Linguistic **I**ssues in **L**anguage **T**echnology – **LiLT**
September 2010

Generating with Discourse Combinatory Categorial Grammar

Crystal Nakatsu
Michael White

Submitted, April 2010
Revised, September 2010
Published by CSLI Publications

Generating with Discourse Combinatory Categorial Grammar

Crystal Nakatsu, *The Ohio State University,* Michael White, *The Ohio State University*

Abstract. This article introduces Discourse Combinatory Categorial Grammar (DCCG) and shows how it can be used to generate multi-sentence paraphrases, flexibly incorporating both intra- and inter-sentential discourse connectives. DCCG employs a simple, practical approach to extending Combinatory Categorial Grammar (CCG) to encompass coverage of discourse-level phenomena, which furthermore makes it possible to generate clauses with multiple connectives and — in contrast to approaches based on Rhetorical Structure Theory — with rhetorical dependencies that do not form a tree. To do so, it borrows from Discourse Lexicalized Tree Adjoining Grammar (D-LTAG) the distinction between structural connectives and anaphoric discourse adverbials. Unlike D-LTAG, however, DCCG treats both sentential and discourse phenomena in the same grammar, rather than employing a separate discourse grammar. A key ingredient of this single-grammar approach is *cue threading*, a tightly constrained technique for extending the semantic scope of a discourse connective beyond the sentence. As DCCG requires no additions to the CCG formalism, it can be used to generate paraphrases of an entire dialogue turn using the OpenCCG realizer as-is, without the need to revise its architecture. In addition, from an interpretation perspective, a single grammar enables easier management of ambiguity across discourse and sentential levels using

LiLT Volume 4, Issue 1, September 2010.
Generating with Discourse Combinatory Categorial Grammar.

standard dynamic programming techniques, whereas D-LTAG has required a potentially complex interaction of sentential and discourse grammars to manage the same ambiguity. As a proof-of-concept, the article demonstrates how OpenCCG can be used with a DCCG to generate multi-sentence paraphrases that reproduce and extend those in the SPaRKy Restaurant Corpus.

1 Introduction

In this article, we introduce Discourse Combinatory Categorial Grammar (DCCG), a simple, practical approach to extending Combinatory Categorial Grammar (CCG; Steedman, 2000, Steedman and Baldridge, 2009) to encompass coverage of discourse-level phenomena. DCCG treats both sentential and discourse phenomena in the same grammar, with no additions required to the CCG formalism. In this way, DCCG can be used with existing CCG chart realization techniques (White, 2004, 2006a,b) to generate paraphrases that are not confined to single sentences, but rather allow for a flexible treatment of both intra- and inter-sentential discourse connectives (such as *but* or *also*).

DCCG follows Discourse Lexicalized Tree Adjoining Grammar (D-LTAG; Webber et al., 2003, Forbes et al., 2003, Webber, 2004, Forbes-Riley et al., 2006) in distinguishing two types of discourse connectives: 1) structural connectives (i.e., coordinating and subordinating conjunctions as well as paired conjunctions such as *not only . . . but also* and *on the one hand . . . on the other hand*), whose discourse arguments must be adjacent to one another; and 2) discourse adverbials (e.g. *then, also, otherwise, instead*), whose first discourse argument is anaphoric and need not be structurally adjacent to its second argument. By following D-LTAG in distinguishing discourse adverbials from structural connectives, it becomes possible to generate clauses with multiple connectives — as long as there is no more than one structural connective — and to handle rhetorical dependencies that do not form a tree, in contrast to traditional generation approaches based on Rhetorical Structure Theory (RST; Mann and Thompson, 1988). However, DCCG departs from D-LTAG in that it extends a sentential grammar to include discourse phenomena, in contrast to the D-LTAG approach where a separate discourse grammar applies to the derivations of the sentential grammar.[1] A key ingredient of our single-grammar approach is the use of

[1] As D-LTAG's proponents have not *advocated* the use of two grammars — instead emphasizing the similarly lexicalized nature of syntactic and discourse processing — it is not clear whether this aspect of their approach is an inherent one, though it is common to existing formalizations and to the single implementation effort to date. To avoid awkward circumlocutions, in the rest of the paper we will

cue threading, a tightly constrained technique in which a cue feature on categories is used to record the presence of an intersentential structural connective in a clause, and to thread this information from the clause to the point in the derivation where the connective's semantics takes scope, thereby extending the connective's scope beyond the sentence. Allowing structural connectives to extend their scope in this way makes it unnecessary to use two grammars to accomplish the same end.

An important advantage of employing a single grammar, as in DCCG, is that there is no need to solve the knotty problem of how to efficiently manage the interaction of the two grammars, since with the single-grammar approach existing dynamic programming techniques can be used off-the-shelf. From the interpretation perspective, a particularly interesting difficulty with using two grammars is that some connectives are ambiguous between sentential and discourse uses, and thus they pose the problem of determining which is the operative grammar when using separate sentence and discourse grammars. In addition, in contrast to D-LTAG, where the treatment of structural connectives in medial position is complicated by the need to reanalyze them as if they appeared initially, DCCG can handle such connectives straightforwardly, since cue threading can be used with the same syntactic categories normally used in sentential grammars.

As a proof-of-concept, we have developed a DCCG to generate comparative descriptions like the ones in the SPaRKy Restaurant Corpus (Walker et al., 2007), as well as additional comparative descriptions that go beyond those produced by SPaRKy. With this hand-crafted grammar, several hundred to several thousand multi-sentence paraphrases can typically be generated from a disjunctive logical form that compactly specifies the possible realizations. These realizations range in quality from excellent to nearly unreadable, and thus a ranking model is needed to separate the wheat from the chaff. As developing good ranking models for these multi-sentence paraphrases is a largely separate task, we leave it for future work.

The rest of the article is structured as follows. Section 2 reviews D-LTAG, including the distinction between structural and anaphoric connectives, as well as the complexities involved with positing separate sentential and discourse grammars. Sections 3 and 4 review CCG syntactic derivations and Hybrid Logic Dependency Semantics (HLDS; Kruijff, 2001, Baldridge and Kruijff, 2002, White, 2006b), respectively, where the latter is the form of compositional semantics used with OpenCCG.

---

leave aside the possibility that D-LTAG might be formalized and implemented in a different way.

Section 5 introduces DCCG, showing how cue threading can be used to extend the semantic scope of a structural connective beyond the sentence, and how anaphoric connectives can be used to handle non–tree structured rhetorical dependencies. Section 6 illustrates how DCCG inputs can be derived from the content plans in the SPaRKy Restaurant Corpus. Section 7 compares our approach to related work on discourse and generation, including those based on RST. Finally, Section 8 concludes with a summary and discussion of future work.

2 D-LTAG

A Discourse Lexicalized Tree Adjoining Grammar (D-LTAG; Webber et al., 2003, Forbes et al., 2003, Webber, 2004, Forbes-Riley et al., 2006) is a Lexicalized Tree Adjoining Grammar (LTAG) that models the discourse structure of a multi-clausal utterance or text. A D-LTAG is paired with a sentential LTAG for the lower level syntactic structure, and together, the two grammars are intended to deliver a complete structural analysis of a multi-clausal discourse.

As noted in the introduction, D-LTAG models two types of discourse connectives: structural connectives and discourse adverbials. Structural connectives include coordinating and subordinating conjunctions as well as paired connectives such as *on the one hand ... on the other hand*. These connectives relate discourse unit arguments that must be found structurally adjacent to one another, and provide a primary tree structure for a discourse. Discourse adverbials, in contrast, are argued to make an anaphoric, rather than structural, connection with the previous discourse. With discourse adverbials, such as *then* or *also*, one discourse unit argument is provided syntactically, while the other is resolved anaphorically, and thus has no structural adjacency restrictions on its potential location in the prior discourse (Webber et al., 2003). Consequently, the rhetorical relations established by discourse adverbials, together with those from structural connectives, need not form a tree.

As an example of both types of connectives, the multi-sentence comparison[2] in (1) has been split into discourse units that correspond to individual clauses (for reference, each clause has been given a unique ID). Using these basic clauses as the minimal discourse units, the structure of a given text can be illustrated as a graph with a primary tree structure, where the leaves of the tree are the basic clauses (or simple

[2]The comparative examples in this paper are from the same domain as those produced by the SPaRKy Sentence Planner. They are mostly similar to the types of sentences SPaRKy produces, in terms of syntax and content, though some go beyond SPaRKy in ways which will be detailed in Section 5.

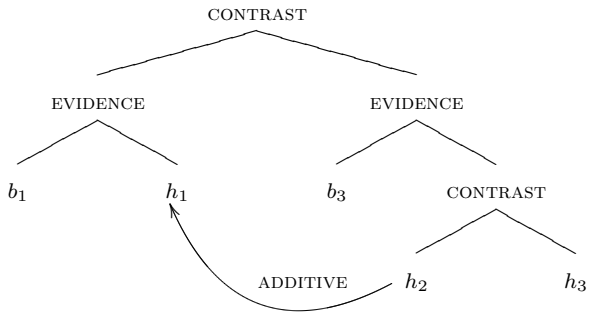

FIGURE 1 The discourse structure of the multi-sentence discourse in (1)

discourse units), and each non-terminal node is considered a complex discourse unit. A non-terminal indicates a structural relation that holds between its child nodes, while arcs from one leaf node back to a previous one indicate an anaphoric relation. Thus, Figure 1 shows the discourse structure for (1), illustrating that the CONTRAST relations signaled by the structural connectives *however*[3] and *while* connect structurally adjacent discourse units, whereas the ADDITIVE relation signaled by the discourse adverbial *also*[4] connects a discourse unit to a non-adjacent antecedent discourse unit by an anaphoric link. Lastly, there are implicit (i.e. not realized with an overt connective) EVIDENCE relations, also between adjacent discourse units.

(1) b_1: Bienvenue is a mediocre restaurant.
h_1: It has poor decor and mediocre food quality.
b_3: However, Sonia Rose is a good restaurant.
h_2: While it also has poor decor,
h_3: it has excellent food quality.

Since D-LTAG is a lexicalized grammar, each discourse connective (of either type) is represented in the lexicon by entries pairing a connective and an elementary tree (either initial or auxiliary). These elemen-

[3]The empirical status of *however* as a structural connective or discourse adverbial remains unclear. We will follow Forbes et al. (2003) in treating *however* as a structural connective — which is the more constrained case — in order to show that it is possible to do so in DCCG.

[4]Although this usage of *also* has not typically been treated on a par with other discourse connectives, as discussed further in Section 7, we focus on it because it plays an important role in making some contrastive statements more natural. For the purposes of showing how discourse adverbials are handled in the grammar, *also* serves equally well as more commonly discussed adverbials such as *then* or *otherwise*.

FIGURE 2 D-LTAG entries for the structural connective *although* in two configurations (Forbes-Riley et al., 2006)

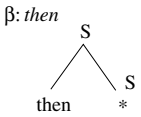

FIGURE 3 LTAG syntactic entry for *then* (Webber et al., 2003)

FIGURE 4 D-LTAG entries for the discourse adverbial *then* (Forbes-Riley et al., 2006)

tary trees are then combined via the LTAG operations of substitution and adjunction, building the discourse into a derived tree that indicates the discourse structure. One benefit of lexicalization is that each discourse connective can specify in its entry whether its discourse unit arguments (D_u) are found structurally or anaphorically. For structural connectives, the lexical entry contains an elementary tree specifying the position of the structural connective relative to its adjacent clauses (i.e. prior to, between or after both clausal discourse units) as in Figure 2. For discourse adverbials, as in Figure 4, the lexical entry contains an auxiliary tree specifying its position relative to the host clause, leaving the anaphoric connection to be resolved in the semantic interpretation.

As mentioned earlier, D-LTAG is part of a dual grammar approach.

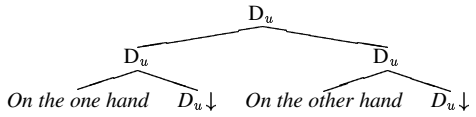

FIGURE 5 D-LTAG initial tree for the parallel contrastive construction (Forbes-Riley et al., 2006)

In this dual grammar approach, discourse connectives are defined by an LTAG with their sentential scope (e.g. *then* in Figure 3 is defined using sentence categories), and then redefined by a D-LTAG for their discourse scope (e.g. *then* in Figure 4 is defined using discourse units). The effect of D-LTAG's redefinition of discourse connectives is magnified in the case of paired discourse connectives. To illustrate, consider the paired connectives *on the one hand* and *on the other hand*. In a sentential LTAG, each of the paired connectives is defined as an idiomatic prepositional phrase that anchors an auxiliary tree. For example, in (2), the paired connectives separately adjoin to (and so have scope over) their respective main clauses (i.e. *on the one hand* adjoins to *Bienvenue has decent decor* and *on the other hand* adjoins to *Sonia Rose has very good decor*). In contrast, in the D-LTAG, both phrases anchor the same initial tree (as in Figure 5), indicating that both main clauses are within the same domain of locality of the CONTRAST relation indicated by the paired anchoring connectives (Webber, 2004).

(2) On the one hand, Bienvenue has decent decor. On the other hand, Sonia Rose has very good decor.

Although D-LTAG successfully expands the semantic scope of discourse connectives beyond their sentential scope, the dual grammar approach adds complexity to processing. For example, since D-LTAG considers clauses as atoms, a clause-medial discourse connective (e.g. adverbs like *however*) must be moved to the initial position of its host clause in order for D-LTAG to process it, and its arguments. Thus, some intermediate processing between the sentential LTAG and D-LTAG stages are required for the two grammars to work together. The complex nature of the intermediate processing (whose exact details are not important for present purposes) is illustrated in the D-LTAG system architecture in Figure 6. Note that with this architecture, only the interpretation direction is considered, and only handling of single-best parses.

While our article focuses on discourse generation, it is worth noting that the existence of ambiguous connectives may also pose a challenge for interpretation under the dual grammar approach. As is well known (see Hirschberg and Litman, 1993, for example), some potential connectives, such as *now*, also have sentential uses, and thus may not always be used in their discourse connective sense. For example, in (3a), *now* is used as a temporal adverb at the sentential level, whereas in (3b), *now* is used as a discourse particle to introduce a different topic at the discourse level. Similarly, in (4a), *still* is used as a temporal adverb meaning continuing up to the relevant time, while in (4b), *still* is

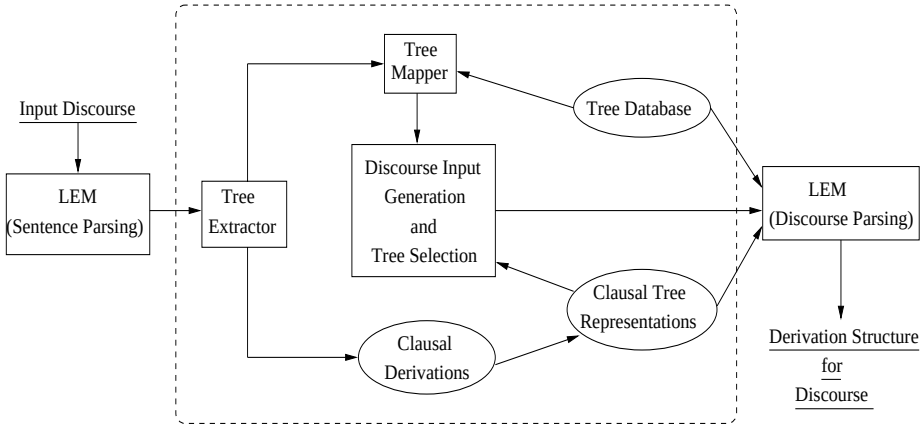

FIGURE 6 D-LTAG parsing system architecture indicating a two pass parsing module, with complex intermediate processing between sentence processing and discourse processing (Forbes et al., 2003)

used a discourse connective whose meaning is much like *nonetheless* or *nevertheless*.[5] Consequently, for interpretation under the dual grammar approach, some mechanism must be devised to determine which is the operative grammar for handling words with such ambiguities. (Language generation does not require such a mechanism, since it begins from an unambiguous semantic representation.)

(3) a. Sonia Rose always has good food. Now we might be able to get a table.

b. No one is suggesting we skip lunch. Now we all love good food, so let's go eat at Sonia Rose.

(4) a. Sonia Rose always has good food. However, it might still be too busy to get a table now.

b. Sonia Rose isn't exactly known for its excellent food quality. However, it might still be our best bet for lunch.

By employing a single grammar, DCCG avoids both of these issues. It eliminates the need for a mapping interface between two grammars and simplifies the treatment of discourse connective ambiguity, since it becomes a "normal" ambiguity that can be resolved using typical single grammar dynamic programming and disambiguation techniques,

[5]We thank one of the reviewers for suggesting a similar example.

rather than one spanning two grammars and thus requiring a special mechanism.

3 Combinatory Categorial Grammar (CCG)

Combinatory Categorial Grammar (CCG; Steedman, 2000, Steedman and Baldridge, 2009) belongs to the family of categorial grammars that take a compositional approach to both syntactic and semantic analysis. It is a unification-based grammar formalism consisting of atomic and functional (complex) categories, and a small set of rules for combining these categories in derivations. As a lexicalized grammar, much of the combinatory information of a language resides in the lexical entries for the words in the language.

CCG also provides a transparent interface to compositional semantics, which is built in parallel with a syntactic derivation. In this section, we will confine our discussion of CCG to syntax, leaving semantics to be discussed in Section 4.

In CCG, an atomic category consists of a single symbol (e.g. s, n, np), which can be the complete category of a lexical entry, such as np for the restaurant name *Bienvenue* in (5a) or n for *decor* in (5b). In addition, atomic categories can be further specified by the use of features, such as person, number, voice, tense, etc. For example, in (5b) *decor* is instantiated with the value mass for the feature num, since it is a mass noun.[6]

(5) a. *Bienvenue* $\vdash \mathsf{np}$
b. *decor* $\vdash \mathsf{n}_{\mathsf{mass}}$
c. *decent* $\vdash \mathsf{n}_{\langle 2\rangle}/_{\diamond}\mathsf{n}_{\langle 2\rangle}$
d. *has* $\vdash (\mathsf{s}\backslash\mathsf{np}_{\mathsf{nom}})/\mathsf{np}_{\mathsf{acc}}$

Functional categories are formed from atomic categories, creating a complex category that specifies its combinatory potential. Functional categories are typically of the form $\alpha/_i\beta$ or $\alpha\backslash_i\beta$, where α and β are categories that may themselves be either atomic or functional, and the subscripts on the slashes represent slash modalities, to be explained below. For example, in (5c), two atomic categories (n) are used to form the functional category $\mathsf{n}/_{\diamond}\mathsf{n}$ as the syntactic entry for *decent*, whereas in (5d), the functional category for *has* (as a main verb) is made up of another functional category $\mathsf{s}\backslash\mathsf{np}_{\mathsf{nom}}$ and the atomic category $\mathsf{np}_{\mathsf{acc}}$, where the leftward NP has nominative case and the rightward NP has accusative case. Note that atomic categories may be co-indexed for

[6] By convention, the feature-value pair on a category is indicated by the value alone, when the feature name is evident from the value.

unification purposes, and thus the fully specified category for *decent*, $\mathsf{n}_{\langle 2\rangle}/_{\diamond}\mathsf{n}_{\langle 2\rangle}$, indicates that the features on the argument and result categories should be unified.

CCG employs a result-first notation, and thus in functional categories, β specifies the argument of the function while α specifies the result. The direction of the slash indicates the direction in which the argument β is sought: the leftward leaning slash ($\backslash$) indicates that the argument is sought to the left of the functor, while the rightward leaning slash ($/$) indicates that the argument is sought to the right of functor. For example, in the category for *has*, $(\mathsf{s}\backslash\mathsf{np})/\mathsf{np}$, α corresponds to $\mathsf{s}\backslash\mathsf{np}$, β corresponds to np and the slash indicates that β is sought to the right of the functor. Left associativity is assumed, and so $(\mathsf{s}\backslash\mathsf{np})/\mathsf{np}$ is equivalent to $\mathsf{s}\backslash\mathsf{np}/\mathsf{np}$.

Once categories are defined for all the words in the lexicon, they can be combined to form larger phrases via the CCG combinators. The most commonly used combinators are forward ($>$) and backward ($<$) functional application, as defined in (6) below. The subscripts located on the slashes in the combinatory rules indicate the use of modalities in CCG (Baldridge, 2002, Baldridge and Kruijff, 2003, Steedman and Baldridge, 2009). These modes are specified in the lexical entry of a word and are used to restrict the application of the combinatory rules, thereby avoiding the overgeneration of incorrect derivations. There exist four modalities in the set $\mathcal{M} = \{\star, \times, \diamond, \cdot\}$, and they restrict the application of the combinatory rules as indicated in (7)-(10) below. Note that the application-only slash modality ($\star$) is only compatible with the application rules in (6), and thus constrains sub-derivations involving application-only slashes to be context-free.

(6) Functional Application

a.	$\mathsf{X}/_i\mathsf{Y}$	Y	$\Rightarrow$	X	(for $i \in \mathcal{M}$) ($>$)
b.	Y	$\mathsf{X}\backslash_i\mathsf{Y}$	$\Rightarrow$	X	(for $i \in \mathcal{M}$) ($<$)

In addition to functional application, CCG has three additional types of combinatory rules, based on the type raising (**T**), composition (**B**), and substitution (**S**) combinators. As with functional application, these rules are applied in a specified direction and most have both a forward and backward version that is employed in analyzing English. Using these combinators, most of the "non-standard" constituents that are hallmarks of CCG (e.g. object-relative clauses) can be built in a manner consistent with the rest of the grammar. Note that with the composition and substitution rules, the modality constraints apply to the modality of the principal functor's slash (i), while the modality of the secondary functor's slash (j) is unified into the result category's slash.

(7) Type Raising

a. $\mathsf{X} \Rightarrow_{\mathbf{T}} \mathsf{Y}/_j(\mathsf{Y}\backslash_j\mathsf{X})$ $(>\mathbf{T})$

b. $\mathsf{X} \Rightarrow_{\mathbf{T}} \mathsf{Y}\backslash_j(\mathsf{Y}/_j\mathsf{X})$ $(<\mathbf{T})$

(8) Harmonic Composition

a. $\mathsf{X}/_i\mathsf{Y}\ \ \mathsf{Y}/_j\mathsf{Z} \Rightarrow_{\mathbf{B}} \mathsf{X}/_j\mathsf{Z}$ (for $i \in \{\diamond, \cdot\}$) $(>\mathbf{B})$

b. $\mathsf{Y}\backslash_j\mathsf{Z}\ \ \mathsf{X}\backslash_i\mathsf{Y} \Rightarrow_{\mathbf{B}} \mathsf{X}\backslash_j\mathsf{Z}$ (for $i \in \{\diamond, \cdot\}$) $(<\mathbf{B})$

(9) Crossed Composition

a. $\mathsf{X}/_i\mathsf{Y}\ \ \mathsf{Y}\backslash_j\mathsf{Z} \Rightarrow_{\mathbf{B}_\times} \mathsf{X}\backslash_j\mathsf{Z}$ (for $i \in \{\times, \cdot\}$) $(>\mathbf{B}_\times)$

b. $\mathsf{Y}/_j\mathsf{Z}\ \ \mathsf{X}\backslash_i\mathsf{Y} \Rightarrow_{\mathbf{B}_\times} \mathsf{X}/_j\mathsf{Z}$ (for $i \in \{\times, \cdot\}$) $(<\mathbf{B}_\times)$

(10) Backward Crossed Substitution

a. $\mathsf{Y}/_j\mathsf{Z}\ \ (\mathsf{X}\backslash_i\mathsf{Y})/_j\mathsf{Z} \Rightarrow_{\mathbf{S}} \mathsf{X}/_j\mathsf{Z}$ (for $i \in \{\times, \cdot\}$) $(<\mathbf{S}_\times)$

The combinatory possibilities of the rules in (6)-(10) are further restricted by the use of features. Two categories undergoing combination are also subject to feature unification. If the argument category (β) of the functional category and the input category (to the functional category) have incompatible values instantiated for a given feature, the functional and input categories are prevented from combining. Otherwise, the features of both categories are unified.

To illustrate feature unification and its effect on category combination, we can combine *decent* (5c) and *decor* (5b) using forward functional application. Since the argument of the functional category for *decent* ($\mathsf{n}_{\langle 2\rangle}/_\diamond\mathsf{n}_{\langle 2\rangle}$) is underspecified for all noun features, it can combine with any noun to the right. When it combines with *decor*, a noun that has a specified *num* (number) value of mass, it unifies the feature values of both n categories (the complex category's argument category and the input category of *decor*). Furthermore, as noted earlier, the subscripted indices on both $\mathsf{n}_{\langle 2\rangle}$ categories in the functor category indicates that the result category inherits all the feature values of the argument category. Thus, after unification, the resulting n category for the phrase *decent decor* inherits the feature value mass for its *num* feature, as in (11).

(11) *decent decor* $\vdash \mathsf{n}_{\mathsf{mass}}$

Lastly, in addition to the core combinatory rules, CCG allows for a small number of unary type-changing rules (Hockenmaier and Steedman, 2002, 2007). These rules can be thought of as zero morphemes, and like morphemes (but unlike type-raising rules), they may add their own semantics. For instance, the type-changing rule in (12) indicates

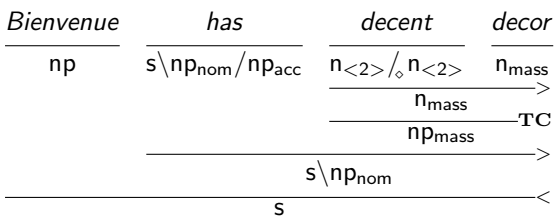

FIGURE 7 CCG derivation of *Bienvenue has decent decor*

that mass nouns can be promoted to the np level without requiring an overt determiner.

(12) $\mathsf{n}_{\mathsf{mass}} \Rightarrow \mathsf{np}_{\mathsf{mass}}$

Given these combinatory rules and the lexical entries in (5), we can derive the rest of the sentence, *Bienvenue has decent decor*. A step-by-step derivation, continuing from *decent decor*, is given in (13), and is shown pictorially in Figure 7.

(13) a. *decent decor* $\vdash \mathsf{np}_{\mathsf{mass}}$
b. *has decent decor* $\vdash \mathsf{s}\backslash\mathsf{np}_{\mathsf{nom}}$
c. *Bienvenue has decent decor* $\vdash \mathsf{s}$

4 Hybrid Logic Dependency Semantics (HLDS)

The OpenCCG[7] library, which provides parsing and realization services for CCG, employs Hybrid Logic Dependency Semantics (HLDS; Kruijff, 2001, Baldridge and Kruijff, 2002) to represent meanings. HLDS is closely related to other computationally-oriented semantic representation frameworks such as Minimal Recursion Semantics (MRS; Copestake et al., 2001, 2005) and is well suited to the needs of natural language generation (White and Baldridge, 2003, White, 2006b). As DCCG is designed to be used computationally with OpenCCG, we also adopt HLDS for DCCG semantic representations.

Hybrid logic (Blackburn, 2000) extends modal logic by enabling direct reference to specific states in the model using *nominals*, a new sort of basic formula that can explicitly name model states. In addition, a nominal can be used with the *satisfaction operator* "@" and a formula to form a new formula. For example, the formula $@_i(p \wedge \langle F\rangle(j \wedge q))$ states that the proposition p, as well as the formula $\langle F\rangle(j \wedge q)$, holds at the state named by the nominal i. Unpacking the sub-formula, it also

[7] http://OpenCCG.sourceforge.net/

$$@_e(\mathbf{have} \wedge \langle \text{MOOD} \rangle \text{decl} \wedge \langle \text{TENSE} \rangle \text{pres} \wedge \langle \text{OWNER} \rangle (b \wedge \mathbf{Bienvenue}) \wedge \langle \text{POSSN} \rangle (d \wedge \mathbf{decor} \wedge \langle \text{DET} \rangle \text{nil} \wedge \langle \text{MOD} \rangle (t \wedge \mathbf{decent})))$$

FIGURE 8 The HLDS logical form for "Bienvenue has decent decor."

states that the state j, where the proposition q holds, is related to the state i via the modal relation F.

To describe linguistic meaning, Baldridge and Kruijff associate each semantic head with a nominal that identifies its discourse referent. Heads are connected to their dependents (which are associated with their own nominal) via dependency relations (characterized as modal relations). For example, the HLDS representation for the sentence *Bienvenue has decent decor* is shown in Figure 8.

In this example, the nominal e identifies the discourse referent for the *having* eventuality, which holds in the present and has a mood of declarative (decl), since it is a statement.[8] It is related to b, the referent for *Bienvenue* (a restaurant name) via the modal relation ⟨OWNER⟩, and to d, the non-definite referent for *decor*, via the modal relation ⟨POSSN⟩ (Possession). Lastly, referent d is in turn related to t, the referent for *decent*, via the modal relation ⟨MOD⟩ (Modifier).

As Blackburn (2000) notes, attribute-value matrices (AVMs), which are more familiar in computational linguistics, are notational variants of hybrid modal logic. Semantically, HLDS representations are descriptions of semantic dependency graphs. White (2006b) shows how these graphs can be translated into the Discourse Representation Structures of Discourse Representation Theory (Kamp and Reyle, 1993), thereby providing the intended natural language meaning. HLDS representations can be given either as a hierarchical expression, as in Figure 8, or in an equivalent flat form, as a conjunction of elementary predications. The flat form enables a monotonic approach to semantic composition as simple conjunction, much as in MRS, which guarantees that semantic inputs can be cleanly decomposed during generation. (For readability, we will only show HLDS representations in hierarchical form.)

During interpretation, logical forms (LFs) are built using unification of syntactic indices and corresponding semantic nominals. Each atomic category (including those in complex categories) is assigned an index variable, via a distinguished index feature. These indices are also associated with corresponding nominals in the semantic representation.

[8]The semantic feature ⟨TENSE⟩pres is meant to stand in for the range of more specific ways that the semantics of the present tense morpheme could be spelled out in order to support inference.

When lexical categories are combined, the application of combinatory rules causes the appropriate nominals to be coindexed, via unification on the categories (White, 2006b).

For example, to build the phrase *decent decor*, we start with the lexical entries for both words, shown in (14). In the definition for *decor*, the index d on the category n is associated with the nominal d in the HLDS representation. For *decent*, both atomic n categories bear an index x, which corresponds to the nominal variable x in the HLDS representation.

(14) a. *Bienvenue* $\vdash \mathsf{np}_b$: $@_b(\mathbf{Bienvenue})$
b. *decor* $\vdash \mathsf{n}_{d,\mathsf{mass}}$: $@_d(\mathbf{decor})$
c. *decent* $\vdash \mathsf{n}_x/_\diamond \mathsf{n}_x$: $@_x(\langle \textsc{Mod} \rangle(t \wedge \mathbf{decent}))$
d. *has* $\vdash \mathsf{s}_e \backslash \mathsf{np}_{x,\mathsf{nom}} / \mathsf{np}_{y,\mathsf{acc}}$:
$@_e(\mathbf{have} \wedge \langle \textsc{tense} \rangle pres \wedge \langle \textsc{Owner} \rangle x \wedge \langle \textsc{Possn} \rangle y)$

When *decent* and *decor* combine, the index d from decor unifies (via feature unification) with the index variable x on the argument category of *decent*. This in turn unifies with the result category's index, so that the resulting category of *decent decor* is n_d. On the semantic side, the nominal variable x for *decent* is coindexed with the newly unified variable d, effectively adding the remaining formulas for *decent* to the semantics for *decor*. The result is (15):

(15) *decent decor* $\vdash \mathsf{n}_{d,\mathsf{mass}}$: $@_d(\mathbf{decor} \wedge \langle \textsc{Mod} \rangle(t \wedge \mathbf{decent}))$

As mentioned in Section 3, mass nouns can be promoted to the np category via a type-changing rule (12), which adds additional semantics. The full rule including the semantics is found in (16):

(16) $\mathsf{n}_{x,\mathsf{mass}} \Rightarrow \mathsf{np}_{x,\mathsf{mass}}$: $@_x(\langle \textsc{det} \rangle nil)$

When (16) is applied to (15), and the appropriate indices are unified and coindexed, the result is (17):

(17) *decent decor* $\vdash \mathsf{np}_{d,\mathsf{mass}}$: $@_d(\mathbf{decor} \wedge \langle \textsc{det} \rangle nil \wedge \langle \textsc{Mod} \rangle(t \wedge \mathbf{decent}))$

Using the remaining lexical entries in (14), and the same unification/coindexation principles as above, the rest of the sentence can be derived. When *has* (14d) is applied to *decent decor* (17), the result is (18). Finally, (18) is applied to *Bienvenue* (14a), resulting in (19). Note that the semantic representation in (19) is essentially the same as the one in Figure 8, missing only the declarative mood semantic feature, which is supplied by the full stop.

(18) *has decent decor* $\vdash \mathsf{s}_e \backslash \mathsf{np}_x$:
$@_e(\mathbf{have} \wedge \langle \textsc{tense} \rangle pres \wedge \langle \textsc{Owner} \rangle x \wedge \langle \textsc{Possn} \rangle (d \wedge \mathbf{decor} \wedge \langle \textsc{det} \rangle \text{nil} \wedge \langle \textsc{Mod} \rangle (t \wedge \mathbf{decent})))$

(19) *Bienvenue has decent decor* $\vdash \mathsf{s}_e$:
$@_e(\mathbf{have} \wedge \langle \textsc{tense} \rangle pres \wedge \langle \textsc{Owner} \rangle (b \wedge \mathbf{Bienvenue}) \wedge \langle \textsc{Possn} \rangle (d \wedge \mathbf{decor} \wedge \langle \textsc{det} \rangle \text{nil} \wedge \langle \textsc{Mod} \rangle (t \wedge \mathbf{decent})))$

5 Discourse Combinatory Categorial Grammar

In this section, we show how Discourse Combinatory Categorial Grammar (DCCG) extends Combinatory Categorial Grammar (CCG) to model both discourse and sentence structure in a single grammar. In Section 5.1, we illustrate how intrasentential structural connectives are treated using standard CCG categories, then discuss how extending this approach across sentences to handle intersentential connectives leads to an undesirable proliferation of categorial ambiguity. In Section 5.2, we introduce *cue threading*, a tightly constrained technique for extending the scope of structural discourse connectives across sentences that works with standard CCG sentential categories and avoids the problem of proliferating categorial ambiguity. In Section 5.3, we show how treating discourse adverbials as anaphoric makes them straightforward to handle in the grammar, irrespective of whether their dependencies form a tree, as well as how this treatment allows multiple connectives to appear in a clause, as long as no more than one connective is structural. We conclude the section with an example illustrating how the approach allows for paraphrases that cross sentence boundaries.

5.1 Standard Categories and Proliferating Ambiguity

To illustrate how standard CCG categories can be used for discourse connectives within sentences, consider the following categories for expressing CONTRAST with *but* and *while*:

(20) a. $\{\textit{while, but}\} \vdash \mathsf{s}_e \backslash_* \mathsf{s}_{e_1} \backslash_* \mathsf{punc}, /_\diamond \mathsf{s}_{e_2}$:
$@_e(\text{contrast-rel} \wedge \langle \textsc{Arg1} \rangle e_1 \wedge \langle \textsc{Arg2} \rangle e_2)$

b. $\textit{while} \vdash \mathsf{s}_e /_* \mathsf{s}_{e_2} /_* \mathsf{punc}, /_\diamond \mathsf{s}_{e_1}$:
$@_e(\text{contrast-rel} \wedge \langle \textsc{Arg1} \rangle e_1 \wedge \langle \textsc{Arg2} \rangle e_2)$

The first category in (20) is for medial *while* or *but*, as in *Bienvenue has decent decor, but Sonia Rose has very good decor.* The category looks for the second argument to the contrast relation (headed by e_2) to the right, then a comma to the left, then the first argument to the contrast relation (headed by e_1) to the left, returning a full clause (headed by e) expressing the contrast relation. The second category in (20) is for an

$$
\begin{array}{l}
@_{c_1}(\mathsf{contrast\text{-}rel} \wedge \langle \textsc{turn} \rangle \mathrm{complete} \wedge \\
\quad \langle \textsc{Arg1} \rangle (h_1 \wedge \mathbf{have} \wedge \underline{\langle \textsc{mood} \rangle \mathrm{decl}} \wedge \\
\qquad \langle \textsc{Owner} \rangle (b_1 \wedge \underline{\mathbf{Bienvenue}}) \wedge \\
\qquad \langle \textsc{Possn} \rangle (d_1 \wedge \mathbf{decor} \wedge \langle \textsc{det} \rangle \mathrm{nil} \wedge \\
\qquad\quad \langle \textsc{Mod} \rangle (e_1 \wedge \mathbf{decent}))) \wedge \\
\quad \langle \textsc{Arg2} \rangle (h_2 \wedge \mathbf{have} \wedge \underline{\langle \textsc{mood} \rangle \mathrm{decl}} \wedge \\
\qquad \langle \textsc{Owner} \rangle (s_1 \wedge \underline{\mathbf{Sonia\_Rose}}) \wedge \\
\qquad \langle \textsc{Possn} \rangle (d_2 \wedge \mathbf{decor} \wedge \langle \textsc{det} \rangle \mathrm{nil} \wedge \\
\qquad\quad \langle \textsc{Mod} \rangle (g_1 \wedge \mathbf{very\_good})))
\end{array}
$$

FIGURE 9 Logical form of *Bienvenue has decent decor. However, Sonia Rose has very good decor.*

initial connective, and is only applicable to *while*, as in *While Bienvenue has decent decor, Sonia Rose has very good decor.*

To account for the full stop, the following category introduces a sentence mood feature:

(21) $. \vdash \mathsf{ts_e} \backslash_* \mathsf{s}_e \ : \ @_e(\langle \textsc{Mood} \rangle \mathrm{decl})$

Typically, the full stop maps a clause to a complete sentence. In (21), we have introduced the category ts, for *text segment*, as the result, where a text segment is one or more complete sentences; having such a category will prove useful when we look at connectives combining multiple sentences.

At this point, we are ready to examine what happens when one tries to employ standard CCG categories for structural connectives that cross sentence boundaries. To illustrate, consider the abbreviated[9] logical form in Figure 9 for (22).

(22) Bienvenue has decent decor. However, Sonia Rose has very good decor.

Since the LF in Figure 9 contains a declarative mood feature on each clause specification, each clause must be realized as a complete sentence (or minimal text segment), and thus the intrasentential categories in (20) do not apply (or at least, will not yield complete realizations). To realize (22), we will instead need a category for *however* that spans sentences, as indicated in Figure 10. The requisite category follows:

[9]LFs in this paper are shown with only those semantic features that pertain to discourse structure, such as ⟨MOOD⟩ and ⟨TURN⟩ (to be discussed in the next subsection). Other semantic features (e.g. ⟨TENSE⟩, ⟨NUMBER⟩, ⟨GENDER⟩, etc.) are present in the LFs and corresponding DCCG grammar used by OpenCCG, but omitted here due to space.

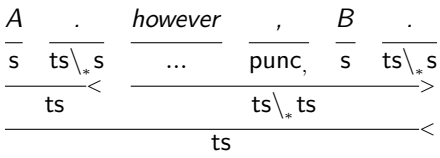

FIGURE 10 Schematic derivation with sentence-initial *however*.

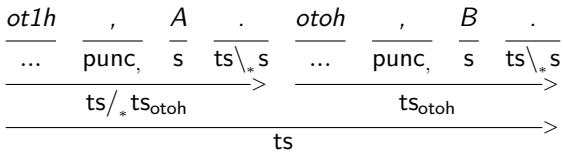

FIGURE 11 Schematic derivation with *on the one hand ... on the other hand*.

(23) $\textit{however} \vdash \mathsf{ts}_e \backslash_* \mathsf{ts}_{e_1} /_* (\mathsf{ts}_{e_2} \backslash \mathsf{s}_{e_2}) /_* \mathsf{s}_{e_2} /_* \mathsf{punc}_{,} \ :$
$@_e(\text{contrast-rel} \wedge \langle \textsc{Arg1} \rangle e_1 \wedge \langle \textsc{Arg2} \rangle e_2)$

In (23), *however* first looks to the right for a comma, then a clause expressing the second argument of the contrast relation, and then the full-stop category; at this point, it looks to the left for a text segment expressing the first argument of the contrast relation, finally returning a text segment expressing the contrast relation itself.

Although this category succeeds in making the desired derivation possible, it has an odd asymmetry, in that the discourse unit for $\langle \textsc{Arg1} \rangle$ can be a text segment, while the discourse unit for $\langle \textsc{Arg2} \rangle$ can only be a clause. We will return to this asymmetry below, after first discussing how paired connectives can be handled.

To express the LF in Figure 9, the paired connectives *on the one hand ... on the other hand* can also be employed, as in (24).

(24) On the one hand, Bienvenue has decent decor. On the other hand, Sonia Rose has very good decor.

The target derivation appears schematically in Figure 11; the requisite categories for the connectives follow in (25). Here, *on the other hand* (abbreviated as *otoh*) is treated as semantically null, similarly to verb particles and case-marking prepositions: it serves to mark the text segment it appears in as one that includes this phrase, using a cue feature (i.e., cue=otoh). Such a category is selected for by *on the one hand* (abbreviated *ot1h*), whose category introduces the CONTRAST relation expressed by the two connectives. Note that *on the other hand* is often found by itself, without being preceded by *on the one hand*; to allow

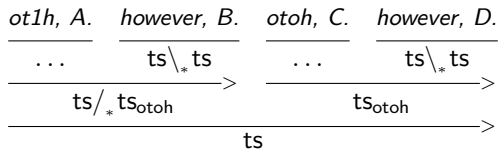

FIGURE 12 Schematic derivation of nested contrasts.

for such solitary uses of *on the other hand*, it must also be assigned the same category as *however* in (23).

(25) a. *on the one hand* $\vdash \mathsf{ts}_e/_*\mathsf{ts}_{e_2,\mathsf{otoh}}/_*(\mathsf{ts}_{e_1}\backslash\mathsf{s}_{e_1})/_*\mathsf{s}_{e_1}/_*\mathsf{punc}_{,}$:
$@_e(\text{contrast-rel} \wedge \langle\text{ARG1}\rangle e_1 \wedge \langle\text{ARG2}\rangle e_2)$

b. *on the other hand* $\vdash \mathsf{ts}_{e,\mathsf{otoh}}/_*(\mathsf{ts}_e\backslash\mathsf{s}_e)/_*\mathsf{s}_e/_*\mathsf{punc}_{,}$

The category for *on the one hand* in (25) is like the one for *however* in (23) insofar as it allows one of the discourse unit arguments of the CONTRAST relation (here, $\langle$ARG2$\rangle$) to be expressed by a potentially extended text segment, while the other (here, $\langle$ARG1$\rangle$) must be expressed by a single clause. With the category given for *on the other hand*, it turns out that the $\langle$ARG2$\rangle$ discourse unit must also be expressed by a single clause. However, these limitations are problematic, as paired connectives are often used to express relations across extended discourses. To illustrate, (26) shows an example of nested contrast relations; schematically, the relations among the clauses are contrast-rel(contrast-rel(A,B), contrast-rel(C,D)).

(26) a. On the one hand, Bienvenue is a mediocre restaurant.
b. However, it has excellent service.
c. On the other hand, Sonia Rose is a good restaurant.
d. However, it has poor decor.

The target derivation for (26) appears in Figure 12. To allow *on the one/other hand* to extend their scope beyond a single clause, they can be given the categories in (27), which have been lifted over a modifying ts\ts category.

(27) a. *ot1h* $\vdash \mathsf{ts}_e/_*\mathsf{ts}_{e_2,\mathsf{otoh}}/_*(\mathsf{ts}_{e_1}\backslash\mathsf{ts}_{e_a})/_*(\mathsf{ts}_{e_a}\backslash\mathsf{s}_{e_a})/_*\mathsf{s}_{e_a}/_*\mathsf{punc}_{,}$:
$@_e(\text{contrast-rel} \wedge \langle\text{ARG1}\rangle e_1 \wedge \langle\text{ARG2}\rangle e_2)$

b. *otoh* $\vdash \mathsf{ts}_{e,\mathsf{otoh}}/_*(\mathsf{ts}_e\backslash\mathsf{ts}_{e_c})/_*(\mathsf{ts}_{e_c}\backslash\mathsf{s}_{e_c})/_*\mathsf{s}_{e_c}/_*\mathsf{punc}_{,}$

While these categories suffice for this example — and suggest that there is no problem in principle with developing an account of structural connectives using standard CCG categories — the need for multiple lexical

categories for these connectives appears problematic; and indeed, when one returns to the categories necessary for *however*, it becomes apparent that there is an undesirable proliferation of lexical category ambiguity. First, we may note that *however* must have a lifted category to likewise extend the scope of its second argument beyond a sentence. More troubling though is the category for medial *however*:

$$(28)\quad \textit{however} \vdash \mathsf{ts}_e \backslash_* \mathsf{ts}_{e_1} /_* (\mathsf{ts}_{e_2} \backslash \mathsf{s}_{e_2}) \backslash \mathsf{np} /_\diamond (\mathsf{s}_{e_2} \backslash \mathsf{np}) \backslash_* \mathsf{punc}_, /_* \mathsf{punc}_, \ : \\ @_e(\text{contrast-rel} \wedge \langle \textsc{Arg1} \rangle e_1 \wedge \langle \textsc{Arg2} \rangle e_2)$$

This category is like the standard one in that it first seeks the surrounding commas, then a verb phrase ($\mathsf{s}\backslash\mathsf{np}$) to the right; but, rather than yielding a verb phrase at this point, the category looks for the subject NP to the left and then yields a category in which s has been replaced by $\mathsf{ts}\backslash_* \mathsf{ts}/_* (\mathsf{ts}\backslash\mathsf{s})$. Accordingly, additional categories will need to be assigned to many other words — for example, words heading clause-initial modifiers, lexically type-raised determiners, auxiliary verbs or negation markers,[10] etc. — in order to interact properly with this result category. For this reason, we will now develop an alternative approach, using *cue threading*, that makes it unnecessary to proliferate category assignments in this way.

5.2 Structural Connectives and Cue Threading

Cue threading is reminiscent of Power et al.'s (1999) *cue storage* technique, shown in Figure 13. With cue storage, discourse connectives signaling rhetorical relations in the upper part of an RST tree percolate down to leaf nodes, which contain propositions expressed as clauses. As a cue store is a stack, multiple connectives can be realized in a single clause, as the example illustrates. Cue threading is similar to cue storage insofar as connectives can be thought of as percolating from where they take scope semantically down to the clause in which they appear during realization (in derivations, connectives actually percolate in the upwards direction, as would be expected in the parsing case). However, cue threading is more constrained than cue storage since it only makes use of a single cue feature, rather than a stack. To handle Power et al.'s example with cue threading, *since* would first connect the basic clauses, then *but* would connect the first sentence with the complex

[10] In sentences where *however* appears after auxiliary verbs or negation markers but still before the verb stem, the CCG category in (28) for medial *however* is sufficient to enable *however* to appear in these syntactic positions, since this category allows *however* to combine with categories headed by verbs in finite or bare form. The main issue presented by words such as auxiliary verbs or negation is that of proliferating categories for these words.

Elixir has no significant side effects. <u>But since</u> the medicine is for you, never give Elixir to other patients.

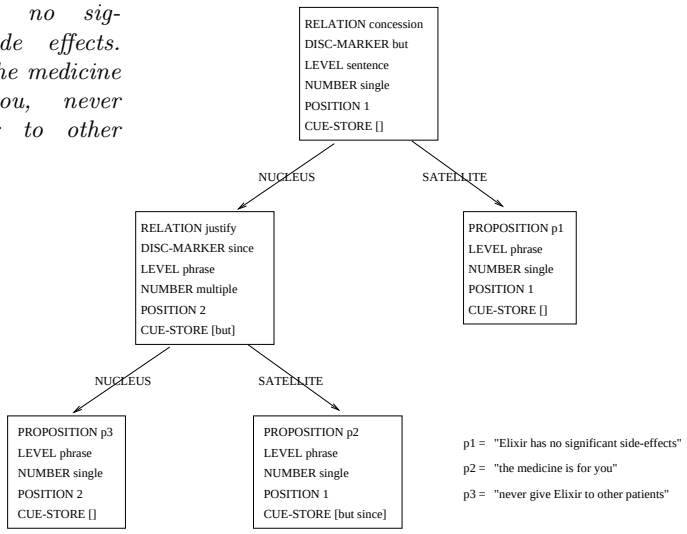

FIGURE 13 A *cue store* allows multiple connectives (underlined) to percolate down to a single clause (Power et al., 1999)

clause *since the medicine is for you, never give Elixir to other patients*, so that only one structural connective is active at the same time. Later in this section, we will see how structural connectives may be combined with discourse adverbials in a single clause; otherwise, structural connectives themselves are constrained to one per clause, which appears to be largely empirically adequate, based on examples in the literature (cf. Webber et al., 2003, Webber, 2004).

However, it may be the case that a given discourse argument is not unique to one particular relation, but rather may be shared between two relations in a multiparent structure (Webber et al., 2003), as in Dinesh et al.'s (2005) example (29), from the Penn Discourse Treebank. Here the discourse unit ArgZ is not considered part of the discourse argument ArgY for *when*, even though it is related to ArgY via the connective phrase *partly because*. This creates a non-hierarchical "M"-shaped discourse structure, as seen in Figure 14, where the same discourse unit (ArgY) is used as an argument for two different discourse relations (SYNCHRONY and JUSTIFICATION), neither of which are subordinate to the other. Under this analysis, we would have to modify our strategy for dealing with intrasentential structural connectives in this arrangement. We leave this issue for future research.

(29) <u>When</u> [**Ms. Evens took her job**]$_{ArgX}$, [*several important divi-*

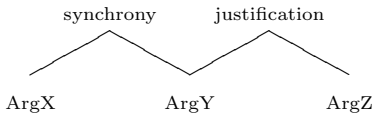

FIGURE 14 The multiparent "M"-shaped discourse structure of (29).

sions that had reported to her predecessor weren't included]$_{ArgY}$ partly because [she didn't wish to be a full administrator]$_{ArgZ}$. (Dinesh et al., 2005)

In addition, our DCCG currently focuses on non-attributed text. Thus any potential problems regarding attributions that lie outside of, but in between, the structural connective and its arguments, as in (30), will also be left for future research.[11]

(30) *The current distribution arrangement ends in March 1990*, although Delmed said **it will continue to provide some supplies of the peritoneal dialysis products to National Medical**, the spokeswoman said. (Dinesh et al., 2005)

We turn now to the explication of the cue threading process. To illustrate cue threading, consider again the logical form in Figure 9. The cue threading derivation using paired contrastive connectives for this LF uses standard CCG categories for the words, with the addition of values for the cue feature. Generalizing the treatment of *on the other hand* in the preceding section, the cue feature is used to mark a clause as containing the structural connective in question. The cue feature is then threaded through the derivation until the point at which the semantic relation for the connective is introduced. For example, the lexical categories for *on the one hand* and *on the other hand* are listed in (31a) and (31b). Both entries are nearly identical, with neither one directly contributing semantic content. Instead, each entry sets a unique value of the cue feature: in (31a), cue=ot1h, while in (31b), cue=otoh. On any category, if the cue is instantiated such that cue=*val* (where *val* is any non-nil value), it indicates an undischarged discourse relation (i.e. one searching for its other argument). A value of cue=nil, however, indicates there are no discourse relations waiting to be discharged.

(31) a. *on the one hand* $\vdash \ \mathsf{s}_{\mathsf{ot1h}}/_{\diamond}\mathsf{s}_{\mathsf{nil}}/_{*}\mathsf{punc},$

b. *on the other hand* $\vdash \ \mathsf{s}_{\mathsf{otoh}}/_{\diamond}\mathsf{s}_{\mathsf{nil}}/_{*}\mathsf{punc},$

[11] We thank one of our reviewers for bringing both (29) and (30) to our attention.

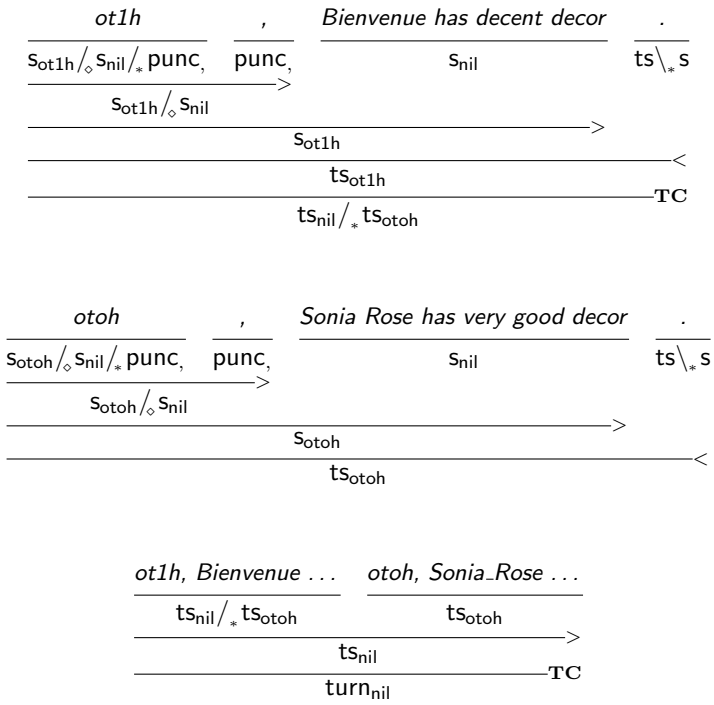

FIGURE 15 A DCCG derivation illustrating the use of cue threading and type-changing rules

The lexical entry for the full stop in (32) has been updated to pass along the value of the cue feature, unifying the value of this feature in the argument and result categories:

(32) $. \vdash \mathsf{ts}_{\mathsf{e,CUE}} \backslash_* \mathsf{s}_{e,\mathsf{CUE}} \;:\; @_e(\langle \textsc{Mood} \rangle \mathrm{decl})$

The derivation involving these paired connectives is shown in Figure 15. Since the main clause in both sentences consists of standard CCG categories, it results in a category of $\mathsf{s}_{\mathsf{nil}}$. The nil value, projected from the verb, indicates that there is no structural connective waiting to be discharged. Using the newly defined discourse connective categories, *on the one hand* combines first with the neighboring comma, then with the $\mathsf{s}_{\mathsf{nil}}$ category of *Bienvenue has decent decor*. Since the complex category for *on the one hand* merely changes the cue value of the input category (from cue=nil to cue=ot1h), it effectively marks the resulting $\mathsf{s}_{\mathsf{ot1h}}$ category as containing an undischarged cue. The sentence then combines with the full stop, which threads the ot1h value of the s category to the resulting $\mathsf{ts}_{\mathsf{ot1h}}$ category. A similar process occurs using the categories

and cue features of *on the other hand* and *Sonia Rose has very good decor*, resulting in the category, $\mathsf{ts}_{\mathsf{otoh}}$.

Next, the $\mathsf{ts}_{\mathsf{ot1h}}$ from the first sentence is type-changed according to the rule in (33), so that a text segment bearing a cue value of ot1h looks to the right for a text segment bearing a cue value of otoh. Furthermore, this rule adds the semantics that there is a CONTRAST relation between the arguments denoted by $\mathsf{ts}_{\mathsf{ot1h}}$ and $\mathsf{ts}_{\mathsf{otoh}}$.

(33) $\mathsf{ts}_{e_1,\mathsf{ot1h}} \Rightarrow \mathsf{ts}_{e,\mathsf{nil}} /_* \mathsf{ts}_{e_2,\mathsf{otoh}}$:
$@_e(\text{contrast-rel} \wedge \langle \text{ARG1} \rangle e_1 \wedge \langle \text{ARG2} \rangle e_2)$

When the newly type-changed category $\mathsf{ts}_{\mathsf{nil}} /_* \mathsf{ts}_{\mathsf{otoh}}$ combines with a $\mathsf{ts}_{\mathsf{otoh}}$, it produces the category $\mathsf{ts}_{\mathsf{nil}}$. The nil value indicates that the connective has been discharged (by finding its other discourse argument), and ends the threading process begun when the structural connectives were first combined in the derivation. It also prevents ts from being type-changed a second time (as the input categories' cue features no longer match).

Note that we have chosen to use a rightward looking type change rule that operates on the ts containing *on the one hand*, instead of a leftward looking type change rule that operates on the ts containing *on the other hand*. The rightward rule mirrors the forward expectation, unique to paired connectives, that if the first half of pair is encountered, marking both the discourse relation and its first argument, then the text segment cannot be considered complete until the second half of the pair, along with the second argument of the discourse relation, is also found.

Lastly, since no further connectives require discharging, the derivation is completed by type changing the result category $\mathsf{ts}_{\mathsf{nil}}$ to the top-level turn category, using the rule in (34). This rule adds a turn completion feature to the semantics that indicates the derivation of one or more complete sentences in a coherent discourse, and enforces a state where no discourse relations are waiting to be discharged.[12]

(34) $\mathsf{ts}_{e,\mathsf{nil}} \Rightarrow \mathsf{turn}_{e,\mathsf{nil}}$: $@_e(\langle \text{TURN} \rangle \text{complete})$

One advantage of using a type changing rule such as (33) to seek out the other half of a paired connective is that we could easily extend (33) the type changing rule to look for ts categories with cue values other than simply otoh. This would allow *on the one hand* to pair with other

[12]Here we are simply using *turn* as a convenient way to ensure that no discourse relations remain undischarged; we leave open the question of how our use of this term relates to other notions of *turn* in the discourse literature.

contrastive connectives such as *at the same time* or *but*, both attested as pairing with *on the one hand* in the Brown Corpus.[13]

Further entries for *on the one/other hand* appear in (35), corresponding to their other possible syntactic positions (i.e. post subject-NP, or clause-finally), as well as the nearly identical entries in (36) for *however*.

(35) *on the other hand* $\vdash$ $\mathsf{s}_{\mathsf{otoh}} \backslash_{\diamond} \mathsf{s}_{\mathsf{nil}} \backslash_{*} \mathsf{punc}$,
on the other hand $\vdash$ $\mathsf{s}_{\mathsf{otoh}} \backslash \mathsf{np} /_{\diamond} (\mathsf{s}_{\mathsf{nil}} \backslash \mathsf{np}) /_{*} \mathsf{punc}, \backslash_{*} \mathsf{punc}$,

(36) *however* $\vdash$ $\mathsf{s}_{\mathsf{however}} /_{\diamond} \mathsf{s}_{\mathsf{nil}} /_{*} \mathsf{punc}$,
however $\vdash$ $\mathsf{s}_{\mathsf{however}} \backslash_{\diamond} \mathsf{s}_{\mathsf{nil}} \backslash_{*} \mathsf{punc}$,
however $\vdash$ $\mathsf{s}_{\mathsf{however}} \backslash \mathsf{np} /_{\diamond} (\mathsf{s}_{\mathsf{nil}} \backslash \mathsf{np}) /_{*} \mathsf{punc}, \backslash_{*} \mathsf{punc}$,

DCCG also contains the type-changing rule in (37) for the more typical solitary structural contrastive connectives. This rule promotes a ts category whose cue=contrastive (where {otoh, however} are subtypes of contrastive and so match) to a complex category searching for a ts (with any value of cue) to its left.

(37) $\mathsf{ts}_{e_2,\mathsf{contrastive}} \Rightarrow \mathsf{ts}_{e,\mathsf{CUE}} \backslash_{*} \mathsf{ts}_{e_1,\mathsf{CUE}}$:
$@_e(\text{contrast-rel} \wedge \langle \textsc{Arg1} \rangle e_1 \wedge \langle \textsc{Arg2} \rangle e_2)$
where {otoh, however} are subtypes of contrastive

Rule (37) propagates the cue value from the leftward argument (headed by e_1) to the parent text segment (headed by e), thereby allowing contrastive or other rhetorical relations to be nested, as shown in Figure 16. In particular, note that *ot1h, A. however, B.* yields a text segment with an undischarged ot1h cue, as shown in Figure 16, given the way rule (37) threads the cue value from the first sentence upwards. Interestingly, cue threading also appropriately restricts these nested relations from being realized by nested paired connectives, as that would result in some clauses bearing multiple structural connectives, as in (38):

(38) *ot1h, ot1h, A. otoh, B. otoh, ot1h, C. otoh, D.

When each clause combines with the structural connective directly preceding them, they result in s categories that bear non-nil cue values. These values then prohibit their host category from combining with additional structural connectives (as on A & C), preventing the generation or interpretation of this schema.

[13] We thank one of our anonymous reviewers for pointing out the existence of non-canonical paired connectives.

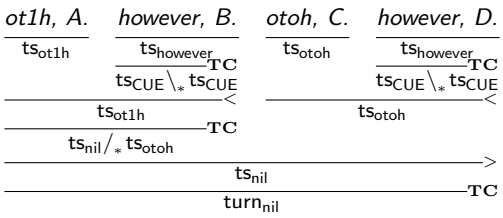

FIGURE 16 A DCCG derivation of nested contrast relations

Returning now to the intrasentential conjunctions that express CONTRAST, their categories remain the same as in the preceding section, except for the addition of the requirement that they combine with clauses having nil values for the cue feature:

(39) a. $\{\textit{while, but}\} \vdash \mathsf{s}_{e,\mathsf{nil}} \backslash_* \mathsf{s}_{e_1,\mathsf{nil}} \backslash_* \mathsf{punc}_{,} /_\diamond \mathsf{s}_{e_2,\mathsf{nil}}$:
$@_e(\text{contrast-rel} \wedge \langle \textsc{Arg}1 \rangle e_1 \wedge \langle \textsc{Arg}2 \rangle e_2)$

b. $\textit{while} \vdash \mathsf{s}_{e,\mathsf{nil}} /_* \mathsf{s}_{e_2,\mathsf{nil}} /_* \mathsf{punc}_{,} /_\diamond \mathsf{s}_{e_1,\mathsf{nil}}$:
$@_e(\text{contrast-rel} \wedge \langle \textsc{Arg}1 \rangle e_1 \wedge \langle \textsc{Arg}2 \rangle e_2)$

Since these categories do not need to look outside the sentence to find both of their discourse arguments, they do not change the cue values of their result categories.

To conclude this section, we address the question of whether it is a necessary move to employ unary type-changing rules in order to handle intersentential discourse connectives in CCG. As noted in the preceding section, the lexicalized categories for connectives offered therein suggest that there is no problem in principle with devising a purely lexicalized approach to discourse connectives; accordingly, the cue threading approach presented in this section appears to yield grammars with coverage equivalent to purely lexicalized alternatives. Nevertheless, as we have seen, the purely lexicalized approach leads to a proliferation of lexical category ambiguity, and while lexical rules might be employed to systematically assign the necessary lexical categories, the cue threading approach is clearly more economical. Similar considerations led Hockenmaier and Steedman (2002, 2007) to make extensive use of type-changing rules in their broad coverage grammar of English, indicating that such rules have an important role to play in practical grammars. Hockenmaier and Steedman further argued that the formal power of the system is unaffected as long as (i) only a finite number of unary rules are employed and (ii) the rules are designed so that they cannot recursively apply to their own output, as is the case here.

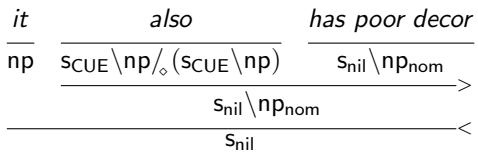

FIGURE 17 A DCCG derivation of a clause including the discourse adverbial *also*.

5.3 Discourse Adverbials and Anaphora Resolution

Unlike structural connectives, which find their discourse arguments via cue threading, discourse adverbials find one argument syntactically, and the other through anaphora resolution. To illustrate how DCCG accomplishes this, consider (1) from Section 2, repeated below:

(1) b_1: Bienvenue is a mediocre restaurant.
h_1: It has poor decor and mediocre food quality.
b_3: However, Sonia Rose is a good restaurant.
h_2: While it also has poor decor,
h_3: it has excellent food quality.

As illustrated by the derivation of the clause for h_2 in Figure 17, the preverbal modifier category for *also* in (40c) below takes a VP category $s_{e,CUE}\backslash np$ as its argument and returns a VP category as its result, adding an ADDITIVE relation to the semantics.

(40) a. $\textit{also} \vdash s_{e,CUE}/_{\diamond}s_{e,CUE}/_{*}punc_{,}$:
$@_e(\langle \text{MOD} \rangle(a \wedge \text{additive-rel} \wedge \langle \text{ARG1} \rangle e_1))$

b. $\textit{also} \vdash s_{e,CUE}\backslash_{\diamond}s_{e,CUE}\backslash_{*}punc_{,}$:
$@_e(\langle \text{MOD} \rangle(a \wedge \text{additive-rel} \wedge \langle \text{ARG1} \rangle e_1))$

c. $\textit{also} \vdash s_{e,CUE}\backslash np/_{\diamond}(s_{e,CUE}\backslash np)$:
$@_e(\langle \text{MOD} \rangle(a \wedge \text{additive-rel} \wedge \langle \text{ARG1} \rangle e_1))$

Since discourse adverbials such as *also* do not necessarily find their discourse arguments in structurally adjacent text segments, they do not use cue threading. Instead, the cue value on discourse adverbials is left underspecfied, as seen in all the lexical entries for *also* in (40). These underspecified values then unify with the cue value of the input category, threading any undischarged structural connectives through. In this way, a discourse adverbial and a structural connective can appear on the same clause (e.g. *However, Bienvenue also has good decor*). In our example, the underspecified cue value of the argument category in (40c) is unified with the *nil* cue value from the input category

($\mathsf{s}_{\mathsf{nil}}\backslash\mathsf{np}_{\mathsf{nom}}$) and then threaded through, resulting in the same category ($\mathsf{s}_{\mathsf{nil}}\backslash\mathsf{np}_{\mathsf{nom}}$). This then allows the subject pronoun *it* to combine with the newly formed VP category to form an $\mathsf{s}_{\mathsf{nil}}$ category.

In each of the semantic representations in (40), *also* establishes an ADDITIVE discourse relation between the head of the modified host clause e and the reference nominal e_1, which is not syntactically bound, and thus must be identified through anaphora resolution. Note that here, the ⟨MOD⟩ relation serves as the inverse of the ⟨ARG2⟩ relation observed with structural connectives. In generation, the antecedent is assumed to be already known.

For our example, the entire semantic representation of (1) is presented in Figure 18. Focusing on lines 19–21 in Figure 18, we can see that the ADDITIVE discourse relation (a_2) on line 20 links the head of the host clause h_2 on line 19 to its antecedent discourse argument h_1 on line 21, which was specified here for generation. An interesting aspect to note is that the unlike the arguments of a structurally established relation (such as CONTRAST on lines 0 or 18) that can be presented in a tree structure, the antecedent argument of an anaphorically established relation (such as ADDITIVE) lies outside of the ADDITIVE subtree and instead resides in a different part of the DLF tree structure. So although h_1 is the antecedent argument of the ADDITIVE relation, only its reference nominal is specified in the ADDITIVE relation subtree, while the rest of its semantics are specified in lines 6–12.

The LF in Figure 18 also includes the EVIDENCE relation (lines 1 & 13). The DCCG realizes this relation simply by placing those text segments in adjacent positions, without the presence of any discourse connective. This is accomplished by the rule in (41). A similar rule can also express the INFER relation, which is used to relate clauses that have no other obvious relation to one other.

(41) $\mathsf{ts}_{\mathsf{nil}} \Rightarrow \mathsf{ts}_{\mathsf{CUE}}\backslash_{*}\mathsf{ts}_{\mathsf{CUE}}$:
$@_e(\text{evidence-rel} \wedge \langle\text{ARG1}\rangle e_1 \wedge \langle\text{ARG2}\rangle e_2)$

At this point we may observe that the ⟨MOOD⟩ feature has been made optional in the LF in Figure 18 (lines 18, 19, 25), using the optionality operator *(?)* from (White, 2006a). Since the ⟨MOOD⟩ feature is introduced by sentence-final punctuation, during realization it has the effect of indicating where sentence boundaries must appear. Making the feature optional therefore allows the realizer some flexibility in choosing where to put sentence breaks. With the DCCG additions shown above, and once the optionality operators are introduced, the OpenCCG realizer generates 660 paraphrases from the LF in Figure 18.

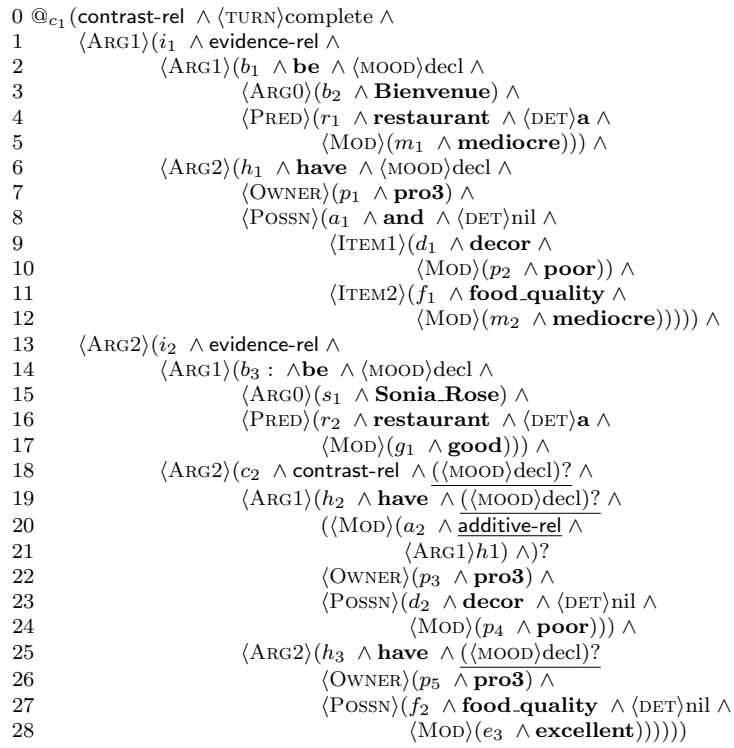

FIGURE 18 The LF used to generate the paraphrases in Figure19, using the optionality marker (*?*) on the ⟨MOOD⟩decl feature. It also features an optional ADDITIVE relation (lines 19–21).

[1.000] Bienvenue is a mediocre restaurant . it has poor decor and mediocre food_quality . however , SoniaRose is a good restaurant . while it also has poor decor , it has excellent food_quality .
[1.000] ot1h , Bienvenue is a mediocre restaurant . it has poor decor and mediocre food_quality . otoh , Sonia_Rose is a good restaurant . it also has poor decor . however , it has excellent food_quality .
[1.000] Bienvenue is a mediocre restaurant . it has poor decor and mediocre food_quality . otoh , Sonia_Rose is a good restaurant . while it also has poor decor , it has excellent food_quality .
[1.000] ot1h , Bienvenue is a mediocre restaurant . it has poor decor and mediocre food_quality . otoh , Sonia_Rose is a good restaurant . while it also has poor decor , it has excellent food_quality .
[1.000] Bienvenue is a mediocre restaurant . it has poor decor and mediocre food_quality . otoh , Sonia_Rose is a good restaurant . it also has poor decor . however , it has excellent food_quality .
. . .
[0.908] ot1h , Bienvenue is a mediocre restaurant . it has poor decor and mediocre food_quality . otoh , Sonia_Rose is a good restaurant . it also has poor decor , but it has excellent food_quality .
. . .
[0.822] Bienvenue is a mediocre restaurant . it has poor decor and mediocre food_quality . however , Sonia_Rose is a good restaurant . ot1h , it also has poor decor . otoh , it has excellent food_quality .
. . .
[0.464] Bienvenue , ot1h , is a mediocre restaurant . it has poor decor and mediocre food_quality . Sonia_Rose is a good restaurant , otoh . it , ot1h , has poor decor , also . it , otoh , has excellent food_quality .
[0.464] Bienvenue is a mediocre restaurant , ot1h . it has poor decor and mediocre food_quality . Sonia_Rose , otoh , is a good restaurant . it , ot1h , has poor decor , also . it has excellent food_quality , otoh .
[0.458] Bienvenue , ot1h , is a mediocre restaurant . it has poor decor and mediocre food_quality . Sonia_Rose is a good restaurant , otoh . it , ot1h , has poor decor , also . it has excellent food_quality , otoh .
[0.458] Bienvenue is a mediocre restaurant , ot1h . it has poor decor and mediocre food_quality . Sonia_Rose is a good restaurant , otoh . it , ot1h , has poor decor , also . it , otoh , has excellent food_quality .
[0.451] Bienvenue is a mediocre restaurant , ot1h . it has poor decor and mediocre food_quality . Sonia_Rose is a good restaurant , otoh . it , ot1h , has poor decor , also . it has excellent food_quality , otoh .

FIGURE 19 DCCG Realizations of Figure 18: The Top 5 Best and Worst paraphrases (first and last 5 paraphrases, respectively), according to n-gram similarity to the target paraphrases (the first two listed)

Figure 19 contains a sample of the generated paraphrases. For illustration purposes, the paraphrases are sorted by a weighted 4-gram precision score,[14] shown in square brackets, which approximates the well-known BLEU score. Here the first two realizations have been used as reference sentences, and thus they receive perfect scores. Overall, the first 5 paraphrases are the top ranked paraphrases, while the last 5 paraphrases are the worst ranked; the middle paraphrases are included to further show the types of paraphrases that can be produced just by allowing contrastive connectives to be realized in several ways and treating sentence boundaries flexibly. In the next section, we will show how many more possibilities may be realized by incorporating the additional text structures and lexicalization options in the SPaRKy corpus.

6 From SPaRKy Content Plans to DLFs

In this section, we illustrate how we derive DCCG disjunctive logical forms (DLFs) from the same content and text plans used by Walker et al.'s (2007) SPaRKy sentence planner. We also evaluate the coverage of our DLFs, and discuss modifications to the existing SPaRKy text plans which enable us to produce a greater variety of natural sounding comparisons.

Figure 20 depicts the relationship between the SPaRKy inputs and outputs, our derived DLFs and the resulting realizations. While the details will be discussed in the remainder of the section, as an overview, this figure illustrates our intent to reuse the SPaRKy content plans and text plans to produce our initial DLFs. These DLFs can then each be used directly by OpenCCG to produce several turn-level realizations, comparable to those found in the SPaRKy Restaurant Corpus. As the figure indicates, the text plan mapper plays much the same role as the SPaRKy sentence planner, and indeed, both components play a role in handling lexicalization and aggregation and generating referring expressions. However, unlike the SPaRKy sentence planner, the text plan mapper does not make specific choices; instead, it simply identifies the range of possible choices, representing them compactly in a single DLF. Note also that the DLFs are more abstract and semantically oriented than SPaRKy's sp-trees and d-trees, where syntactic dependency relations are employed and sentence boundaries are fixed.

Given that our approach does not involve sentence plans per se, we also do not rank them as does SPaRKy. Instead, we plan to rank the

[14]Note that while using n-gram precision scores against target realizations is well suited to grammar testing, such scores normally cannot be used in applications, where target sentences are usually not already available.

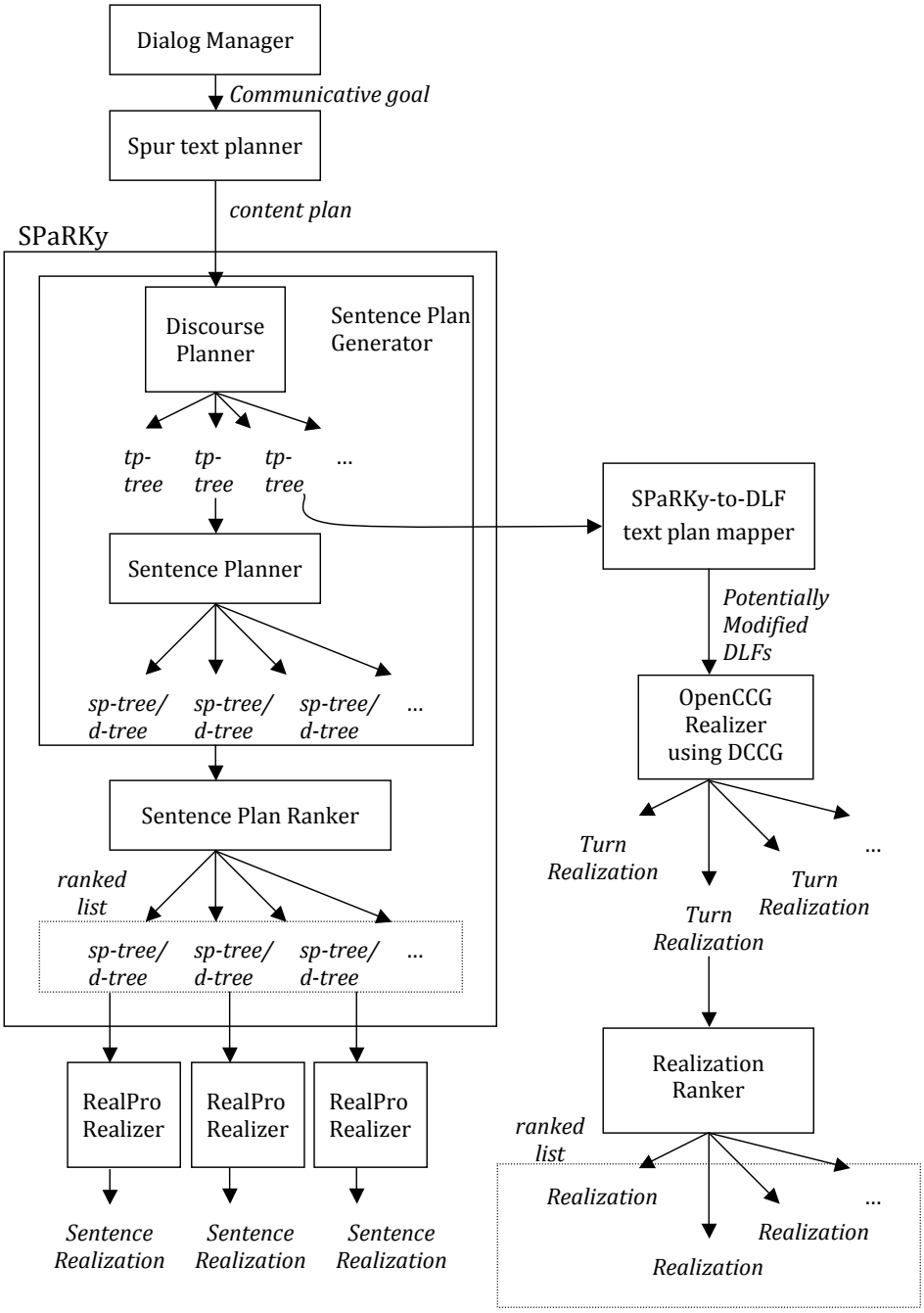

FIGURE 20 The relationship between the SPaRKy's text plans and DLFs. On the left side is the SPaRKy Sentence Planner, depicted as part of the MATCH Spoken Language Generator architecture. On the right side is our alternative approach using DLFs mapped from the SPaRKy text plan and OpenCCG as the generator.

final realizations produced by OpenCCG, using our DCCG grammar.[15] In a previous study on the SPaRKy Restaurant Corpus (Nakatsu, 2008), we found that developing a good ranker for the paraphrases in the corpus is a non-trivial task, especially a ranker that does well with contrastive connective choices. In a comparison, we found that a simple ranker trained only on n-gram features yielded similar rankings as SPaRKy's (Walker et al., 2007) more sophisticated ranker trained additionally on concept and tree features. However, we also found that most of the examples involving contrastive connectives in the SPaRKy Restaurant Corpus received low ratings by the human judges. Upon further inspection, the low ratings were not necessarily attributed directly to the use of a contrastive connective but to some other aspect of the sentence. Nevertheless, the large proportion of low rated examples containing contrastive connectives caused the n-gram ranker to negatively weight any use of a contrastive connective. Thus, we hypothesize that if our DCCG can be used to produce additional realizations that (with additional judgments) receive higher ratings than those already in the SPaRKy Restaurant Corpus, rankers may not learn that contrastive connectives are a priori indicators of dispreferred realizations, and thus better learn when they may be used naturally. For this reason, in this paper we focus on how the realizations found in the SPaRKy Restaurant Corpus can be generated with a DCCG, as well as additional realizations that go beyond those in this corpus, and leave the development of better rankers for this domain for future work.

6.1 Sparky Restaurant Corpus

The SPaRKy Restaurant Corpus was generated by the MATCH Spoken Language Generator (Walker et al., 2007), shown on the left side of Figure 20. It consists of a dialog manager, the SPUR text planner (Walker et al., 2004), the SPaRKy sentence planner (Walker et al., 2007), and the RealPro surface realizer (Lavoie and Rambow, 1997).

The corpus contains realizations for 3 dialogue strategies:

- Recommend (REC): recommend an entity from a set of entities
- Compare-2 (C2): compare 2 entities
- Compare-3(C3): compare 3 or more entities

Each strategy contains several different content plans, each consisting of a set of assertions and the rhetorical (discourse) relations that exist between two assertions, as in Figure 21. Following Rhetorical Structure Theory (RST; Mann and Thompson, 1988), each rhetorical rela-

[15] In practice, it makes sense to use an integrated ranker, at least for generating an initial n-best list, if not for the final ranking.

tion indicates which of the two related assertions is the nucleus (i.e., the main claim that is "more essential to the writer's purpose") and which is the satellite (i.e., the supplementary information that is less important and thus substitutable/deletable without altering the main message of the text). SPaRKy employs three of the RST relations depending on the presentation strategy employed: CONTRAST, ELABORATION and JUSTIFY. Content plans from the RECOMMEND strategy exclusively employ the RST relation JUSTIFY while those from COMPARE-2 use CONTRAST and ELABORATION. COMPARE-3 content plans consist mainly of CONTRAST and ELABORATION relations, though 5 of the COMPARE-3 content plans exclusively use the JUSTIFY relation instead. In addition, SPaRKy specifies the INFER relation between assertions whose relations were not specified by the content planner. It does so after it has taken the content plan as input and created text plans in the discourse planner, where the content plan assertions are ordered according to like topics (i.e., by restaurant or by attribute). This results in text plan trees (tp-trees) such as in Figure 22. These tp-trees are then input into the sentence planner which creates sp-tree and dependency tree (d-tree) pairs. Together, each pair supplies information as to how the basic assertions are aggregated (i.e. grouped into sentences via clause combining operations), and indicates the specific lexical items to be used and their syntactic relations. These sp-tree and d-tree pairs are then ranked by the sentence plan ranker, and input to the RealPro realizer, which realizes one output for each sp-tree/d-tree pair.

The 8 basic assertions used in the SPaRKy content plans can be grouped into 6 types (as shown in Figure 23): offer, highest-ranked, price, located, cuisine, and have. As discussed further in Section 6.2, SPaRKy text plans can be characterized as coming in two types: *back-and-forth* and *serial*. Depending on the assertion type and text plan type, assertions can be realized as a number of different surface forms, ranging from full sentences to sentence fragments. While the back-and-forth text plan type only realizes assertions as full clauses, in the serial text plan type, a have assertion, for example, can be conjoined as a main clause with either a proper noun or pronoun as subject (e.g. *Monsoon/It has decent decor*); subordinated as a preposition phrase, headed by *with* (e.g. *with mediocre decor*) and attached clause finally or NP finally (in relative clauses); or predicated as a VP (e.g. *has decent decor*) joined with the subject form of a CUISINE predicate. The possible surface forms for each type of assertion are listed in Figure 24.[16]

[16]Note that unlike the last four assertion types, offer is only realized as a main clause with a proper name subject, because it only appears discourse-initially in the SPaRKy text plans, and is never combined with other assertions at the surface

strategy:	compare2
items:	Bienvenue, Sonia_Rose
relations:	contrast(nucleus:1,nucleus:2)
content:	1. assert(has-att(Bienvenue,decor(decent)))
	2. assert(has-att(Sonia_Rose,decor(very_good)))

(a) The single relation content plan corresponding to the LF in Figure 9

strategy:	compare2
items:	Hallo_Berlin, Meskerem
relations:	contrast(nucleus:1,nucleus:2)
	contrast(nucleus:3,nucleus:4)
content:	1. assert(has-att(Hallo_Berlin,service(mediocre)))
	2. assert(has-att(Meskerem,service(decent)))
	3. assert(has-att(Hallo_Berlin,cuisine(German)))
	4. assert(has-att(Meskerem,cuisine(African)))

(b) A multiple relation content plan

FIGURE 21 Examples of COMPARE-2 content plans used by SPaRKy to generate contrastive descriptions in the SPaRKy Restaurant Corpus

In summary, each of the 3 strategies contain 30 content plans, for a total of 90 content plans. From each content plan, 2-19 intermediate stage text plans are created. Then, from those text plans, 1-10 sentence plans and dependency plan pairs are generated, to make a total of 16 or 20 sentence plan and dependency plan pairs per content plan.[17] Four sentence plans were discarded due to duplication upon realization, leaving 1756 realizations[18] in the corpus. After the sentence and dependency plan pairs were generated, each one was realized by the RealPro surface realizer, resulting in a corpus of 1756 realizations.

6.2 SPaRKy Content and Text Plan Mapping

To generate the SPaRKy Restaurant Corpus using our DCCG, we map the information from the SPaRKy content plans and text plans to DCCG disjunctive logical forms (DLFs). We use SPaRKy's text plan trees (tp-trees), as in Figure 22, to map the discourse structure to a

level.

[17]Since this study is concerned with the output generated by SPaRKy, we did not include the 90 handcrafted template realizations used in the evaluation of the SPaRKy generator in the corpus.

[18]The total number of realizations reported here is inconsistent with the information reported in (Walker et al., 2007). In corresponding with the authors of that paper, it is not clear why this is the case; however, the difference in reported amounts is quite small, and so should not affect the outcome of this study.

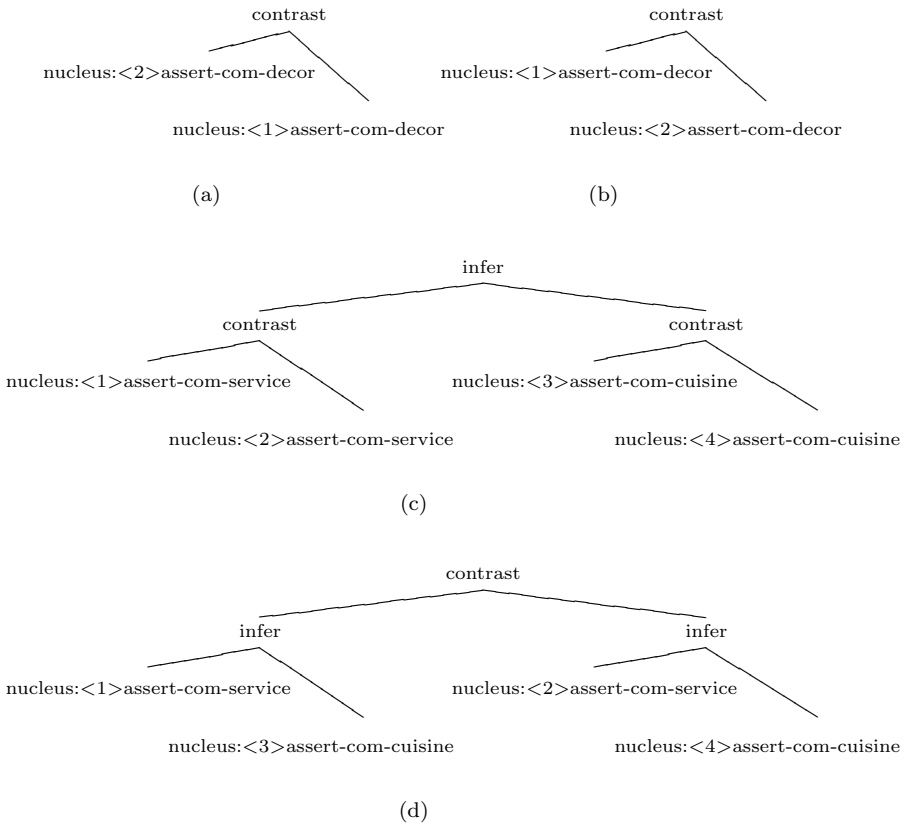

FIGURE 22 The tp-trees built by SPaRKy from the content plans in Figure 21. Figures (a) and (b) show the tp-trees for the content plan in Figure 21a. Figure (c) is a *back-and-forth* tp-tree, while Figure (d) is a *serial* tp-tree, for the content plan in Figure 21b.

Type	Assertion *Example*
offer	assert-com-list_exceptional *Bond Street and Komodo offer exceptional value among the selected restaurants.*
highest-ranked	assert-com-list_top *There are 5 highest ranked restaurants.*
price	assert-com-price *Ferrara's price is 17 dollars.*
locate	assert-com-nbhd *L'ecole is located in TriBeCa SoHo.*
cuisine	assert-com-cuisine *Da Andrea is an Italian restaurant.*
have	assert-com-food_quality *Rain has very good food quality*
	assert-com-service *Uguale has very good service.*
	assert-com-decor *Bienvenue has good decor.*

FIGURE 23 Assertion Types

DLF, and then fill in the clausal arguments with information from the asserted propositions in the content plan.

Initially, we select a content plan and use the associated SPaRKy text plans as the starting point for mapping to our DLFs. As noted earlier, SPaRKy text plans come in two types: *back-and-forth* and *serial*. These types refer to the arrangement of the clausal arguments, in terms of restaurant attribute groupings. Due to the differing arrangements, both types require different mapping strategies.

In back-and-forth text plans, restaurant attributes are grouped by attribute type, and so all the restaurants are compared by one attribute first, then by a second attribute next, etc. For example, Figure 22c is a text plan that produces the realization (among others) in (42). However, because of the continually changing restaurant subjects in back-and-forth text plans, these text plans leave little room for individual assertions to be realized as anything less than main clauses.

(42) Hallo Berlin has mediocre service but Meskerem has decent service. Hallo Berlin is a German restaurant, while Meskerem is an African restaurant.

In serial text plans, all the attributes of one restaurant are grouped together, followed by all the attributes of the next restaurant, etc. Fig-

Type	Surface Forms *Example*
offer	main clause; proper noun subject *Bond Street and Komodo offer exceptional value among the selected restaurants.*
highest-ranked	main clause; expletive-*there* subject *There are 5 highest ranked restaurants.*
price	main clause; proper noun subject *Ferrara's price is 17 dollars.* main clause; pronoun subject *Its price is 20 dollars.*
locate	main clause; proper noun subject *L'ecole is located in TriBeCa SoHo.* main clause; pronoun subject *It is located in TriBeCa SoHo.*
cuisine	main clause; proper noun subject *Da Andrea is an Italian restaurant.* main clause; pronoun subject *It is an Indian restaurant.* demonstrative NP subject, attached to *have*-VP *This Chinese, Latin American restaurant ...* relative clause, attached to NP subject of *have* assertions *..., which is a Chinese restaurant, ...*
have	main clause; proper noun subject *Uguale has very good service.* main clause; pronoun subject *It has good decor.* *with*-PP, attached clause finally or NP finally *..., with very good food quality.* VP with *cuisine*-type as subject *... has very good food quality*

FIGURE 24 Possible surface forms for each assertion type

ure 22d is an example of a serial text plan that produces the realization (among others) in (43). Since all of the attributes are grouped around a single restaurant, additional discourse strategies are possible, including the generation of subject NP anaphora and the realization of basic assertions as subordinate clauses or other sentence fragments, which can be combined with neighboring assertions into a single sentence.

(43) Hallo Berlin has mediocre service, and it is a German restaurant. Meskerem, which is an African restaurant, has decent service.

The back-and-forth and serial text plan classifications only apply to text plans that relate 2 or more different restaurant attributes. Thus, single-attribute text plans that only contrast one type of restaurant attribute fall into neither text plan type. However, since the assertions involved in single-attribute text plans do not have the freedom to be realized as anything less than main clauses, they are mapped in a manner similar to back-and-forth text plans.

6.2.1 Mapping Back-and-Forth Text Plans

The SPaRKy-to-DLF mapping strategy employed for back-and-forth text plans is a simple top-down, order preserving, recursive rewrite of the tp-tree into a DLF. It consists of the following 3 basic steps, where Step 2 can be repeated as many times as is necessary, or omitted in the case of single-attribute text plans. These steps, illustrated in Figure 25, will be further detailed in the remainder of this subsection.

1. Map root node.
2. Map child relations to the argument relation slots. Repeat/Omit this step as necessary.
3. Map content plan clause representations (terminal leaves) to the remaining argument relation slots.

Step 1 begins with determining the relation stored in the root node of the tp-tree. This root relation is mapped to the top level relation in the LF. In addition, at this level, the semantic feature ⟨TURN⟩complete is specified to indicate that at this level the realization must be complete. This first step is illustrated using a case where the root node of the tp-tree is INFER, as in the back-and-forth tp-tree in Figure 22c. This relation is mapped to the discourse relation in the corresponding LF as infer-rel, as depicted in Figure 25a. The feature (⟨MOOD⟩decl)?, with the optionality operator, is also added, as it is for all relations, to indicate that a sentence break is possible (subject to the grammar's constraints). In this case, note that the whole turn can be expressed as

$@_{i_0}$(infer-rel $\wedge$ ⟨TURN⟩complete $\wedge$ (⟨MOOD⟩decl)? $\wedge$
⟨ARG1⟩(...) $\wedge$
⟨ARG2⟩(...))

(a) Step 1: Map root node.

$@_{i_0}$(infer-rel $\wedge$ ⟨TURN⟩complete $\wedge$ (⟨MOOD⟩decl)? $\wedge$
⟨ARG1⟩(c_0 $\wedge$ (contrast-rel $\trianglelefteq$ infer-rel) $\wedge$ (⟨MOOD⟩decl)? $\wedge$
⟨ARG1⟩(...) $\wedge$
⟨ARG2⟩(...)) $\wedge$
⟨ARG2⟩((c_1 $\wedge$ contrast-rel $\trianglelefteq$ infer-rel) $\wedge$ (⟨MOOD⟩decl)? $\wedge$
⟨ARG1⟩(...) $\wedge$
⟨ARG2⟩(...)))

(b) Step 2: Map child relations to the argument relation slots.

$@_{i_0}$(infer-rel $\wedge$ ⟨TURN⟩complete $\wedge$ (⟨MOOD⟩decl)? $\wedge$
⟨ARG1⟩(c_0 $\wedge$ (contrast-rel $\trianglelefteq$ infer-rel) $\wedge$ (⟨MOOD⟩decl)? $\wedge$
⟨ARG1⟩(hsN_1 $\wedge$ **have** $\wedge$ (⟨MOOD⟩decl)? $\wedge$
⟨OWNER⟩(rn_1 $\wedge$ **Hallo_Berlin**) $\wedge$
⟨POSSN⟩(s_1 $\wedge$ **service** $\wedge$ ⟨DET⟩nil $\wedge$
⟨MOD⟩(sq_1 $\wedge$ **mediocre**))) $\wedge$
⟨ARG2⟩(hsN_2 $\wedge$ **have** $\wedge$ (⟨MOOD⟩decl)? $\wedge$
⟨OWNER⟩(rn_2 $\wedge$ **Meskerem**) $\wedge$
⟨POSSN⟩(s_2 $\wedge$ **service** $\wedge$ ⟨DET⟩nil $\wedge$
⟨MOD⟩(sq_2 $\wedge$ **decent**)))) $\wedge$
⟨ARG2⟩(c_1 $\wedge$ (contrast-rel $\trianglelefteq$ infer-rel) $\wedge$ (⟨MOOD⟩decl)? $\wedge$
⟨ARG1⟩(bcN_3 $\wedge$ **be** $\wedge$ (⟨MOOD⟩decl)? $\wedge$
⟨ARG0⟩(rn_3 $\wedge$ **Hallo_Berlin**) $\wedge$
⟨PRED⟩(c_3 $\wedge$ **restaurant** $\wedge$ ⟨DET⟩nil $\wedge$
⟨MOD⟩(cq_3 $\wedge$ **German**))) $\wedge$
⟨ARG2⟩(bcN_4 $\wedge$ **be** $\wedge$ (⟨MOOD⟩decl)? $\wedge$
⟨ARG0⟩(rn_4 $\wedge$ **Meskerem**) $\wedge$
⟨PRED⟩(c_4 $\wedge$ **restaurant** $\wedge$ ⟨DET⟩nil $\wedge$
⟨MOD⟩(cq_4 $\wedge$ **African**)))))

(c) Step 3: Map content plan clause representations (terminal leaves) to the remaining argument relation slots.

FIGURE 25 Step-by-step mapping from back-and-forth tp-tree in Figure 22c to DLF

a single sentence, using *and* to express the INFER relation, as well as two separate sentences separated by a full stop. With the single-sentence option, the grammar allows the ⟨MOOD⟩decl feature to be realized only at this top-level, not at the lower levels. Otherwise, when expressing the INFER relation as a simple juxtaposition of separate sentences, the grammar allows⟨MOOD⟩decl to be realized in the arguments (⟨ARG1⟩ and ⟨ARG2⟩) of infer-rel, and prevents its realization at the top-level.

In Step 2, the left subtree provides the contents of ⟨ARG1⟩, while the right subtree provides the contents of ⟨ARG2⟩. This step can be repeated as many times as necessary, in the event that the back-and-forth tp-tree contains more levels of non-terminal nodes, or omitted entirely in the case of single-attribute text plans. In this example, both child nodes are CONTRAST nodes which are mapped to the discourse relations for each argument as in Figure 25b. In our DCCG, CONTRAST relations are always realized with an overt contrastive connective. However, since CONTRAST can be implied by simple juxtaposition and no use of an overt connective, we specify the alternative usage of an INFER relation, which will either not realize any connective, or will realize the sentence level conjunction, *and*. Both "." (full stop) and *and* are realized in the SPaRKy Restaurant Corpus for text plans using a CONTRAST relation. In our DLFs, we indicate the possible use of either relation using the one-of operator ($\veebar$) as in Figure 25b.

Lastly, in Step 3, the leaves of the tp-tree that correspond to the asserted propositions from the original content plan are mapped to the remaining argument slots in the LF. As with the non-terminal nodes, the left leaf provides the contents of ⟨ARG1⟩, while the right leaf provides the contents of ⟨ARG2⟩. Each leaf is then mapped to a corresponding template LF specification depending on the attributes of the assertion (e.g. *price*, *decor*, etc.). This final step is illustrated in Figure 25c.

6.2.2 Mapping Serial Text Plans

To create a compact DLF for serial text plans, we require 5 steps, where the first two steps are similar to those used in mapping back-and-forth text plans. As in the previous subsection, these steps will be further detailed in the remainder of this subsection, and are illustrated in Figures 26 and 27.

1. Map root node.
2. Map child relations to the argument relation slots. Repeat/Omit this step as necessary.
3. Map the possible sentence fragment LFs to a DLF for each leaf.
4. Add multi-clausal combinations to last sibling leaf.

5. Map references from the node ID access list to the appropriate argument relation slots.

As in back-and-forth text plans mapping, Step 1 starts with mapping the root relation of the serial text plan. Using the serial text plan in Figure 22d as an illustration, we first map its root, as shown in Figure 26a, adding in the ⟨TURN⟩complete and the optional (⟨MOOD⟩decl)? features. In this example, the root is a CONTRAST relation, so as in the earlier mapping example, we indicate the possible use of either the CONTRAST or INFER relation using the one-of operator ($\veebar$).

In Step 2, we first identify the children of each root child, i.e. the root's grandchild nodes. If the grandchild nodes are leaves (as in our example), we skip directly to Step 3. Otherwise, we continue with Step 2 as described in the previous mapping example, and repeat this step again.

Step 3 is where the serial text plan mapping diverges notably from back-and-forth text plan mapping. Because serial text plans allow for more surface form possibilities for any given assertion than back-and-forth text plans, we map the DLF partial clause specifications corresponding to the different sentence form possibilities for each leaf to separate DLF node structures. In addition, we separate out any shared nodes that are common to many disjuncts, such as the restaurant name node, and use reference nominals in the DLF clause specification instead. This way, the common node is defined only once, which minimizes the number of unique nodes used in the entire DLF. Step 3 is repeated $n-1$ times, where n is the number of sibling leaves.

To illustrate this step, we will map the ⟨ARG1⟩ (or leftmost branch) of the root node. Since there are only two sibling leaves in this branch, we only need to execute Step 3 once. Thus, Figure 26c shows the DLF structure for the disjuncts of the leftmost leaf of the tp-tree (labeled nucleus:⟨1⟩assert-com-service) in Figure 22d. Because this leaf ⟨1⟩ is of the have assertion type and there are 4 possible surface forms for this type, we map four possible disjuncts. Of the four disjuncts, the first one defines a new node (including its lexical predication), whereas the last three disjuncts reference the nodes to which the ⟨ELABREL⟩ relation connects. These nodes are later defined in the disjunct set in Step 4 (Figure 27a).

Note that each node is named by a code that indicates the type of leaf, type of disjunct, and the original SPaRKy leaf ID, as listed in Figure 28. For example, in the node name hsN_1, the first two letters (hs) indicate that this node forms a DLF disjunct for a have-service leaf. The following capital letter (N) indicates that the surface form

$@_{c_0}((\textsf{contrast-rel} \trianglerighteq \textsf{infer-rel}) \wedge \langle \text{TURN} \rangle \text{complete} \wedge (\langle \text{MOOD} \rangle \text{decl})? \wedge$
$\langle \text{ARG1} \rangle (\quad \ldots \quad) \wedge$
$\langle \text{ARG2} \rangle (\quad \ldots \quad))$

(a) Step 1: Map root node.

(Not Applicable to this example, though possible for other serial text plans).

(b) Step 2: Map child relations to the argument relation slots.

Disjunctive LF:

$(@_{hsN_1}(\textbf{have} \wedge (\langle \text{MOOD} \rangle \text{decl})? \wedge$	
$\langle \text{OWNER} \rangle (rn_{13}) \wedge$	*Hallo Berlin has mediocre service*
$\langle \text{POSSN} \rangle (s_1)) \trianglerighteq$	
$@_{bcN_3}(\langle \text{ELABREL} \rangle (hsW_1)) \trianglerighteq$	[$_{bcN_3}$ *... with mediocre service*]
$@_{bcP_3}(\langle \text{ELABREL} \rangle (hsW_1)) \trianglerighteq$	[$_{bcP_3}$ *... with mediocre service*]
$@_{c_3}(\langle \text{ELABREL} \rangle (hsW_1)))$	[$_{c_3}$ *... with mediocre service*]

Shared nodes referenced in the above DLF:

$@_{rn_{13}}(\textbf{Hallo\_Berlin})$	*Hallo Berlin*
$@_{hsW_1}(\textbf{with} \wedge$	*with mediocre service*
$\langle \text{ARG1} \rangle (s_1))$	
$@_{s_1}(\textbf{service} \wedge \langle \text{DET} \rangle \text{nil} \wedge$	*mediocre service*
$\langle \text{MOD} \rangle (sq_1 \wedge \textbf{mediocre}))$	

Node ID Access List: [hsN_1]

(c) Step 3: Map the possible sentence fragment LFs to a DLF for each leaf. Above is the DLF for the left-most clause of Fig. 22d.

FIGURE 26 Step-by-step mapping from serial tp-tree in Figure 22d to DLF (continued in Figure 27)

Disjunctive LF:

$(@_{bcN_3}(\mathbf{be} \wedge (\langle \text{MOOD} \rangle \text{decl})? \wedge$	
$\quad \langle \text{ARG0} \rangle (rn_{13}) \wedge$	*Hallo Berlin is a German restaurant*
$\quad \langle \text{PRED} \rangle (c_3)) \trianglelefteq$	
$@_{i_0}(\mathsf{infer\text{-}rel} \wedge (\langle \text{MOOD} \rangle \text{decl})? \wedge$	
$\quad \langle \text{ARG1} \rangle (hsN_1) \wedge$	*Hallo Berlin has good service (.|and)*
$\quad \langle \text{ARG2} \rangle (bcP_3)))$	*it is a German restaurant*
$(@_{rn_{13}}(\langle \text{GENREL} \rangle (bcN_3)))?$	$[_{rn_{13}}$ *..., which is a German restaurant,*$]$

Potentially Shared Nodes (in the case of 3 or more sibling leaves):

$@_{bcP_3}(\mathbf{be} \wedge (\langle \text{MOOD} \rangle \text{decl})? \wedge$	
$\quad \langle \text{ARG0} \rangle (p_3 \wedge \mathbf{pro3}) \wedge$	*it is a German restaurant*
$\quad \langle \text{PRED} \rangle (c_3))$	
$@_{cq_3}(\mathbf{German})$	*German*

Shared nodes referenced in the above DLF:

$@_{c_3}(\mathbf{restaurant} \wedge \langle \text{DET} \rangle \text{a} \wedge$	*a German restaurant*
$\quad \langle \text{MOD} \rangle (cq_3))$	

Node ID Access List: [hsN_1, bcN_3, i_0]

(a) Step 4: Add multi-clausal combinations to last sibling leaf. Above is the DLF for the leaf labelled nucleus:⟨3⟩assert-com-cuisine in Fig. 22d.

$$@_{c_0}((\mathsf{contrast\text{-}rel} \trianglelefteq \mathsf{infer\text{-}rel}) \wedge \langle \text{TURN} \rangle \text{complete} \wedge (\langle \text{MOOD} \rangle \text{decl})? \wedge \langle \text{ARG1} \rangle (hsN_1 \trianglelefteq bcN_3 \trianglelefteq i_0) \wedge \langle \text{ARG2} \rangle (hsN_2 \trianglelefteq bcN_4 \trianglelefteq i_1))$$

(b) Step 5: Map references from the node ID access list to the appropriate argument relation slots. Above, the access list from Step 4 is mapped to ⟨ARG1⟩, while the access list for the right branch of the root (mapping steps not shown) is mapped to ⟨ARG2⟩.

FIGURE 27 Step-by-step mapping from serial tp-tree in Figure 22d to DLF (continued from Figure 26)

Code	Assertion type	Code	Disjunct type
hd	have-decor	N	proper (N)ame
hf	have-food-quality	P	(P)ronoun
hs	have-service	W	(W)ith
bc	be-cuisine (restaurant type)	T	(T)his
bn	be-neighborhood (location)		
bp	be-price		

FIGURE 28 Node ID Legend

resulting form this disjunct will be a full sentence clause with a proper noun subject. The final subscript ($_1$) is the original SPaRKy content plan ID for this leaf. These codes are generated for all the leaves in a sibling set at the beginning of step 3, when the assertion types for all of the leaves are also determined.

In addition to the disjuncts, the shared nodes referenced within them are listed underneath. They include the node for the restaurant name, which is shared by the disjunct set in Step 4 (Figure 27a), the with predicate node, and the node for the object NP.

Lastly in Step 3, in order to make it possible to generate references to nodes defined elsewhere, during the first iteration of the third step, we create a node ID access list which we then update during each iteration of the third step, as well as during the fourth step. This access list contains the node IDs for any main clause with a proper noun subject, which can be modified by a disjunct from each of its sibling leaves to form a single sentence covering all of these leaves. In addition, during the fourth step, we also add the node IDs of the multi-sentence combinations, headed by INFER relation nodes, to the access list. In our example, the only ID added from our single iteration of Step 3 is hsN_1 (main clause with proper noun subject for the have-service leaf).

Next, in Step 4, we add the potential multi-sentence combinations of all the leaves sharing the same parent to the disjuncts of the rightmost leaf of that sibling set (henceforth referred to as the rightmost leaf), and shift potentially shared disjuncts from the DLF as explained further below. Our example in Figure 27a shows the augmented DLF for the rightmost leaf of this sibling set, labelled nucleus:⟨3⟩assert-com-cuisine in Figure 22d. This augmented DLF is built in a similar fashion to the leaf in Step 3. Initially, disjuncts leading to the first 3 of the 4 possible surface forms of cuisine assertion type (as listed in Figure 24) are built into the DLF. This includes the disjuncts for realizing the leaf as a main clause with a proper noun (restaurant name) as subject, main clause with a pronoun as subject, and as a demonstrative subject

NP. The fourth possible surface form, realization as a relative clause, is characterized as an optional modification of the shared restaurant name node shown in the previous Figure 26c. In our grammar, the LF for subject relative clauses has a ⟨GenRel⟩ relation between the NP head and the main clause semantics of the relative clause, where the subject of that relative clause references the NP head node to which it is attached. Thus, this fourth possibility lies outside of the disjunction because when it is selected, it references the first of the disjuncts.

At this point, Step 4 diverges from Step 3 in that it makes a number of alterations to the disjunction formed by the Step 3 process. First, we delete any disjuncts that would not be realized due to the number of the sibling leaves. In our example, leaf ⟨1⟩ is a cuisine assertion in a 2-leaf sibling set, and so it will not be realized solely as a subject NP. That disjunct is only realized if the cuisine assertion occurs as a non-initial leaf in a serial text plan with 3 or more sibling leafs. Since this is not the case in our example, we delete the subject NP disjunct.

Next, we add to the disjunct set any multi-sentence combinations related by an INFER relation. In our example, there are only two siblings, so they can only be joined as full sentences by being placed one after the other, in leaf order. This results in the disjunct headed by the node labelled i_0, where i refers to the connecting relation, INFER. Note that in this combination, the first argument of the INFER relation is a reference to the have-service assertion, realized as a full clause with a proper noun subject, whereas the second (and final) argument of the relation is realized as a full clause with an anaphoric pronoun subject, since both sentences refer to the same restaurant.

If there were 3 or more sibling leaves, there would be more possible multi-sentence combinations, all of which would reference the rightmost leaf's disjunct for realizing main clause with a pronoun subject. Since this disjunct would be referenced or shared between many disjuncts, we move this disjunct out of the DLF partial clause specification and into the shared nodes section, so as to be with the other shared nodes. Although this disjunct is not shared in our example, in order to be uniform with the other cases, we label it as *potentially shared*, as it could be shared if there were more multi-sentence combinations. Similarly, the node for the cuisine modifier *German* is included here, as in other cases it could be shared by a demonstrative subject NP.

Lastly, we update the node ID access list started in Step 3, with the node IDs for the main clause with a proper noun subject, and the node IDs for all multi-sentence combinations. In our example, this includes the node bcN_3 (main clause with proper noun subject for the cuisine leaf), and i_0, the only INFER relation node.

Finally, in Step 5, we use the node ID access list completed in Step 4 to map disjunctive references to the possible leaf assertion combinations. All the references for a given set of sibling leaves are placed as disjuncts into the relevant argument slot of the root node mapping. In Steps 3 and 4, our example covered the left branch of the root, which corresponds to the first argument of the root CONTRAST relation. Thus, the node IDs collected in Step 4 will be placed into the ⟨ARG1⟩ node in Figure 27b.

To map the right branch of the root, we repeat Steps 3 and 4 on that branch of the tp-tree, and place the node IDs from the corresponding access list into the ⟨ARG2⟩ node, which we have filled in Figure 27b.

6.2.3 Beyond Binary Branches

The mapping steps detailed above work best for binary branching trees. However, there exist text plans with more than two branches on any non-terminal level, and so additional mapping steps are needed to account for all the variations produced by SPaRKy.

SPaRKy text plans can have 2-5 branches from any given non-terminal node, where each branch represents a discourse unit of one or more clauses. Multiple branch nodes (referring to nodes with 3 or more children) can happen at any level of the tree, and on more than one level at a time. Trees can contain multiple siblings at the leaf level, in additional to multiple branching at a higher level.

When a node in a SPaRKy text plan contains more than 3 branches, SPaRKy probabilistically chooses one of several clause combining operations on adjacent branches, to create a new discourse unit, represented in a sentence plan as a new binary branch. For example, if there are 3 leaf nodes from the same parent node, then SPaRKy may choose to merge the first two clauses into a single sentence via conjunction or subordination, or juxtapose each as full sentences (i.e. ended by a ".", and placed adjacent to each other). Then SPaRKy combines the new 2-clause discourse unit with the third clause, again, by conjunction, subordination, or juxtaposition. SPaRKy continues in this fashion until all the branches have been combined with one another, so that the sentence plan is a binary tree. The probabilistic nature of the clause combining is such that many different sentence plan trees (sp-trees) can result from the same tp-tree.

In our DLFs, instead of recreating these probabilistic clause combinations, we use disjunctions to compactly specify the alternative sentence forms that each leaf can be realized as, whether there are only two leaves or up to five leaves to combine. In this way, both lexicalization and aggregation decisions are deferred to the realizer, which

can take both grammatical constraints and ranking preferences into account when choosing surface forms.

6.3 Evaluation of the Mapping Process

To demonstrate the feasibility of our approach, we set out to exactly reproduce the majority of the contrastive realizations in the SPaRKy Restaurant Corpus, in order to illustrate the breadth of our DLF mapping approach in comparison to a comparable generator.

We started by excluding 685 of the realizations in SPaRKy Restaurant Corpus since their content plans did not include the CONTRAST relation. This included the 30 RECOMMEND content plans and the 5 COMPARE-3 content plans that do not employ the CONTRAST relation. We also initially excluded another 5 content plans (totaling 100 realizations) that compare 4 or 5 restaurants in one long utterance because we believe the information should be aggregated in a more compact form for the listener to understand. Lastly, as we conducted our regression testing, we became aware of and excluded an additional 36 realizations (across 8 different content plans) that have errors in either subject-predicate agreement with respect to the copula or infelicitously employed pronominal anaphora. Added together, we have excluded a total of 821 realizations, leaving us with 935 target SPaRKy contrastive realizations for our regression testing.

Using OpenCCG's regression testing framework,[19] we have verified that we can reproduce 92% of the 935 target SPaRKy contrastive realizations. The remaining 8% of contrastive realizations are likely missing due to the limitations of our text plan mapping algorithm (though search errors could also account for a portion of these). SPaRKy uses a bottom-up approach to creating sentence plans, starting with a tp-tree and applying clause combining operations to each node in the tree probabilistically. Our text plan mapping method, on the other hand, uses a one-pass top-down traversal to create DLFs, which include various alternatives for combining specifications for the leaves of the tp-trees. In developing the algorithm, we have incorporated the most frequent patterns used to create SPaRKy realizations, but have found that enumerating all of the possible patterns becomes a time consuming and error-prone process, especially when the content plan contains 4 or more different sibling leaves (assertions). This can be a similar problem for other large text plans, i.e. those that compare 4 or more restaurants. In addition, the larger the resulting DLF is (that is,

[19] For regression testing, OpenCCG uses n-grams from a given target sentence to guide the search, as illustrated in Figure 19, along with beam search to narrow the search space.

the more patterns that a DLF must account for), the longer it takes for OpenCCG to realize all the possible outputs for a given DLF or search for a target realization in the case of regression testing. These difficulties suggest that our one-pass algorithm does not scale well as the DLFs become larger, and thus that adding another, bottom-up pass, as in the SPaRKy approach, should be explored in future work.

We hasten to point out that in the end, using the above mapping techniques, we can actually convert every targeted contrastive SPaRKy text plan to a disjunctive HLDS representation, including those that yielded the ungrammatical or infelicitous realizations that were then excluded and unreproduced.[20] The resulting DLFs can then be realized using OpenCCG, yielding several hundred to several thousand complete realizations for each DLF, with quality ranging from very natural to quite unnatural sounding realizations. Separating the desirable natural sounding realizations from the unnatural ones is a problem for ranker, which we leave for future work. The realizations produced from our DLFs cover the majority of the SPaRKy corpus, plus additional possibilities that go beyond SPaRKy, to which we now turn.

6.4 Modifications to the SPaRKy Content Plans

In the interest of generating texts that use contrastive connectives in more natural ways, we extend and refine existing SPaRKy text plan features as a result of our mapping process. This includes realizing contrastive connectives in serial text plans, and restricting referring expression generation. We also modify other SPaRKy content and text plan features by adding to the text plan itself. This includes adding the ADDITIVE and MERELY relations into the tree when appropriate, along with adding the EVIDENCE relation and summary clause semantics required to incorporate summary statements in serial text plans.

Contrastive Connectives in Serial Text Plans In SPaRKy, contrastive connectives are only realized for the back-and-forth text plans and text plans that contrast only one type of assertion (i.e. only one restaurant attribute). In our DCCG realizations, we additionally realize contrastive connectives for serial text plans. That is, we can overtly realize CONTRAST relations that are higher up in the text plan, relating two complex discourse units. For example, using the same text plan as in Figure 22d from Section 6.1, we can produce the realization in

[20] We do not currently have the ability to map the larger 4- or 5-restaurant text plans. Since they were initially excluded from the testing, no algorithm was developed to map these text plans. However, it would not be difficult to extend the current algorithm to map these restaurant plans, although the difficulty in enumerating all the plans remains.

(44).[21] In addition to this version that includes the contrastive connective *however*, we can also produce this and other SPaRKy realizations without the use overt contrastive connectives.

(44) Hallo Berlin is a German restaurant, with mediocre service. However, Meskerem, which is an African restaurant, has decent service.

Referring Expression Generation In these back-and-forth text plans, unlike SPaRKy, we do not generate pronouns for the restaurant subjects. Due to the nature of the assertion order in these text plans, the nominative references constantly change from one restaurant to the other. This results in discourse where a pronoun and its antecedent would always be separated by another restaurant referent, thus making it difficult to retrieve the correct antecedent for a given restaurant pronoun.

On the other hand, for serial text plans, we do adopt a similar strategy as that used in SPaRKy by allowing restaurant references to be realized as a pronoun if the antecedent has just occurred in the preceding discourse unit. Due to the nature of serial text plans, where all assertions about the same restaurant are grouped together, no other nominative reference intervenes between a given pronoun and its antecedent.

As illustrated in the previous Section 6.2.2, we chose to create separate disjuncts for main clause specifications with proper noun subjects and for those with pronoun subjects. While we could have used the one-of operator ($\veebar$) to separate the disjuncts at the subject noun specification, e.g. $b_2 \wedge$ (**Bienvenue** $\veebar$ **pro3**), this would allow for the infelicitous realization of cataphora, wherein the first quality in a serial leaf sibling set was realized with a pronoun, and its "antecedent" (the proper noun) was realized in a subsequent sentence. In order to constrain pronoun reference to anaphoric reference alone, we specify the order of the main clause types such that main clauses with pronouns can only occur after a main clause with the proper noun referent in a

[21] The version of (44) that does not contain a contrastive connective does not appear in the SPaRKy Restaurant Corpus, though it should be possible generate it using SPaRKy. We have chosen to use this realization, generated by our DCCG, as our example rather than the more awkward one produced by SPaRKy in (43), repeated here:

(43) Hallo Berlin has mediocre service, and is a German restaurant. Meskerem, which is an African restaurant, has decent service.

serial text plan. In addition, we also do not allow several main clauses with the same proper noun subject to occur adjacent to one another.

Additional Relations We employ two additional relations, ADDITIVE and MERELY, both of which are further discussed in Section 7. Following Striegnitz (2004), we use the ADDITIVE connectives, *too* and *also*, to relate a pair of discourse units that meets her definition of *also-parallelism*:

> **Definition 5.2 (Also-parallelism).** Let S be a sentence, e the eventuality it describes, and let $a_1, ..., a_n$ be the entities that are specified as participants of e by S. Furthermore, let p be the sort of e as specified by S and $q_1, ..., q_n$ the properties that S uses to describe $a_1, ..., a_n$. S is *also-parallel with respect to the context* iff the context provides an eventuality e' with participants $b_1, ..., b_n$ such that b_i and a_i are alternatives for one i and q_j holds of b_j for all $j \neq i$. If the above conditions hold, we will also say that eventuality e is *also-parallel* to eventuality e'.

To illustrate this definition, consider (45) below. Prior to the marking of the additive relation (represented by *also*), the LF for (45b) is as in Figure 29b. If the S under consideration is (45b), then e corresponds to hsN_2, and a_1 and a_2 correspond to the argument entities rn_2 and d_2, respectively. The property q_1 is simply **Bienvenue**, while the more complex property q_2 is **decor** $\wedge \langle \text{MOD} \rangle (dq1 \wedge$ **decent**$)$. The context is given by (45a), in which e' corresponds to hsN_1, and b_1 and b_2 correspond to rn_2 and d_2, respectively. Here, b_1 (satisfying **Sonia_Rose**) is an alternative restaurant to a_1, and if we assume that dq_1 and dq_2 are co-referential (i.e. both are discourse referents for a unique attribute value, namely the one satisfying **decent**), then q_2 holds for both a_2 and b_2. Thus we can then say that S (45b) is *also-parallel* with respect to the context provided by S' (45a). Given this, we posit that an ADDITIVE relation holds between S and S', where S' is the sentence containing e'.

(45) a. Sonia Rose has decent decor.
b. Bienvenue also has decent decor.

Though the definition above allows us to provide alternatives for any q_i, we currently restrict the alternative set to the set of restaurants in the SPaRKy Restaurant Corpus. The insertion of the ADDITIVE relation into a tp-tree can be seen in Figure 31.

The MERELY relation, which can be realized by the discourse connectives *merely*, *just* or *only*, is also optionally added between tp-tree leaves which contain identical attributes, but in this case, differing values. That is, the MERELY relation can only be employed when the sec-

$@_{hsN_1}$(**have** $\wedge$ ($\langle$MOOD$\rangle$decl)? $\wedge$
$\langle$OWNER$\rangle$(rn_1 $\wedge$ **Sonia_Rose**) $\wedge$
$\langle$POSSN$\rangle$(d_1 $\wedge$ **decor** $\wedge$ $\langle$DET$\rangle$nil $\wedge$
$\langle$MOD$\rangle$(dq_1 $\wedge$ **decent**)))

(a)

$@_{hsN_2}$(**have** $\wedge$ ($\langle$MOOD$\rangle$decl)? $\wedge$
$\langle$OWNER$\rangle$(rn_2 $\wedge$ **Bienvenue**) $\wedge$
$\langle$POSSN$\rangle$(d_2 $\wedge$ **decor** $\wedge$ $\langle$DET$\rangle$nil $\wedge$
$\langle$MOD$\rangle$(dq_2 $\wedge$ **decent**)))

(b)

FIGURE 29 Prior to adding the ADDITIVE relation, the LF specifications corresponding to (45a) and (45b) are in (a) and (b), respectively.

ond value is worse on the scale of desirability (i.e. is poorer) than the first value, as is the case with the values *very good* and *decent* in (46).

(46) a. Sonia Rose has very good decor.
b. However, Bienvenue merely has decent decor.

However, it cannot be employed when the second value is located clearly on the other side of a neutral point in the scale from the first value, as in (47). This restriction is not exercised in our text plan modifications since the linguistic neutral point (which does not necessarily correspond to any numeric calculation of a midpoint) intuitively seems to be near *mediocre* or *decent* on the scale, and there are no terms attested in the SPaRKy Restaurant Corpus indicating a worse gradation than *mediocre*. Instead, we generate the MERELY relation when the first criterion is met, and leave more fine-grained filtering to the automatic ranker.

(47) a. Sonia Rose has very good decor.
b. # However, Bienvenue merely has poor decor.

In addition, regardless of the addition of the MERELY relation, in serial text plans where the attributes of the same restaurant have values on the opposite side of the neutral point, or one of the values is the midpoint while the other is not, we also specify the alternate use of the CONTRAST relation in addition to the existing INFER relation supplied by the original SPaRKy tp-tree.

Lastly, when both values are on the negative side of the scale in the MERELY relation, it would be odd if the more positive item were presented first as in (48). Instead, the positive item should be last as in (49).[22]

(48) a. Sonia Rose has mediocre decor.
b. # However, Bienvenue merely has poor decor.

[22]We thank one of our anonymous reviewers for the examples in (48) and (49).

Numeric Range	Linguisic Term
8-13	mediocre
14-16	decent
17-19	good
20-22	very_good
23-25	excellent
26-28	superb

FIGURE 30 Desirability Scale Mapping

(49) a. Sonia Rose has dreadful decor.
b. However, Bienvenue merely has poor decor.

Figure 31 illustrates the insertion of MERELY relations to a tp-tree. In this example, there are two MERELY relations: one instance relating the second summary statement to the first summary statement, and a second instance relating the food-quality assertions. Figure 31 also illustrates the alternation between INFER and CONTRAST in the right subtree due to one of the values being the midpoint *decent*, while the other is *mediocre*.

Summarizing Statements We have added summarizing statements to serial text plans which map the average of the property ratings for each restaurant to a degree of desirability, using the same mapping scale that SPaRKy uses for each individual property.[23] These summary statements only apply to the subset of content plans that mention more than one type of have assertion, since these are the only assertion types that are mapped to the desirability scale. Based on the corpus, the SPaRKy desirability scale maps a numeric rating (with an attested range of 8-28; 28 being best) to a linguistic 6-point scale, given in Figure 30. To illustrate our summary statement, we use the example in (50), with the summary statements in italics. Here we can see that based on the SPaRKy scale, Sonia Rose, a restaurant with one *very good* quality (rating=20) and one *decent* quality (rating=16), averages out to be a *good* restaurant (rating=18). On the other hand, Bienvenue, which also has a *decent* quality (rating=16) but in addition has a *mediocre* quality (rating=10), averages out to be a *mediocre* restaurant (rating=13).

In addition, (50) also illustrates the other modifications and additions discussed in this subsection. The contrastive connective, *however*, that joins the two restaurants in a serial text plan, is indicated in sans serif font, while the anaphoric pronouns (in this case, *it*) are indicated

[23]Naturally, one could easily substitute other methods of determining a summary-level rating during content planning, if desired.

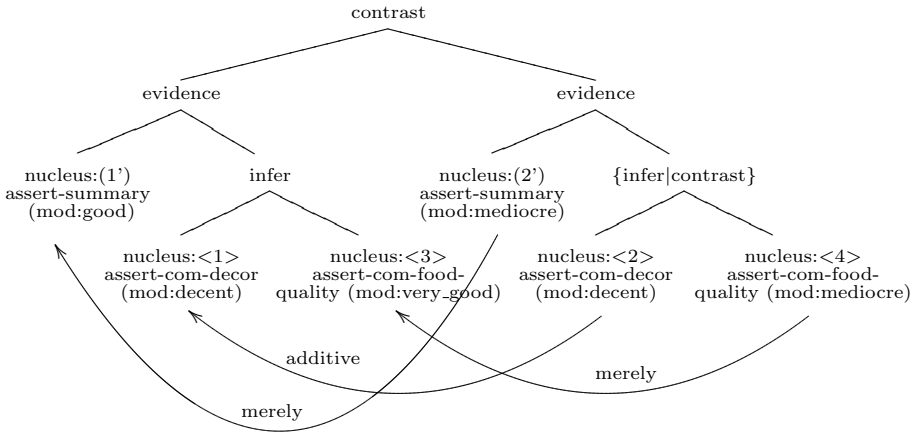

FIGURE 31 SPaRKy tp-tree altered with new relations and summary statements, corresponding to Example 50.

in bold font. Lastly, the connectives *also* and *only*, which represent the additional relations, ADDITIVE and MERELY, respectively, are indicated in small caps.

(50) (1'): *Sonia Rose is a good restaurant.*
<1>: **It** has decent decor and
<3>: very good food quality.
(2'): However, *Bienvenue is just a mediocre restaurant.*
<2>: While **it** ALSO has decent decor,
<4>: **it** ONLY has mediocre food quality.

7 Related Work

In terms of its discourse theoretical basis, DCCG is most closely related to D-LTAG. In general, as Webber (2006) observes, discourse grammars vary in their theoretical style, from wholly based on dependency relations (e.g. Halliday and Hasan 1976) to adherence to a completely constituent-based model (e.g. Rhetorical Structure Theory [RST], Mann and Thompson 1988; Linguistic Discourse Model, Polanyi 1988, Polanyi and van den Berg 1996). Dependency-based discourse theories are advantageous because they allow discourse relations to exist between non-adjacent discourse units, lifting restrictions on which clauses can serve as discourse arguments of a given relation. Compu-

tationally, this requires a large search space when attempting to determine the locations of the discourse units, as they can potentially be found anywhere in the text. However, similar to the process of determining the referent of a coreferential pronoun in coreference resolution, the location of the antecedent discourse unit is generally limited by other factors such as recency effects or the presence of competing discourse units. In contrast, constituency-based discourse models narrow the computational search space by determining the location of both discourse arguments using constituent-forming rules. A problem with this approach, though, is that some discourse relations may exist between two non-adjacent constituents. This is explicitly barred in continuous constituency models such as RST, the discourse model that has been most frequently used for generation tasks. D-LTAG (and thus DCCG, in this respect) lies between these two extremes. It is a mixed dependency/constituency model, in that structural connectives are treated in a constituency-based manner while anaphoric discourse adverbials are treated in a dependency-based manner. Discourse GraphBank (Wolf and Gibson, 2005) is another type of mixed dependency/constituency model, where the discourse model is treated as a shallow graph of discourse units linked to one or more previous discourse units. Segmented Discourse Representation Theory (SDRT; Asher and Lascarides, 2003) is another theory that, like D-LTAG, assigns a hierarchical structure to discourse, but one that is not constrained to be a tree. It is interesting to note that previous work on discourse parsing with RST and SDRT (Soricut and Marcu, 2003, Baldridge and Lascarides, 2005) has employed separate discourse segmentation and discourse parsing stages, mirroring the distinction between the separate syntactic and discourse grammars in D-LTAG.

Since our disjunctive LFs are based on the SPaRKy text plans, which draw their relations from the RST taxonomy, our structural relations are also based on RST relations.[24] However, neither RST nor SPaRKy describe relations corresponding to our use of the anaphoric connectives, *also/too* and *merely/just/only*, requiring us to look elsewhere for relation definitions.

With respect to *also/too*, RST has associated *also* with the MULTINUCLEAR RESTATEMENT relation, which holds when "an item [discourse unit] is primarily a reexpression of one linked to it; the items are of comparable importance to the purposes of [the] W[riter]" (Mann and

[24]The usage of these relations in the DLFs differs from RST in that DLF's do not mark the RST nucleus/satellite distinctions on their discourse relation arguments, although SPaRKy text plans do so. The nucleus/satellite distinction could easily be added if it proved useful.

Taboada, 2010). While this is clearly not the same relation we signify in our use of *also*, it follows the general schema of all RST relations in that this relation only holds between two adjacent discourse units. This schema is potentially problematic for RST since in other usages, the clauses that license *also* need not be adjacent to the clauses that contain *also*, as was exemplified previously in (50). However, since RST is not a lexicalized theory, there may be no need for it to account for all discourse licensed adverbials.

The Penn Discourse Treebank (PDTB) Tagging Manual (Prasad et al., 2008) has similarly reported that its annotators have identified *also* as conveying the SPECIFICATION relation, though only for 1 token out of 1746. SPECIFICATION is a subtype of their RESTATEMENT relation, wherein Arg2 describes the situation described in Arg1 in more detail, and is thus the most similar relation to RST's MULTINUCLEAR RESTATEMENT relation. However, given the uniqueness of this assignment, it is possible that this tag may be considered noise, though more investigation is required to be certain. In addition, the PDTB tagging manual reports three other relations that have been annotated for *also*. The relation LIST, which applies when Arg1 and Arg2 are members of a list defined in the prior discourse, was annotated for 10 out of 1746 tokens of *also* (0.6%), while the bulk of the *also* tokens (1733 tokens, or 99.3%) were labelled as CONJUNCTION, where the situation described in Arg2 provides additional, discourse new, information that is related to the situation described in Arg1. Lastly, two instances of *also* were annotated as representing two relations – CONJUNCTION and SYNCHRONY, where SYNCHRONY indicates that both arguments overlap in time.

Although it could be argued that our use of *also* is similar to that described by the CONJUNCTION relation (in that in introduces the new information in the form of the alternative subject), this relation also applies when the entire argument (and not just one aspect) is new. This is outside the scope of the *also-parallel* case we describe in Section 6.

In the theoretical pragmatics literature, and elsewhere as well, *also* and *too* have frequently been classed as additive particles (König, 1991, Krifka, 1998, Striegnitz, 2004, Creswell et al., 2002). Although the class of additive particles may sometimes be taken to include additional adverbials such as *in addition* or *moreover* (Creswell et al., 2002), it can also refer to just those adverbials which satisfy a much narrower condition. For example, according to Krifka (1998), additive particles express that the predication to which they are added holds for at least one alternative of the expression in focus. This condition is similar to, though slightly broader than, the *also-parallelism* condition that we are employing in our DLFs.

With respect to *only*, there have been many analyses of it and its exclusivity implications/presuppositions (Horn, 1969, Rooth, 1985, König, 1991, Rooth, 1992, Roberts, Forthcoming). While *only* undoubtedly carries this interpretation, we are interested here in a different, relational sense of *only*.[25] In our examples in Section 6, we find that the presence of *only* in our comparisons serves to ameliorate the felicity of the utterance when bringing into focus a quality that is worse compared with a previous quality. The same effect is also achieved using *merely* or *just*. To our knowledge, previous research on *only* has not addressed this particular relational meaning.

As noted in Section 2, while the usages of *also* and *only* that we have focused on have not typically been treated on a par with other discourse connectives, they nevertheless serve equally well as more commonly discussed adverbials such as *then* or *otherwise* for the purposes of showing how discourse adverbials are handled in our single grammar approach.

Returning to the issue of one versus two grammars, we observe that in its use of a single grammar for both the sentential and discourse levels, DCCG is reminiscent of G-TAG (Danlos, 2000). However, G-TAG employs no mechanism like cue threading, instead expressing the semantics of a discourse connective directly in its elementary tree, which is allowed to span sentences. As such, it is unclear whether it could be used to analyze paired intersentential connectives, each of whose arguments can span multiple sentences. In addition, like D-LTAG, it requires special post-processing of connectives in medial position.

Similarly to G-TAG, Banik's (2009a, 2009b) discourse-level Tree Adjoining Grammar also incorporates discourse and syntax structure into a single grammar. Currently, the focus of this grammar is to account for the periphrastic generation of parentheticals and referring expressions, issues which we have not yet addressed. Although the grammar description includes a brief mention of a representation for discourse connectives and their constraints on syntax, the connectives' treatment is not fully explained. Furthermore, it is unclear whether the discourse-

[25] Although *only*, *even*, *also*, *too*, etc. are usually analyzed as particles in the theoretical literature, Stede has suggested that they might also convey the same meaning as more typical discourse connectives (Webber et al., 1999), as in the following example:

(1) a They laid waste to the park land.
b *Moreover*, they began cutting down trees.
b' They *even* began cutting down trees.

level TAG allows for discourse connectives to take scope over multiple sentences, for multiple connectives occurring on a single clause, or for connectives in sentence medial positions.

As discussed in Section 5, our cue threading technique itself is reminiscent of Power et. al.'s (1999) *cue store*, a stack of discourse cues accumulated from higher levels in a rhetorical structure tree. The use of a stack of cues, rather than a single cue feature, as in DCCG, strikes us as underconstrained. While multiple connectives are allowed, there is no mechanism to prevent multiple structural connectives from occurring in the same basic clause. The issue is avoided in later descriptions of their generation system, where no more than one connective per elementary proposition is considered (Power et al., 2003, p. 240). A perhaps more important difference is that Power et al.'s focus is to investigate the relationship between rhetorical structure and document structure, rather than the relationship between discourse and sentential syntax, about which they say relatively little; conversely, issues of document structure lie largely outside the focus of our work.

Cue threading also resembles Cooper storage for quantifiers (Cooper, 1983). Cooper storage enables a quantifier to take wider scope than the surface syntactic position of its NP. In the same way, cue threading enables structural connectives to have wider scope than the clause to which they are syntactically bound. On the other hand, Cooper storage allows multiple quantifiers to be in storage at any given time, leading to scope ambiguities (which are attested, in this case). Cue threading avoids these scope ambiguities by only keeping track of a single undischarged structural connective at once.

Finally, to compare our approach to SPaRKy (Walker et al., 2007), we first note that SPaRKy, along with most approaches to discourse generation (i.e. Hovy, 1988, Scott and de Souza, 1990, Walker et al., 2002, Williams and Reiter, 2008), is based on Rhetorical Structure Theory (RST; Mann and Thompson, 1987). By treating discourse adverbials as anaphoric relations, we transcend the limitations of RST's constituent tree structure. In another difference, SPaRKy employs arbitrary transformations on clausal syntactic dependency trees to produce its paraphrases, rather than a declarative, bi-directional grammar,[26] which we suggest may be a more scalable solution. Lastly, as discourse adverbials have not been employed in SPaRKy, its use of contrastive connectives may be less natural than desired in many cases.

[26] A bi-directional grammar is one that can be used both for parsing and generating natural language.

8 Conclusions and Future Work

In this paper, we have shown how a simple cue threading technique enables a lexicalized grammar such as CCG to be extended to handle structural discourse connectives — leaving discourse adverbials to be handled via anaphora resolution — without resorting to the use of two separate grammars, as in D-LTAG. While in this paper we use the resulting Discourse CCG, or DCCG, to generate paraphrased comparisons, in principle we could also use our DCCG in OpenCCG to parse these same comparisons and others like them. However, currently we do not employ an ambiguity resolution module, which can result in a huge number of parses for a given comparison, or an anaphora resolution module, which results in underspecified reference nominals for anaphoric elements in the parse tree. Both such modules are areas for future research.

We have also shown how DCCG can be used to generate comparative descriptions that go beyond ones in the SPaRKy Restaurant Corpus, including clauses with multiple connectives and non–tree structured rhetorical dependencies, unlike traditional RST-based approaches. In future work, we plan to collect ratings data that will allow us to train a ranker for use with this DCCG. We also plan to investigate more flexible approaches to ordering the arguments of discourse connectives, together with constraints to ensure that the antecedents of discourse adverbials are always realized in the preceding text.

Acknowledgements

We thank Bonnie Webber, Mark Steedman, Johanna Moore, Craige Roberts, Erhard Hinrichs, Alex Lasacarides, Donia Scott, Markus Dickinson, and the anonymous reviewers for helpful comments and discussion, as well as audiences in Edinburgh and Bloomington where earlier versions of this work have been presented. In addition, we thank Marilyn Walker, Amanda Stent and François Mairesse for allowing and supporting our use of the SPaRKy inputs and the SPaRKy Restaurant Corpus. This work was supported in part by NSF grant IIS-0812297.

References

Asher, Nicholas and Alex Lascarides. 2003. *Logics of Conversation*. Cambridge University Press.

Baldridge, Jason. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Baldridge, Jason and Geert-Jan Kruijff. 2002. Coupling CCG and Hybrid Logic Dependency Semantics. In *Proc. ACL-02*.

Baldridge, Jason and Geert-Jan Kruijff. 2003. Multi-Modal Combinatory Categorial Grammar. In *Proc. ACL-03*.

Baldridge, Jason and Alex Lascarides. 2005. Probabilistic head-driven parsing for discourse structure. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 96–103. Ann Arbor, Michigan: Association for Computational Linguistics.

Banik, Eva. 2009a. Extending a surface realizer to generate coherent discourse. In *Proceedings of the Short Papers of the Joint conference of the Association for Computational Linguistics and the Asian Federation of Natural Language Processing (ACL-IJCNLP-09), Singapore*.

Banik, Eva. 2009b. Parenthetical constructions - an argument against modularity. In *Proceedings of the workshop on Grammar Engineering Across Frameworks,at ACL-IJCNLP-09, Singapore*.

Blackburn, Patrick. 2000. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL* 8(3):339–625.

Cooper, Robin. 1983. *Quantification and Syntactic Theory*. Dordrecht: D. Reidel.

Copestake, Ann, Dan Flickinger, Carl Pollard, and Ivan Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation* 3(4):281–332.

Copestake, Ann, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*, pages 132–139.

Creswell, Cassandre, Katherine Forbes, Eleni Miltsakaki, Rashmi Prasad, Aravind K. Joshi, and Bonnie Webber. 2002. The discourse anaphoric properties of connectives. In *Proceedings of 4th Discourse Anaphora and Anaphor Resolution Colloquium (DAARC)*. Lisbon, Portugal.

Danlos, Laurence. 2000. G-TAG: A lexicalized formalism for text generation inspired by tree adjoining grammar. In *Tree Adjoining Grammars: Formalisms, Linguistic Analysis, and Processing*. CSLI.

Dinesh, Nikhil, Alan Lee, Eleni Miltsakaki, Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2005. Attribution and the (non-)alignment of syntactic and discourse arguments of connectives. In *CorpusAnno '05: Proceedings of the Workshop on Frontiers in Corpus Annotations II*, pages 29–36. Morristown, NJ, USA: Association for Computational Linguistics.

Forbes, K., E. Miltsakak, R. Prasad, A. Sarkar, A. Joshi, and B Webber. 2003. D-LTAG system: Discourse parsing with a lexicalized tree-adjoining grammar. *Journal of Logic, Language and Information* 12:261–279(19).

Forbes-Riley, Katherine, Bonnie Webber, and Aravind Joshi. 2006. Computing discourse semantics: The predicate-argument semantics of discourse connectives in D-LTAG. *Journal of Semantics* 23(1):55–106.

Halliday, M. A. K. and Ruqaiya Hasan. 1976. *Cohesion in English*, vol. 9. London: Longman.

Hirschberg, Julia and Diane Litman. 1993. Empirical studies on the disambiguation of cue phrases. *Computational Linguistics* 19(3):501–530.

Hockenmaier, Julia and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of Third International Conference on Language Resources and Evaluation*, pages 1974–1981.

Hockenmaier, Julia and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics* 33(3):355–396.

Horn, L. R. 1969. A presuppositional analysis of only and even. *CSL* 5:98–107.

Hovy, Eduard H. 1988. Planning coherent multisentential text. In *Proceedings of the 26th annual meeting on Association for Computational Linguistics*, pages 163–169. Morristown, NJ, USA: Association for Computational Linguistics.

Kamp, Hans and Uwe Reyle. 1993. *From Discourse to Logic*. Kluwer.

König, Ekkehard. 1991. *The Meaning of Focus Particles :A Comparative Perspective*. London; New York: Routledge.

Krifka, Manfred. 1998. Additive particles under stress. In D. Strolovitch and A. Lawson, eds., *Proceedings of SALT VIII*, pages 111–129. Ithaca, NY: CLC Publications.

Kruijff, Geert-Jan M. 2001. *A Categorial Modal Architecture of Informativity: Dependency Grammar Logic & Information Structure*. Ph.D. thesis, Charles University.

Lavoie, B. and O. Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the 5th. Conference on Applied Natural Language Processing*, pages 265–268. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Mann, William C. and Maite Taboada. 2010. Rhetorical structure theory. `http://www.sfu.ca/rst/`.

Mann, William C. and Sandra A. Thompson. 1987. Rhetorical structure theory: A theory of text organization. Tech. Rep. ISI/RS-87-190, University of Southern California Information Sciences Instuture.

Mann, William C. and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Towards a functional theory of text organization. *TEXT* 8(3):243–281.

Nakatsu, Crystal. 2008. Learning contrastive connectives in sentence realization ranking. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, pages 76–79. Columbus, Ohio: Association for Computational Linguistics.

Polanyi, Livia. 1988. A formal model of the structure of discourse. *Journal of Pragmatics* 12:601–638.

Polanyi, Livia and Martin H. van den Berg. 1996. Discourse structure and discourse interpretation. In *Proceedings of the 10th Amsterdam Colloquium*, pages 113–131. University of Amsterdam, The Netherlands.

Power, R., C. Doran, and D. Scott. 1999. Generating embedded discourse markers from rhetorical structure. In *Proc. EWNLG-99*.

Power, Richard, Donia Scott, and Nadjet Bouayad-Agha. 2003. Document structure. *Computational Linguistics* 29(2):211–260.

Prasad, R, E Miltsakaki, N Dinesh, A Lee, A Joshi, L Robaldo, and B Webber. 2008. The penn discourse treebank 2.0 annotation manual. Technical Report IRCS-08-01, Institute for Research in Cognitive Science, University of Pennsylvania.

Roberts, Craige. Forthcoming. Only, presupposition and implicature. to appear in the Journal of Semantics; the most recent 2006 manuscript is available from http://ling.osu.edu/∼croberts/only.pdf.

Rooth, Mats. 1985. *Association with Focus*. Ph.D. thesis, UMass, Amherst.

Rooth, Mats. 1992. A theory of focus interpretation. *Natural Language Semantics* 1:75–116.

Scott, Donia R. and Clarisse Sieckenius de Souza. 1990. Getting the message across in RST-based text generation. In *Current research in natural language generation*, pages 47–73. San Diego, CA, USA: Academic Press Professional, Inc. ISBN 0-12-200735-2.

Soricut, Radu and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of HLT-NAACL 2003*, pages 149–156. Edmonton.

Steedman, Mark. 2000. *The Syntactic Process*. MIT Press.

Steedman, Mark J. and Jason Baldridge. 2009. Combinatory Categorial Grammar. In R. Borsley and K. Borjars, eds., *Constraint-based approaches to grammar: alternatives to transformational syntax*. Oxford: Blackwell. To appear.

Striegnitz, Kristina. 2004. *Generating Anaphoric Expressions — Contextual Inference in Sentence Planning*. Ph.D. thesis, University of Saalandes & Universit de Nancy.

Walker, M., A. Stent, F. Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research (JAIR)* 30:413–456.

Walker, Marilyn A., Owen C. Rambow, and Monica Rogati. 2002. Training a sentence planner for spoken dialogue using boosting. *Computer Speech and Language* 16:409–433.

Walker, M. A., S. J. Whittaker, A. Stent, P. Maloor, J. D. Moore, M. Johnston, and G Vasireddy. 2004. Generation and evaluation of user tailored responses in multimodal dialogue. *Cognitive Science* 28(5):811–840.

Webber, Bonnie. 2004. D-LTAG: Extending lexicalized TAG to discourse. *Cognitive Science* 28(5):751–779.

Webber, Bonnie. 2006. Accounting for discourse relations: Constituency and dependency. In *Intelligent Linguistic Architectures*, pages 339–360. CSLI Publications.

Webber, Bonnie, Alistair Knott, Matthew Stone, and Aravind K. Joshi. 1999. What are little texts made of? A structural and presuppositional account using lexicalised TAG. In *Proceedings of International Workshop on Levels of Representation in Discourse (LORID'99)*. Edinburgh, UK.

Webber, Bonnie, Matthew Stone, Aravind Joshi, and Alistair Knott. 2003. Anaphora and discourse structure. *Computational Linguistics* 29(4).

White, Michael. 2004. Reining in CCG Chart Realization. In *Proc. INLG-04*.

White, Michael. 2006a. CCG chart realization from disjunctive logical forms. In *Proc. INLG-06*.

White, Michael. 2006b. Efficient Realization of Coordinate Structures in Combinatory Categorial Grammar. *Research on Language and Computation* 4(1):39–75.

White, Michael and Jason Baldridge. 2003. Adapting Chart Realization to CCG. In *Proc. EWNLG-03*.

Williams, Sandra and Ehud Reiter. 2008. Generating basic skills reports for low-skilled readers*. *Nat. Lang. Eng.* 14(4):495–525.

Wolf, Florian and Edward Gibson. 2005. Representing discourse coherence: A corpus-based study. *Computational Linguistics* 31(2):249–288.