1 Overview

Continuized **C**ombinatory **C**ategorial **G**rammars supplement traditional applicative categorial grammars with a small number of combinators that allow expressions to take scope over their compositional contexts (e.g. Shan & Barker 2006, Barker & Shan 2008, 2014). Though CCCGs offer robust accounts of in-scope binding (ibid.), the treatment of dynamic binding (i.e. cross-sentential and donkey anaphora) proposed in Barker & Shan 2008 overgenerates (Charlow 2010; see Barker & Shan 2014 for in-depth discussion). Conversely, the related theory of de Groote 2006 (which relies on continuations, but is not a CCG) handles dynamic binding elegantly, but does not offer any general account of scope-taking. In addition, both Barker & Shan 2008 and de Groote 2006 analyze dynamic binding by indefinites, but not by expressions such as *exactly one linguist*, which differs in important ways (cf. Kamp & Reyle 1993).

This paper presents an explicit CCCG account of dynamic binding inspired by the computer science notion of a *monad* (e.g. Moggi 1989, Wadler 1992, 1994, 1995, Shan 2002). Monads were developed as an abstraction for structuring extensions to the pure lambda calculus; they enable us to modularly add *side effects* (cf. Shan 2005) to a programming language or natural language grammar. It is proposed that dynamic binding can be captured by recognizing two kinds of linguistic side effects: **state** and **nondeterminism**. State corresponds to the ability to introduce discourse referents, while nondeterminism means the ability to track multiple computations in parallel, which facilitates a referential treatment of indefinite expressions.

Concretely, a monad determines a pair of combinators $\langle \eta, \star \rangle$, one for wrapping values inside trivially effectful structures, and another for composing programs with potential side effects. Our proposal is to replace the standard lifting and lowering operations of CCCGs with (respectively) $\star$ and η. Combined with some intuitive lexical entries, this has the automatic effect of producing a CCCG that countenances side effects. Crucially, given the underlying continuations-based scaffolding, the problem of integrating well-motivated combinatorial approaches to scope with insights from dynamic semantics reduces to finding a monad for state and nondeterminism. Though we focus on the case of dynamic binding here, any system of effects can be grafted onto a continuized CCG in this way.

2 Continuations

Simplifying somewhat,[1] we take it that a standard CCCG consists of three polymorphic combinators, **Lift**, **Triv**, and **Scope**, in addition to the basic categorial slashes that encode forward and backward function application. These combinators are defined in Figure 1a (along with a characterization of the slashes' behavior). The inference rules make use of a type constructor K parameterized by two concrete types. Formally, $K\,a\,r ::= r/(r/a)$; the r/a argument to something of type $K\,a\,r$ is called its *continuation*, and represents the denotation of its context. Intuitively, expressions with type $K\,a\,r$ behave like things of type a within compositional contexts of type r, the 'result type' of the computation. For example, extensional generalized quantifiers have type $K\,e\,t$, since they behave locally as individuals but quantify over constituents of type t.

As for combinators, **Lift** turns any expression into a function on its *continuation*; it is a polymorphic generalization of the familiar lifting procedure that converts individuals to the corresponding principal ultrafilters (cf. Montague 1974). **Scope** defines the compositional process whereby two continuized expressions are combined, giving the left scope over the right. Finally, **Triv** is a trivial continuation used to delimit contexts of evaluation and expose underlying semantic values. Shan & Barker 2006 show that these three operations suffice to derive cases of inverse scope (that is, where an expression scopes over something to its left).

3 Adding side effects

Continuized grammars need not place inherent restrictions on r, the type of the context. For simple cases where only boolean scope-taking is at-issue (cf. Barwise & Cooper 1981), we can set $r ::= t$. But in general, we may be interested in contexts that denote in richer semantic spaces, as is typically the case with questions, focus, attitude reports, dynamic updates, etc. In such cases, we enrich the return type to match the phenomenon of interest.

We propose to model the sorts of semantic enrichments motivated by these phenomena using monads (Wadler 1994, Liang et al. 1995), which serve much the same purpose in capturing the semantics of programming language *side effects* like IO. A monad is a triple $\langle M, \eta, \star \rangle$ of a type construc-

---

$$\frac{\Gamma \vdash f : b/a \quad \Delta \vdash e : a}{\Gamma \cdot \Delta \vdash f e : b} \, / \qquad \frac{\Delta \vdash e : a \quad \Gamma \vdash f : a\backslash b}{\Delta \cdot \Gamma \vdash f e : b} \, \backslash$$

$$\overline{\varepsilon \vdash \lambda x.\, x : a/a} \ \textbf{Triv} \qquad \overline{\varepsilon \vdash \lambda x k.\, k\, x : K a r / a} \ \textbf{Lift} \qquad \overline{\varepsilon \vdash \textbf{S} : (K b r / K a r) / K (b/a) r} \ \textbf{Scope}$$

(a) Continuized CCG without side effects, fixing a result type r.

$$\frac{\Gamma \vdash f : b/a \quad \Delta \vdash e : a}{\Gamma \cdot \Delta \vdash f e : b} \, / \qquad \frac{\Delta \vdash e : a \quad \Gamma \vdash f : a\backslash b}{\Delta \cdot \Gamma \vdash f e : b} \, \backslash$$

$$\overline{\varepsilon \vdash \eta : \mathsf{M} a / a} \ \eta \qquad \overline{\varepsilon \vdash (\star) : K a \mathsf{M} r / \mathsf{M} a} \ \star \qquad \overline{\varepsilon \vdash \textbf{S} : (K b r / K a r) / K (b/a) r} \ \textbf{Scope}$$

(b) Continuized CCG with side effects, fixing a monad $\langle \mathsf{M}, \eta, \star \rangle$ and a result type r.

Figure 1: Continuized CCGs with and without side effects, for a fixed result type r and monad $\langle \mathsf{M}, \eta, \star \rangle$. In both grammars, $\textbf{S}\, m\, n := \lambda k.\, m\, (\lambda f.\, n\, (\lambda x.\, k\, (f\, x)))$.

tor M, an injection function η of type $a \to \mathsf{M} a$ (given any type a), and a recipe for sequencing programs $\star$, of type $\mathsf{M} a \to (a \to \mathsf{M} b) \to \mathsf{M} b$ (given any types a, b). The monadic functors η and $\star$ are required to satisfy the following three *monad laws*, which ensure that ηx represents a *trivial* injection of x into the monad (left and right unit), and that order of evaluation, but not relative embedding, matters for program sequencing (a form of associativity):

Definition 1. For any monad $\langle \mathsf{M}, \eta, \star \rangle$:

Left unit:	$\eta x \star k = k x$
Right unit:	$m \star \eta = m$
Associativity:	$(m \star k) \star c = m \star \lambda x.\, k\, x \star c$

Monads provide an elegant bridge between side effects and continuations, because the type of $(\star)$ is equivalent to $\mathsf{M} a \to K a (\mathsf{M} b)$. The bind function is thus itself a means of lifting effectful programs into continuized programs. Similarly, the unit function doubles as a default continuation, always available to be passed to $K a (\mathsf{M} b)$ programs without generating any new side effects. In this way, we can construct continuized grammars directly from monads by trading in **Lift** for $\star$ and **Triv** for η. Conveniently, because **Scope** operates at the level of already continuized programs, it doesn't change.

In fact, **Lift** is a theorem of any monadic grammar, when we restrict the result type to $\mathsf{M} r$. Generally, for any monad $\langle \mathsf{M}, \eta, \star \rangle$ and result type r:

Fact 1. $\Gamma \vdash x : a \Rightarrow \Gamma \vdash \lambda k.\, k\, x : K a (\mathsf{M} r)$

Fact 2. $\textbf{S}\, ((\star)\, m)\, ((\star)\, n)\, \eta$
$\qquad = m \star \lambda f.\, n \star \lambda x.\, \eta\, (f\, x)$

4 Finding the dynamic monad

We treat dynamic semantics as a case study for the monadic approach. Following Shan (2001), we characterize dynamic systems as those that recognize some forms of mutable state and nondeterminism. The former provides a mechanism by which expressions can manipulate the discourse context, usually through the introduction of discourse referents. The latter guarantees that indefinites are handled on a par with referential expressions, but whose referents are generated "randomly".

In classic formulations of dynamic semantics (Heim 1982, Kamp 1981, Groenendijk & Stokhof 1991, Dekker 1994), sentences denote relations on sequences. Non-empty relations correspond to truthful updates. Indefinites (and perhaps disjunctions) generate nonfunctional (in our terms, "nondeterministic") relations, with inputs matched against multiple outputs. Sentential conjunction amounts to relation composition, which pipes the sequences output by the left conjunct pointwise into the right.

(1)

$$[\![\text{someone}]\!] = \lambda k i.\, \bigcup_{x \in \mathsf{person}} k\, x\, (i + x)$$

$$[\![\text{someone left}]\!] = \lambda i.\, \{ i + x : x \in \mathsf{person} \wedge x \in \mathsf{left} \}$$

The denotations in (1) illustrate the way in which the standard dynamic approach puts state and nondeterminism are put front and center. Sentences are in fact *nothing but* nondeterministic modifications of state; even truth and falsity are derivative notions. Borrowing from the functional programming literature, we advocate a different perspective: treating nondeterminism and state modification as *side effects*, associated with semantic values, rather than replacing them.

2

Here is the standard monad for state manipulation. Assume that γ is the type of "evaluation contexts". For our purposes, we set γ equal to the type of discourse referent sequences.

Definition 2 (The State monad).

$$
\begin{aligned}
\mathsf{M}\,a &::= \gamma \to a * \gamma \\
\eta\,x &:= \lambda i.\langle x, i\rangle \\
m \star k &:= \lambda i.\,k\,(m\,i)_0\,(m\,i)_1
\end{aligned}
$$

To keep things simple, we model dref introduction as sequence extension (cf. de Groote 2006, Unger 2012, Charlow 2014) and dref retrieval as sequence projection. To dynamically charge a monadic individual (of type $\mathsf{M}\,e$), we pass the individual through a continuation that simply returns its underlying value and appends it to the output sequence (Definition 3). For instance $(\eta\,\mathsf{j})^{\triangleright} \star \lambda xi.\langle x, i+x\rangle = \lambda i.\langle \mathsf{j}, i+\mathsf{j}\rangle$.

Again keeping things simple, we assume a single projection function $\cdot_{\top}$ that returns as its value the most topical referent (say, for concreteness, the last) in a sequence.

Definition 3 (Dref introduction).

$$
m^{\triangleright} := m \star \lambda x.\eta\,x\,(i+x)
$$

Definition 4 (Dref retrieval).

$$
\mathsf{he} := \lambda i.\langle i_{\top}, i\rangle
$$

As an example, consider the sentence *Al left* (call its denotation **X**). Al is first injected into the state monad with η, then shifted so as to introduce himself to the discourse record, then sequenced into a continuation representing his nuclear scope. The result is a function from states to truth if Al left and falsity if he didn't, paired with an updated state which now lists Al as a dref.

$$
(\eta\,\mathsf{a})^{\triangleright} \star \lambda x.\eta\,(\mathsf{left}\,x) = \lambda i.\langle \mathsf{left}\,\mathsf{a}, i+\mathsf{a}\rangle
$$

Obversely, the sentence *he was tired* (call its denotation **Y**) reads in a discourse state, chooses the most topical referent in that state, and then returns the referent together with the original state, unmodified.

$$
\mathsf{he} \star \lambda x.\eta\,(\mathsf{tired}\,x) = \lambda i.\langle \mathsf{tired}\,i_{\top}, i\rangle
$$

When we sequence the first sentence into the second, the dref introduced by the proper name is immediately picked up by the pronoun.

$$
\mathbf{X} \star \lambda p.\mathbf{Y} \star \lambda q.\eta\,(p \wedge q)
$$
$$
= \lambda i.\langle \mathsf{left}\,\mathsf{a} \wedge \mathsf{tired}\,\mathsf{a}, i+\mathsf{a}\rangle
$$

$$
\begin{aligned}
\mathsf{john} &:= \mathsf{j} & \mathsf{so} &:= \lambda i.\{\langle x, i\rangle \mid \mathsf{person}\,x\} \\
\mathsf{left} &:= \mathsf{left} & \mathsf{he} &:= \lambda i.\{\langle i_{\top}, i\rangle\} \\
\mathsf{not} &:= \lambda mi.\{\langle \neg \exists \pi \in mi.\pi_0, i\rangle\} \\
\mathsf{a} &:= \lambda ci.\{\langle x, i'\rangle \mid \langle \mathsf{T}, i'\rangle \in c\,x\,i\} \\
\mathsf{every} &:= \lambda ck.\,\mathsf{not}\,(\mathsf{a}\,c \star \lambda x.\,\mathsf{not}\,(k\,x)) \\
\mathsf{ev.ling} &:= \lambda k.\,\mathsf{not}\,(\mathsf{a.ling} \star \lambda x.\,\mathsf{not}\,(k\,x)) \\
\mathsf{no.ling} &:= \lambda k.\,\mathsf{not}\,(\mathsf{a.ling} \star k)
\end{aligned}
$$

Figure 2: Lexicon for CCCG specialized to State-Set Monad

Thus the State monad allows us to see discourse modification as a side effect of composition. However, to incorporate indefinites, which introduce drefs *indeterminately*, we need to add a layer of nondeterminism to the basic State monad. This we do in Definition 5, which lays out what we will call the State-Set monad.[2]

Definition 5 (The State-Set monad).

$$
\begin{aligned}
\mathsf{M}\,a &::= \gamma \to (a * \gamma) \to t \\
\eta\,x &:= \lambda i.\{\langle x, i\rangle\} \\
m \star k &:= \lambda i.\bigcup_{\pi \in mi} k\,\pi_0\,\pi_1
\end{aligned}
$$

Monadic meanings are now functions from discourse states to *sets* of values, tagged with potentially updated output states. Sequential combination still pipes the output state of one program in as input to another, but it now does so pointwise. In the next section, we will see how this provides a robust effects regime for modeling dynamic binding. Finally, to round out the formal fragment, we define a fourth combinator that shifts a State-Set monadic update into a dref-introducing update.

$$
\overline{\varepsilon \vdash \lambda mk.m\,(\lambda xi.\,k\,x\,(i+x)) : \mathsf{M}\,e/\mathsf{M}\,e}^{\blacktriangleright}
$$

5 Examples

Figure 2 defines a small lexicon tailored for a CCCG built around the State-Set monad. We argue that this grammar offers a powerful vantage point for analyzing dynamic phenomena that interact with patterns of scope-taking.

For instance, Figure 3 provides a derivation of cross-sentential anaphora. One property of the derivation worth highlighting is that both clauses are evaluated, i.e. applied to the default η continuation, before they are conjoined. This delimits the

---

2 Also known as the Parser monad (Hutton & Meijer 1998).

scope of quantificational operators in the clauses, including that of the indefinite in the first sentence. Nevertheless, the indefinite's side effect — its introduction of a nondeterministic discourse referent — lives on to influence the evaluation of the pronoun in the second conjunct.

Other operators, however, may *quantify* over the nondeterminism of indefinites in their scope, gobbling up their side effects in the process. Negation is the classic example. As defined in Figure 2, **not** evaluates its prejacent, discards any discourse referents generated in the process, and then checks that none of the alternative updates resulted in true boolean values. As is standard in dynamic systems, we can use this state-capturing behavior to build up other dynamically closed meanings. For example, the universal quantifier defined in Figure 2 will display quite different behavior than the indefinite of Figure 3 when evaluated at the clause boundary. Plugging in the unit continuation will do two things: (i) as before, it will close off the nuclear scope of the universal; but (ii) this time, we will be left with a pure (effect-free) computation, equivalent to $\eta\,(\forall x.\,\mathsf{person}\,x \Rightarrow \mathsf{left}\,x)$.

Importantly this does not preclude the possibility of donkey anaphora. Returning to the lexical entry for **every**, note that the restrictor c here acquires a kind of monadic scope, via $\star$, over the nuclear scope k. This means that any side effects inside c will influence the context of evaluation for k. However, as in other dynamic systems, the wide-scoping negation will discharge whatever side effects are accrued in the process.

6 Discussion

Compare PLA, where only sentences are imbued with context change potential. Necessary since in PLA and standard dynamic treatments of anaphora, discourse-level content and truth-conditional content are conflated—i.e. a sentence denotes a nonempty relation on sequences iff the sentence is true. Thus: standard dynamic techniques (DPL, DMG) not reducible to monads.

Some upshots: no dynamic conjunction, completely standard model theory (cf. de Groote 2006). "Contexts of evaluation" are constructed on the fly. Variable-free, directly compositional (Jacobson 1999).

Monads as a natural way to extend a continuations-based grammar with tools for dynamic binding and exceptional scope. In the end: you have functional application, plus the functors from whichever monads are implicated in a given language. Effects recognized in the types.

There is no need to settle on a single ("the") grammar. Different and quite varied side effects regimes can be modularly grafted onto a simple applicative ("pure") core. Lexical entries that would seem incongruous in a flat-footed standard perspective integrate seamlessly in a single grammar.

Again, any sufficiently "well-behaved" (see Section **??**) regime of semantic enrichment (e.g. focus-sensitivity, alternative-generation, intensionality) may be accommodated along similar lines. Furthermore, because of the inherent modularity of the monadic approach, various side effects may be added to or subtracted from the grammar without adjusting the basic compositional machinery or the lexical entries that are not sensitive to the effects in question. Therefore, we expect these results will be of interest for natural language semanticists of many stripes, in addition to categorial grammarians working on donkey anaphora and/or scope-taking.

Theory extends to scope islands, wide range of exceptional binding configurations Charlow 2014. Extends to pair-list phenomena, functional quantification: Bumford to appear. Crossover, superiority less clear (cf. Shan & Barker 2006, Barker & Shan 2008).

References

Barker, Chris & Chung-chieh Shan. 2008. Donkey Anaphora is In-Scope Binding. *Semantics & Pragmatics* 1(1). 1–46. http://dx.doi.org/10.3765/sp.1.1.

Barker, Chris & Chung-chieh Shan. 2014. *Continuations and Natural Language*. Oxford: Oxford University Press.

Barwise, Jon & Robin Cooper. 1981. Generalized quantifiers and natural language. *Linguistics and Philosophy* 4(2). 159–219. http://dx.doi.org/10.1007/BF00350139.

Bumford, Dylan. to appear. Incremental quantication and the dynamics of pair-list phenomena. *Semantics & Pragmatics* 8(9).

Charlow, Simon. 2010. Two kinds of binding out of DP. Unpublished ms.

Charlow, Simon. 2014. *On the semantics of exceptional scope*: New York University Ph.D. thesis.

Dekker, Paul. 1994. Predicate Logic with Anaphora. In Mandy Harvey & Lynn Santelmann (eds.), *Proceedings of Semantics and Linguistic Theory 4*, 79–95. Ithaca, NY: Cornell University.

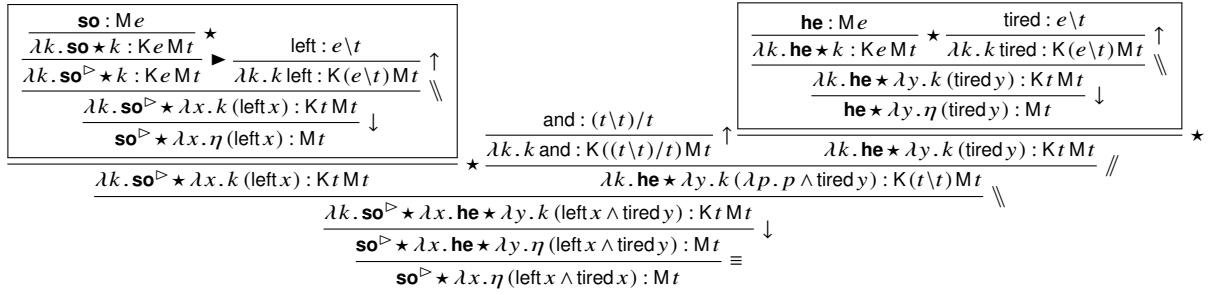Groenendijk, Jeroen & Martin Stokhof. 1991. Dynamic predicate logic. *Linguistics and Philosophy* 14(1).

$$\cfrac{\cfrac{\cfrac{\dfrac{\textbf{so}:\mathsf{M}e}{\lambda k.\,\textbf{so}\star k:K\,e\,\mathsf{M}t}\,\star}{\lambda k.\,\textbf{so}^{\triangleright}\star k:K\,e\,\mathsf{M}t}\blacktriangleright \quad \dfrac{\text{left}:e\backslash t}{\lambda k.\,k\,\text{left}:K(e\backslash t)\mathsf{M}t}\,\uparrow}{\dfrac{\lambda k.\,\textbf{so}^{\triangleright}\star \lambda x.\,k\,(\text{left}\,x):K\,t\,\mathsf{M}t}{\textbf{so}^{\triangleright}\star\lambda x.\,\eta\,(\text{left}\,x):\mathsf{M}t}\,\downarrow}}{\lambda k.\,\textbf{so}^{\triangleright}\star\lambda x.\,k\,(\text{left}\,x):K\,t\,\mathsf{M}t}\,\star \quad \cfrac{\cfrac{\dfrac{\text{and}:(t\backslash t)/t}{\lambda k.\,k\,\text{and}:K((t\backslash t)/t)\mathsf{M}t}\,\uparrow \quad \cfrac{\cfrac{\dfrac{\textbf{he}:\mathsf{M}e}{\lambda k.\,\textbf{he}\star k:K\,e\,\mathsf{M}t}\,\star\quad\dfrac{\text{tired}:e\backslash t}{\lambda k.\,k\,\text{tired}:K(e\backslash t)\mathsf{M}t}\,\uparrow}{\dfrac{\lambda k.\,\textbf{he}\star\lambda y.\,k\,(\text{tired}\,y):K\,t\,\mathsf{M}t}{\textbf{he}\star\lambda y.\,\eta\,(\text{tired}\,y):\mathsf{M}t}\,\downarrow}}{\lambda k.\,\textbf{he}\star\lambda y.\,k\,(\text{tired}\,y):K\,t\,\mathsf{M}t}\backslash\backslash}{\lambda k.\,\textbf{he}\star\lambda y.\,k\,(\lambda p.\,p\wedge\text{tired}\,y):K(t\backslash t)\mathsf{M}t}}{\lambda k.\,\textbf{he}\star\lambda y.\,k\,(\text{tired}\,y):K\,t\,\mathsf{M}t}\,/\!/}\,\star}{\cfrac{\dfrac{\lambda k.\,\textbf{so}^{\triangleright}\star\lambda x.\,\textbf{he}\star\lambda y.\,k\,(\text{left}\,x\wedge\text{tired}\,y):K\,t\,\mathsf{M}t}{\textbf{so}^{\triangleright}\star\lambda x.\,\textbf{he}\star\lambda y.\,\eta\,(\text{left}\,x\wedge\text{tired}\,y):\mathsf{M}t}\,\downarrow}{\textbf{so}^{\triangleright}\star\lambda x.\,\eta\,(\text{left}\,x\wedge\text{tired}\,x):\mathsf{M}t}\,\equiv}$$

Figure 3: Cross-sentential anaphora: deriving *someone$_i$ left; he$_i$ was tired.*

39–100. http://dx.doi.org/10.1007/BF00628304.

de Groote, Philippe. 2006. Towards a Montagovian account of dynamics. In Masayuki Gibson & Jonathan Howell (eds.), *Proceedings of Semantics and Linguistic Theory 16*, 1–16. Ithaca, NY: Cornell University.

Heim, Irene. 1982. *The Semantics of Definite and Indefinite Noun Phrases*: University of Massachusetts, Amherst Ph.D. thesis.

Hutton, Graham & Erik Meijer. 1998. Monadic parsing in Haskell. *Journal of Functional Programming* 8. 437–444. http://dx.doi.org/10.1017/S0956796898003050.

Jacobson, Pauline. 1999. Towards a Variable-Free Semantics. *Linguistics and Philosophy* 22. 117–184. http://dx.doi.org/10.1023/A:1005464228727.

Kamp, Hans. 1981. A theory of truth and semantic interpretation. In Jeroen Groenendijk, Theo Janssen & Martin Stokhof (eds.), *Formal Methods in the Study of Language*, 277–322. Mathematical Centre Amsterdam.

Kamp, Hans & Uwe Reyle. 1993. *From Discourse to Logic*. Dordrecht: Kluwer Academic Publishers.

Liang, Sheng, Paul Hudak & Mark Jones. 1995. Monad Transformers and Modular Interpreters. In *22nd ACM Symposium on Principles of Programming Languages (POPL '95)*, 333–343. ACM Press.

Moggi, Eugenio. 1989. Computational lambda-calculus and monads. In *Proceedings of the Fourth Annual Symposium on Logic in computer science*, 14–23. Piscataway, NJ, USA: IEEE Press.

Montague, Richard. 1974. The proper treatment of quantification in ordinary English. In Richmond Thomason (ed.), *Formal Philosophy*, chap. 8, 247–270. New Haven: Yale University Press.

Shan, Chung-chieh. 2001. A Variable-Free Dynamic Semantics. In Robert van Rooy & Martin Stokhof (eds.), *Proceedings of the Thirteenth Amsterdam Colloquium*, University of Amsterdam.

Shan, Chung-chieh. 2002. Monads for natural language semantics. In Kristina Striegnitz (ed.), *Proceedings of the ESSLLI 2001 Student Session*, 285–298.

Shan, Chung-chieh. 2005. *Linguistic Side Effects*: Harvard University Ph.D. thesis.

Shan, Chung-chieh & Chris Barker. 2006. Explaining Crossover and Superiority as Left-to-right Evaluation. *Linguistics and Philosophy* 29(1). 91–134. http://dx.doi.org/10.1007/s10988-005-6580-7.

Unger, Christina. 2012. Dynamic Semantics as Monadic Computation. In Manabu Okumura, Daisuke Bekki & Ken Satoh (eds.), *New Frontiers in Artificial Intelligence JSAI-isAI 2011*, vol. 7258 Lecture Notes in Artificial Intelligence, 68–81. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-32090-3_7.

Wadler, Philip. 1992. Comprehending monads. In *Mathematical Structures in Computer Science*, vol. 2 (special issue of selected papers from 6th Conference on Lisp and Functional Programming), 461–493.

Wadler, Philip. 1994. Monads and composable continuations. *Lisp and Symbolic Computation* 7(1). 39–56. http://dx.doi.org/10.1007/BF01019944.

Wadler, Philip. 1995. Monads for functional programming. In Johan Jeuring & Erik Meijer (eds.), *Advanced Functional Programming*, vol. 925 Lecture Notes in Computer Science, 24–52. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/3-540-59451-5_2.