Clitic climbing in Grecia Salentina Greek: A dynamic account

Conference Paper · July 2009

CITATIONS
0

READS
40

1 author:

Stergios Chatzikyriakidis
University of Gothenburg
74 PUBLICATIONS **242** CITATIONS

SEE PROFILE

Clitic climbing in Grecia Salentina Greek: A dynamic account

STERGIOS CHATZIKYRIAKIDIS
King's College, London
kafouroutsos@hotmail.com

1. Introduction

Grecia Salentina Greek (GSG) is one of the two varieties of Greek spoken in Southern Italy, the other one being Calabrian Greek. GSG is spoken in a number of villages[1] around the area of Lecce, Southeast Italy.

The clitic system of GSG resembles to a high degree that of Standard Modern Greek in terms of positioning . Clitics are proclitic with non-imperative verbs and enclitic with imperatives:

(1) *Ton* *gapa*
him.CL-ACC loves
'He loves him.'

(2) **Gapa* *ton*
loves him.CL-ACC
'He loves him.'

(3) *Grafe* *to*
write.IMP-2SG it.CL-ACC
'Write it!'

(4) **To* *grafe*
it.CL-ACC write.IMP-2SG
'Write it!'

The only difference noted in the literature (Ralli, 2006) is that GSG unlike SMG does not allow free ordering in clitic clusters in imperative environments:

(5) *Δos* *mu* *to*
give.IMP-2SG me.CL-GEN it.CL-ACC
'Give it to me.' [SMG]

(6) *Δos* *to* *mu*
give.IMP-2SG it.CL-ACC me.CL-GEN
'Give it to me.' [SMG]

[1]The exact number is hard to define. The standard assumption is that the number is 9. However, in a number of these villages like e.g. Zollino GSG is not spoken anymore.

(7) *Do* *mu* *to*
give.IMP-2SG me.CL-GEN it.CL-ACC
'Give it to me.' [GSG]

(8) **Do* *to* *mu*
give.IMP-2SG it.CL-ACC me.CL-GEN
'Give it to me.'

GSG is also the only Modern Greek dialect exhibiting clitic climbing with non-auxiliary verbs, a fact that has not been noticed in the literature so far. In specific, GSG allows clitic climbing with two verbs of the restructuring class,[2] the verbs *sotzo* and *spitseo*, 'can' and 'finish' respectively. However, unlike CC in languages like Italian (Cinque 2002, 2006; Cardinaletti & Shlonsky, 2004 among others), CC is obligatory in GSG:

(9)

a. *To* *sotzume* *avorasi.*
it.CL-ACC can buy.INF

b. **Sotzume* *to* *avorasi.*
can it.CL-ACC buy.INF

c. **Sotzume* *avorasi* *to.*
can buy.INF it.CL-ACC
'We can buy it.'

(10)

a. *To* *spitseo* *tse* *torisi* *avri*
it.CL-ACC finish.1SG COMP see.INF tomorrow

b. **Spitseo* *tse* *to* *torisi* *avri*
finish.1SG COMP it.CL-ACC see.INF tomorrow

c. **Spitseo* *tse* *torisi* *to* *avri*
finish.1SG COMP see.INF it.CL-ACC tomorrow
'I will finish seeing it tomorrow.'

It is crucial to note that these two verbs are the only ones of the restructuring class that still subcategorize for an infinitive. Such a fact is crucial in understanding the unavailability of climbing with the rest of the verbs of the same class, since all the other restructuring verbs use the subjunctive marker *na* as the complementation strategy:

(11)

a. *Telume* *no(na-to)* *avorasume.*
want SUBJ-it.CL-ACC buy.1PL

b. **To* *telume* *na* *avorasume.*
it.CL-ACCwant SUBJ buy.1PL
'We want to buy it.'

[2]Restructuring verbs include modal, motion and aspectual verbs.

(12)

a. *Ancigneo* *na* *to* *toro*
begin.1SG SUBJ it.CL-ACC see.INF

b. **To* *Ancigneo* *na* *toro*
it.CL-ACC begin.1SG SUBJ see.INF
'I begin to see it.'

It seems that the unavailability of the infinitival strategy in the rest of the verbs of the restructuring class is at least one of the reasons that climbing is only available for just these two verbs. Assuming monoclausality is what really lies behind CC constructions, the subjunctive strategy is incompatible with a monoclausal interpretation since disjoint reference is always possible given that the verb folowing the subjunctive marker is finite. What other semantic or historical reasons may be behind this rather idiosyncratic property of GSG is something that I do not know. However, it does not seem that anything semantic is really at play here. There is no generalization across classes and verbs similar in meaning (compare *spicceo* and *ancigneo* for example) behave differently with respect to climbing. Surprisingly, Romanian which like GSG uses the subjunctive strategy but retains the infinitive with a limited number of verbs, does not allow climbing except with the verb *a putea* 'can':

(13)

a. *O* *pot* *vedea*
her.CL-ACC can see.INF

b. **Pot* *vedea* *o*
can see.INF her.CL-ACC
'I can see her.'

I do not have any historical story of why some verbs retained the infinitive and some did not, and as such this issue will not be discussed here. On the other hand, I will try to provide an account of obligatoriness of CC in GSG within the DS framework. But before doing that, let us first take a look at the most prominent analyses of CC in the literature.

2. Approaching Clitic Climbing

The first thing one should think of when giving an analysis of CC, is what is the problem with such constructions, i.e. what makes them problematic to linguistic theory. The problem can be simply stated as a locality violation, with clitics being attached to a verbal host other than the one they constitute arguments of. A number of different accounts have been proposed by the years in different grammatical frameworks. Earlier approaches within the GB/Minimalist tradition assume that CC is a case of restructuring. In all these approaches, a restructuring rule is invoked, in effect transforming a biclausal structure into a monoclausal sone. Such an approach can be found for example in Rizzi (1983), where a restructuring rule is posited to account for CC (see also Manzini (1983) for a similar treatment):

(14) Rizzi's restructuring rule - My formalization

V (P) V.INF → V.CMLX

The above rule transforms a biclausal structure consisting of a verb and an infinitive into a monoclausal structure where the two verbs are assumed to form a verbal complex. However, the fact that in some CC languages a number of adverbs can intervene between the verb and the infinitive caused serious problems to such accounts. On the other hand, Kayne (1989) working within the GB/Barriers framework claimed that

CC is the result of clitic movement out of the infinitival VP. The exact reasoning is then that in non-CC languages the VP consitutes a barrier for movement and thus movement is debarred, while in CC languages the IP L-marks the VP and the latter is not considered a blocking category anymore (in the sense of Chomsky, 1986).

Recent approaches within the minimalist program propose that clitic climbing occurs when one of the verbs appears not as a fully fledged verb heading its own VP, but rather as an instantiation of an FP within the richly articulated FP structure of the clause proposed by Cinque (1999). In that sense optional climbing is caused when one verb can be inserted either as a functional or a lexical verb:

(15) Functional and lexical instantiation of a verb

a) [CP...[FP...[FP V_{restr}[FP...[VP V]]]]] Climbing case

b) [CP...[FP...[FP[VP V_{restr} [CP...[FP...[FP[VP V]]]]]]] Non-climbing case

Cardinaletti and Shlonsky (2004) propose that a) is involved in CC constructions and b) in non-CC constructions. Cinque (2006) on the other hand argues that a) is involved in both cases.

Within HPSG, CC has been considered to be an argument sharing phenomenon (Miller and Sag 1997, Monachesi (1998, 1999) among others). All the HPSG analyses concur on the latter claim . The assumption is that the climbing inducing verb subcategorizes for an infinitive plus its arguments:

(16)
$$\begin{bmatrix} \text{HEAD V} \\ \text{VCLASS modal} \vee \text{aspectual} \vee \text{motion} \\ SUBJ\langle NP \rangle \\ \text{COMPS L} \bigoplus \quad \langle \text{ V} \begin{bmatrix} \text{CLTS} \{\} \\ \langle \text{ NP } \rangle \\ \text{COMPS L} \end{bmatrix} \end{bmatrix}$$

Argument sharing explains why the clitic can climb in CC constructions but does not however have anything to say with respect to restructuring effects found in CC environments like for example unavailability of infinitival negation when CC has taken place (example below from Italian):

(17) **Lo vuole non vedere*
it.CL-ACC want NEG buy.INF
'I want to not see it.'

(18) *Vuole non vederlo*
want NEG buy.INF it.CL-ACC
'I want to not see it.'

Furthermore, it is not clear to me what subcategorization for an infinitive plus its arguments means and furthermore why non-restructuring verbs are not able to do this kind of subcategorization.

I will not go into the exact details of all these approaches since it is not my attention to refine on any of these analyses, but rather to provide an alternative DS analysis. Therefore, in what follows I will present a DS analysis of CC, arguing that the functional vs non-functional distinction assumed in the recent minimalist literature can receive a better interpretation and formalization once a shift towards a parsing oriented framework is done. But first an introduction to the Dynamic Syntax (DS) framework.

3. An informal introduction to Dynamic Syntax

3.1 Basic intuitions behind Dynamic Syntax

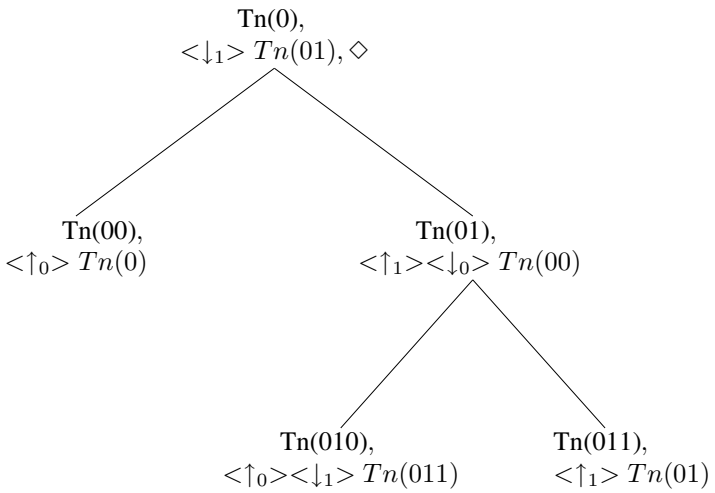
The basic assumption behind DS is that natural languages are interpreted via an incremental, word-to-word, left-to-right cumulative construction of transparent semantic representations with the upper goal to finally construct a logical form of type t (?Ty(t)). Such an interpretation is driven by means of monotonic tree growth, representing the attempt to model the way information is processed in a time-linear, word-to-word manner. However, tree-structures in DS are considerably different from those found in derivational or declarative frameworks like minimalism or HPSG respectively, in that they are not inhabited by words as such, but rather from the representations of those words (Fodor, 1975). Furthermore, the tree structure corresponding to the representation of the ending result of parsing a natural language string is a semantic representation assigned to this natural language string with respect to some context. This semantic representation does not correspond to word order but rather represents argument structure. However, the incremental left-to-right parsing via an array of successive, monotonically growing tree structures, handles word order through the mere definition of incremental parsing. The partial tree structures or the history of parsing stages are used to capture word order phenomena, since this whole process is totally dependent on the way words are ordered. In order for all these intuitions to be carried out, a number of formal tools are employed.

3.2 The formal framework in a glance

3.2.1 LOFT, Tree decorations, Requirements

The parsing process is represented by means of binary trees underpinned by the Logic Of Finite Trees (LOFT, Blackburn and Meyer - Viol, 2001). Left branches are addressed conventionally by adding 0 to the value of the mother node, while right branches by adding 1. The position of a given node is expressed using the predicate Tn (standing for treenode) followed by the actual treenode address. Furthermore two basic tree modalities, $<\uparrow>$ and $<\downarrow>$, standing for the mother and daughter relationship respectively, allow relations between the trees to be represented:

(19)



Notice that a given treenode can be addressed from the perspective of a different treenode. For example $<\uparrow_0><\downarrow_1> Tn(011)$ in the 010 node reads as follows: You will find treenode 011 if you first go up the 0 mother relation and then go down the 1 daughter relation. The $\diamond$ symbol, found in the 0 node in our example is called the pointer, and its basic function is to track which node is the current node under construction any time during the parsing process.[3] Nodes in DS are inhabited by a set of labels, conventionally called "Tree Decorations". The basic elements comprising this set are:

a. **Formula value decorations**. These are represented using the predicate Fo followed by the representation of the entity in brackets, e.g. Fo(John').[4]

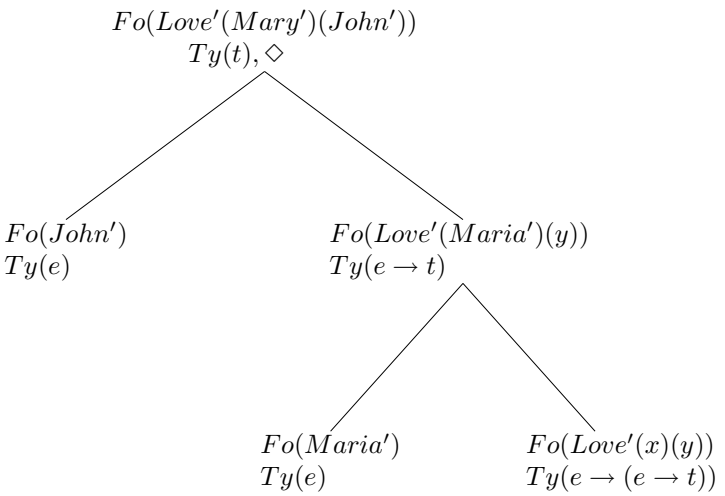
[3]The Pointer function is also one of the ways to account for ungrammaticality in DS. For instance, if the pointer is at a given node which has an outstanding requirement for a type e expression to be found, and the next word that is parsed does not provide such a Type, by providing e.g. a Ty(e→t) expression, the parse will abort rendering the whole string ungrammatical.

[4]The prime indicates that the concept and not the word John is represented.

b. **Type value decorations**. These are represented using the predicate Ty followed by the type of the word/concept in question in brackets, e.g. Ty(e).[5]

A basic concept in the DS framework is that of requirements. Requirements can be seen as goals that need to be achieved. Requirements have the general form ?La$_i$[6] (e.g. ?Ty(e)). In order for a given parse to be successful, no outstanding requirements must exist in the ending tree. In that respect, requirements can be also seen as a device explaining ungrammaticality. Example (19) shows a complete tree in DS. Notice that no outstanding requirements exist[7]:

(20)



3.2.2 Computational - Lexical - Pragmatic rules/actions

The parsing process is driven by three kinds of rules/actions: a) Computational b) Lexical and c) Pragmatic rules/actions. The former are general computational devices, comprising the basic tree construction mechanism. They always involve an input and an output description. The former designates where the pointer must be along with information about the node that the pointer is on or other nodes with respect to the pointer node, while the latter shows the transformation of the input in terms of requirements, adding nodes, pointer movement etc. An example of a computational rule, the rule of COMPLETION, is shown below:

(21) $$\frac{\{...\{Tn(n), ...\}, \{<\uparrow_i>, Tn(n), .., Ty(X) .., \Diamond\}...\}}{\{...\{Tn(n), ..., <\downarrow_i> Ty(X), ..., \Diamond\}, \{<\uparrow_i> Tn(n), ..., Ty(X), ...\}...\}}$$[8]

The above rule moves the pointer to the mother node as soon as any type of requirement is satisfied on any of the daughters of that mother node. The top part reads as follows: There is a node with treenode address Tn(n) and another one dominated by Tn(n) ($<\uparrow_i>, Tn(n)$) that has a satisfied type requirement and also bears the pointer. The output description (second line) presents a situation where the pointer has moved to the Tn(n) addressed node, with an additional statement that records the daughter's satisfied requirement ($<\downarrow_i> Ty(X)$). There are a number of procedural computational rules like the one we have just seen but we won't go into the rest of them for reasons of space. The interested reader is referred to Kempson et al. (2001), Cann et al (2005) for a detailed presentation of a number of computational rules. Additional computational rules will be introduced if needed.

[5]The difference between DS and most of the formal semantic theories with respect to typing is twofold. Firstly, DS has an additional type (cn) standing for common noun, and furthermore there is no recursion on types. Types are a rather closed set. For a more detailed discussion on DS typing see Kempson et al. (2001) and Cann et al. (2005).

[6]Where La stands for label and i$\geqslant$1. For a formal presentation of declarative units in DS consult Kempson et al. (2001: chapter 7).

[7]The lambda terms in the Fo formulas have been excluded for ease of exposition.

[8]Where i=(0,1,*).

On the other hand, lexical rules are basically entries associated with a given word providing instructions on how the parsing must or must not proceed. There are no general rules regarding the content of these instructions. They rather depend on the syntactic nature of these words. However, there is a generalized schema involved in the way these words introduce their content. This general procedural schema followed by a sample lexical entry is shown below:

(22) Lexical entry format

IF	Trigger
THEN	Actions
ELSE	Elsewhere statement

(23) Sample lexical entry for ***Bill***

IF	$?Ty(e)$
THEN	$\texttt{put}(Ty(e), Fo(Bill'), [\downarrow]\bot)$
ELSE	abort

Example (23) reads as follows: If you are in a node that has a type e requirement, then decorate this node with a type e value and a formula value representing the concept given by the word *Bill.* In any other case abort. A proper noun like *Bill* in English will be able to get parsed as soon as a node has a requirement for a type e. This will allow a word like *Bill* to be parsed either as a subject or as an object in English. Other languages with overt noun case marking will have further restrictions for their equivalent entry for *Bill* that will ensure that the proper noun will be parsed in the relevant node depending on case marking. For example we can associate a given case marking with a fixed structural position by means of tree modalities as shown below:

(24) Structural accusative lexical entry

IF	$?Ty(e), \langle\uparrow_1\rangle ?Ty(e \rightarrow t)$
THEN	$\texttt{put}(Ty(e), Fo(x'), [\downarrow]\bot)$
ELSE	abort

The above entry will block a noun of the above type to be parsed in the subject node (00) simply because the condition $?Ty(e \rightarrow t)$ is not going to be satisfied.[9]

Lastly, pragmatic actions involve contextual information providing information with respect to the parsing process. One very basic rule is the rule of SUBSTITUTION which updates a formula metavariable into a proper formula referring to some entity in the context.[10] I won't discuss any other pragmatic actions in this paper but the interested reader is directed to Kempson et al. (2001) and Cann et al. (2005) for further information on pragmatic actions.

3.2.3 Underspecification, LINK

Central within the DS framework is the concept of underspecification, the idea that parts of natural language may not be fully specified by the time they enter into the parsing procedure. Of course underspecification is not in itself a new concept in linguistics. It has been extensively used the last 20 years by formal semanticists to deal with ambiguity and anaphora resolution. What is novel however, is that underspecification is moved into the area of syntax,[11] making the syntax the dynamic part rather than the semantics. DS uses two types of underspecification: a) Content underspecification and b) Structural underspecification.

[9] In DS binary trees, as already mentioned, do not encode word order but rather represent argument structure. In that respect the subject node is always in the same position no matter what the actual word order is. This position is the 00 node. Given that, it is clear why the condition is not satisfied.

[10] See Kempson et al. (2001) and Cann et al. (2005) for a formal definition of the rule of SUBSTITUTION.

[11] There is however a similar notion, the notion of functional uncertainty in LFG (Bresnan, 2001)

With respect to content underspecification, DS employs the use of metavariables, functioning as mere content placeholders with a requirement that substitution of the metavariable will take place at later stages of the parse. A classic example of content underspecification is pronouns. DS assumes that the lexical entry for a pronoun, say *he*, involves the projection of a metavariable as the value the Formula takes. This metavariable must be updated as soon as a proper formula value is provided by the context or by the natural language string itself. The metavariable comes with person and case restrictions depending on the pronoun. A requirement that a proper Fo value must be found ensures that the node which bears the metavariable will eventually get a proper formula value. The lexical entry for the pronoun *he* is shown below:

(25) Lexical entry for *he*

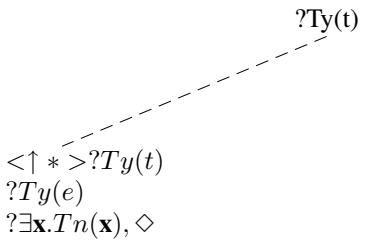
IF	$?Ty(e), \langle\uparrow_1\rangle ?Ty(t)$
THEN	$\texttt{put}(Ty(e), Fo(U_{male'}), ?\exists \mathbf{x}.Fo(\mathbf{x}), [\downarrow]\bot)$
ELSE	$abort$

Structural underspecification on the other hand is represented in DS by employing a set of rules, the family of ADJUNCTION rules.[12] *ADJUNCTION effectively introduces a node, which position in the tree is not yet fixed at the time of its introduction. To be more specific, the rule of *ADJUNCTION projects such an unfixed node from the initial ?Ty(t) node which bears a requirement for an expression of type e to be found at that node:[13]

(26) $$\frac{\{...\{Tn(a), ...?Ty(t), \Diamond\}\}}{\{...\{Tn(a), ...?Ty(t)\}, \{<\uparrow *> Tn(a), ?\exists x.Tn(x), ..., ?Ty(e), \Diamond\}\}}$$

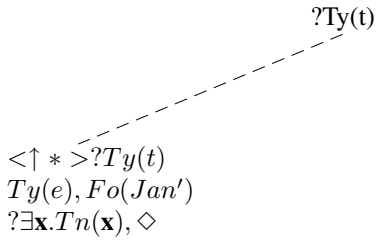
The effect of the rule is shown schematically below:

(27)



An NP can be parsed on that unfixed node satisfying the type e requirement:

(28) Parsing *ton Jani* 'John' on an unfixed node[14]



[12] I will present two of the various variants of the ADJUNCTION rule here. Additional *ADJUNCTION rules will be introduced later on if needed.

[13] The kleene star operator is a way to encode underspecification in the modal language. $<\uparrow^*> X$ reads as: X holds at a node above me or at the current node. Using the opposite modality, i.e the daughter modality, the kleene star denotes the notion of dominance plus reflexiveness. The pure notion of dominance is encoded by means of the kleene plus operator (+). In that respect $<\uparrow^+> X$ reads as: There is a node above me where X holds. We will see later on how we will exploit both of the operators in our analysis.

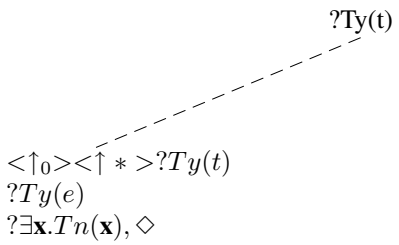
[14] I ignore determiners for the moment.

The $?\exists\mathbf{x}.Tn(\mathbf{x})$ restriction will ensure that the node must be fixed at a later stage in the parse.[15] The underspecified relation $<\uparrow * >?Ty(t)$ will enable the NP to be parsed in different structural positions. The *ADJUNCTION rule is a natural way to encode this intuition. The *ADJUNCTION rule will account for long scrambling cases as well, since the tree modality used does not restrict the full NP to apply in the local domain. A variant of *ADJUNCTION however, *LOCAL ADJUNCTION will do just that, i.e. it will restricts the potential fixing sites of the node to local nodes. The rule is shown below:

(29) $$\frac{\{...\{Tn(a),...,?Ty(t),\Diamond\}...\}}{\{...\{Tn(a),?Ty(t),\}...\{<\uparrow_0><\uparrow * > Tn(a),?\exists x.Tn(x),...,?Ty(e),\Diamond\}...\}}$$

The effect of the rule in tree notation is the following:

(30)



Notice that the modality has changed from $<\uparrow^*>$ to $<\uparrow_0><\uparrow_1^*>$. This will ensure that the NP in question is parsed in the local propositional domain. [16] The two rules are used for long and short distance scrambling effects respectively. We will see later on the relevance of these rules with respect to clitics.

While the *ADJUNCTION rules involve the creation of an unfixed node that has a requirement for a specified treenode address in the tree under construction to be found, LINK structures involve the construction of a second tree structure independently of the initial one, which however posits a requirement for a shared term between the two trees. In order for LINK structures to be modelled, we need to introduce two new modal operators, $< L >$ and $< L^{-1} >$. The former refers to a tree structure which is linked, as it is shown schematically in (33), by means of an arrow, with the current node, while the latter refers to that node. LINK rules are also a family of rules, sharing the characteristics just mentioned. For demonstration purposes we will present one of them. The latter comes in the form of two rules, the rules of TOPIC STRUCTURE INTRODUCTION and TOPIC STRUCTURE REQUIREMENT [17] respectively. These two rules are used by Cann et al. (2005) to account for Hanging Topic Left dislocation structures(HTLD). The first rule effectively creates a LINK transition between the initial node and a top node with a *type e* requirement. The rule is shown below:

(31) $$\frac{\{\{Tn(0),?Ty(t),\diamond\}\}}{\{\{Tn(0),?Ty(t)\}\},\{< L > Tn(0),?Ty(e),\diamond\}}$$

Notice that the above rule does not mention anything about a shared term. That is because there is no shared term at the time the rule applies. The requirement for a shared term is introduced via the second rule the rule of TOPIC STRUCTURE REQUIREMENT shown below. This rule applies as soon as the dislocated argument is parsed (*as for Mary* in our example):[18]

[15] $?\exists x.Tn(x)$ reads as: There is a requirement for a proper treenode address (fixed) to be found. If the latter does not happen, then the parse cannot be completed as at least one outstanding requirement will exist in the tree.

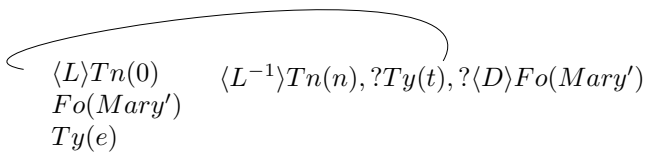
[16] Assuming that all argument nodes are the 0 nodes and an additional propositional domain will involve a type t in one of the argument nodes, this rule will exclude cases where the NP is associated with an argument in the next propositional domain.

[17] The prototypical LINK rule is the rule of LINK *ADJUNCTION used by Cann et al. (2005) to account for relative clauses. We choose to present the TOPIC STRUCTURE INTRODUCTION and TOPIC STRUCTURE REQUIREMENT rules instead, which are basically a variant of the prototypical LINK rule. The interested reader is referred to Cann et al. (2005: 85-94) for the prototypical LINK rule.

[18] The D modality stands for the downward modality that encodes the kleene star operator and can furthermore extend over LINK structures In that respect D is defined as D=$\{\downarrow_0,\downarrow_1,\downarrow,\downarrow^*,L\}$.

(32) $$\frac{\{\{Tn(0), ?Ty(t)\}, \{< L > Tn(0), Fo(a), Ty(e), \diamond\}}{\{\{Tn(0), ?Ty(t), ? < D > Fo(a), \diamond\}\}, \{< L > Tn(0), Fo(a), Ty(e)\}}$$

After the introduction of these two rules, we get the following:

(33) 

In an HTLD construction, the dislocated NP will be parsed in the node where the LINK begins. After TOPIC STRUCTURE REQUIREMENT has applied, a requirement for the same Formula value provided in the first tree will be put in the LINKed tree. This will ensure that a copy of the formula *Mary* will also be provided by the linguistic string, for example a resumptive pronoun in English. There are a number of important things with respect to the general LINK rule, but we won't discuss them here since these are not relevant to the scope of the paper. The interested reader is however referred to Kempson et al. (2001) and Cann et al. (2005) for more information on the various LINK rules.

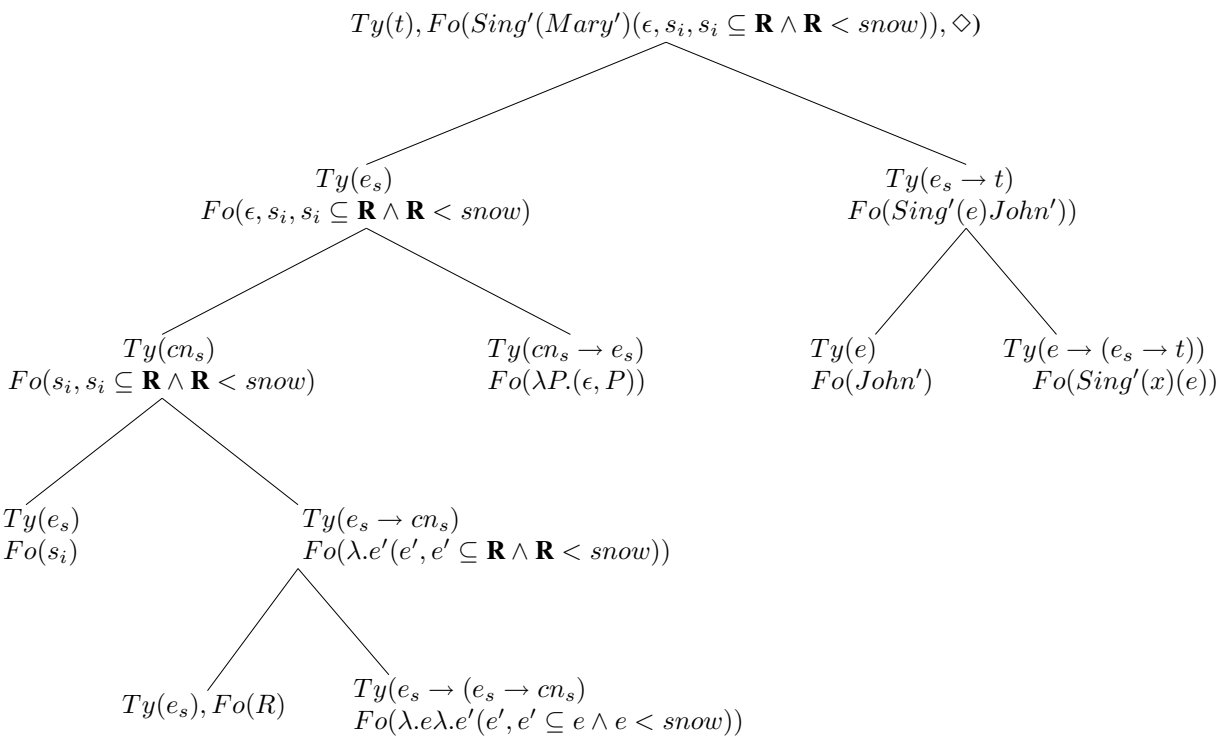
3.3 Newest developments in DS - Tense and Aspect

In both Kempson et al. (2001) and Cann et al. (2005), no attempt to address tense or aspect is made. Tense is encoded as a diacritic in Cann et al (2005) (e.g. Tns(Past)), noting that a proper analysis of tense is pending in the framework. Recent advances within DS however assume a treatment of tense based on the introduction of an explicit situation argument, introducing a situation in which the formula value the proposition expresses will be true.[19] This situation argument node is then the locus of all tense and aspect information. In order to for this situation argument node to be represented two additional nodes are added to the standard DS propositional spine. A node standing for the situation argument, which is assumed to be of type es (with *s* standing for situation),[20] and a functor node of type $e_s \rightarrow t$. The situation argument node in line with quantified NPs (see Kempson et al., 2001; Cann et al., 2005 among others) is assumed to involve complex structure. The example below shows the structure assumed in Cann (forthcoming) after the complete parse of the sentence like 'Mary sang':

(34) Parsing *Mary sang*

[19]The situation argument node in DS was first introduced in Gregoromichelaki (2006).

[20]In line with Gregoromichelaki (2006: 196) we assume that values in D_{Ty} involve Ty(e) as a general type with subtypes e_i for individuals, e_s for situations etc. Cann (forthcoming) uses the notation e_w which is nothing more than a notational variant.



In the above structure, the intransitive verb sing is taken to be a two-place predicate, subcategorizing for both a subject and an event/situation argument. Let us see what the complex situation argument does. In the lowest e_s node, the reference time metavariable **R** is introduced. This will combine with the semantic specifications given for the past tense in the lowest functor node ($Fo(\lambda.e\lambda.e'(e', e' \subseteq e \wedge e < snow))$), to return a formula value in which the first lambda bound variable (e) is substituted by **R** ($Fo(\lambda.e'(e', e' \subseteq \mathbf{R} \wedge \mathbf{R} < snow))$). This new formula states that the remaining lambda bound variable e' is contained within or holds at R ($e' \subseteq \mathbf{R}$) and that the reference time precedes the utterance time ($\mathbf{R} < snow$)). In the intemediate e_s node, a situation s_i is introduced. This situation will substitute the remaining lambda bound variable (e') to return the formula value $Fo(s_i, s_i \subseteq \mathbf{R} \wedge \mathbf{R} < snow)$, in effect providing a situation that will bear the given tense/aspect specifications. The last step involves quantifying over the last formula we have obtained. In the example above, the situation with the given specifications is existentially quantified to return a formula value which roughly states that a past situation exists ($Fo(\epsilon, s_i, s_i \subseteq \mathbf{R} \wedge \mathbf{R} < snow)$). The last step involves substituting this last formula, for the lambda bound e in the formula value for sing to get the well-formed type t formula $Ty(t), Fo(Sing'(John')(\epsilon, s_i, s_i \subseteq \mathbf{R} \wedge \mathbf{R} < snow))$.

Tense/aspect information are assumed to be projected mainly from verbs, both auxiliary and content verbs. However, a number of other elements can be taken to provide such information, like modality/tense particles/markers or even negation markers. With this said, I stop the discussion on the newest developments on tense/aspect in DS, noting that relevant additional details will be introduced if needed and directing the interested reader to Cann (forthcoming) for a detailed discussion of this approach.

4. A dynamic account of Clitic Climbing

4.1 Clitics in DS

A number of approaches have been proposed for clitics in DS (Bouzouita 2008a, 2008b; Chatzikyriakidis 2009a, 2009b, forthcoming; Kempson & Chatzikyriakidis 2009). In all these analyses, positioning restrictions are defined as restrictions on the current parse state, while the actions projected by the clitic vary depending on the level of underspecification involved in each case. For example, 1st/2nd person accusative clitics in Spanish have been proposed (Kempson & Cann, 2007; Kempson & Chatzikyriakidis, 2009) to project locally unfixed nodes, a proposal largely motivated by the syncresis found in these clitic forms. On

the other hand, 3rd person accusative clitics in the same language are treated as projecting fixed structure, in effect building and decorating the direct object node. There are a number of interesting predictions that such a proposal makes, especially with respect to the PCC but this is something that will not bother us in this paper. The interested reader is however directed to Kempson and Cann (2007) and Kempson and Chatzikyriakidis (2009) for an analysis of the PCC in DS. In this paper I will use the Italian 3rd person accusative neuter clitic *to* as a role model and I will not deal with clitic clusters or person case restrictions. The role language used will be Italian but the same account will be easily extendable to other CC languages as well (see Chatzikyriakidis 2010 for a DS analysis of optional climbing.). As already said, positioning restrictions are defined as restrictions on the current parse state. For example the trigger shown below, effectively captures proclisis associated with clitics in Italian:[21]

(35) Trigger ensuring proclisis

IF	$?Ty(t)$	
THEN	IF	$[\downarrow_1^+]?Ty(x)$
	THEN ...	
ELSE	abort	

The above reads as follows: If you are in a node with a type t requirement, then if all functor nodes (nodes with a Tn address) below that node bear a type requirement, then the clitic can be parsed. This means that no verbal type has been parsed yet, since in case it did we would have at least one fixed node.[22] The next step is to define the actual actions projected by the clitic. Given a fixed node analysis for 3rd person accusative clitics, *to* will basically build the direct object node, project a type value and a formula metavariable on that node and return the pointer to the most local type t node above it (gofirst(?Ty(t))). Furthermore, an additional trigger is added, specified as a disjunction in the embedded IF part of the algorithm (OR), positing that the clitic can also be parsed in case an imperatival feature is present in the type t requiring node:[23]

(36) Lexical entry for the third person accusative clitic *to*[24]

IF	$?Ty(t)$	
THEN	IF	$[\downarrow_1^+]?Ty(x)$
	OR	
	IF	$+IMP$
	THEN	$\texttt{make}(\langle\downarrow_1\rangle);\texttt{go}(\langle\downarrow_1\rangle);$
		$\texttt{make}(\langle\downarrow_0\rangle);\texttt{go}(\langle\downarrow_0\rangle)$
		$\texttt{put}(Ty(e), Fo(U_x), ?\exists\mathbf{x}.Fo(\mathbf{x}), gofirst(?Ty(t)))$
ELSE	Abort	

Having sketched the way clitics are treated in DS, it is time to move to the actual analysis of CC in DS.

4.2 The analysis

The problem posed by CC in GSG, given our account of clitics and assuming a biclausal structure is involved in CC constructions, is that the clitic ends up projecting a type e value in the direct object node

[21]The same trigger is a general proclitic trigger and as such will work for languages with similar clitic positioning restrictions (Spanish, Italian, French to name a few.)

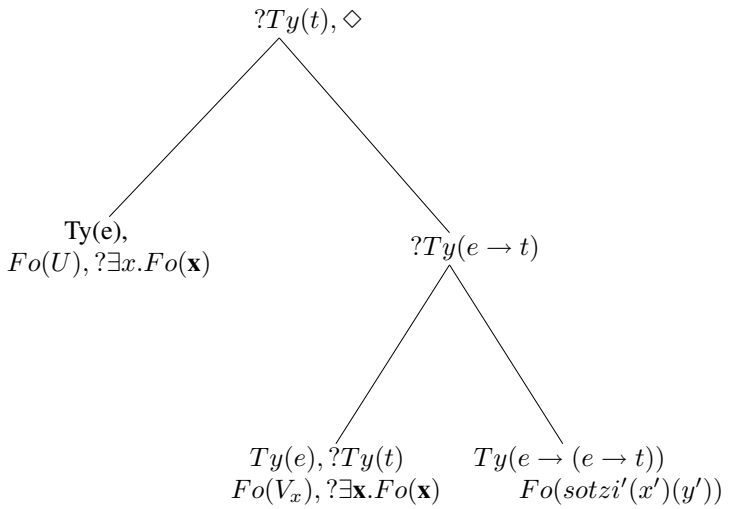
[22]Note that such an entry will also correctly predict *to* to be possible after dative clitics in clitic clusters like *mu to*. Assuming that *mu* will project an unfixed node (following Kempson and Cann, 2007; Kempson and Chatzikyriakidis, 2009), the clitic *lo* can be parsed.

[23]Note that the feature +IMP are used as diacritics and do not by any means constitute any serious attempt to give an analysis of imperatives and infinitives in DS.

[24]The subscript x in the formula metavariable U stands for the restrictions on metavariable update that the clitic will bear. These will not be specified in this paper but will have to be assumed in a more complete analysis to prevent overgeneration, e.g. avoiding a situation where *it* is updated by a formula value specified as female.

projected by the modal that however bears a type t requirement, standing for the type the infinitival clause will receive. The partial tree after the modal *sotzo*, 'can', in *to sotzo vorasi*, 'I can buy it', is parsed is shown below:

(37) Parsing *sotzi* in *to sotzi vorasi*



Notice that both a type e value and type t requirement exist in the direct object node. Such a partial tree cannot lead to a well-formed parse. The reason is simple. Satisfying the type t requirement will lead to a situation where the node carries two distinct, incompatible type values, something obviously not allowed by the system. Leaving the requirement unsatisfied will not do any better, since an outstanding requirement will always exist in the tree. Since a complete tree representing a successful parse, as already mentioned in the introduction to the framework, must not have any outstanding requirements, the parse will never be completed in case the type t requirement does not get satisfied. This is the situation we get assuming climbing inducing verbs are parsed like regular verbal complement verbs, i.e. when a biclausal structure is assumed to be involved in CC constructions. However, CC has been argued convincingly to involve a monoclausal rather than a biclausal structure (see Cinque 2006 for an extensive discussion). The next question that comes to mind is how DS can capture this fact, or more practically how such a thing is formalizable in DS. The answer that I will propose, basically assumes that climbing inducing verbs behave like auxiliary verbs in that they do not project a verbal functor type, but rather project their semantics inside a complex situation argument node. According to Ronnie Cann (forthcoming) English auxiliaries are taken to project tense and aspect in the situation argument node complex, while they further project a type value and a formula metavariable in the predicate node (011 node). The same assumptions can be used to analyze auxiliaries eço/ixa 'have/had' in SMG. In SMG these two auxiliaries are used to form the present and past perfect respectively. In the presence of a clitic in perfect constructions, climbing of the clitic before the auxiliary is obligatory:

(38)

a. *To* *exo* *δesi*
it.CL-ACC have.1SG tied.PST-PRTCPL

b. **Exo* *to* *δesi*
have.1SG it.CL-ACC tied.PST-PRTCPL
'I have it tied.' [SMG]

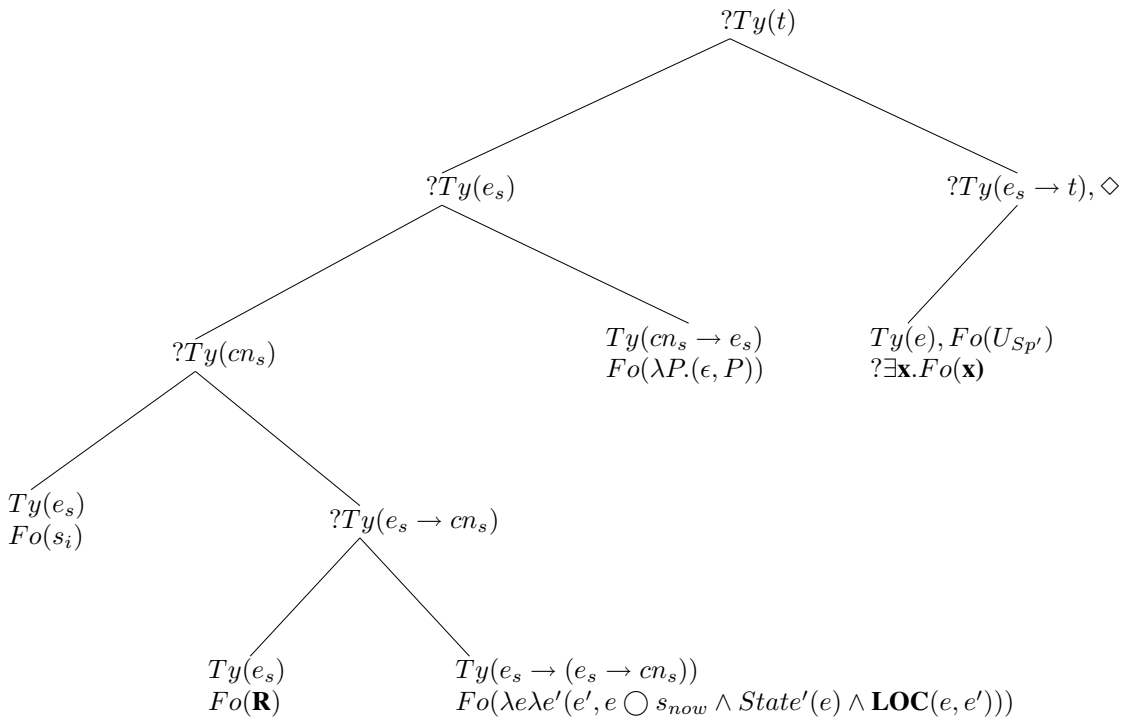
In line with Cann's (forthcoming) analysis for auxiliaries in English, I assume that in SMG auxiliary *eço*, projects all the relevant semantic information in the complex situation argument node and thus no verbal type value needs to be projected from the auxiliary. However, unlike English which projects a formula

metavariable (Fo(U)) and a type value containing a metavariable (Ty(U→(e_s → t))) in the 011 node, SMG auxiliaries do not project the 011 at all. They do however project the subject node along with a formula metavariable and a type value. The reason for the latter move is that subject agreement in SMG is encoded in the auxiliary. Thus, all the relevant information that make pro-drop possible in SMG should be encoded in there. They eventually as in the English case leave the pointer at the situation functor node. The lexical entry plus the result of parsing *exo*, 'have', is shown below:

(39) Lexical entry for *exo* 'have.1SG' in SMG

IF	?Ty(t)
THEN	$\texttt{make}(\langle\downarrow_0\rangle);\texttt{go}(\langle\downarrow_0\rangle);\texttt{put}(?Ty(e_s))$
	$\texttt{make}(\langle\downarrow_1\rangle);\texttt{go}(\langle\downarrow_1\rangle);\texttt{put}(Ty(cn \rightarrow e_s), Fo(\lambda P.(\epsilon, P)))$
	$\texttt{go}(\langle\uparrow_1\rangle);\texttt{make}(\langle\downarrow_0\rangle);\texttt{go}(\langle\downarrow_0\rangle);\texttt{put}(?Ty(cn_s))$
	$\texttt{make}(\langle\downarrow_0\rangle);\texttt{go}(\langle\downarrow_0\rangle);\texttt{put}(Ty(e_s), freshput(s))$
	$\texttt{make}(\langle\downarrow_0\rangle);\texttt{go}(\langle\downarrow_1\rangle);\texttt{put}(?Ty(e_s \rightarrow cn))$
	$\texttt{make}(\langle\downarrow_0\rangle);\texttt{go}(\langle\downarrow_0\rangle);\texttt{put}(Ty(e_s, Fo(\mathbf{R}))$
	$\texttt{go}(\langle\uparrow_0\rangle);\texttt{make}(\langle\downarrow_1\rangle);\texttt{go}(\langle\downarrow_1\rangle);\texttt{put}(Ty(e_s \rightarrow (e_s \rightarrow cn)))$
	$\texttt{put}(Fo(\lambda e\lambda e'(e', e \bigcirc s_{now} \wedge State'(e) \wedge \mathbf{LOC}(e, e'))))$
	$\texttt{go}(\langle\uparrow_1\rangle\langle\uparrow_1\rangle\langle\uparrow_0\rangle\langle\uparrow_0\rangle);\texttt{make}(\langle\downarrow_0\rangle);\texttt{go}(\langle\downarrow_0\rangle);\texttt{put}(Ty(e), Fo(U_{Speaker'}), ?\exists.\mathbf{x}.Fo(\mathbf{x}))$
	$\texttt{go}(\langle\uparrow_1\rangle)\texttt{make}(\langle\downarrow_1\rangle);\texttt{go}(\langle\downarrow_1\rangle);\texttt{put}(?Ty(e_s \rightarrow t))$
ELSE	abort

(40) The effect of parsing *exo*



The auxiliary introduces both the reference time R and a fresh situation s_i. Furthermore, it introduces the tense/aspect specifications for the present perfect ($Fo(\lambda e\lambda e'(e', e \bigcirc s_{now} \wedge State'(e) \wedge \mathbf{LOC}(e, e'))$), where $\bigcirc$ stands for the overlap relation while **LOC** expresses an underspecified relation between the event and the reference points that enables the various perfect readings to be generated.[25] Additionally, the auxiliary projects the formula $Fo(\lambda P.(\epsilon, P))$ which will basically existentially quantify over the formula value that will emerge after combining the formula value of the intermediate e_s node with the formula value of the

[25] See Cann (2009) for details and motivation as regards the **LOC** feature.

?Ty(e_s →cn) requiring node. The rest of the structure projected is rather straightforward. Assuming this entry for auxiliaries and furthermore the entries we have given for clitics in SMG, climbing is predicted to be the only option with auxiliaries. Let us see why. Assuming a clitic has been parsed first, the pointer is left at the type t requiring node. This node is then the trigger for parsing the auxiliary (see lexical entry 39) and the rest follows naturally. On the other hand, assuming that the auxiliary has been parsed first the pointer is left at the situation functor node. In case a clitic comes into parse, the parse will abort, since the initial triggering point I have given for clitics, i.e. a type t requiring node, will not be satisfied given that the pointer will be at the situation functor node (?Ty(e_s →t)). Notice that the pointer cannot move up via COMPLETION, since no type or formula will be satisfied in the situation functor node in parsing an auxiliary. In that respect, the only option is for the clitic to appear proclitic to the auxiliary according to fact. Notice furthermore, that the auxiliary *exo*, 'have', in SMG unlike its English counterpart (Cann, forthcoming) does not project a type value and a formula metavariable in the 011 node. This is because assuming a type value and a formula metavariable are projected by the auxiliary in the 011 node, VP ellipsis will be predicted to be possible with auxiliaries in SMG contrary to fact. Given the projection of a formula metavariable in the 011 node by the auxiliary, substituting this formula metavariable with a value provided by the context (which is always a possibility for metavariables), can give us a well formed sentence. This works nicely for English but will fallaciously predict VP ellipsis with auxiliaries to be possible in SMG. The examples below show the relevant facts for English and SMG:

(41) *Have you hit John? Yes, I have*

(42) *Exis xtipisi ton Jani? *Ne, eço (SMG)*
have hit the$_{acc}$ John$_{acc}$ yes have
'Have you hit John? Yes, I have.'

So far, so good. The question how is this account relevant to restructuring verb climbing? The answer is that an analogous account can straightforwardly be put forth for restructuring verbs by simply making the following two assumptions: a) Climbing inducing verbs do not project any verbal functor type and b) The semantics of restructuring verbs are captured in the complex situation argument node similarly to auxiliaries. Let us illustrate this claim using the GSG verb *sotzo* 'can'. As already said, following Cann (forthcoming) I take modals to behave like auxiliaries in that they project their semantics in the situation argument node rather than projecting a verbal type. However, remember that modals are content verbs and more than tense and aspect information will be needed to capture their semantics. Fortunately, there is a way in which this can be done. Remember that aspect and tense information where introduced in the Ty($e_s \rightarrow (e_s \rightarrow cn)$) node of the complex situation argument node and percolate up to the Ty($e_s \rightarrow cn$), where they combine with the situation Fo(s_i)) provided by the intermediate e_s node. Now, assuming that this situation (Fo(s_i) can also be evaluated with respect to possible worlds we immediately get a solution to our problem. The only thing that we will have to further assume is that a 'world' parameter is also projected as part of a complex argument involving a situation and a world parameter, both independently quantified by the right quantifier in each case. The assumption I am going to make is that modal verbs project such a complex situation argument involving a situation and a world parameter. Then, the next step is the use of possible world semantics. For example, the lexical entry for *must* can be seen as specifying that the proposition expressed by the infinitive plus its arguments is true in all contextualy given possible worlds accessible from the default world.[26] The same can be argued to be the case for the ability modal *sotzo* 'can', the difference being that the domain of quantification in this case is ability worlds, a subset of the set of possible worlds rather than just possible worlds. *Sotzo* under such an approach, will project a complex Fo value in the internal e_s node, encoding both a situation and a world parameter (Fo(s_i,w_i)). This world parameter must be a member of the set of ability

[26] A fact already suggested to me by Ronnie Cann (p.c).

contexts which in turn are a subset of the set of contextually accessible worlds. The next thing we need to take care of is the form of quantification quantifying over these possible worlds. Since we are dealing with the set of all ability contexts, what we need is universal quantification. In that respect, we posit a tau term, instead of an epsilon term to capture the universal quantification properties of the world parameter projected by *sotzo*. Furthermore, tense/aspect specifications are also going to be included. Under the account just sketched, the only difference between modals and auxiliaries is that the former introduce a complex situation argument including both a situation and a world parameter in the intermediate e_s node, whereas the latter projects only a situation parameter. One further difference between the two is the node where the pointer is assumed to be left. We have seen that the pointer is left at the $e_s \rightarrow t$ node after an auxiliary is parsed. However, leaving the pointer at the same node in the case of restructuring verbs will basically predict that infinitives have two distinct parsing trigges, a type t and a type $e_s \rightarrow t$ requiring trigger. The type t requiring trigger is independently needed for constructions where the infinitive functions as the complement of a regular complement taking verb. In that case, and assuming that the complement taking verb will decorate the direct object node with a type t requirement and will leave the pointer there with no other nodes existing below that node, the trigger for the infitive must be the type t node. In order to avoid redundancy, I posit that the trigger for infinitives is a type t requiring node in all cases [27]. Lastly, I further assume that restructuring verbs in GSG further project the predicate node, in contrast to SMG auxiliaries and similarly to English auxiliaries and modals. The reason for this is that VP ellipsis is possible with restructuring verbs in GSG:

(43) *To* *sotzi* *vorasi?* *Ne,* *sotzi*
it.CL-ACC can.3SG buy.INF yes can.3SG
'Can he buy it? Yes, he can.'

Assuming that *sotzi* projects a formula metavariable in the predicate node along with a type value, then given standard DS assumptions (Kempson et al., 2001; Cann et al., 2005) on metavariable update, this update can be done via the context in which case no overt input is needed.

Putting all these assumptions together, one gets the lexical entry for 'sotzo' shown below:

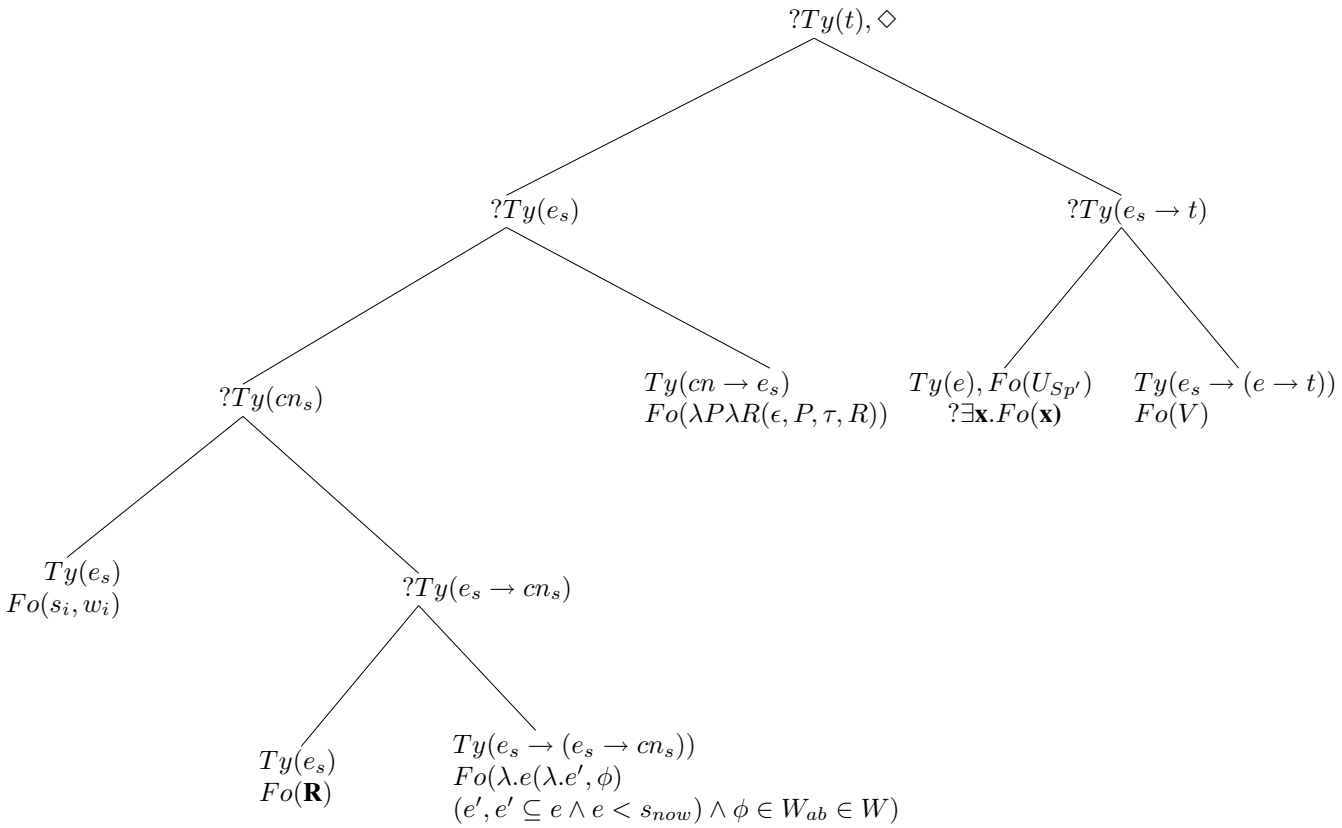
(44) Lexical entry for sotzo.1SG, 'can' in GSG

IF	?Ty(t)
THEN	$\texttt{make}(\langle\downarrow_0\rangle);\texttt{go}(\langle\downarrow_0\rangle);\texttt{put}(?Ty(e_s))$
	$\texttt{make}(\langle\downarrow_1\rangle);\texttt{go}(\langle\downarrow_1\rangle);\texttt{put}(Ty(cn \rightarrow e_s), Fo(\lambda P\lambda R(\epsilon, P, \tau, R)))$
	$\texttt{go}(\langle\uparrow_1\rangle);\texttt{make}(\langle\downarrow_0\rangle);\texttt{go}(\langle\downarrow_0\rangle);\texttt{put}(?Ty(cn))$
	$\texttt{make}(\langle\downarrow_0\rangle);\texttt{go}(\langle\downarrow_0\rangle);\texttt{put}(Ty(e_s), freshput(w_i, s_i))$
	$\texttt{make}(\langle\downarrow_1\rangle);\texttt{go}(\langle\downarrow_1\rangle);\texttt{put}(?Ty(e_s \rightarrow cn))$
	$\texttt{make}(\langle\downarrow_0\rangle);\texttt{go}(\langle\downarrow_0\rangle);\texttt{put}(Ty(e_s), Fo(R))$
	$\texttt{go}(\langle\uparrow_0\rangle;)\texttt{make}(\langle\downarrow_1\rangle);\texttt{go}(\langle\downarrow_1\rangle);\texttt{put}(Fo(\lambda.e(\lambda.e', \phi)$
	$(e', e' \subseteq e \wedge e < snow) \wedge \phi \in W_{ab} \in W))$
	$\texttt{go}(\langle\uparrow_1\rangle\langle\uparrow_0\rangle\langle\uparrow_0\rangle\langle\uparrow_0\rangle);\texttt{make}(\langle\downarrow_1\rangle);\texttt{go}(\langle\downarrow_1\rangle);\texttt{put}(?Ty(e_s \rightarrow t)$
	$\texttt{make}(\langle\downarrow_0\rangle);\texttt{go}(\langle\downarrow_0\rangle);\texttt{put}(Ty(e), Fo(U_{Speaker'}), ?\exists.\mathbf{x}.Fo(\mathbf{x}))$
	$\texttt{go}(\langle\uparrow_1\rangle\langle\uparrow_0\rangle\langle\uparrow_0)\texttt{put}(Ty(e_S \rightarrow (e_s \rightarrow e_s)), Fo(V)), gofirst(?Ty(t))$
ELSE	abort

The result of parsing *sotzo* is shown below:

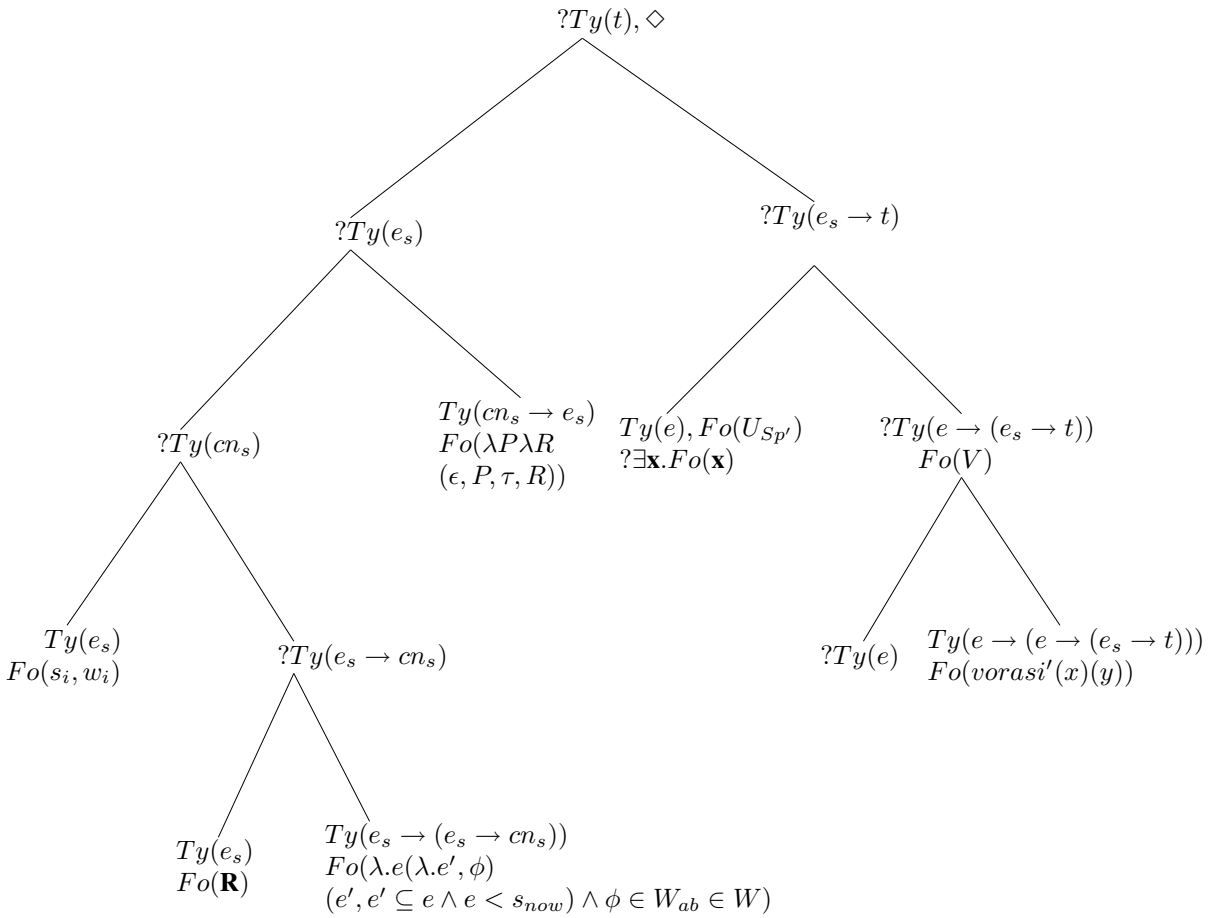
(45) Parsing *sotzo*

[27] A further welcomed result is that such treatment can further enable us to distinguish between infinitives and past participles without actually referring to any other of their properties. For example, under such an analysis, an infinitive will always be impossible after an auxiliary has been parsed first, since the pointer in that case will be at the situation functor node.



The intermediate Ty(e_s) node has a complex formula value introducing both a situation and a world parameter (Fo(s_i,w_i)). The lowest functor node on the other hand (Ty(e_s →(e_s → cn_s)) contains three lambda bound variables ($Fo(\lambda.e(\lambda.e', \phi)(e', e' \subseteq e \wedge e = snow) \wedge \phi \in W_{ab} \in W)$). The first variable (e) is to be substituted by the reference time metavariable **R**. Then, the other two (e' and ϕ) stand for the two variables that will be substituted by the two parameters of the complex Fo value of the internal e_s node, s_i and w_i respectively. In that sense, e (**R** after substitution) is taken to hold at a time e', where time e' is the same as the utterance time s_{now}, while ϕ is taken to belong to the set of ability contexts, which in turn are a subset of the contextually accessible worlds W ($\phi \in W_{ab} \in W$). Then, at the 001 node, the form of quantification for each of the arguments is introduced. The situation parameter is associated with existential while the world parameter with universal quantification (Fo(λPλR.(ϵ,P,τ,R))). The subject node is further projected (010 node) and a type value and a formula metavariable are posited in the same node. Lastly, the predicate node is also built and decorated with a type value and a formula metavariable. The pointer is left at the type t requiring node. Given the structure projected by *sotzo* in (45), the clitic cannot be parsed after *sotzo* has already done so. The trigger of the accusative clitic will not be satisfied because a functor node will bear a type value (the 011 node), while the trigger for genitive clitics will not be satisfied given that a number of fixed nodes will exist after parsing *sotzo.* Parsing of the clitic after the infinitive is not possible for the same reasons. The infinitive builds the 0111 node and projects a verbal type and a formula value on that same node. Furthermore, it also builds the direct object node and decorates it with a type e requirement. Lastly, it returns the pointer to the type t requiring node:

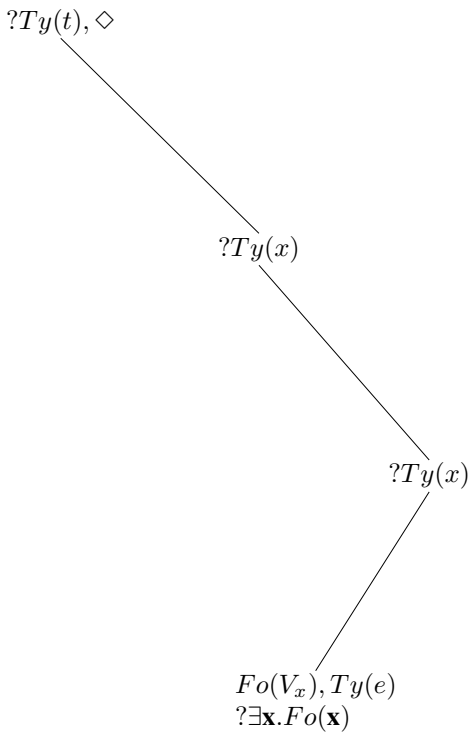
(46) Parsing *vorasi* 'to buy' in **sotzo vorasi to* 'I can buy it'



It is at that point that the clitic comes into parse. The lexical entry for the clitic *to* 'it' specifies that no functor node with a type value must exist in order for the parsing process to proceed. However, this is not true in the tree above, since two functor nodes with type values exist (the 011 and 0111 nodes). Thus, CC is the only option in the presence of *sotzo* in GSG. In a CC case like the one shown below, the clitic is parsed first, building and decorating the direct object node with a type value and a formula metavariable:

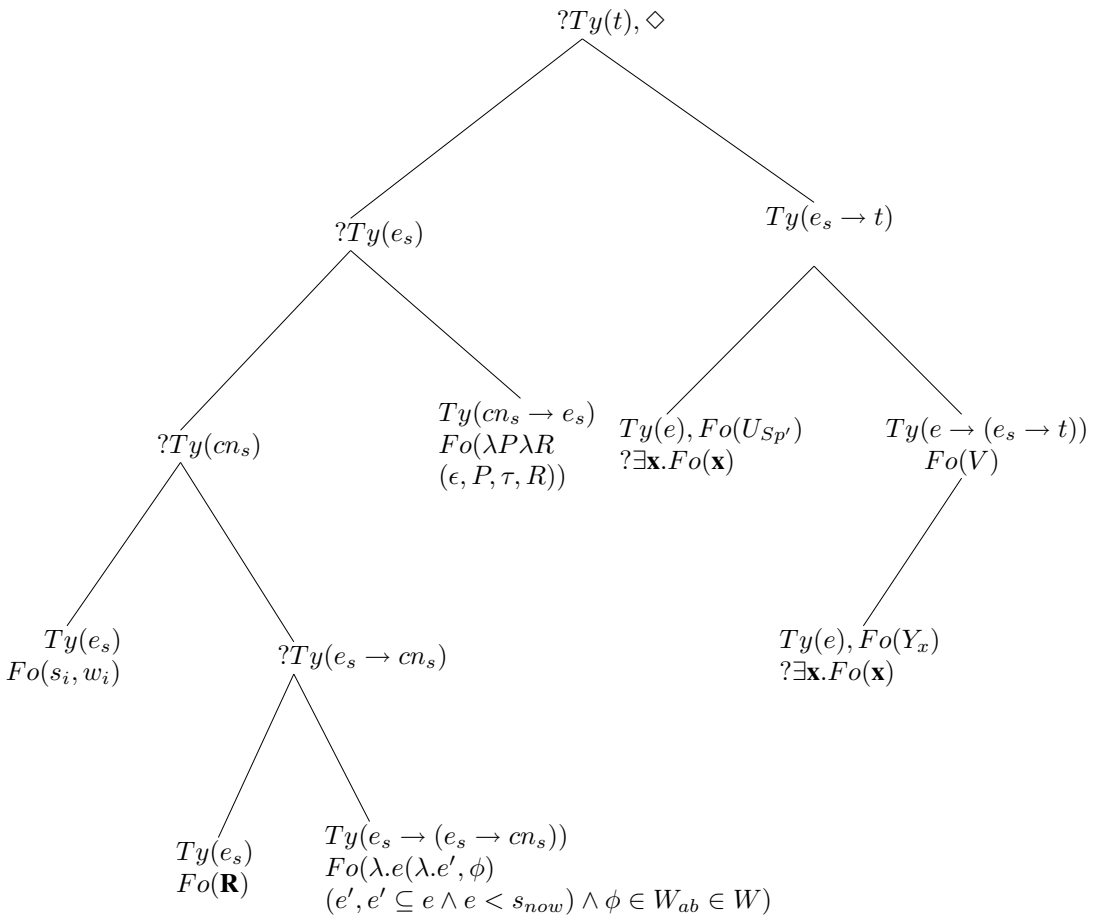
(47) *To* *sotzo* *vorasi*
it.CL-ACC can.1SG buy.INF
'I can buy it.'

(48) Parsing *to* 'it' in *to sotzo vorasi* 'I can buy it.'



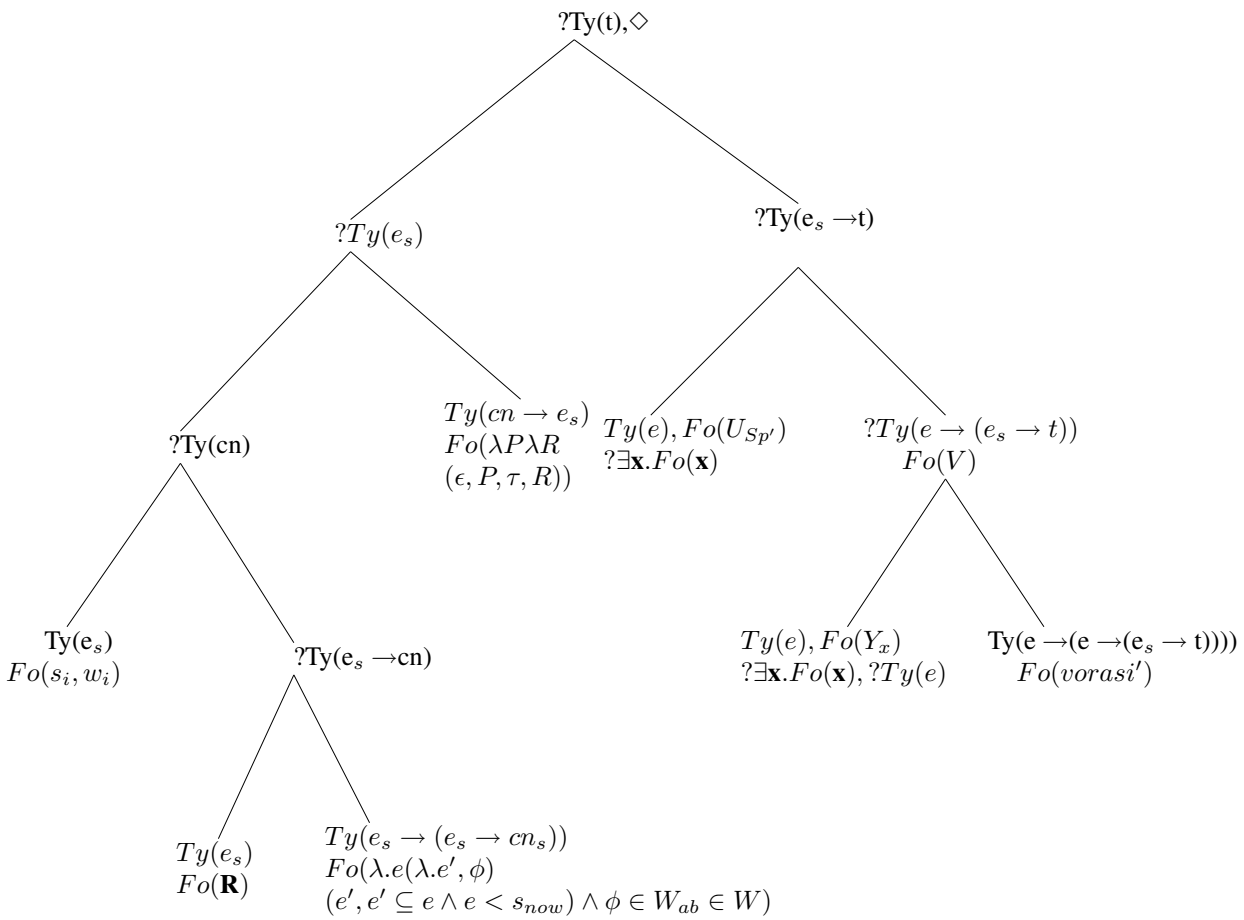
With the pointer being at the type t requiring node, *sotzi* comes into parse:

(49) Parsing *sotzo* 'can' in *to sotzo vorasi* 'I can buy it'



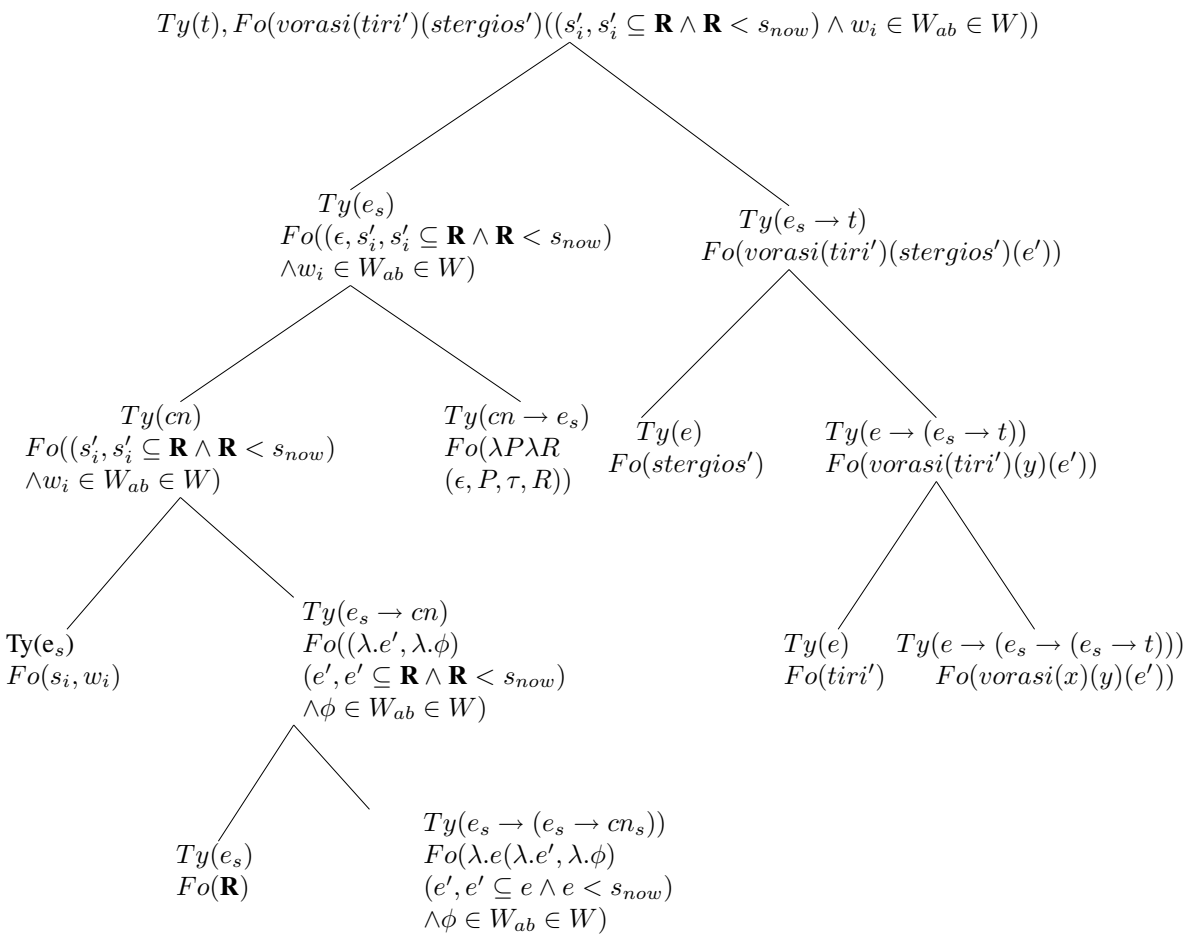
The pointer is again at the type t requiring node. The infinitive comes into parse, projecting a type plus a formula value in the 0111 node. It further builds the direct object node and decorates it with type e requirement:

(50) Parsing *vorasi* 'can' in *to sotzo vorasi* 'I can buy it'



At that point the rules of ELIMINATION, THINNING, COMPLETION and metavariable SUBSTITUTION will apply getting rid of any requirements that have been satisfied, combining formula and type values via functional application and modus ponens respectively and providing proper Fo values for the metavariables from the context. The result of all these processes is the well-formed parse shown below:

(51) Completing the parse



The difference between optional climbing languages on the one hand and GSG on the other is that in the latter, infinitives are incompatible with clitics whereas in optional climbing languages infinitives can host clitics. In that sense, the lexical entries in these languages will have to involve a trigger that will capture enclitic positioning with infinitives as well (see Chatzikyriakidis 2009, forthcoming, in preparation for a DS analysis of optional CC).

The two verbs climbing inducing verbs in GSG can form multiple climbing constructions, in which the clitic climbs across two verbs:

(52) *(T)o* *sotzo* *spiccetsi* *tse* *di* *avri*
it.CL-ACC can finish.INF COMP see.INF tomorrow
'I can finish seeing it tomorrow.'

(53) **Sotzo* *spiccetsi* *tse* *to* *di* *avri*
can spiccetsi COMP it.CL-ACC see.INF tomorrow
'I can finish seeing it tomorrow.'

(54) **Sotzo* *spiccetsi* *tse* *di* *to* *avri*
can spiccetsi COMP see.INF it.CL-ACC tomorrow
'I can finish seeing it tomorrow.'

Climbing in the intermediate position is also banned in GSG, in contrast to languages like Italian where intermediate climbing is permitted:[28]

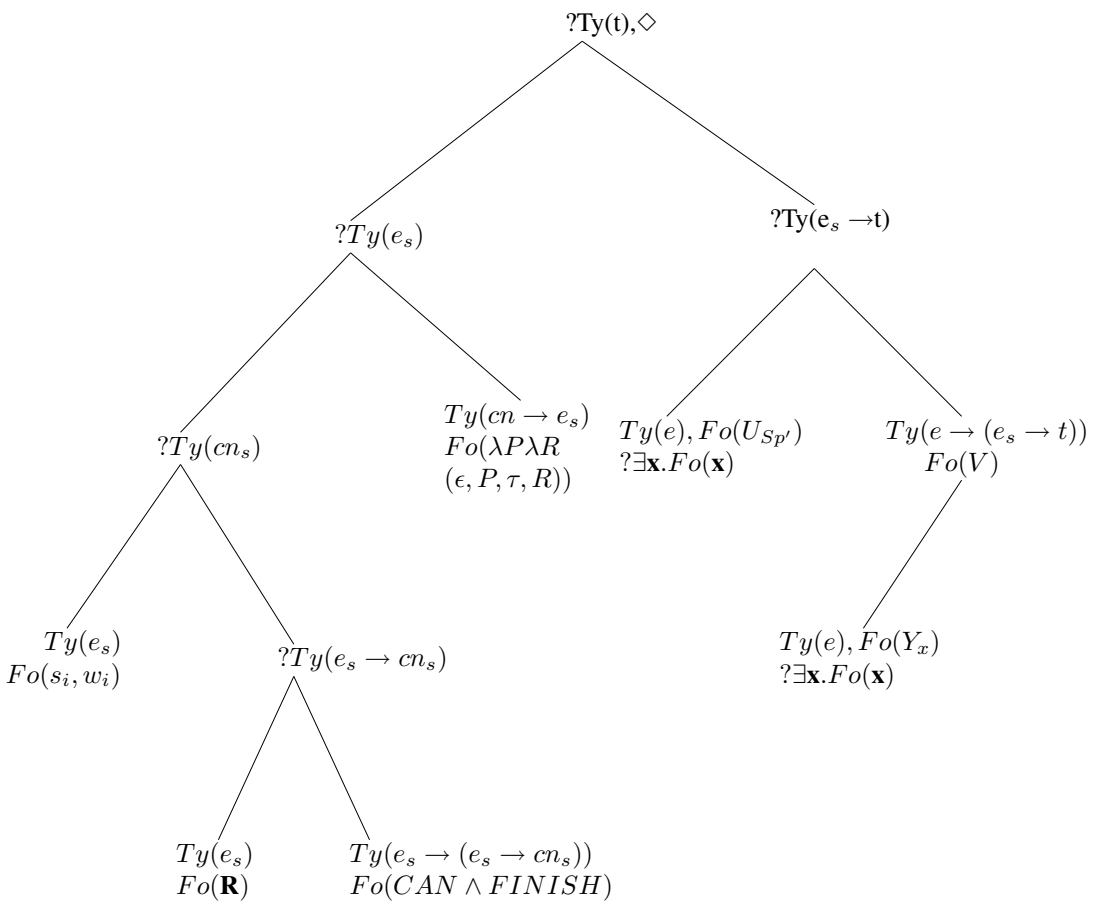
[28] A number of restrictions apply in case of multiple climbing in Italian. See Cardinaletti & Shlonsky (2004) for an extensive discussion on the issue.

(55) *Sotzo to spiccetsi tse di avri
can it.CL-ACC finish.INF COMP see.INF tomorrow
'I can finish seeing it tomorrow.'

(56) *Sotzo spiccetsi to tse di avri
can finish.INF it.CL-ACC COMP see.INF tomorrow
'I can finish seeing it tomorrow.'

The account sketched so far correctly predicts the above facts. Let us see why. The restructuring infinitive is assumed to be parsed in the same sense as *sotzo*, i.e. as encoding its semantics in the situation argument node. The difference between finite and infinitive restructuring verbs lies in the fact that infinitives in contrast to finite verbal forms will not introduce a freshput situation or world parameter in the internal e_s node but will rather depend on the situation or world already projected by the finite verb. Tense, aspect or world specifications will be projected on the relevant nodes. If these nodes already contain such specifications, a combination of the two specifications is done using a form of generalized conjunction in the sense of Cann (forthcoming), effectively combining two formulae of the same type. The example below ilustrates the result of parsing *sotzo spiccetsi* 'can finish':[29]

(57) Parsing *sotzo spitsetsi*

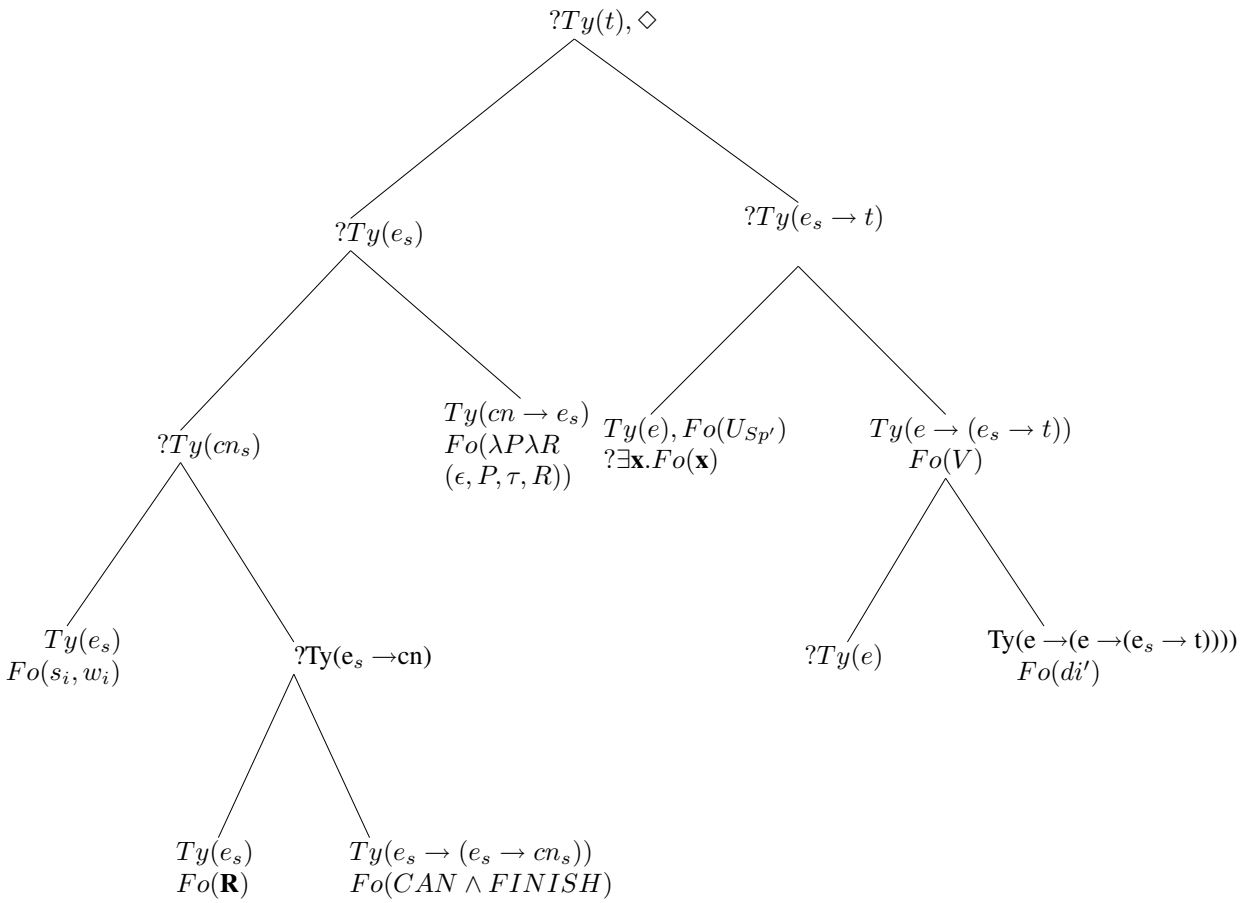


Example (53) is predicted to be ungrammatical, since the clitic trigger $[\downarrow_1^+]$?Ty(x) is again not satisfied (a verbal type and a number of fixed nodes exist). Thus the parse wil abort. In (54) the clitic will come into

[29]Note that the exact formal specifications of what both *sotzo* and *spitsetsi* contribute to the 0001 node are not shown. This is because the exact semantics have not been fully worked out yet in this case. The statement *CAN* ∧ *FINISH* is only a diacritic for what the exact semantics projected by *sotzo* and *spitsetsi* will be.

parse after the lexical infinitive will project its structure, i.e. the rest of the propositional template plus a verbal type in the 0111 node:

(58) After parsing *di* 'see.INF' in **sotzo spiccetsi tse di to avri*



The clitic cannot be parsed since in the above partial tree a functor type is present (Ty(e →(e→)(e$_s$ →t)))), thus the triggering restrictions we have given for clitics do not get satisfied. The trigger for clitics ($[\downarrow_1^+]?Ty(x)$) will fail due to the presence of two functor types (in the 011 and 0111 nodes). Similarly in (55) and (56), parsing of a clitic is not possible, since again one functor type will be present (in the 011 node). The only option in that respect under our account is multiple CC in accordance with the facts. The account I have proposed correctly predicts the climbing facts in GSG. Furthermore, CC with auxiliary verbs in SMG is also accounted correctly within the same reasoning. There are a number of other welcoming results this account has to offer for a number of other phenomena found in CC languages, notably unavailability of negating infinitives in CC contexts [30] and auxiliary switch in Italian but these will not be discussed here for reasons of relevance. The reader interested in the way DS can account for these phenomena is directed to Chatzikyriakidis (2009, forthcoming).

5. Conclusion

In this paper, I have presented a first sketch of a DS account of CC in GSG. I have argued that the phenomenon of CC can receive a straightforward explanation and formaliation once one shifts into a dynamic perspective. In particular, I provided an analysis of restructuring verbs as auxiliary-like verbs based on the auxiliary analysis given by Cann (forthcoming). Under this analysis, restructuring verbs do not project a verbal type but rather project their semantics inside the complex situation argument node. This assumption

[30] Such a fact is precluded independently for GSG, since infinitives cannot be negated in general in GSG or Calabrian Greek (Katsoyannou, 1995).

straightforwardly captures the phenomenon of CC. Multiple climbing was accounted using the exact same reasoning, the difference being that more than one projects semantic information in the situation argument node in this case. The obligatoriness of CC in GSG is attributed to a general ban of clitics from appearing with infinitives. This auxiliary-like analysis of the specific two verbs in GSG is not available to any other of the verbs belonging to the class of restructuring verbs in GSG. Thus, all other verbs of the restructuring verbs will be parsed as regular complement taking verbs and as such will not induce climbing according to fact.

Acknowledgements

I'm indebted to my supervisor, Ruth Kempson, for invaluable comments and constructive critique on various parts of this paper. I also thank the audiences at MGDLT4 and LAGB 2009 for useful comments and discussion. Particularly I want to thank Ronnie Cann and Wilfried Meyer-Viol for helpful comments. The Arts and Humanities Research Council (AHRC), the Leventis Foundation and the Alexandros Miaris Board are gratefully thanked for providing partial funding related to parts of the work presented in this paper. Normal disclaimers apply.

References

Blackburn P. & Meyer - Viol W. (1994). Linguistics, Logic and Finite Trees. *Bulletin of Interest Group of Pure and Applied Logics 2,* 2-39.

Bouzouita M. (2008a). *The Diachronic Development of Spanish Clitic Placement.* Phd Thesis, King's College, London.

Bouzouita M. (2008b). At the Syntax-Pragmatics Interface: Clitics in the History of Spanish. In: Cooper R. & Kempson R. (eds), *Language Evolution and Change.* Cambridge: Cambridge University Press.

Cardinaletti A. & Shlonsky U. (2004). Clitic Positions and Restructuring in Italian. *Linguistic Inquiry 35*, 519-557.

Cann R., Kempson R. & Marten L. (2005). *The Dynamics of Language*. Oxford: Elsevier.

Cann R. (forthcoming). Towards an Account of the English Auxiliary System: Building Interpretations Incrementally. In: Kempson R. & Gregoromichelaki E. (eds), *The Dynamics of Lexical Interfaces*. CSLI publications.

Chatzikyriakidis S. (2006).*Clitics in Modern Greek : A Dynamic Account*. MSc thesis, King's college London.

Chatzikyriakidis S. (2009a). Clitics in Grecia Salentina Greek: A Dynamic Account. *Lingua 119*, 1939-1968.

Chatzikyriakidis S. (2009b). *Clitic Climbing: A dynamic account*. Ms, King's College, London.

Chatzikyriakidis S. In preparation. *Clitics in Four Dialects of Modern Greek : A dynamic account*. Phd thesis, King's college, London.

Chatzikyriakidis S. (forthcoming). A Dynamic Account of Clitic Climbing: A First Sketch. In: Kempson R. & Gregoromichelaki E. (eds), *The Dynamics of Lexical Interfaces*. CSLI publications.

Chomsky N. (1986). *Barriers*, MIT Press.

Cinque G. (1999). *Adverbs and Functional heads: A Cross- Linguistic Perspective*. New York, Oxford University press.

Cinque G. (2001). Restructuring and the order of aspectual and root modal heads. In: Cinque G. & Salvi G. (eds.), *Current studies in Italian Syntax.* Elsevier, 137-155.

Cinque G. (2006). *Restructuring and Functional Heads*. Oxford Studies in Comparative Syntax, Oxford university press.

Fodor J.A. (1975). *Language of Thought*. Cambridge: Harvard University Press.

Katsoyannou M. (1995). *Le Parler Greco de Galliciano (Italie): Description d'une Langue en Voie de Disparition*. Phd thesis, University of Paris VII.

Kayne, R., (1989). Null Subjects and Clitic Climbing. In: Jeggli O. & Safir, K.J (eds), *The Null Subject Parameter*. Dordrecht, pp. 239-261.

Kempson R. & Cann R. (2007). *Production Pressures, Syntactic Change and the Emergence of Clitic Pronouns*. Ms., King's college London.

Kempson R., Meyer-viol W. & Gabbay, D., (2001). *Dynamic Syntax: The Flow of Language Understanding*. Blackwell Publishing.

Kempson R. & Chatzikyriakidis S. (2009). *The Person Case Constraint as a Treegrowth Restriction.* Ms, King's College, London.

Manzini R. (1983). *Restructuring and Reanalysis*. Phd thesis, MIT.

Miller P. & Sag I. (1997). French Clitic Movement without Clitics or Movement. *Natural Language and Linguistic Theory 15*, 573-639.

Monachesi, P., (1993). Object Clitics and Clitic Climbing in Italian HPSG Grammar. In: *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics.* Utrecht, 437-442.

Monachesi P. (1998). Italian Restructuring Verbs: a Lexical Analysis. In: Hinrichs, E., Kathol, A., Nakazawa, T. (eds.), Complex Predicates in Nonderivational Syntax. *Syntax and Semantics 30*, Academic Press, San Diego, 313-368.

Monachesi P. (1999). The Syntactic Structure of Romanian Auxiliary (and Modal) Verbs. In: Bouma G. et al. (eds), *Constraints and resources in natural language syntax and semantics*. CSLI publications, Stanford

Ralli A. (2006). Syntactic and morphosyntactic phenomena in Modern Greek dialects: The state of the art. *Journal of Greek Linguistics 7*, 121-159.

Rizzi L. (1982). Issues in Italian Syntax. *Studies in Generative grammar 11.* Foris Publications, Dordrecht.