A Type-Theoretic Account of Neg-Raising Predicates in Tree Adjoining Grammars

Laurence Danlos, Philippe De Groote, Sylvain Pogodalla

A Type-Theoretic Account of Neg-Raising Predicates in Tree Adjoining Grammars*

Laurence Danlos[1,2,3],
Philippe de Groote[4,5,6], and
Sylvain Pogodalla[4,5,6]

[1] Université Paris Diderot (Paris 7), Paris, F-75013, France
[2] ALPAGE, INRIA Paris–Rocquencourt, Paris, F-75013, France
[3] Institut Universitaire de France, Paris, F-75005, France
[4] INRIA, Villers-lès-Nancy, F-54600, France
[5] Université de Lorraine, LORIA, UMR 7503, Vandœuvre-lès-Nancy, F-54500, France
[6] CNRS, LORIA, UMR 7503, Vandœuvre-lès-Nancy, F-54500, France
{laurence.danlos,philippe.degroote,sylvain.pogodalla}@inria.fr

Abstract. Neg-Raising (NR) verbs form a class of verbs with a clausal complement that show the following behavior: when a negation syntactically attaches to the matrix predicate, it can semantically attach to the embedded predicate. This paper presents an account of NR predicates within Tree Adjoining Grammar (TAG). We propose a lexical semantic interpretation that heavily relies on a Montague-like semantics for TAG and on higher-order types.

Keywords: Tree Adjoining Grammar, semantics, λ-calculus, type theory, neg-raising

1 Introduction

Neg-Raising (NR) verbs form a class of verbs with a clausal complement that show the following behavior: when a negation syntactically attaches to the matrix predicate, it can semantically attach to the embedded predicate, as the implication of (1c) by (1b) shows. This corresponds to the NR reading of this predicate.

(1) a. Marie pense que Pierre partira
Mary thinks that Peter will leave

b. Marie ne pense pas que Pierre partira
Mary does not think that Peter will leave

c. Marie pense que Pierre ne partira pas
Mary thinks that Peter will not leave

* This work has been supported by the French agency Agence Nationale de la Recherche (ANR-12-CORD-0004).

Such an implication does not always hold. Some contexts make it impossible to consider the negation as having scope over the embedded predicate only [1, 2]. This corresponds to the non-NR reading of the predicate.

This paper aims at providing an account of NR predicates within Tree Adjoining Grammar (TAG) [3]. We propose a lexical semantic interpretation that heavily relies on a Montague-like semantics for TAG and on higher-order types. As a base case, our approach lexically provides both NR and non-NR readings to NR predicates. We implement our proposal in the Abstract Categorial Grammar (ACG) framework [4] as it offers a fairly standard interface to logical formal semantics for TAG. However, our approach could be implemented in other synchronous frameworks such as Synchronous TAG [5, 6, 7].

Not all sentence-embedding predicates show a NR behavior [8, 9]. In order to test whether a predicate is NR, we can look at its interaction with Negative Polarity Items (NPI). Such items need to be in the scope of a negative operator in order for the utterance to be felicitous as the contrast between (2a) and (2b) shows. This contrast also shows up when the NPI occurs within the *positive* clause embedded under a *negative* matrix clause as in (3).[7] On the other hand, non-NR predicates do not allow NPI[8] in a positive embedded clause even if it is itself in a negative context (4b).

(2) a. Pierre n'est pas dans son assiette
Peter doesn't feel good

b. *Pierre est dans son assiette
Peter feels good

(3) a. Marie pense que Pierre n'est pas dans son assiette
Mary thinks that Peter doesn't feel good

b. Marie ne pense pas que Pierre soit dans son assiette
Marie doesn't think that Peter feels good

c. *Marie pense que Pierre est/soit dans son assiette
Mary thinks that Peter feels good

(4) a. Marie affirme que Pierre n'est pas dans son assiette
Mary claims that Peter doesn't feel good

b. *Marie n'affirme pas que Pierre est/soit dans son assiette
Mary doesn't claim that Peter feels good

The modeling we propose takes into account several constraints or properties of NR-predicates. First, the availability of the different readings should not introduce spurious ambiguities, in particular when no negation occur. To achieve this, we make use of fine-grained typing.

[7] We do not discuss here the use of subjunctive in the embedded clause when the matrix predicate is in negative form as it does not add to the constraints we describe.

[8] At least some NPI. Some other NPI seems to perfectly occur in a similar position. See [2].

Second, we want to give an account of *NR cyclicity*. This phenomenon occurs when a NR predicate embeds another NR predicate: a negation at the matrix level will semantically cycle down to the most embedded NR predicate, giving rise to several possible interpretations as (5) shows with the interpretations (5a–c). We achieve this effect by making use of higher-order types. In particular, the clausal argument is type-raised so that it can further be modified. Note however that according to [10, 2] we may want to block NR cyclicity when a NR desire predicate embeds a NR belief predicate. This amounts to force non-NR readings of the predicate.

(5) Jeanne ne pense pas que Marie veuille que Pierre parte
Jeanne doesn't think that Mary wants Peter to leave

a. $\neg$(**think** (**want** (**go p**) **m**) **j**)

b. **think** ($\neg$(**want** (**go p**) **m**)) **j**

c. **think** (**want** ($\neg$(**go p**)) **m**) **j**

A similar effect on forcing specific readings occurs with NPI. When a NPI occurs in a positive embedded clause, it forces a NR reading of the (negated) matrix predicate [11]. Then, the negation has scope over the embedded sentence, but not over the whole sentence.

On the other hand, some adverbial discourse connectives (ADCs) force a NR reading. [12] discusses the syntax-semantics mismatches of arguments of an ADC in French and introduces three principles. It illustrates in particular the phenomena with complex sentences that include an ADC (e.g. *par contre (however)*) in the matrix clause and shows that this ADC can have scope: over the whole sentence (principle 1) as in (6), over the embedded clause only (principle 2) as in (7), or over the negation of the embedded clause with a NR reading of the matrix predicate (principle 3) as in (8).

(6) Fred ira à Dax pour Noel. Jane pense, par contre, qu'il n'ira pas.
Fred will go to Dax for Christmas. Jane thinks, however, that he will not go.

(7) Fred ira à Dax pour Noel. Jane pense, par contre, que Pierre n'ira pas.
Fred will go to Dax for Christmas. Jane thinks, however, that Peter will not go.

(8) Fred ira à Dax pour Noel. Jane ne pense pas, par contre, que Pierre ira.
Fred will go to Dax for Christmas. Jane doesn't think, however, that Peter will go.

Our long term goal is to provide a semantic and discourse analysis of these phenomena within TAG and D-STAG [13] that would account for these interactions. However, in this article, we only deal with the NR and non-NR readings of NR predicates. We propose a simply typed λ-calculus approach at the semantic level, while we take the standard analysis of NR predicates as auxiliary trees allowing for adjunction at the syntactic level.

2 Verbs with Clausal Arguments in TAG

In TAG, verbs that have sentential arguments usually are represented as auxiliary trees (see [14, Section 6.7] for English and [15, Chap. 3, Section 1.2] for French) in order to allow for describing long distance dependencies with multiple embeddings as in (9). Negation is analyzed with adjunction as well[9]. Fig. 1(a) and 1(b) show the TAG analysis of (1b) and Fig. 1(c) shows the corresponding derivation tree γ_0: the auxiliary tree of *ne pas* is adjoined to the V node of the auxiliary tree of *pense que*, and the latter is adjoined to the S node of the initial tree of *partira*. The initial trees of *Marie* and *Pierre* are substituted to the *NP* nodes of the trees of *pense que* and *partira* resp.

(9) a. Quelle fille Paul pense-t-il que Bob sait que Jean aime ? [15, p. 234]

b. What did Bill tell Mary that John said? [14, p. 43]

3 Type-Theoretic Perspective on TAG

3.1 Abstract Categorial Grammars

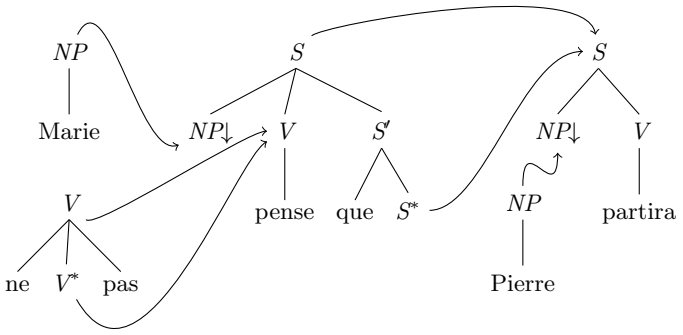
ACGs provide a framework in which several grammatical formalisms may be encoded. They generate languages of linear λ-terms, which generalize both string and tree languages. A key feature is to provide the user direct control over the parse structures of the grammar, the *abstract language*, which allows several grammatical formalisms to be defined in terms of ACG, in particular TAG [16]. In this perspective, derivation trees of TAG are straightforwardly represented as terms of the abstract language, while derived trees (and yields) are represented by terms of the object language. We refer the reader to [4, 17] for the details and introduce here only few relevant definitions and notations.

A *higher-order linear signature* defines a finite set of atomic types and a finite set of typed (possibly with complex types $\alpha \to \beta$) constants. It is also called a *vocabulary*. $\Lambda(\Sigma)$ is the set of λ-terms built on Σ, and for $t \in \Lambda(\Sigma)$ such that t has type α, we note $t : \alpha$

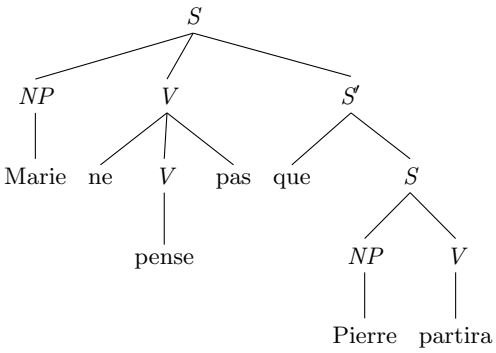
An *abstract categorial grammar* is a quadruple $\mathscr{G} = \langle \Sigma, \Xi, \mathscr{L}, s \rangle$ where Σ and Ξ are two higher-order linear signatures, which are called the *abstract vocabulary* and the *object vocabulary* respectively. $\mathscr{L} : \Sigma \longrightarrow \Xi$ is a lexicon from the abstract vocabulary to the object vocabulary. It is a homomorphism.[10] We note $t{:=}\, u$ if

[9] For sake of clarity, we use a simplified version here.

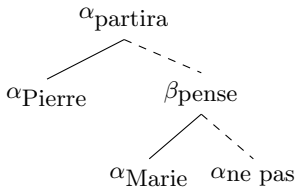
[10] In addition to defining $\mathscr{L}$ on the atomic types and on the constants of Σ, we have:



(a) Tree operations



(b) Derived tree c_0



(c) Derivation tree γ_0

Fig. 1. TAG analysis of *Marie ne pense pas que Pierre partira*

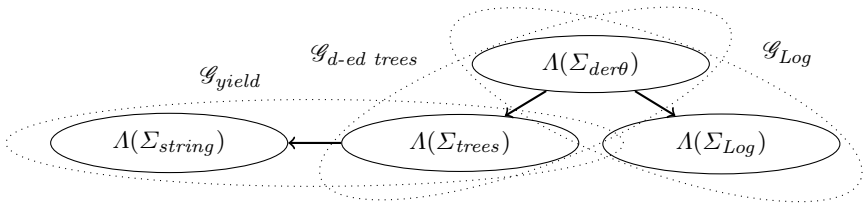


Fig. 2. ACG architecture for TAG

$\mathscr{L}(t) = u$. $s \in \mathscr{T}_\Sigma$ is a type of the abstract vocabulary, which is called the *distinguished type* of the grammar.

Since there is no structural difference between the abstract and the object vocabulary as they both are higher-order signatures, ACGs can be combined in different ways. Either by making the abstract vocabulary of an ACG the object vocabulary of another ACG, so that we modularize the relation between the derivation structures and the strings, as with $\mathscr{G}_{yield}$ and $\mathscr{G}_{d\text{-}ed\ trees}$ in Fig. 2. Or by having a same abstract vocabulary shared by several ACGs in order to make two object terms (for instance a derived tree and a logical formula) share the same underlying structure, as do $\mathscr{G}_{d\text{-}ed\ trees}$ and $\mathscr{G}_{Log}$ in Fig. 2. This is indeed what we use here to provide the semantic readings of TAG analyzed expressions, even if the proposed lexical semantics could be implemented in other approaches, in particular synchronous approaches [5, 6, 18].

3.2 Derivation Trees and Derived Trees

In this paper, we focus on the encoding of TAG derivation trees, TAG derived trees, and the associated logical representation. The ACG $\mathscr{G}_{d\text{-}ed\ trees} = \langle \Sigma_{der\theta}, \Sigma_{trees}, \mathscr{L}_{d\text{-}ed\ trees}, S \rangle$ encodes the relation between derivation trees and derived trees. Table 1 sketches the lexicon to analyze (1b). It relies on $\Sigma_{der\theta}$ whose atomic types include S, V,[11] NP, S_A, $V_A \ldots$ where the X types stand for the categories X of the nodes where a substitution can occur while the X_A types stand for the categories X of the nodes where an adjunction can occur. They later are interpreted as functional types. For each elementary tree it contains a constant whose type is based on the adjunction and substitution sites as Table 1 shows. It additionally contains constants $\beta^{Id}_{\mathrm{X}} : X_A$ that are meant to provide a fake auxiliary tree on adjunction sites where no adjunction actually takes place in a TAG derivation.

The other signature, Σ_{trees}, has τ the type of trees as unique atomic type. Then, for any X of arity n belonging to the ranked alphabet describing the elementary trees of the TAG, we have a constant $X_n : \underbrace{\tau \to \cdots \to \tau}_{n \text{ times}} \to \tau$.

The relation between these vocabularies is given by the lexicon $\mathscr{L}_{d\text{-}ed\ trees}$ where $\mathscr{L}_{d\text{-}ed\ trees}(X_A) = \tau \to \tau$ and for any other type X, $\mathscr{L}_{d\text{-}ed\ trees}(X_A) = \tau$. Then, the derivation tree γ_0 of Fig. 1(c) and the corresponding derived tree c_0

– if $\alpha \to \beta$ is a type build on Σ then $\mathscr{L}(\alpha \to \beta) = \mathscr{L}(\alpha) \to \mathscr{L}(\beta)$;

– if $x \in \Lambda(\Sigma)$ (resp. $\lambda x.t \in \Lambda(\Sigma)$ and $t\,u \in \Lambda(\Sigma)$) then $\mathscr{L}(x) = x$ (resp. $\mathscr{L}(\lambda x.t) = \lambda x.\mathscr{L}(t)$ and $\mathscr{L}(t\,u) = \mathscr{L}(t)\,\mathscr{L}(u)$);

with the proviso that for any constant $c : \alpha$ of Σ we have $\mathscr{L}(c) : \mathscr{L}(\alpha)$.

[11] We follow [15] and we do not use VP categories. Using it would not change our analysis.

of Fig. 1(b) are represented and related as follows:

$\gamma_0 = \alpha_{\text{partira}}\ (\beta_{\text{pense}}\ \beta_S^{Id}\ \beta_{\text{ne pas}}\ \alpha_{\text{Marie}})\ \beta_V^{Id}\ \alpha_{\text{Pierre}}$
$c_0 = \mathscr{L}_{d\text{-}ed\ trees}(\gamma_0)$
$\quad = S_3(NP_1\ \text{Marie})(\ V_3\ \text{ne}(\ V_1\ \text{pense})\text{pas})\ (S'_2\ \text{que}(S_2(NP_1\ \text{Pierre})(\ V_1\ \text{partira})))$

Parallel (or synchronous) to this interpretation as derived trees, we can also interpret terms representing derivation trees as logical formulas representing the associated meanings.

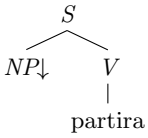
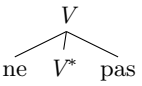
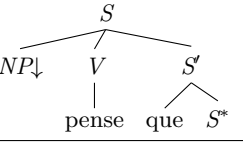
Abstract constants of $\Sigma_{der\theta}$	Their images by $\Sigma_{der\theta}$ and the corresponding TAG trees
$\alpha_{\text{Marie}} : NP$	$c_{Marie} = NP_1\ \text{Marie} : \tau$ $\gamma_{Marie} =$
$\alpha_{\text{partira}} : S_A \to V_A \to NP \to S$	$c_{partira} = \lambda^{\circ} a\ v\ s.s\ (S_2\ s\ (v\ (V_1\ \text{partira})))$ $: (\tau \multimap \tau) \multimap (\tau \multimap \tau) \multimap \tau \multimap \tau$ $\gamma_{partira} =$ 
$\beta_{\text{ne pas}} : V_A$	$c_{ne\ pas} = \lambda^{\circ} x.V_3\ \text{ne}\ x\ \text{pas} : \tau \to \tau$ $\gamma_{ne\ pas} =$ 
$\beta_{\text{pense}} : S_A \to V_A \to NP \to S_A$	$c_{pense\ que} = \lambda^{\circ} s\ v\ x\ y.s\ (S_3\ x(v\ (V_1\ \text{pense}))\ (S'_2\ \text{que}\ y))$ $: (\tau \to \tau) \to (\tau \to \tau) \to \tau \to (\tau \to \tau)$ $\gamma_{pense\ que} =$ 

Table 1. TAG as ACG: the $\mathscr{L}_{d\text{-}ed\ trees}$ lexicon

3.3 Building Semantic Representations

In order to define the translation of terms denoting derivation trees into a logical formula with the ACG $\mathscr{G}_{Log} = \langle \Sigma_{der\theta}, \Sigma_{Log}, \mathscr{L}_{Log}, S \rangle$, we need to define the interpretation of each atomic type and of each constant, and then to consider the homomorphic extension of this interpretation. In other worlds, we have to define the semantic recipe of each lexical item.

The higher-order signature Σ_{Log} for the logical representation defines the following typed constants:

$$\mathbf{m} : e \qquad \neg : t \to t \qquad \mathbf{go} : e \to t \qquad \mathbf{think} : t \to e \to t$$

And we consider the following interpretation $\mathscr{L}_{Log}$:

$$\begin{array}{rl} S := t & \alpha_{\text{Marie}} := \mathbf{m} \\ NP := e & \alpha_{\text{partira}} := \lambda p\ v\ s.p\ (\lambda f.f\ (v\ (\mathbf{go}\ s))) \\ V := e \to t & \beta^{Id}_{\text{X}} := \lambda x.x \\ V_A := t \to t & \beta_{\text{ne pas}} := \lambda p.\neg\ p \\ S_A := ((t \to t) \to t) & \beta_{\text{pense}} := \lambda p\ v\ s\ c.p\ (\lambda f.f\ (v\ (\mathbf{think}\ (c\ (\lambda x.x))\ s))) \\ \quad \to (t \to t) \to t & \beta'_{\text{pense}} := \lambda p\ v\ s\ c.p\ (\lambda f.\mathbf{think}\ (f\ (c\ v))\ s) \end{array}$$

We model the NR and non-NR readings with two possible auxiliary trees for NR predicates[12]: one with the non-NR reading (β_{pense}) and one with the NR reading (β'_{pense}), both with the same type $S_A \to V_A \to NP \to S_A$.

Remark 1. An important point is the type interpreting S_A. In previous works [17], S_A was interpreted with a $t \to t$ type, a function from truth values to truth values where the parameter corresponded to the truth value associated with the embedded clause. This gave the following interpretation:

$$\beta'_{\text{pense}} := \lambda p\ v\ s\ c.p\ (\mathbf{think}\ (v\ c)\ s)$$

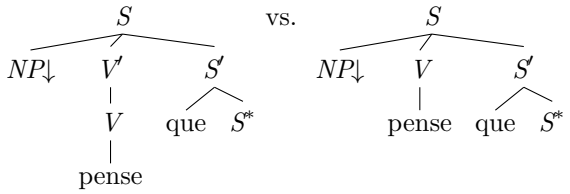
where the c parameter conveys the meaning of the embedded clause. However, in this setting, this meaning cannot further be subject to changes, in particular by a negation provided by the v parameter (interpreting the V_A adjunction site of the verb *pense*). While this would provide us with a reading where the negation scopes over the embedded clause only, it cannot cycle down to a clause more deeply embedded under another NR predicate.

In order to model the NR cyclicity we then need that the semantic argument of the negation can occur arbitrarily far away from the matrix verb as in (5). With $S_A := ((t \to t) \to t) \to ((t \to t) \to t)$, the modification stipulated by the second argument v of $[\![\beta'_{\text{pense}}]\!]$ (which will be replaced by the semantics of the auxiliary tree adjoined to the V_A node of β'_{pense}, typically the negation) will be given as argument to the "raised" clausal argument c so that it can possibly cycle down if c itself represents a NR predicate.

In $[\![\beta_{\text{pense}}]\!]$, v directly has scope over the **think** predicate and c, applied to the identity $\lambda x.x$, is not modified.

Remark 2. Another difference in the interpretation of type modeling the adjunction occurs in the interpretation of $[\![V_A]\!]$. We usually have $V_A := (e \to t) \to (e \to t)$ for V modifiers interpretations. It should not be surprising that we need to actually consider two V adjunction sites since their semantic contribution definitely differ. Technically, it amounts to duplicate the V node to create two adjunction sites in the elementary trees for NR predicates:

[12] We use the $\beta_{\text{NR predicate}}$ notation for constants to be interpreted with the non-NR reading, and $\beta'_{\text{NR predicate}}$ for those to be interpreted with the NR reading.



where one of the sites is dedicated to the negation and the other one to usual V adjunctions such as auxiliaries. We would have

$$\beta'_{\text{pense}} : \quad S_A \rightarrow V'_A \rightarrow V_A \rightarrow NP \rightarrow S_A$$
$$\beta'_{\text{pense}} := \lambda p\ v'_1\ v_2\ s\ c.p\ (\lambda f.(v'_1\ (\lambda y.\mathbf{think}\ (f\ (c\ v_2))\ y)\ s))$$

This would prevent the meaning of auxiliaries (represented by the parameter v'_1) to modify the meaning of the embedded clauses, contrary to the meaning of the negation (represented by the parameter v_2). However, for sake of simplicity, we drop this additional adjunction site as we do not provide any example using it in this paper .

We now are in position to give examples of interpretations. (1b) can be given the derivation trees and the associated meanings:

$$\gamma_0 = \alpha_{\text{partira}}(\beta_{\text{pense}}\ \beta_S^{Id}\ \beta_{\text{ne pas}}\ \alpha_{\text{Marie}})\beta_V^{Id}\ \alpha_{\text{Pierre}}$$
$$\gamma_1 = \alpha_{\text{partira}}(\beta'_{\text{pense}}\ \beta_S^{Id}\ \beta_{\text{ne pas}}\ \alpha_{\text{Marie}})\beta_V^{Id}\ \alpha_{\text{Pierre}}$$
$$\mathscr{L}_{Log}(\gamma_0) = \neg(\mathbf{think}\ (\mathbf{partira}\ \mathbf{p})\ \mathbf{m})$$
$$\mathscr{L}_{Log}(\gamma_1) = \mathbf{think}\ (\neg(\mathbf{partira}\ \mathbf{p}))\ \mathbf{m}$$

Extending in a straightforward way the given lexicon with other NR predicates, we can give (5) the following derivation trees:[13]

$$\gamma'_0 = \alpha_{\text{partira}}(\beta_{\text{veut}}(\beta_{\text{pense}}\beta_S^{Id}\beta_{\text{ne pas}}\alpha_{\text{Jeanne}})\beta_V^{Id}\alpha_{\text{Pierre}})\beta_V^{Id}\ \alpha_{\text{Marie}}$$
$$\gamma'_1 = \alpha_{\text{partira}}(\beta_{\text{veut}}(\beta'_{\text{pense}}\beta_S^{Id}\beta_{\text{ne pas}}\alpha_{\text{Jeanne}})\beta_V^{Id}\alpha_{\text{Pierre}})\beta_V^{Id}\ \alpha_{\text{Marie}}$$
$$\gamma'_2 = \alpha_{\text{partira}}(\beta'_{\text{veut}}(\beta_{\text{pense}}\beta_S^{Id}\beta_{\text{ne pas}}\alpha_{\text{Jeanne}})\beta_V^{Id}\alpha_{\text{Pierre}})\beta_V^{Id}\ \alpha_{\text{Marie}}$$
$$\gamma'_3 = \alpha_{\text{partira}}(\beta'_{\text{veut}}(\beta'_{\text{pense}}\beta_S^{Id}\beta_{\text{ne pas}}\alpha_{\text{Jeanne}})\beta_V^{Id}\alpha_{\text{Pierre}})\beta_V^{Id}\ \alpha_{\text{Marie}}$$

[13] In the described architecture, the semantic ambiguities are derived from "derivation ambiguities". We can avoid this in considering an intermediate level between $\Sigma_{der\theta}$ and Σ_{trees}. β_{pense} and β'_{pense} would map on the same term of this intermediate level, and the latter would be considered as the actual derivation tree representation level. The upper part would then be considered as lying within the semantic device. However this intermediate level would not provide any additional modeling capability so we do not consider it here.

and the associated meanings:

$$\mathscr{L}_{Log}(\gamma'_0) = \neg(\textbf{think}\ (\textbf{want}\ (\textbf{go}\ \textbf{m})\ \textbf{p})\ \textbf{j})$$
$$\mathscr{L}_{Log}(\gamma'_1) = \textbf{think}\ (\neg\ (\textbf{want}\ (\textbf{go}\ \textbf{m})\ \textbf{p}))\ \textbf{j}$$
$$\mathscr{L}_{Log}(\gamma'_2) = \neg(\textbf{think}\ (\textbf{want}\ (\textbf{go}\ \textbf{m})\ \textbf{p})\ \textbf{j})$$
$$\mathscr{L}_{Log}(\gamma'_3) = \textbf{think}\ (\textbf{want}\ (\neg(\textbf{go}\ \textbf{m}))\ \textbf{p})\ \textbf{j}$$

However, this produces some spurious ambiguities. For instance, assigning a NR reading to the intermediary NR predicate when the matrix NR predicate has a non-NR reading (as for γ'_2) is useless. Similarly, when no negation occur in the matrix predicate, using the NR readings yields the same result as using the non-NR one as the following possible derivations and their interpretations for (1a) shows:

$$\gamma''_0 = \alpha_{\text{partira}}(\beta_{\text{pense}}\ \beta_S^{Id}\ \beta_V^{Id}\ \alpha_{\text{Marie}})\beta_V^{Id}\ \alpha_{\text{Pierre}}$$
$$\gamma''_1 = \alpha_{\text{partira}}(\beta'_{\text{pense}}\ \beta_S^{Id}\ \beta_V^{Id}\ \alpha_{\text{Marie}})\beta_V^{Id}\ \alpha_{\text{Pierre}}$$
$$\mathscr{L}_{Log}(\gamma''_0) = \textbf{think}\ (\textbf{go}\ \textbf{p})\ \textbf{m}$$
$$\mathscr{L}_{Log}(\gamma''_1) = \textbf{think}\ (\textbf{go}\ \textbf{p})\ \textbf{m}$$

4 Improvements

4.1 Avoiding Spurious Ambiguities

In order to avoid spurious ambiguities, we refine the types for S and V adjunction sites. There are several ways to present this refinement using records and dependent types as proposed in [19, 20] that are rather similar to feature structures. As it would involve additional notations, we prefer to keep the atomic types we have used so far, but instead of the atomic type S_A and V_A we now have $S_A[\text{Neg} = no]$, $S_A[\text{Neg} = yes]$, $V_A[\text{Neg} = no]$, and $V_A[\text{Neg} = yes]$ as atomic types.

With these types, an auxiliary tree for a NR predicate will have type:

$$\beta : S_A[\text{Neg} = m] \to V_A[\text{Neg} = n] \to NP \to S_A[\text{Neg} = r] \text{ with } (m, n, p) \in \{\text{yes}, \text{no}\}^3$$

While in principle we could instantiate these auxiliary trees with all the possible combinations, removing some of them will prevent us from getting unwanted readings.

The accepted combinations for non-NR readings are given in Table 1(a) and the ones for NR readings in Table 1(b). The tables show the resulting type $S_A[\text{Neg} = r]$ for each combination of $S_A[\text{Neg} = m]$ and $V_A[\text{Neg} = n]$ values. When a cell is empty, it means there is no term with this type combination.

(a) Combinations for non-NR readings

	$S_A[\text{Neg} = no]$	$S_A[\text{Neg} = yes]$
$V_A[\text{Neg} = no]$	$S_A[\text{Neg} = no]$	$S_A[\text{Neg} = no]$
$V_A[\text{Neg} = yes]$	$S_A[\text{Neg} = no]$	$S_A[\text{Neg} = no]$

(b) Combinations for NR readings

	$S_A[\text{Neg} = no]$	$S_A[\text{Neg} = yes]$
$V_A[\text{Neg} = no]$		$S_A[\text{Neg} = yes]$
$V_A[\text{Neg} = yes]$	$S_A[\text{Neg} = yes]$	$S_A[\text{Neg} = yes]$

Table 2. Allowed feature combinations

We now have four constants for the non-NR reading of *pense que*, but only three for its NR readings:

$$\beta^0_{\text{pense}} : S_A[\text{Neg} = no] \rightarrow V_A[\text{Neg} = no] \rightarrow NP \rightarrow S_A[\text{Neg} = no]$$
$$\beta^1_{\text{pense}} : S_A[\text{Neg} = yes] \rightarrow V_A[\text{Neg} = no] \rightarrow NP \rightarrow S_A[\text{Neg} = no]$$
$$\beta^2_{\text{pense}} : S_A[\text{Neg} = no] \rightarrow V_A[\text{Neg} = yes] \rightarrow NP \rightarrow S_A[\text{Neg} = no]$$
$$\beta^3_{\text{pense}} : S_A[\text{Neg} = yes] \rightarrow V_A[\text{Neg} = yes] \rightarrow NP \rightarrow S_A[\text{Neg} = no]$$
$$\beta'^0_{\text{pense}} : S_A[\text{Neg} = yes] \rightarrow V_A[\text{Neg} = no] \rightarrow NP \rightarrow S_A[\text{Neg} = yes]$$
$$\beta'^1_{\text{pense}} : S_A[\text{Neg} = no] \rightarrow V_A[\text{Neg} = yes] \rightarrow NP \rightarrow S_A[\text{Neg} = yes]$$
$$\beta'^2_{\text{pense}} : S_A[\text{Neg} = yes] \rightarrow V_A[\text{Neg} = yes] \rightarrow NP \rightarrow S_A[\text{Neg} = yes]$$

We also need to adapt the types of the other terms and set the type of β^{Id}_S, β^{Id}_V, and $\beta_{\text{ne pas}}$ to $S_A[\text{Neg} = no]$, $V_A[\text{Neg} = no]$, and $V_A[\text{Neg} = yes]$ respectively. For α_{partira} to accept any kind of parameter, it now comes in four forms $\alpha^{m,n}_{\text{partira}}$ whose types are $S_a[\text{Neg} = m] \rightarrow V_A[\text{Neg} = n] \rightarrow NP \rightarrow S$ with $(m, n) \in \{yes, no\}^2$.

With this setting, the derivation tree analyzing (1a) that uses a NR predicate with two fake adjunctions $\beta^{Id}_{S[\text{Neg}=no]}$ and $\beta^{Id}_{V[\text{Neg}=no]}$ cannot make use of any of the constants associated with the NR reading, but only of β^0_{pense}. So that it now has the single analysis:

$$\gamma'''_0 = \alpha^{no,no}_{\text{partira}}(\beta^0_{\text{pense}}\ \beta^{Id}_{S[\text{Neg}=no]}\ \beta^{Id}_{V[\text{Neg}=no]}\ \alpha_{\text{Marie}})\beta^{Id}_{V[\text{Neg}=no]}\ \alpha_{\text{Pierre}}$$

Similarly, if a negated NR predicate embeds another NR predicate as in (5), and if we have a non-NR reading for the matrix predicate (as for *pense que* in γ'_0 and γ'_2), the type of $\beta^2_{\text{pense}}\ \beta^{Id}_{S[\text{Neg}=no]}\ \beta_{\text{ne pas}}\ \alpha_{\text{Jeanne}}$ necessarily is $S_A[\text{Neg} = no]$, hence cannot serve as first argument of any NR reading β'^i_{veut} of *veut que* if the latter is not negated. This avoids the $\mathscr{L}_{Log}(\gamma'_2)$ reading of the sentence.

4.2 Enforcing the NR Reading

A similar technique can be used to enforce the NR reading when a NPI occurs in the embedded clause. S_A and NP types are declined as $S_A[\text{NPI} = b]$ and

$NP[\text{NPI} = b]$ with $b \in \{yes, no\}$. $[\text{NPI} = yes]$ means it licenses a NPI in the clause. Only NR readings should allow for a $S_A[\text{NPI} = yes]$ resulting type:

$$\beta'_{\text{pense}} : S_A[\text{NPI} = m] \rightarrow V_A[\text{Neg} = n] \rightarrow NP[\text{NPI} = p] \rightarrow S_A[\text{NPI} = yes]$$

if at least one of the m, n, and p is set to yes[14] while non-NR readings only allow for a $S_A[\text{NPI} = no]$ type.

Then, together with the constants:

$$\alpha^0_{\text{est } \ldots \text{ assiette}} : S_A[\text{NPI} = no] \rightarrow V_A[\text{Neg} = yes] \rightarrow NP[\text{NPI} = no] \rightarrow S$$
$$\alpha^1_{\text{est } \ldots \text{ assiette}} : S_A[\text{NPI} = yes] \rightarrow V_A[\text{Neg} = no] \rightarrow NP[\text{NPI} = no] \rightarrow S$$
$$\beta^{Id}_{S[\text{NPI}=no]} : S_A[\text{NPI} = no]$$

the (positive) embedded clause *Pierre est/soit dans son assiette* requires the adjunction of an auxiliary tree $\gamma^{(4)}$ of type $S_A[\text{NPI} = yes]$ to be analyzed as:

$$\gamma^{(5)} = \alpha^1_{\text{est } \ldots \text{ assiette}}\ \gamma^{(4)}\ \beta^{Id}_{V[\text{Neg}=no]}\ \alpha_{\text{Pierre}}$$

Because it has type $S_A[\text{NPI} = yes]$, the $\gamma^{(4)}$ auxiliary tree can only be the result of a β'_{pense} application (NR reading). For instance, the actual analysis of (3b) would be:

$$\gamma^{(6)} = \alpha^1_{\text{est } \ldots \text{ assiette}}\ (\beta'_{\text{pense}}\ \beta^{Id}_{S[\text{NPI}=no]}\ \beta_{\text{ne pas}}\ \alpha_{\text{Marie}})\ \beta^{Id}_{V[\text{Neg}=no]}\ \alpha_{\text{Pierre}}$$

This approach could also be used to model the blocking of NR cyclicity when a NR desire predicate embeds a NR belief predicate [10, 2], or to model the readings forced by embedded discourse connectives.

5 Related Works

Our approach models NR predicates at the syntax-semantics interface. This contrasts with the semantic/pragmatic approaches that explains NR behaviors from presupposition. [2] unifies different trends from this vein using *soft presuppositions* [21]. Because we aim at articulating the semantics of NR predicates with discourse connectives that are lexically introduced as in [12], the syntax-semantics approach allows us to have them modeled at a same level, at least as a first approximation.

Another syntax-semantics interface perspective in TAG on NR predicates is proposed in [11] for German. Its modeling relies on multicomponent TAG [22] with an underspecified semantic representation [23]. The motivation for using multicomponents does not only depend on the analysis of NR predicates. It is motivated in the first place by the modeling of scrambling and other word-order phenomena in German. As it also shows useful to analyze long-dependencies, it is

[14] We should be more careful with the licensed combinations, but this is enough to show how to force the reading.

used for NR predicates as well. Then, both ambiguities and relative scoping are managed at the semantic representation level with the underspecified framework whereas we model them at the syntax-semantics interface level and at the logical representation level respectively.

It is also worth noting that the constraints we model using the fine tuning of types with $[\text{Neg} = m]$ and $[\text{NPI} = n]$ extensions is also present in [11]. The latter implements specific computations on the NEG features of the verbal spine that extend the usual notion of unification of TAG. The different combinations we propose in Table 1(a) and 1(b) for instance closely relate to this computation.

Conclusion

We have presented an account of NR predicates within TAG that relies on a Montague-like semantics for TAG. The different properties of NR predicates are rendered at different levels: the ambiguity of the readings is modeled by lexical ambiguity; the scoping and cyclicity properties are modeled through the lexical semantics and the higher-order interpretation of adjunction nodes; spurious ambiguities are avoided using fine-grained types for terms representing derivation trees. This provides us with a base layer where to account for interactions with discourse connectives and discourse representation.

References

[1] Bartsch, R.: "Negative Transportation" gibt es nicht. Linguistische Berichte **27** (1973)
[2] Gajewski, J.R.: Neg-Raising and Polarity. Linguistics and Philosophy **30**(3) (2007) 289–328
[3] Joshi, A.K., Schabes, Y.: Tree-Adjoining Grammars. In Rozenberg, G., Salomaa, A., eds.: Handbook of formal languages. Volume 3. Springer (1997)
[4] de Groote, P.: Towards Abstract Categorial Grammars. In: Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Proceedings of the Conference. (2001) 148–155
[5] Shieber, S.M., Schabes, Y.: Synchronous tree-adjoining grammars. In: Proceedings of the 13th International Conference on Computational Linguistics. Volume 3., Helsinki, Finland (1990) 253–258
[6] Nesson, R., Shieber, S.M.: Simpler TAG semantics through synchronization. In: Proceedings of the 11th Conference on Formal Grammar, Malaga, Spain, CSLI Publications (29–30 July 2006)
[7] Storoshenk, D.R., Frank, R.: Deriving syntax-semantics mappings: node linking, type shifting and scope ambiguity. In hye Han, C., Satta, G., eds.: Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Framework (TAG+11). (2012) 10–18
[8] Horn, L.R.: Neg-raising predicates: Towards an explanation. In: Proceedings of CLS 11. (1975)
[9] Horn, L.R.: A natural history of negation. University of Chicago Press (1989)
[10] Horn, L.R.: Negative transportation: Unsafe at any speed? In: Proceedings of CLS 7. (1971)

[11] Lichte, T., Kallmeyer, L.: Licensing German Negative Polarity Items in LTAG. In: Proceedings of the Eighth International Workshop on Tree Adjoining Grammar and Related Formalisms, Sydney, Australia, Association for Computational Linguistics (July 2006) 81–90
[12] Danlos, L.: Connecteurs de discours adverbiaux : Problèmes à l'interface syntaxe-sémantique. Lingvisticæ Investigationes **36**(2) (2013)
[13] Danlos, L.: D-STAG: a Formalism for Discourse Analysis Based on SDRT and Using synchronous TAG. In de Groote, P., Egg, M., Kallmeyer, L., eds.: 14th conference on Formal Grammar - FG 2009. Volume 5591 of LNCS/LNAI., Springer (2011) 64–84
[14] XTAG Research Group: A Lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-01-03, IRCS, University of Pennsylvania (2001)
[15] Abeillé, A.: Une grammaire électronique du français. CNRS Éditions (2002)
[16] de Groote, P.: Tree-Adjoining Grammars as Abstract Categorial Grammars. In: TAG+6, Proceedings of the sixth International Workshop on Tree Adjoining Grammars and Related Frameworks, Università di Venezia (2002) 145–150
[17] Pogodalla, S.: Advances in Abstract Categorial Grammars: Language Theory and Linguistic Modeling. ESSLLI 2009 Lecture Notes, Part II. (2009)
[18] Shieber, S.M.: Unifying synchronous tree adjoining grammars and tree transducers via bimorphisms. In: In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006. (2006) 377–384
[19] de Groote, P., Maarek, S.: Type-theoretic extensions of abstract categorial grammars. In: New Directions in Type-Theoretic Grammars, proceedings of the workshop. (2007) 18–30 `http://let.uvt.nl/general/people/rmuskens/ndttg/ndttg2007.pdf`.
[20] de Groote, P., Yoshinaka, R., Maarek, S.: On two extensions of abstract categorial grammars. In Dershowitz, N., Voronkov, A., eds.: Logic for Programming, Artificial Intelligence, and Reasoning, 14th International Conference, LPAR 2007, Yerevan, Armenia, October 15-19, 2007, Proceedings. Volume 4790 of Lecture Notes in Computer Science., Springer (2007) 273–287
[21] Abusch, D.: Presupposition triggering from alternatives. Journal of Semantics **27**(1) (2010) 37–80
[22] Weir, D.J.: Characterizing Mildly Context-Sensitive Grammar Formalisms. PhD thesis, University of Pennsylvania (1988)
[23] Kallmeyer, L., Romero, M.: Scope and Situation Binding for LTAG. Research on Language and Computation **6**(1) (2008) 3–52