Computational Syntax-Semantics Interface with Type-Theory of Acyclic Recursion for Underspecified Semantics

Roussanka Loukanova
Stockholm University
loukanova@gmail.com

Abstract

The paper provides a technique for algorithmic syntax-semantics interface in computational grammar with underspecified semantic representations of human language. The technique is introduced for expressions that contain NP quantifiers, by using computational, generalised Constraint-Based Lexicalised Grammar (GCBLG) that represents major, common syntactic characteristics of a variety of approaches to formal grammar and natural language processing (NLP). Our solution can be realised by any of the grammar formalisms in the CBLG class, e.g., Head-Driven Phrase Structure Grammar (HPSG), Lexical Functional Grammar (LFG), Categorial Grammar (CG). The type-theory of acyclic recursion $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, provides facility for representing major semantic ambiguities, as underspecification, at the object level of the formal language of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, without recourse of meta-language variables. Specific semantic representations can be obtained by instantiations of underspecified $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-terms, in context. These are subject to constraints provided by a newly introduced feature-structure description of syntax-semantics interface in GCBLG.

1 Introduction

Ambiguity permeates human language, in all of its manifestations, by interdependences, across lexicon, syntax, semantics, discourse, context, etc. Alternative interpretations may persist even when specific context and discourse resolve or discard some specific instances in syntax and semantics. We present computational grammar that integrates lexicon, syntax, types, constraints, and semantics. The formal facilities of the grammar have components that integrate syntactic constructions with semantic representations. The syntax-semantic interface, internally in the grammar, handles some ambiguities as phenomena of underspecification in human language.

We employ a computational grammar, which we call Generalised Constraint-Based Lexicalised Grammar (GCBLG). The formal system GCBLG uses feature-value descriptions and constraints in a grammar with a hierarchy of dependent types, which covers lexicon, phrasal structures, and semantic representations. In GCBLG, for the syntax, we use feature-value descriptions, similar to that in Sag et al. (2003), which are presented formally in Loukanova (2017a) as a class of formal languages designating mathematical structures of functional domains of linguistics information. GCBLG is a generalisation from major lexical and syntactic facilities of frameworks in the class of Constraint-Based Lexicalist Grammar (CBLG) approaches. To some extend, this is reminiscence of Vijay-Shanker and Weir (1994). We lift the idea of extending classic formal grammars to cover semantic representations with semantic underspecification via syntax-semantics interface within computational grammar.

We introduce the technique here for varieties of grammar formalisms from the CBLG approach, in particular: Categorial Grammar (CG), e.g., see Moortgat (1996); Head-Driven Phrase Structure Grammar (HPSG), e.g., see Pollard and Sag (1994); Lexical Functional Grammar (LFG), e.g., see Bresnan (2001), Dalrymple (2001), Kroeger (2004); Tree-Adjoining Grammar (TAG), e.g., see Joshi et al. (1975); Joshi (1987); and other grammar approaches, such as the Grammatical Framework GF, see Ranta (2011). The valance features that we use here with corresponding semantic representations can be translated directly into HPSG, LFG, and Categorial Grammar (CG).

Proceedings of the IWCS 2019 Workshop on Computing Semantics with Types, Frames and Related Structures, pages 37–48
Gothenburg, Sweden, May 24, 2019.

The grammar rules and constraints in GCBLG, in syntax and lexicon, carry semantic representations. The formal language of the semantic representations is a specialised feature-value encoding of terms of the formal language of acyclic recursion $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, see Moschovakis (2006).

2 Overview of Moschovakis Type-Theory of Acyclic Recursion

2.1 Syntax of L^{λ}_{ar}

$\mathsf{Types}_{\mathrm{L}^{\lambda}_{\mathrm{ar}}}$ is the smallest set defined recursively, e.g., by presenting the rules in Backus-Naur form:

$$\tau ::= \mathsf{e} \mid \mathsf{t} \mid \mathsf{s} \mid (\tau \to \tau) \tag{Types}$$

Typed Vocabulary of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$: For each type $\tau \in \mathsf{Types}$, $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ has typed constants, and variables.
Constants K: a denumerable (e.g., finite) set, $K_\tau = \{\mathsf{c_0}^\tau, \ldots, \mathsf{c}_k^\tau\}$, $K = \bigcup_\tau K_\tau$
Pure variables PV: a denumerable set: $\mathsf{PV}_\tau = \{\mathsf{v_0}, \mathsf{v_1}, \ldots\}$; $\mathsf{PV} = \bigcup_\tau \mathsf{PV}_\tau$
Recursion (memory) variables RV: a denumerable set: $\mathsf{RV}_\tau = \{\mathsf{r_0}, \mathsf{r_1}, \ldots\}$; $\mathsf{RV} = \bigcup_\tau \mathsf{RV}_\tau$
The sets of constants and variables of both kinds are mutually distinct: $K \neq \mathsf{RV} \neq \mathsf{PV}$.

Terms of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$: The set of the terms of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ is $\mathsf{Terms} = \cup_{\tau \in \mathsf{Types}} \mathsf{Terms}_\tau$, where for every $\tau \in \mathsf{Types}$, the terms in Terms_τ are defined recursively as follows:

- *Constants*: If $\mathsf{c} \in K_\tau$, then $\mathsf{c} \in \mathsf{Terms}_\tau$, denoted by $c : \tau$ and c^τ
- *Variables*: If $x \in \mathsf{PV}_\tau \cup \mathsf{RV}_\tau$, then $x \in \mathsf{Terms}_\tau$, denoted by $x : \tau$ and x^τ
- *Application Terms*: If $A \in \mathsf{Terms}_{(\sigma \to \tau)}$ and $B \in \mathsf{Terms}_\sigma$, then $A(B) \in \mathsf{Terms}_\tau$, denoted by $A(B) : \tau$ and $[A(B)]^\tau$
- *λ-abstraction terms:* If $x \in \mathsf{PV}_\sigma$, and $A \in \mathsf{Terms}_\tau$, then $\lambda(x)(A) \in \mathsf{Terms}_{(\sigma \to \tau)}$, denoted by $\lambda(x)(A) : (\sigma \to \tau)$ and $[\lambda(x)(A)]^{(\sigma \to \tau)}$
- *Recursion terms:* For any $n \geq 0$, if $A_i \in \mathsf{Terms}_{\sigma_i}$ $(i = 0, \ldots, n)$ and $p_i \in \mathsf{RV}_{\sigma_i}$ $(i = 1, \ldots, n)$ are such that $p_1, \ldots, p_n$ are pairwise different, and the sequence $\{\, p_1 := A_1, \ldots, p_n := A_n \,\}$ satisfies the Acyclicity Constraint, i.e., it is *acyclic*, then A_0 where $\{p_1 := A_1, \ldots, p_n := A_n\} \in \mathsf{Terms}_{\sigma_0}$. The type assignment of the recursion term is denoted by:

$$A_0 \text{ where } \{\, p_1 := A_1, \ldots, p_n := A_n \,\} : \sigma_0 \tag{1a}$$

$$[A_0 \text{ where } \{\, p_1 := A_1, \ldots, p_n := A_n \,\}]^{\sigma_0} \tag{1b}$$

Acyclicity Constraint (AC): the sequence of assignments $\{p_1 := A_1, \ldots, p_n := A_n\}$ is *acyclic* iff there is a function $\mathsf{rank} : \{p_1, \ldots, p_n\} \longrightarrow \mathbb{N}$ such that, for all $p_i, p_j \in \{p_1, \ldots, p_n\}$,

$$\text{if } p_j \text{ occurs freely in } A_i \text{ then } \mathsf{rank}(p_j) < \mathsf{rank}(p_i) \tag{2}$$

We use the meta-symbol $\equiv$ for identity between expressions, e.g., $E_1 \equiv E_2$, and in abbreviations.

The sets $\mathsf{FreeV}(A)$ and $\mathsf{BoundV}(A)$, respectively of the free and bound variables of a term A, are defined by structural recursion, in the usual way, with the exception of the recursion terms. For any given recursion term $A \equiv [A_0 \,\mathsf{where}\, \{p_1 := A_1, \ldots, p_n := A_n\}]$, all occurrences of $p_1, \ldots, p_n \in \mathsf{RV}$ in A are bound, and all other free (bound) occurrences of variables (constants) in $A_0, \ldots, A_n$ are also free (bound) in A.

The reduction calculus of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ has a set of reduction rules that reduce each $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-term A to its unique, up to congruence, canonical form $\mathsf{cf}(A)$, i.e., $A \Rightarrow \mathsf{cf}(A)$. Informally, for every $A, B \in \mathsf{Terms}$:

$A \approx B$ iff (1) A and B are proper terms and their denotations are equal and computed by the same algorithm determined by $\mathsf{cf}(A)$ and $\mathsf{cf}(B)$; (2) or, A and B are immediate, and A and B have the same denotations.

See Moschovakis (2006) and Loukanova (2016, 2019, 2018), for details on the denotational and algorithmic semantics of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$. For the first representation of semantic underspecification with the theory of acyclic algorithms, see Loukanova (2007).

3 Underspecified Terms and Universal Syntax

Definition 1 (Underspecified $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-Terms)**.** *For any $A \in \mathsf{Terms}$, we call A an* underspecified term*, in case $\mathsf{FreeV}(A) \cap \mathsf{RV} \neq \varnothing$, i.e., when A has free occurrences of recursion variables; otherwise A is* specified*.*

We represent some of the semantic ambiguities of natural language sentences by rendering them into underspecified $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-terms. In this paper, we consider a class of typical ambiguous sentences, which have different scope readings, due to occurrences of multiple quantifier NPs in them. For any such sentence Φ, direct representation of alternative readings, by a set of different $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-terms, is available, e.g., $\Phi \xrightarrow{\mathsf{render}} A_i$, $i = 1, \ldots, n$, for some $n \geq 1$. Without any specific context, all of these alternative readings can be potentially viable. Instead of rendering Φ into the set of these specific terms, e.g., by some (syntactic or other) analyses, we render Φ into a single, underspecified term A that represents the set of the alternatives. When context information is available, e.g., by data driven methods, individual or all A_i can be derived from A. That can be done by instantiating the free recursion variables of A, and thus instantiating A, via expanding A by adding recursion assignments that bind its free recursion variables. We impose constraints over the free recursion variables $\mathsf{FreeV}(A) \cap \mathsf{RV}$, e.g., due to the syntactic structure of A, via syntax-semantic analysis, to prevent undesirable alternative instantiations and denotational interpretations $\mathsf{den}^{\mathfrak{A}}(A)(g)$.

Informally, we can restrict the instantiations of A by constraints over possible bindings of recursion variables that occur in A. For any given $A, R \in \mathsf{Terms}$ and $p \in \mathsf{RV}$, the relation $(R\ \mathsf{rBind}\ p)$ holds between R and p in A, when R *recursively binds* p in A, via a sequence of recursion assignments and/or by λ-abstraction across recursion assignments. Thus, rBind provides *specification relation* between underspecified and specified $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-terms. The formal treatment of rBind, which is not in the subject of this paper, is based on the binding relation introduced in Loukanova (2017b). Here, we focus on the technique of rendering natural language expressions into underspecified terms, via syntax-semantics interface.

For example, the terms in (3d)–(3f) render the sentence "Fido barks", via unordered, i.e., abstract, universal syntax. (3c) renders the verb "barks", which is of lexical type *verb* in the lexicon, to the $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-term T_b. T_b has in its where-assignment, the term $barks : (\widetilde{\mathsf{e}} \to \widetilde{t})$, which is not a constant, but a complex term that caries information about time in relation to possible, underspecified time of potential utterance, see Loukanova (2011b). While the term $T_b : \widetilde{t}$ is of sentential type, it is unsaturated because it has a free recursion variable p that fills the argument slot of b, and which is without any constraint over it, regarding possible binding of p.

- NP word in lexicon; semantically specified:

$$[\text{Fido}]_{\text{NP}} \xrightarrow{\mathsf{render}} T_i \equiv i \ \mathsf{where}\ \{\, i := \mathit{fido} \,\} : \widetilde{e} \tag{3a}$$

- V lexeme in lexicon; rendered to a constant; semantically specified:

$$[\text{bark}]_{\text{V}} \xrightarrow{\mathsf{render}} T^{(\widetilde{\mathsf{e}} \to \widetilde{t})}_{lex-b} \equiv \mathit{bark} : (\widetilde{\mathsf{e}} \to \widetilde{t}) \tag{3b}$$

- Inflected V word in lexicon; semantically underspecified term carrying information about time; unsaturated for $p \in \mathsf{FreeV}(T^{\widetilde{t}}_b)$; it becomes constrained VP in the sentence analysis:

$$[\text{barks}]_{\text{V}} \xrightarrow{\mathsf{render}} T^{\widetilde{t}}_b \equiv b(p) \ \mathsf{where}\ \{\, b := \mathit{barks} \,\} : \widetilde{t} \tag{3c}$$

- Sentence: universal, unordered syntax with SynSem; semantically underspecified; SynSem constrained:

$$\{\, [\text{Fido}]_{\text{NP}}, [\text{barks}]_{\text{VP}} \,\}_{\text{S}} \xrightarrow{\mathsf{render}} T_1 \equiv b(p) \ \mathsf{where}\ \{\, i := \mathit{fido},\ b := \mathit{barks} \,\} : \widetilde{t} \ \ \mathsf{such\ that}\ \{\, (i\ \mathsf{rBind}\ p) \,\} \tag{3d}$$

- Universal, unordered syntax with SynSem; semantically specified by the SynSem constraint:

$$\{\, [\text{Fido}]_{\text{NP}}, [\text{barks}]_{\text{VP}} \,\}_{\text{S}} \xrightarrow{\mathsf{render}} T^{1}_{\text{univ}} \equiv b(p) \ \mathsf{where}\ \{\, p := i,\ i := \mathit{fido},\ b := \mathit{barks} \,\} \tag{3e}$$

- Semantically specified by the SynSem constraint; reduced chain assignments:

$$\{\, [\text{Fido}]_{\text{NP}}, [\text{barks}]_{\text{VP}} \,\}_{\text{S}} \xrightarrow{\mathsf{render}} T \equiv T_{\text{univ}} \equiv b(p) \ \mathsf{where}\ \{\, p := \mathit{fido},\ b := \mathit{barks} \,\} \tag{3f}$$

- Surface ordered syntax with SynSem; semantically specified by the SynSem constraint:

$$\{\,[\text{Fido}]_{\text{NP}}, [\text{barks}]_{\text{VP}}\,\}_{\text{S}} \xrightarrow{\mathsf{render}} T \equiv T_{\text{univ}} \equiv b(p) \text{ where } \{\,p := \mathit{fido},\ b := \mathit{barks}\,\} \tag{3g}$$

An alternative syntax-semantics analysis of the same sentence "Fido barks", can be obtained by using a λ-term $[T'_b]^{(\widetilde{\mathsf{e}}\to\widetilde{t})}$ rendering a VP like "barks", as in (4), the type of which, via $\lambda(x)$-abstraction, reflects that it is of a functional type. This analysis results in the same term for rendering the sentence, as in (3g), but with different intermediate reductions of the canonical forms. The analysis uses intermediate steps with γ^*-reduction, or, alternatively, γ-reduction, introduced in Loukanova (2019, 2018) for canonical forms cf_{γ^*} and cf_{γ}, correspondingly. These reduction extensions of the classic reduction from Moschovakis (2006), provide computational simplifications in many cases, like this one.

- Inflected V word of type *verb* in lexicon; $p' \in \mathsf{FreeV}(T'_b)$, $p'(x)$ designates an actant; semantically underspecified:

$$[\text{barks}]_{\text{VP}} \xrightarrow{\mathsf{render}} [T'_b]^{(\widetilde{\mathsf{e}}\to\widetilde{t})} \equiv [\lambda(x)(b(p'(x))) \text{ where } \{\,b := \mathit{barks}\,\}] : (\widetilde{\mathsf{e}} \to \widetilde{t}) \tag{4}$$

In (5a)–(5e), we present stages of syntax-semantics analysis, of a sentence with one quantifier NP and an intransitive verb, via unordered, i.e., abstract, universal syntax-semantics interface. (5b) renders the verb "barks", which is of lexical type *verb* (a subtype of the type *word*) in the lexicon, to the $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-term T_b. The term T_b has in its where-assignment, the term $\mathit{barks} : (\widetilde{\mathsf{e}} \to \widetilde{t})$, which is not a constant, but a complex term that caries information about time in relation to possible time of a potential utterance, see Loukanova (2011b). While the term $T_b : \widetilde{t}$ is of sentential type, it is unsaturated because it has a free recursion variable p that fills the argument slot of b, and which is without any constraint over it, regarding possible binding of p. The syntactic structure of the sentence saturates the syntactic argument of the VP, $[\text{barks}]_{\text{VP}}$, with the subject NP, e.g., $[\text{every dog}]_{\text{NP}}$.

- NP phrase, in grammar with SynSem; semantically specified:

$$\begin{aligned}[\text{Every, dog}]_{\text{NP}} &\xrightarrow{\mathsf{render}} [K]^{((\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}})\to\widetilde{\mathsf{t}})} \\ &\equiv [Q(d)]^{((\widetilde{\mathsf{e}}\to\widetilde{\mathsf{t}})\to\widetilde{\mathsf{t}})} \text{ where } \{\,Q := \mathit{every},\ d := \mathit{dog}\,\}\end{aligned} \tag{5a}$$

- Inflected V word in lexicon; semantically underspecified term carrying information about time; unsaturated for $p \in \mathsf{FreeV}(T^{\widetilde{t}}_b)$; it becomes constrained VP in the sentence analysis:

$$[\text{barks}]_{\text{VP}} \xrightarrow{\mathsf{render}} [T_b]^{\tilde{\mathsf{t}}} \equiv b(p) \text{ where } \{\,b^{(\tilde{\mathsf{e}}\to\tilde{\mathsf{t}})} := \mathit{barks}^{(\tilde{\mathsf{e}}\to\tilde{\mathsf{t}})}\,\} : \tilde{\mathsf{t}} \tag{5b}$$

- Sentence: universal, unordered syntax with SynSem; semantically underspecified; SynSem constrained:

$$\begin{aligned}\{\,[\text{Every dog}]_{\text{NP}}, [\text{barks}]_{\text{VP}}\,\}_{\text{S}} \xrightarrow{\mathsf{render}} [T]^{\tilde{\mathsf{t}}} \equiv i \text{ where } \{ \\ k := Q(d),\ Q := \mathit{every},\ d := \mathit{dog}, \\ h := b(p),\ b := \mathit{barks}\,\} \\ \text{such that } \{\,(k \ \mathsf{rBind}\ p)\,\}\end{aligned} \tag{5c}$$

- Universal, unordered syntax with SynSem; semantically specified by SynSem constraint:

$$\begin{aligned}\xrightarrow{\text{process}} [T]^{\tilde{\mathsf{t}}} \equiv i \text{ where } \{\, i := k(\lambda(x)(h(x))) \approx k(h), \\ k := Q(d),\ Q := \mathit{every},\ d := \mathit{dog}, \\ h := \lambda(x)(b(p'(x))),\ p' := \lambda(x)(x),\ b := \mathit{barks}\,\} \\ \text{such that } \{\,(k \ \mathsf{rBind}\ p')\,\}\end{aligned} \tag{5d}$$

- Surface ordered syntax with SynSem; semantically specified by the SynSem constraint:

$$\begin{aligned}\{\,[\text{Every dog}]_{\text{NP}}\ [\text{barks}]_{\text{VP}}\,\}_{\text{S}} \xrightarrow{\mathsf{render}} [T]^{\tilde{\mathsf{t}}} \equiv i \text{ where } \{\, i := k(\lambda(x)(h(x))) \approx k(h), \\ k := Q(d),\ Q := \mathit{every},\ d := \mathit{dog}, \\ h := \lambda(x)(b(p'(x))),\ p' := \lambda(x)(x),\ b := \mathit{barks}\,\} \\ \text{such that } \{\,(k \ \mathsf{rBind}\ p')\,\}\end{aligned} \tag{5e}$$

Note that the tree structures of the syntactic analyses in Sect. 6 have unordered daughters, and in essence, these are three demential graphs.

4 Syntax-Semantics Interface in GCBLG by the Type-Theory of Acyclic Recursion

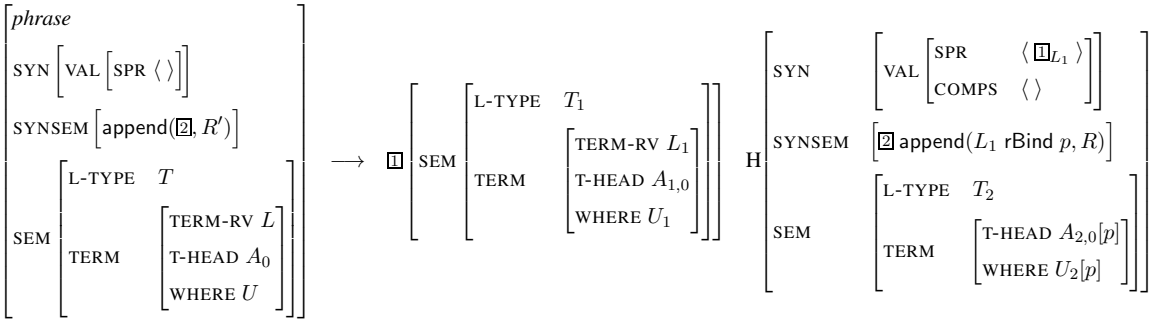
A formal background on generalised GCBLG is given in Loukanova (2017a). We use a formal feature-value language for type-theoretical descriptions of computational syntax of human language. Sag et al. (2003) is a detailed introduction to formal grammar of human language, by providing and using theoretical linguistics. In this version of generalised GCBLG, semantic representations of human language expressions, are provided by $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-terms of the formal language of acyclic recursion $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, by using feature-value structures, via the technique introduced in Loukanova (2011a). Here, we provide two of the major grammatical rules of GCBLG, enhanced with semantic representation, and a new, additional feature-value description on constraints over semantic representations, via syntax-semantics interface. For this, we introduce a new feature SYNSEM of type *synsem*, with values of type *list-of(propositions)*.

The analysed natural language expressions are rendered into $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-terms, in canonical forms, by using feature-value representations of the recursion terms, according to the following rule:

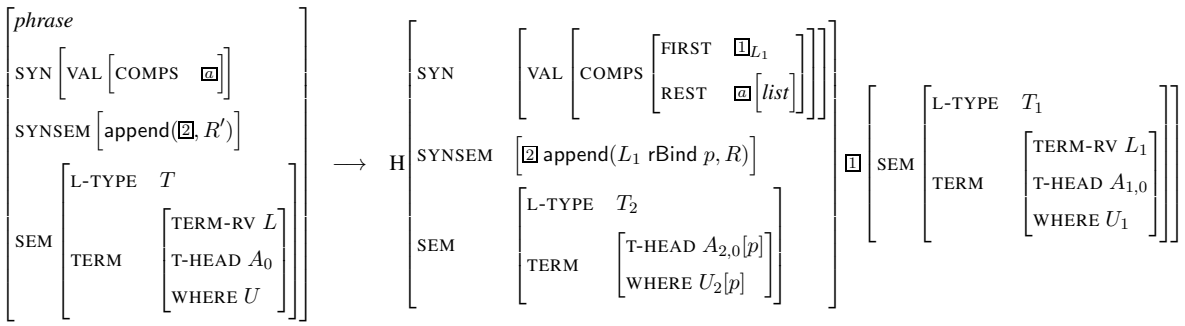
Rendering syntactic structures into $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-terms: Assume that a natural language expression E is analysed by GCBLG as a feature-value description $F(A)$. Assume that, in $F(A)$, the value of the feature T-HEAD is a $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-term A_0, and the value of the feature WHERE is a sequence $\overrightarrow{p} := \overrightarrow{A}$ of where-assignments. Then, $E \xrightarrow{\mathsf{render}} A_0 \text{ where } \{ \overrightarrow{p} := \overrightarrow{A} \}$. The value of the feature L-TYPE is the $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-type of A_0. We use recursion variables to designate the rendering terms in the feature-value descriptions, which can be used in the combined terms.

The rules HSR and HCR1 take as inputs expressions that have semantic representations in canonical forms and generate phrases with semantic representations that are in canonical forms too. The values of T-HEAD and WHERE of the left hand side of the rules are determined by: the semantic types T_1 and T_2 of the daughter nodes; the values of T-HEAD and WHERE in the daughters' feature structures on the right hand side; and the definition of the canonical form $\mathsf{cf}(A)$ of each term A.

Head Specifier Rule (HSR): (6)



Head Complement Rule (HCR1): (7)



In both rules (6)–(7), $p \in \mathsf{RV}$, $A_{2,0}[p]$, $U_2[p]$ indicate that p may fill up some argument slots in some of the sub-terms, and the following cases can be realised:

Case1: $T_i \equiv (\sigma \to \tau)$ and $T_j \equiv \sigma$, for $i, j \in \{1, 2\}$ and $i \neq j$, $T \equiv \tau$

There are two sub-cases, (8) and (9), for the term $[A_0 \ \mathsf{where} \ U]$ in (6)–(7):

(1) if $A_{j,0}$ is immediate, then:

$$A_0 \equiv A_{i,0}(A_{j,0}) \text{ and } U \equiv \mathsf{append}(U_1, U_2) \tag{8}$$

(2) otherwise, i.e., if $A_{j,0}$ is proper, then:

$$A_0 \equiv A_{i,0}(q_0) \text{ and } U \equiv \mathsf{append}(\{q_0 := A_{j,0}\}, \mathsf{append}(U_1, U_2)) \tag{9}$$

Case2: $T_2 : \widetilde{\mathsf{t}}$; and $T_1 : ((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{t}})$ or $T_1 : ((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to ((\widetilde{\mathsf{e}} \to \widetilde{\mathsf{t}}) \to \widetilde{\mathsf{t}}))$

The binding constraints in the value of the feature SYNSEM provide suitable terms for A_0 and U, by combining the parts $A_{j,0}, U_j$, for $j = 1, 2$, and adding new constraints, as needed. The detailed specifications use the definition of the relation rBind.

Case2 covers expressions such as NPs, VPs, and sentences S, where some argument slots are bound by sub-terms of the renderings of NP quantifiers, via recursion variables. The technique is exemplified by the analysis of the sentence "Every cat hugs some dog", which represents a general pattern for multiple quantifier scopes. It is presented in Fig. 3.

5 Scope Underspecification and Specification

Scope Underspecification: The feature-value structural description given in Fig. 3 is a pattern of one of the possible ways to represent multiple semantic scopes of quantification. The syntax-semantics interface is provided by the HSR and HCR1 rules of GCBLG, while the formal calculi of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, provides terms for algorithmic semantics. A sentence that has occurrences of multiple quantifier NPs can be rendered into a single $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-term that represents the multiple possibilities simultaneously. It is underspecified, by having constrained free recursion variables filling up the argument slots that are bound by the corresponding NPs.

Scope Specification: The binding relation rBind in $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ provides a syntax-semantics facility to constrain the possible bindings in specific scopes, via restricting free recursion variables occurring in an underspecified term. An underspecified term can be expanded into specified terms only if they satisfy the rBind-constraints. Thus, constraints expressed by rBind also restrict the possible interpretations of the free recursion variables. For instance, the semantic rendering in Fig. 3 can be expanded into specified semantic representation, which has to satisfy the constraints imposed by the structural combinations. Fig. (4) presents one of the available possibilities to instantiate the free recursion variables, i.e., to bind them by the corresponding NP quantifiers.

6 Grammar Analyses with Syntax-Semantics Interface in GCBLG

In this section, we provide several analyses of typical sentences with head verbs taking NPs as their syntactic, and corresponding semantic, arguments. We exemplify the classes of intransitive and transitive head verbs having NP subjects and single NP complements.

Note that in the analysis in Fig. 1, we do not render the NP that is a proper name into a term of a generalised quantifier, since that would introduce unnecessary complexity in the analysis of expressions of this kind. Furthermore, the term $[T_b]^{\tilde{\mathsf{t}}}$ that renders the VP is as in (5b). Optionally, we can use a λ-abstraction term $[T'_b]^{(\widetilde{\mathsf{e}} \to \widetilde{t})}$, and then, the sentence gets rendered to an algorithmically equivalent term by the γ^*-reduction calculus of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$, see Loukanova (2019, 2018).

In Fig. 2, we present a syntax-semantics analysis of a sentence with a single quantifier NP, in the subject position of the intransitive V. Optionally, we can use a λ-abstraction term $[T'_b]^{(\widetilde{\mathsf{e}} \to \widetilde{t})}$, resulting in an algorithmically equivalent term for the sentence, by the γ^*-reduction calculus of $\mathrm{L}^{\lambda}_{\mathrm{ar}}$.

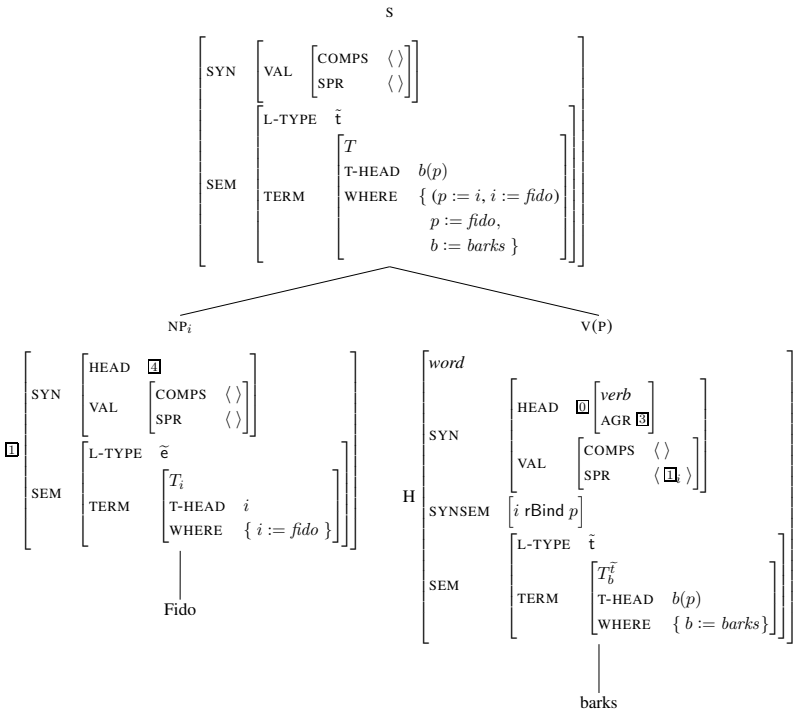


Figure 1: A Proper Name as the Specifier of an Intransitive Verb

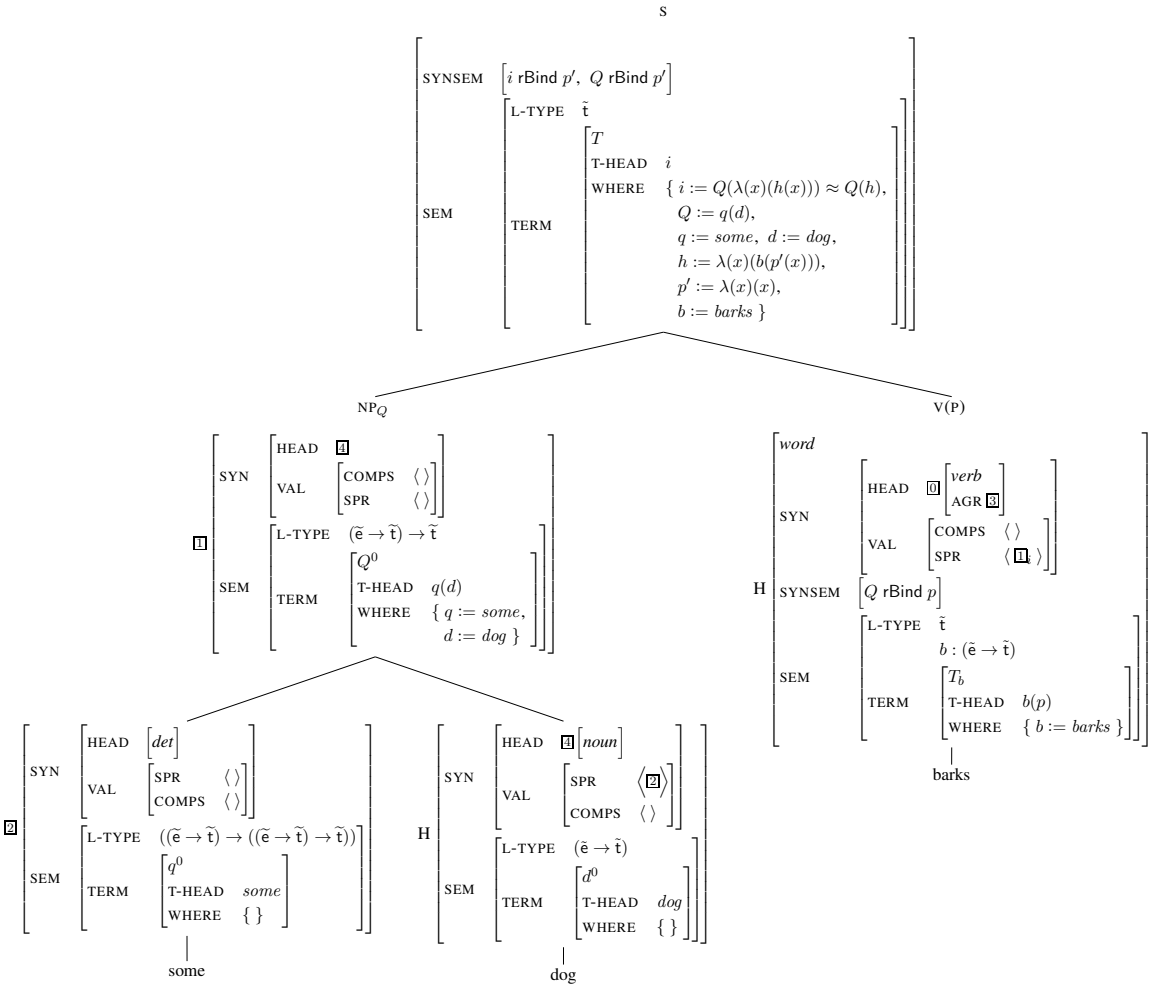


Figure 2: Quantifier NP in Subject Position, Specified

In Fig. 3, we present a syntax-semantics analysis of a sentence with two quantifier NPs, in subject and compliment positions of the head verb V. In Fig. 4, we present its specification to one of the alternative scope distributions. In it, we have specified the VP at the intermediate level of the analysis, in the node marked by (n_1) VP. Pragmatically, it is more viable for this node to be underspecified as it is in 3, and the specification is at the node (n_0) S, at the sentence level, e.g., when disambiguating information is available at that level.

In Fig. 5, we present an optional analysis of the same sentence, by rendering the head verb to a λ-term, which is congruent to $\lambda(x_{d_2})\lambda(x_{d_1})h(p'_1(x_{d_1}))(p'_2(x_{d_2}))$. In such a case, there is a correspondence between the syntactic and semantic saturation types of the V and VP expressions. But, the term that renders the sentence at the node (n_0) S, is not algorithmically equivalent to the one in Fig. 3, and naturally reflect on the algorithmic steps that are used during the analyses for filling up arguments. Similarly, the specification terms corresponding to these two options are not algorithmically equivalent.

7 Conclusions and Future Work

We have presented how the formal language of the theory of acyclic recursion $\mathrm{L}^{\lambda}_{\mathrm{ar}}$ can be used for semantic representations of natural language via syntax-semantics interface in computational grammar. We have introduced the technique by generalised GCBLG that employs feature-value descriptions. GCBLG is type theoretical by its hierarchy of constraints, which are of dependent types, for the syntactic compositions. The feature-value descriptions embed semantic representations by the higher-order $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-terms. We have focused on two of the major grammar rules for saturation of syntactic and semantic arguments, for underspecified semantic representations of multiple quantifier scopes. A sentence that has two (or more) quantifier NPs, with ambiguous semantic scopes, can be rendered into a single, underspecified $\mathrm{L}^{\lambda}_{\mathrm{ar}}$-term A. The key idea is that A has free recursion variables saturating arguments that can be bound by corresponding quantifier NPs in alternative orders. The phrasal rules of GCBLG introduce restrictions over possible bindings via recursion assignments in syntax-semantics interface.

We foresee extending and implementing the technique for computational syntax-semantics interface in lexical and phrasal structures, for broader grammatical constructions.

References

Bresnan, J. (2001). *Lexical-Functional Syntax*. Oxford: Blackwell Publishers.

Dalrymple, M. (2001). *Lexical Functional Grammar*, Volume 34 of *Syntax and Semantics*. New York: Academic Press.

Joshi, A. K. (1987). An introduction to tree adjoining grammars. *Mathematics of Language 1*, 87–115.

Joshi, A. K., L. S. Levy, and M. Takahashi (1975, February). Tree adjunct grammars. *J. Comput. Syst. Sci. 10*(1), 136–163.

Kroeger, P. (2004). *Analyzing Syntax: A Lexical-Functional Approach*. Cambridge: Cambridge University Press.

Loukanova, R. (2007, August). Typed Lambda Language of Acyclic Recursion and Scope Underspecification. In R. Muskens (Ed.), *Workshop on New Directions in Type-theoretic Grammars*, ESSLLI 2007, Dublin, Ireland, pp. 73–89. The Association for Logic, Language and Information.

Loukanova, R. (2011a). Semantics with the Language of Acyclic Recursion in Constraint-Based Grammar. In G. Bel-Enguix and M. D. Jiménez-López (Eds.), *Bio-Inspired Models for Natural and Formal Languages*, pp. 103–134. Cambridge Scholars Publishing.

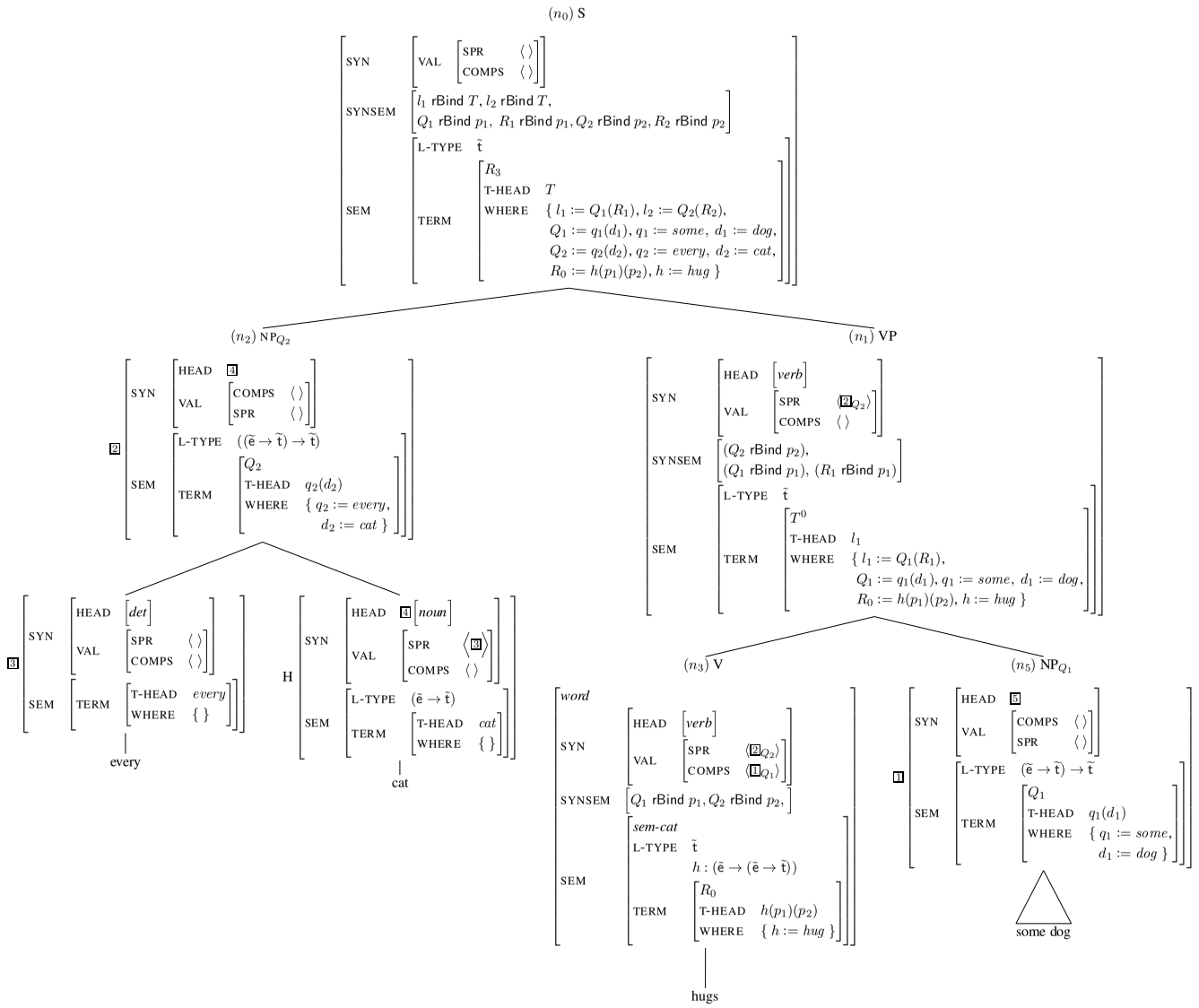


Figure 3: Underspecified Scope: Quantifier NP as Subject and Quantifier NP as Complement

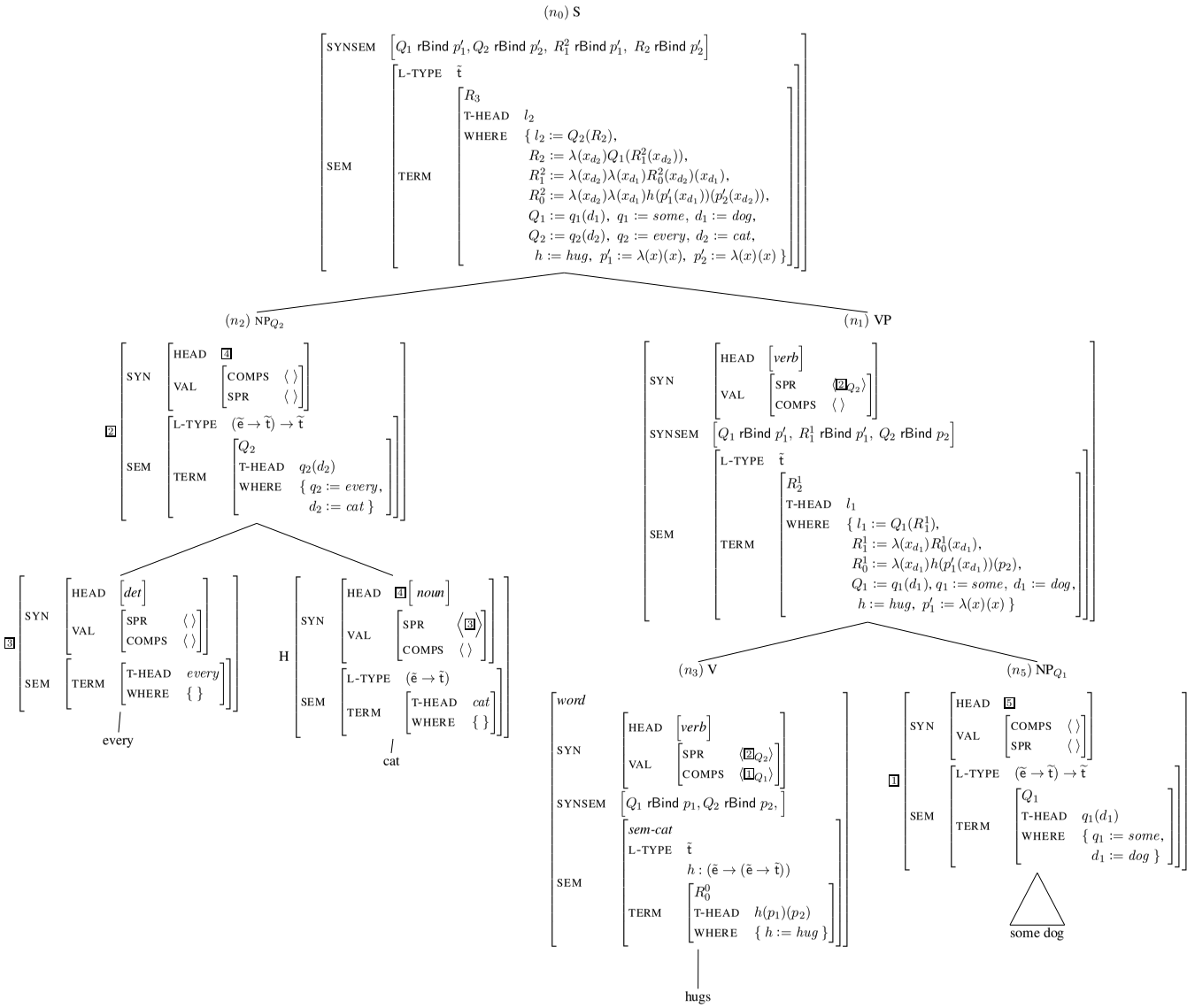


Figure 4: Specification: Quantifier NP as Subject and Quantifier NP as Complement: de dicto

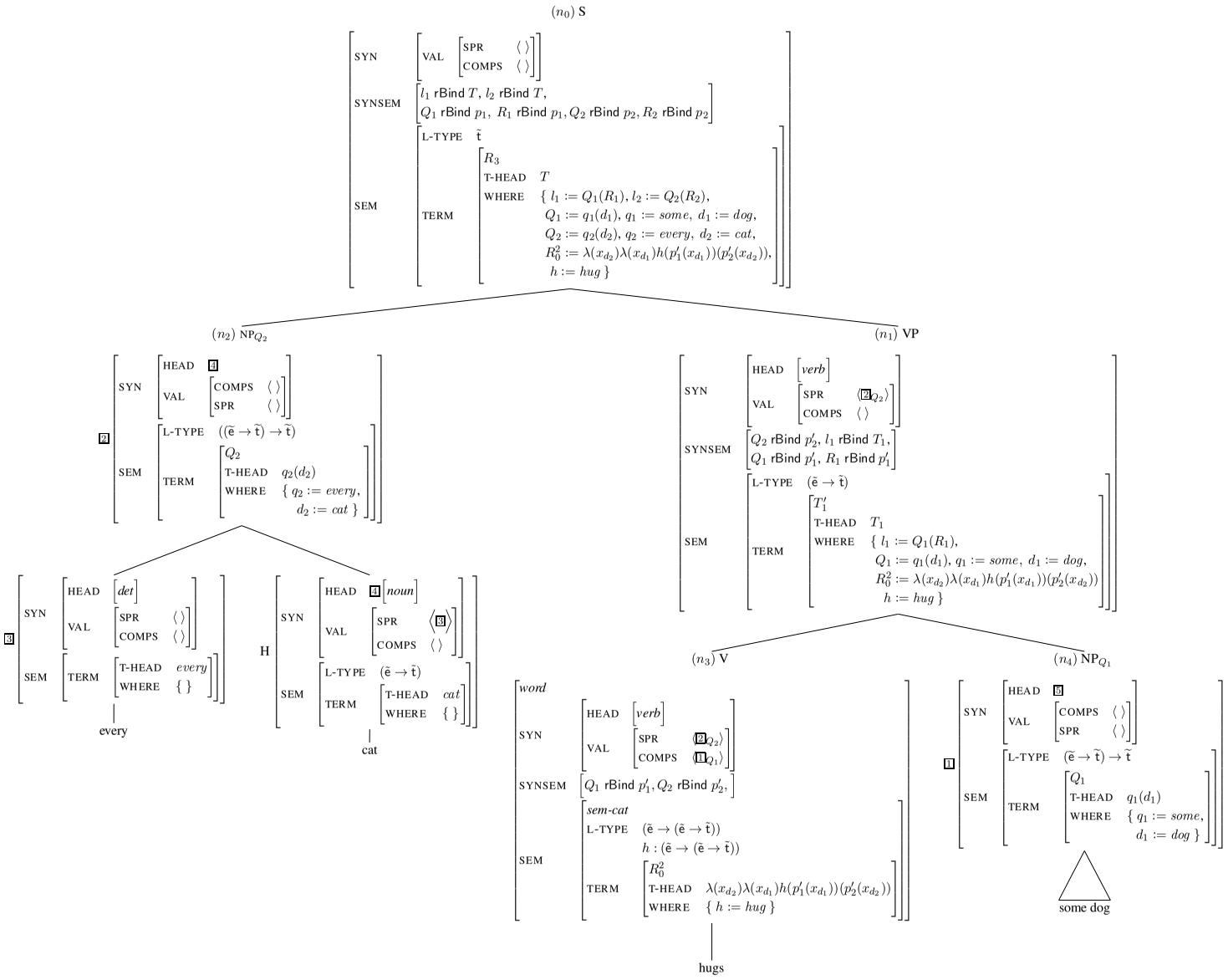


Figure 5: Underspecified Scope: Quantifier NP as Subject and Quantifier NP as Complement: via λ-abstraction

Loukanova, R. (2011b). Syntax-Semantics Interface for Lexical Inflection with the Language of Acyclic Recursion. In G. Bel-Enguix, V. Dahl, and M. D. Jiménez-López (Eds.), *Biology, Computation and Linguistics — New Interdisciplinary Paradigms*, Volume 228 of *Frontiers in Artificial Intelligence and Applications*, pp. 215–236. Amsterdam; Berlin; Tokyo; Washington, DC: IOS Press.

Loukanova, R. (2016). Relationships between Specified and Underspecified Quantification by the Theory of Acyclic Recursion. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal 5*(4), 19–42.

Loukanova, R. (2017a). An Approach to Functional Formal Models of Constraint-Based Lexicalized Grammar (CBLG). *Fundamenta Informaticae 152*(4), 341–372.

Loukanova, R. (2017b). Binding Operators in Type-Theory of Algorithms for Algorithmic Binding of Functional Neuro-Receptors. In M. Ganzha, L. Maciaszek, and M. Paprzycki (Eds.), *Proceedings of the 2017 Federated Conference on Computer Science and Information Systems*, Volume 11 of *Annals of Computer Science and Information Systems*, pp. 57–66. Polish Information Processing Society.

Loukanova, R. (2018). γ-Reduction in Type Theory of Acyclic Recursion. to appear.

Loukanova, R. (2019). Gamma-star canonical forms in the type-theory of acyclic algorithms. In J. van den Herik and A. P. Rocha (Eds.), *Agents and Artificial Intelligence*, Cham, pp. 383–407. Springer International Publishing.

Moortgat, M. (1996). Categorial type logics. In J. van Benthem and A. ter Meulen (Eds.), *Handbook of Logic and Language*, pp. 93–177. Amsterdam: Elsevier.

Moschovakis, Y. N. (2006). A Logical Calculus of Meaning and Synonymy. *Linguistics and Philosophy 29*(1), 27–89.

Pollard, C. and I. A. Sag (1994). *Head-Driven Phrase Structure Grammar*. Chicago, IL: University of Chicago Press.

Ranta, A. (2011). *Grammatical Framework: Programming with Multilingual Grammars*. Stanford: CSLI Publications.

Sag, I. A., T. Wasow, and E. M. Bender (2003). *Syntactic Theory: A Formal Introduction*. Stanford, California: CSLI Publications.

Vijay-Shanker, K. and D. J. Weir (1994, November). The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory 27*(6), 511–546.