \mathscr{L}_{dsyntRel}(\mathscr{L}_{dsynt}(\gamma_{tiic}^{dst}))\\
&= \mathscr{L}_{dsyntRel}(\gamma_{tiic}^{ds}) = \gamma_{tiic}^{dt}\\
\mathscr{L}_{sem}(\gamma_{cit}^{str}) &= \mathscr{L}_{sem}(\gamma_{tiic}^{str}) = \gamma_{cit}^{s}
\end{aligned} \tag{15}$$

(15) shows that both expressions share the same predicative semantic graph (γ_{cit}^{s}), but have different sementic representations (γ_{cit}^{str} and γ_{tiic}^{str}) that lead to different deep-syntactic structures (γ_{cit}^{dt} and γ_{tiic}^{dt}). Hence, by controlling the types of constants, Σ_{dst} controls how structures combine with respect of several constraints, among which their theme-rheme opposition.
This example (with INVITE) is one illustration of the role of the thematic opposition. Theme-rheme opposition is not limited to active/passive distinction. Another example of its implication can be found in the next section (subsection 4.3.6) where it determine the apparition (or not) of a copula.

4.3.6 Example from Σ_s to Σ_{dt}

This section present another example, where two SemR share the same SemS but have different thematic markers. Figure 19 shows that two abstract constants exist for ‘son’, son^{dst}_{rtr} and son^{dst}_{rrr}. None of them allow ‘son’ to have both its arguments typed as theme. In figure 22a[2], with son^{dst}_{rtr}, the first argument is theme and the second rheme (as well as ‘son’ itself) which lead to the copula expression “ *X is Y's son* ". In figure 22b[2], with son^{dst}_{rrr}, all constants are labelled as rheme, leading to the nominal expression “ *X, Y's son* ".

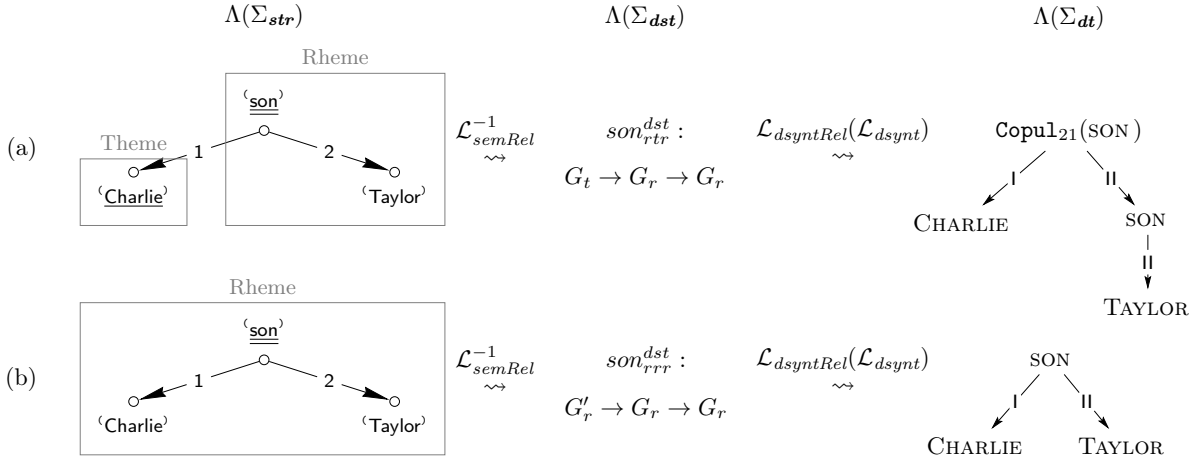


Figure 22: Example of a generation of two DSyntS sharing the same SemS but with different Sem-CommS

5 Results

This encoding of MTT into ACGs enables the representation of concepts such as LFs, semantic paraphrasing, surface-syntactic paraphrasing, the communicative structure (theme-rheme opposition) and its role, as well as some deep-syntactic paraphrasing. It also allows the representation and handling of several lexical phenomena, among which synonymy, idiomatic expressions, and modifiers behavior. Several modifiers can be used to modify the same predicate. ACGs of this encoding, except the one made of Σ_s, Σ_{str}, and $\mathscr{L}_{sem}$, are from the specific class of ACGs of order 2, which parsing is decidable and polynomial.
All ACGs are encoded in the software ACGtk[3]. Lexicons and vocabularies used here have between 30 and 50 lexemes approximately. These ACGs are tested on 63 terms build upon $\Lambda(\Sigma_{dst})$. Tests scripts perform analysis operations toward Σ_s as well as transduction operations toward Σ_{st}. For 63 input terms, 430 output terms (*i.e.*, surface-syntactic structures) are generated and manually verified.

[2]For reading purposes, λ-terms representing SemR and DSyntR are not given here, all constants necessary to built these terms are given in the extracts of signatures and lexicons of section 4. The traduction as graphs or trees of these complex λ-terms is represented instead.

[3]A link to the files will be available in the final version, but is not given here to preserve anonymity.

6 Perspectives

The communicative structure of MTT does not contain only the theme-rheme opposition, but also six other communicative opposition (Mel'čuk et al. 2012), such as the giveness (opposition between given, new and neutral), or focalization (opposition between focalized and non-focalized). This work currently takes only the thematic opposition into account and none of the six other ones. We are also aiming at extending the communicative structure to all levels of this encoding, and not only to the semantic to deep-syntax interface.
On top of that, the encoding presented here is a toy grammar, built with examples from the litterature and the help of the linguist Alain Polguère. Among the perspectives for this work, we aim to find or build a corpora in order to extend our input data. It would be possible to extract data from SUD annotated corpora (Gerdes et al. 2018) or from the RL-fr (ATILF 2025) to extend our coverage. Syntactic structures of MTT are dependency structures that roughly correspond to the ones of SUD. Labels on relations are different, but tree structures of MTT surface-syntactic representations are very similar to SUD representations. Syntactic representations of the encoding presented here are in accordance with the ones of annotated corpora from Grew (Guillaume 2021).
As an additinal line of research, we observe similarities between the encoding of constants of the same grammatical class (transitives verbs, nouns, etc.). Therefore, it is likely that we'll work with generation tools (or build them) to have a grammar with better coverage. We are working on the idea of writing a metagrammar for this purpose.
Lastly, all ACGs except the one made of Σ_s, Σ_{str}, and $\mathscr{L}_{sem}$ are of order 2. This means that all lexicons except $\mathscr{L}_{sem}$ can be parsed, and that generation and analysis can both be used between Σ_{str} and Σ_{st} (as we wanted). However, between Σ_s and Σ_{str}, only analysis can be applied. This limitation is not bothersome, since Σ_{str} enable the representation of SemR, and that is the starting point of the generation. Σ_s is only a simplification of Σ_{str}, that enable the representation of SemS only.

7 Conclusion

This chapter presented how we added the theme-rheme opposition of the Sem-CommS to Cousin (2023a)'s encoding of MMT into ACG. This addition adds control over the structures by composing ACGs that bear thematicity information in their constant's typing. It enables the differentiation of SemR sharing the same SemS but not the same Sem-CommS, and hence the generation of different DSyntS associated to these SemR.
In contrast to previous encodings (Cousin 2023b; Cousin 2023a), this encoding is mostly of order 2, and not of higher order. This enables, on top of polynomial ACG complexity, that all used lexicons (except $\mathscr{L}_{sem}$) can be parsed. Consequently, this encoding can be used for generation as well as analysis purposes, from Σ_{str} to Σ_{st}. It is the case for every transition between two signatures, independently of whether the whole transition between semantic and surface-syntax is considered, or only an intermediate transition. Theme-rheme opposition is accounted for in semantic and deep-syntactic representation levels, what enables a more accurate SSyntS generation from a SemR. Additionally, lexemes only constrain their mandatory dependents, and do so via the type of abstract constants encoding them.

References

ATILF (2025). *Réseau Lexical du Français (RL-fr)*. ORTOLANG (Open Resources and TOols for LANGuage) –www.ortolang.fr. URL: https://hdl.handle.net/11403/lexical-system-fr/v3.2.

Bohnet, Bernd, Andreas Langjahr, and Leo Wanner (June 2000). "A development Environment for an MTT-Based Sentence Generator". In: *INLG'2000 Proceedings of the First International Conference on Natural Language Generation*. Ed. by Michael Elhadad. Mitzpe Ramon, Israel: Association for Computational Linguistics, pp. 260–263. DOI: 10.3115/1118253.1118292. URL: https://aclanthology.org/W00-1436/.

Cousin, Marie (June 2023a). "Meaning-Text Theory within Abstract Categorial Grammars: Towards Paraphrase and Lexical Function Modeling for Text Generation". In: *IWCS 2023 - 15th International Conference on Computational Semantics*. Ed. by Maxime Amblard and Helen Breitholtz. Nancy, France: Association for Computational Linguistics. URL: https://inria.hal.science/hal-04104453.

– (June 2023b). "Vers une implémentation de la théorie sens-texte avec les grammaires catégorielles abstraites". In: *18e Conférence en Recherche d'Information et Applications – 16e Rencontres Jeunes Chercheurs en RI – 30e Conférence sur le Traitement Automatique des Langues Naturelles – 25e Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues*. Ed. by Marie Candito, Thomas Gerald, and José G Moreno. Paris, France: ATALA, pp. 72–86. URL: https://inria.hal.science/hal-04100197.

– (Aug. 2025a). "Dependency Structures Representation: Meaning Text Theory's Deep-Syntax Encoding With Abstract Categorial Grammars". In: *18th Meeting on the Mathematics of Language - MOL 2025*. Thomas Graf and Matthew Hayden and Jeffrey Heinz and Gary Mar. Stony Brook (NY), United States. URL: https://inria.hal.science/hal-05225895.

– (June 2025b). "Syntaxe en dépendance avec les grammaires catégorielles abstraites : une application à la théorie sens-texte". In: *Actes de CORIA-TALN-RJCRI-RECITAL 2025. Actes des 32ème Conférence sur le Traitement Automatique des Langues Naturelles (TALN), volume 1 : articles scientifiques originaux*. Marseille, France: Association pour le Traitement Automatique des Langues, pp. 716–729. URL: https://talnarchives.atala.org/TALN/TALN-2025/55.pdf.

de Groote, Philippe (July 2001). "Towards Abstract Categorial Grammars". In: *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*. Toulouse, France: Association for Computational Linguistics, pp. 252–259. DOI: 10.3115/1073012.1073045. URL: https://aclanthology.org/P01-1033.

Gerdes, Kim et al. (Nov. 2018). "SUD or Surface-Syntactic Universal Dependencies: An annotation scheme near-isomorphic to UD". In: *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*. Ed. by Marie-Catherine de Marneffe, Teresa Lynn, and Sebastian Schuster. Brussels, Belgium: Association for Computational Linguistics, pp. 66–74. DOI: 10.18653/v1/W18-6008. URL: https://aclanthology.org/W18-6008/.

Groote, Philippe de and Sylvain Pogodalla (2004). "On the expressive power of Abstract Categorial Grammars: Representing context-free formalisms". In: *Journal of Logic, Language and Information* 13.4. http://www.springerlink.com/content/1572-9583/, pp. 421–438. DOI: 10.1007/s10849-004-2114-x. URL: https://inria.hal.science/inria-00112956.

Guillaume, Bruno (Apr. 2021). "Graph Matching and Graph Rewriting: GREW tools for corpus exploration, maintenance and conversion". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. Ed. by Dimitra Gkatzia and Djamé Seddah. Online: Association for Computational Linguistics, pp. 168–175. DOI: 10.18653/v1/2021.eacl-demos.21. URL: https://aclanthology.org/2021.eacl-demos.21/.

Iordanskaja, Lidija, Richard Kittredge, and Alain Polguère (1991). "Lexical Selection and Paraphrase in a Meaning-Text Generation Model". In: *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Ed. by Cécile L. Paris, William R. Swartout, and William C. Mann. Boston, MA: Springer US, pp. 293–312. ISBN: 978-1-4757-5945-7. DOI: 10.1007/978-1-4757-5945-7_11. URL: https://doi.org/10.1007/978-1-4757-5945-7_11.

Kanazawa, Makoto (June 2007). "Parsing and Generation as Datalog Queries". In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, pp. 176–183. URL: https://aclanthology.org/P07-1023.

– (May 2017). "Parsing and generation as Datalog query evaluation". In: *IfCoLog Journal of Logics and Their Applications* 4.4, pp. 1103–1211. URL: https://makotokanazawa.ws.hosei.ac.jp/publications/Kanazawa_M.pdf.

Kruijff-Korbayova, Ivana and Mark Steedman (June 2003). "Discourse and Information Structure". In: *Journal of Logic, Language and Information* 12, pp. 249–259. DOI: 10.1023/A:1024160025821.

Lareau, François et al. (May 2018). "GenDR: A Generic Deep Realizer with Complex Lexicalization". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA). URL: https://aclanthology.org/L18-1478.

Mel'cuk, Igor (Oct. 2001). *Communicative Organization in Natural Language: The semantic-communicative structure of sentences*. ISBN: 9789027230607. DOI: 10.1075/slcs.57.

Mel'čuk, I.A. et al. (2012). *Semantics: From Meaning to Text*. Vol. 1. Semantics: From Meaning to Text. John Benjamins Publishing Company. ISBN: 9789027205964.

– (2013). *Semantics: From Meaning to Text*. Vol. 2. Semantics: From Meaning to Text. John Benjamins Publishing Company. ISBN: 9789027206022.

– (2015). *Semantics: From Meaning to Text*. Vol. 3. Semantics: From Meaning to Text. John Benjamins Publishing Company. ISBN: 9789027259332.

Mel'Čuk, Igor and Alain Polguère (July 2021). "Les fonctions lexicales dernier cri". In: *La Théorie Sens-Texte. Concepts-clés et applications*. Ed. by Sébastien Marengo. Dixit Grammatica. L'Harmattan, pp. 75–155. URL: https://hal.archives-ouvertes.fr/hal-03311348.

Milićević, J. (2007). *La paraphrase: modélisation de la paraphrase langagière*. Sciences pour la communication. Lang. ISBN: 9783039111978. DOI: 10.3726/978-3-0352-0096-6.

Milićević, Jasmina (2006). "A short guide to the Meaning-Text linguistic Theory". In: *Journal of Koralex* 8, pp. 187–233. URL: https://olst.ling.umontreal.ca/static/pdf/IntroMTTJM.pdf.

Pogodalla, Sylvain (2017). "A syntax-semantics interface for Tree-Adjoining Grammars through Abstract Categorial Grammars". In: *Journal of Language Modelling* 5.3, pp. 527–605. DOI: 10.15398/jlm.v5i3.193. URL: https://hal.inria.fr/hal-01242154.

Polguère, Alain (1990). “Structuration et mise en jeu procédurale d’un modèle linguistique déclaratif dans un cadre de génération de texte”. PhD thesis. URL: https://olst.ling.umontreal.ca/static/pdf/PolguerePhD1990.pdf.

Ranta, Aarne (Mar. 2004). “Grammatical Framework”. In: *J. Funct. Program.* 14, pp. 145–189. DOI: 10.1017/S0956796803004738.

Salvati, Sylvain (2005). “Problèmes de filtrage et problème d’analyse pour les grammaires catégorielles abstraites”. Thèse de doctorat dirigée par Philippe de Groote. PhD thesis. Institut National Polytechnique de Lorraine, 1 vol. (VIII–127 p.) URL: http://www.theses.fr/2005INPL050N.

Steedman, Mark (2024). *Combinatory Categorial Grammar: An Introduction*. Course notes from ESSLLI 2024. URL: https://homepages.inf.ed.ac.uk/steedman/papers/ccg/esslli24.pdf.

Wanner, Leo et al. (2010). “MARQUIS: Generation of User-Tailored Multilingual Air Quality Bulletins”. In: *Applied Artificial Intelligence* 24.10, pp. 914–952. DOI: 10.1080/08839514.2010.529258. eprint: https://doi.org/10.1080/08839514.2010.529258. URL: https://doi.org/10.1080/08839514.2010.529258.