# Computing All Quantifier Scopes with CCG

**Miloš Stanojević**
School of Informatics
University of Edinburgh
m.stanojevic@ed.ac.uk

**Mark Steedman**
School of Informatics
University of Edinburgh
steedman@inf.ed.ac.uk

## Abstract

We present a method for computing all quantifer scopes that can be extracted from a single CCG derivation. To do that we build on the proposal of Steedman (1999, 2011) where all existential quantifiers are treated as Skolem functions. We extend the approach by introducing a better packed representation of all possible specifications that also includes node addresses where the specifications happen. These addresses are necessary for recovering all, and only, possible readings.

## 1 Introduction

Quantifiers often introduce a peculiar type of semantic ambiguity. Take for instance the following sentence: Every farmer owns a donkey. This sentence has two readings: a *wide reading* where there is one donkey that all farmers share and *narrow reading* where each farmer has a different donkey. If we express these readings as first-order logic they would look as follows:

*Wide*:
$\exists a\,[donkey'(a) \wedge \forall b\,[farmer'(b) \Rightarrow own'(b,a)]]$
*Narrow*:
$\forall b\,[\exists a\,[donkey'(a) \wedge (farmer'(b) \Rightarrow own'(b,a))]]$

From these formulas it is clear where the name for different readings come from. In the *wide* reading the existential quantifier takes the wide scope i.e. it contains the universal quantifier. In the *narrow* reading the existential quantifier's scope does not cover the universal quantifier.

Any theory of the syntax-semantics interface needs to account for the fact that quantifiers can introduce scope ambiguity. Early approaches to this problem involved either representing the two meanings with distinct logical forms like the above, obtained from the surface string either by treating *every farmer* and *a donkey* as generalized quantifiers or "quantifying in" in either order to a proposition containing distinguished variables (Montague, 1973), or via equivalent structure-changing operations of "quantifier raising" (May, 1985). Later approaches decoupled scope from syntactic derivation by the use of "storage" to pass scope information (Cooper, 1983; Keller, 1988). However, all of these approaches overgenerate unattested readings for certain examples involving coordination, first noted by (Geach, 1970) and considered in section 3 below. The approach of (Steedman, 2011) can be thought of as reuniting a storage-like account with surface-compositional syntactic derivation.

## 2 Computing Scope with CCG

Steedman (1999, 2011) introduces a different view of existential quantifiers, according to which the only true quantifiers are universal quantifiers and that existential quantifiers can be treated as generalized Skolem terms in the following way:

$$\textit{Wide}: \quad \forall b\left[farmer'(b) \Rightarrow own'(b, \mathrm{sk}^{\{\}}_{donkey'})\right]$$
$$\textit{Narrow}: \quad \forall b\left[farmer'(b) \Rightarrow own'(b, \mathrm{sk}^{\{b\}}_{donkey'})\right]$$

Here, $\mathrm{sk}^{\alpha}_{\beta}$ represents the Skolem function whose arguments are variables of type $\alpha$ and whose result is of type $\beta$.

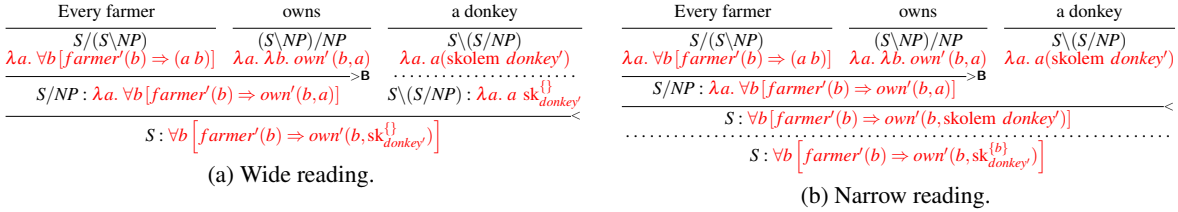
In the wide scope reading $\mathrm{sk}^{\{\}}_{donkey'}$ is a Skolem constant (Skolem function with no arguments). This means that it will produce only a unique value of type $donkey'$, somewhat like a proper name. In the narrow scope reading $\mathrm{sk}^{\{b\}}_{donkey'}$ is a Skolem function that has the variable $b$ bound by the universal as its argument. This function will produce a different value for each $b$, in other words there will be a different $donkey'$ for each $farmer'$.

Other non-universal generalized quantifiers are also treated as Skolem terms. Steedman (2011) also discusses negation which we do not present here, but our approach naturally extends to it. We do not deal with intentionality.

This view of quantifiers allows for a simple



*Proceedings of the 14th International Conference on Computational Semantics*, pages 33–37
June 17–18, 2021. 




(a) Wide reading.

(b) Narrow reading.

Figure 1: Two different readings.

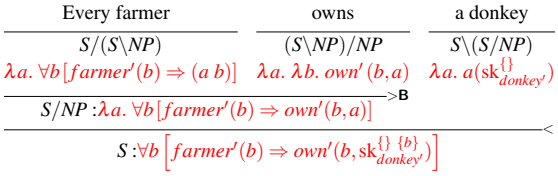



Figure 2: Packed representation.

syntax-semantics interface. CCG derivations for wide and narrow readings are presented in Figure 1, for one of the two derivation trees allowed by CCG. Syntactic component of these two trees is the same, only the semantics differ. Semantic entry for all words are the usual lambda expressions except for the indefinite articles whose entry is $\lambda a.\lambda b.b(\text{skolem } a)$. Here *skolem a* is a *underspecified* Skolem term of type $a$. An underspecified Skolem term becomes a Skolem function/constant when it is *specified*. Skolem specification is marked in the derivation tree with a dotted underline, and influences only the logical form, converting an underspecified Skolem terms by giving it as arguments all universally bound variables into whose scope it has been brought by the derivation so far. In Figure 1a that set is empty, so the result of Skolem specification is a Skolem constant, yielding the wide scope reading. In Figure 1b, that set includes the single variable $b$. By choosing to specify at a different point in the derivation, we get a different narrow-scope reading for the sentence.

In order to prevent overgeneration of unattested readings, we must impose a further rather natural constraint on Skolem specification requiring that any embedded unspecified Skolem terms are specified at the same time in the same environment. Thus we get the following readings for "every farmer owns a donkey that ate a hat":

$\forall b[farmer'(b) \Rightarrow own'(b, \text{sk}^{\{\}}_{\lambda a.donkey'(a) \wedge ate'(a,\text{sk}^{\{\}}_{hat'})})]$

$\forall b[farmer'(b) \Rightarrow own'(b, \text{sk}^{\{b\}}_{\lambda a.donkey'(a) \wedge ate'(a,\text{sk}^{\{b\}}_{hat'})})]$

$\forall b[farmer'(b) \Rightarrow own'(b, \text{sk}^{\{b\}}_{\lambda a.donkey'(a) \wedge ate'(a,\text{sk}^{\{\}}_{hat'})})]$

However, we exclude a fourth reading with a wide-scope Skolem constant donkey eating multiple farmer-dependent hats:[1]

$\#\forall b[farmer(b) \Rightarrow own'(b, \text{sk}^{\{\}}_{\lambda a.donkey'(a) \wedge ate'(a,\text{sk}^{\{b\}}_{hat'})})]$

To ensure that all available readings are obtained, it is inefficient to choose all possible specification points in the derivation, because most of them yield duplicate results where there has been no change in the set of scoping variables. To eliminate such redundancy, Steedman (2011) proposed a *packed representation* presented in Figure 2 where the Skolem term is associated with multiple bindings. At points in the derivation where the binding environment of the function changes, a new argument combination is introduced.

## 3 Problems with Taking Scope over Coordination

The proposal of Steedman (2011) was implemented by Kartsaklis (2010) and it works quite well for examples that we have seen so far. However, coordination poses some challenges for the packed representation. Consider coordination of two universal quantifiers in Figure 3a. Here $NP^{\uparrow}$ is a shorthand for a type-raised $NP$. For a moment ignore additional annotations in the arguments of the Skolem functions. In this example, the specification of *an apple* will either happen before it is combined with the universals, or after. This means that either it will be in the scope of both or none. The only two readings are given in Figure 3b. However, if we were to unpack the packed formula by computing all combinations of Skolem arguments we would get four readings, including the impossible reading of *an apple* being within scope of one universal quantifier but not the other. We stress that this is a problem arising from the packed representation, not the theory of scope itself.

It may look like the solution to this problem is simple: take all Skolems stemming from the

[1] This condition was inadvertently omitted from the original proposal.

same noun phrase and combine their arguments in order i.e. first arguments of both Skolems go together and second arguments of both Skolems go together. While this solves the example in Figure 3, it does not work on that in Figure 4 where we coordinate one universal and one existential quantifier. Here there is no clear correspondence between arguments of two Skolem functions: one of them has two different arguments ($\{\}$ and $\{a\}$) while the other one has only one possible argument ($\{\}$). Of course, in principle the difference in the number of possible combinations could be significantly larger and it may not be clear how to combine them. To solve this problem we need a more principled solution that directly reflects the mechanism of the non-packed derivations.

## 4 Proposed Solution

The packed representation can be seen as a dynamic programming approach to computing all possible orders of specifications of Skolem terms. However, the packed representation of Steedman (2011) that we considered so far is incomplete: from a given packed representation we cannot reconstruct the non-packed representations that are encoded in it. That is caused by the missing information of the location in the tree where the specification was done. We extend the packed representation with this information: whenever a new argument combination is added, together with it we add the Gorn address of the current node. For instance $\mathrm{sk}_{apple'}^{\{\}^{trrl}\ \{a\}^{t}}$ from Figure 4a signifies that there are two possible arguments for this Skolem function: an empty argument list specified at Gorn address $trrrl$ ($top \rightarrow right \rightarrow right \rightarrow right \rightarrow left$) and a non-empty argument $\{a\}$ at address $t$ ($top$). We know that all the Gorn addresses for a given Skolem function will be on a single path from the root of the tree to the determiner that introduced it into derivation. This means that, for a given function, we can sort all addresses by their height in the tree.

Assume we have a Skolem function with $k$ possible argument sets $e_1, e_2, \ldots, e_k$ sorted by the height of their Gorn addresses $g_1, g_2, \ldots, g_k$ such that $g_k$ is closest to the root of the tree. We can say that every argument set $e_i$ corresponds to the specializations done on any node $g$ for which it holds $g_i \leq g < g_{i+1}$.[2] In other words $g$ can be any node between $g_i$ and $g_{i+1}$, including $g_i$ but excluding $g_{i+1}$. If we take again $\mathrm{sk}_{apple'}^{\{\}^{trrl}\ \{a\}^{t}}$ as an example we can say that the argument $\{\}$ corresponds to specialization of Skolem function for nodes $trrl$, $trr$ and $tr$.

Additional important point is that we know for certain that $g_1$ is the address of the leaf of the tree because that is the first point in the derivation where the specification can be done. This is important because in the cases of coordination the logical formula can have copies of the Skolem term that comes from the same noun phrase. We can use the Gorn address of the leaf to identify the Skolem terms that originate in the same noun phrase. Steedman (2011) uses a special index to keep track of this information, but that index is not necessary in our representation due to the existance of Gorn addresses.

Now we can define unpacking of the new version of the packed representation. We will illustrate it with the example packed formula from the top node of Figure 4a: $\forall a \left[man'(a) \Rightarrow eat'\left(a, \mathrm{sk}_{apple'}^{\{\}^{trrl}\ \{a\}^{t}}\right)\right] \wedge eat'\left(\mathrm{sk}_{woman'}^{\{\}^{tlrrl}}, \mathrm{sk}_{apple'}^{\{\}^{trrl}}\right)$

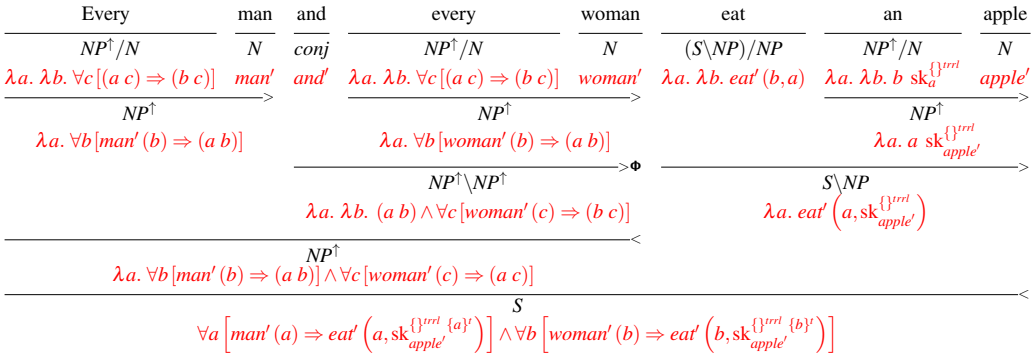
**step 1** Group Skolem terms by the NP they belong to. For that we can use the first Gorn address that specifies the leaf node. In the example that would give $\{\mathrm{sk}_{woman'}^{\{\}^{tlrrl}}\}$ for the first NP and $\{\mathrm{sk}_{apple'}^{\{\}^{trrl}\ \{a\}^{t}}, \mathrm{sk}_{apple'}^{\{\}^{trrl}}\}$ for the second.

**step 2** For each group of the Skolems extract the unique Gorn addresses where specification changes. In this example that would be $\{tlrrl\}$ for the first noun phrase and $\{trrl, t\}$ for the second.

**step 3** Compute the Cartesian product of the sets of Gorn addresses. That will give all possible combinations of specification points. Each combination will correspond to one possible reading of the sentence. In the example that will give $\{(tlrrl, trrl), (tlrrl, t)\}$.

**step 4** To transform each entry to a reading we filter the Skolem arguments by the Gorn address. Let us consider how we extract the reading for entry $(tlrrl, t)$. Filtering arguments for the first noun phrase Skolem term $\{\mathrm{sk}_{woman'}^{\{\}^{tlrrl}}\}$ with $tlrrl$ is easy because there is only one entry that matches it exactly. Filtering arguments for the second noun phrase is more interesting because there are two copies of it. We need to

[2] For simplicity, when $g_k \neq t$ we can consider $g_{k+1} = t$ in order to have a complete coverage to the root of the tree.



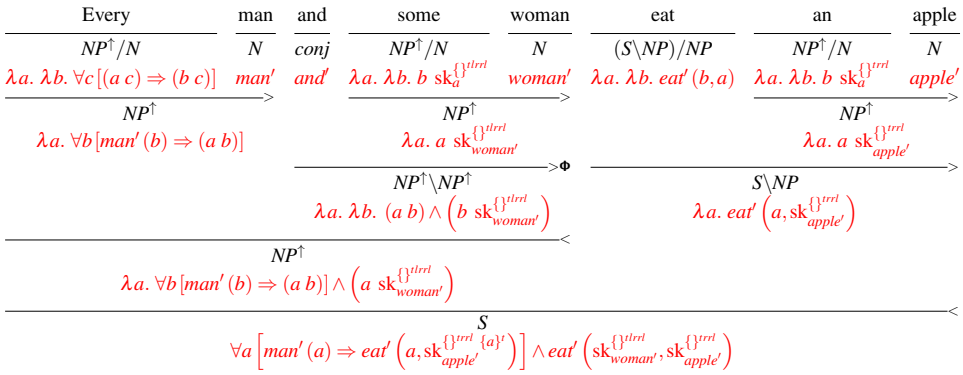

(a) Packed derivation.

$$\forall a \left[ man'(a) \Rightarrow eat'\left(a, \mathrm{sk}_{apple'}^{\{\}^{trrl}}\right)\right] \wedge \forall b \left[ woman'(b) \Rightarrow eat'\left(b, \mathrm{sk}_{apple'}^{\{\}^{trrl}}\right)\right]$$

$$\forall a \left[ man'(a) \Rightarrow eat'\left(a, \mathrm{sk}_{apple'}^{\{a\}^{t}}\right)\right] \wedge \forall b \left[ woman'(b) \Rightarrow eat'\left(b, \mathrm{sk}_{apple'}^{\{b\}^{t}}\right)\right]$$

(b) Readings.

Figure 3: Coordination with two universal quantifiers.




(a) Packed derivation.

$$\forall a \left[ man'(a) \Rightarrow eat'\left(a, \mathrm{sk}_{apple'}^{\{\}^{trrl}}\right)\right] \wedge eat'\left(\mathrm{sk}_{woman'}^{\{\}^{tlrrl}}, \mathrm{sk}_{apple'}^{\{\}^{trrl}}\right)$$

$$\forall a \left[ man'(a) \Rightarrow eat'\left(a, \mathrm{sk}_{apple'}^{\{a\}^{t}}\right)\right] \wedge eat'\left(\mathrm{sk}_{woman'}^{\{\}^{tlrrl}}, \mathrm{sk}_{apple'}^{\{\}^{trrl}}\right)$$

(b) Readings.

Figure 4: Coordination with one universal and one existential quantifier.

select for specification on node $t$. In the first copy $\mathrm{sk}_{apple'}^{\{\}^{trrl}\ \{a\}^{t}}$ we just select argument $\{a\}$ since it corresponds to node $t$. In the second copy $\mathrm{sk}_{apple'}^{\{\}^{trrl}}$ we select for $\{\}$ because it covers all nodes from $trrl$ to the root including $t$.

## 5 Conclusion

This approach is really just a full dynamic programming representation of the unpacked representations that could easily be extracted from this representation. We do not have to explicitly encode all the nodes where specification happens, but only for the places where that specification changes the existing result and we also encode exactly at which places in the tree this happens.

Here we have described how to get all possible readings from a single CCG derivation. However, in some cases there can be alternative CCG derivations that can provide additional readings. To get those readings we can apply the same method on all alternative derivations either by chart parsing, as described in (Steedman, 2011), or by recovering alternative derivations with the *tree-rotation* operation (Niv, 1994; Stanojević and Steedman, 2019)

Evang and Bos (2013) show that there is a strong preference for subject to take scope over object.

Our representation of Skolem terms could be extended to encode the information of the type of noun phrase they originate from. With this extension we could rank the extracted readings by *subject* > *object* preference.

The implementation of our approach is available at https://github.com/stanojevic/CCG-Quantifiers.

## Acknowledgments

This work was supported by ERC H2020 Advanced Fellowship GA 742137 SEMANTAX grant.

## References

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

Donald Davidson and Gilbert Harman, editors. 1972. *Semantics of Natural Language*. Reidel, Dordrecht.

Kilian Evang and Johan Bos. 2013. Scope Disambiguation as a Tagging Task. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Short Papers*, pages 314–320, Potsdam, Germany. Association for Computational Linguistics.

Peter Geach. 1970. A program for syntax. *Synthèse*, 22:3–17. Reprinted as Davidson and Harman 1972:483–497.

Dimitrios Kartsaklis. 2010. Wide-coverage CCG parsing with quantifier scope. Master's thesis, University of Edinburgh.

William Keller. 1988. Nested Cooper storage. In Uwe Reyle and Christian Rohrer, editors, *Natural Language Parsing and Linguistic Theory*, pages 432–447. Reidel, Dordrecht.

Robert May. 1985. *Logical Form*. MIT Press, Cambridge, MA.

Richard Montague. 1973. The Proper Treatment of Quantification in Ordinary English. In K. J. J. Hintikka, J. M. E. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*, pages 221–242. Springer Netherlands, Dordrecht.

Michael Niv. 1994. A Psycholinguistically Motivated Parser for CCG. In *32nd Annual Meeting of the Association for Computational Linguistics*, pages 125–132, Las Cruces, New Mexico, USA. Association for Computational Linguistics.

Miloš Stanojević and Mark Steedman. 2019. CCG Parsing Algorithm with Incremental Tree Rotation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 228–239, Minneapolis, Minnesota. Association for Computational Linguistics.

Mark Steedman. 1999. Alternating Quantifier Scope in CCG. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 301–308, USA. Association for Computational Linguistics.

Mark Steedman. 2011. *Taking Scope: The Natural Semantics of Quantifiers*. The MIT Press.