Two souls of disjunction: Making dynamic semantics (more) explanatory*

Patrick D. Elliott†
www: patrickdelliott.com
Frankfurt Semantics Colloquium

January 17, 2019

1. Introduction

- there are two broad traditions addressing the semantics and pragmatics of disjunction, with little overlap.
- The *scalar implicature* literature (Sauerland 2004, a.o.) is concerned with deriving exclusive readings and ignorance inferences, and the logical environments in which they arise, while retaining inclusive disjunction as the basic meaning of natural language "or".
- The literature on *dynamic semantics* (Heim 1983, Beaver 2001) is largely concerned with deriving facts concerning presupposition projection in disjunctive sentences (Karttunen 1973).
- Both approaches to disjunction seem necessary, given the data, but it is not obvious that the two are even compatible – concretely, the scalar implicature literature takes as its starting point that the basic meaning of natural language *or* is inclusive logical disjunction, whereas dynamic semantics departs from this orthodoxy.
- Relatedly, dynamic semantics has been repeatedly criticized (see, e.g., Schlenker 2009, 2010) because the dynamic entry for disjunction can't be derived from logical disjunction (the same criticism applies to the other logical connectives).
- In this talk we will ultimately aim to reconcile the pragmatic and dynamic approaches to disjunction by integrating *exhaustification* into a dynamic framework.
- We'll argue that we can get away with just lifting propositional disjunction, once exhaustification enters the picture.
- Relatedly, one of our main goals will be to boost the explanatory power of dynamic semantics (wrt. presupposition projection). We'll aim to accomplish this by presenting a novel take on dynamic

*This work has previously been presented at an internal ZAS workshop. Special thanks to Paul Marty, Matthew Mandelkern, and Daniel Rothschild for helpful discussion.
†handout @ https://patrl.keybase.pub/handouts/frankfurt.pdf

semantics which makes use of the *state monad* (see, e.g., Charlow 2014).

- In this new fragment, dynamic connectives are not stated as primitives, but rather are derived by type-lifting propositional connectives in a systematic way.
- We'll present evidence that this move is independently necessary from facts concerning presupposition in disjunctive sentences, which the orthodox dynamic approach is ill-equipped to handle.

2. Presupposition projection and disjunction

- Karttunen (1973) observed that in a sentence such as (1-3), the presupposition in the second disjunct triggered by *stopped vaping* (*that Paul did vape*) is not inherited by the complex disjunctive sentence.

(1) Paul never vaped or Paul stopped vaping.

(2) Either neither Paul nor Sophie have vaped, or Paul stopped vaping.

(3) Either there is no bathroom, or the bathroom is downstairs.

- In order to capture such data, Karttunen proposed the generalization in (4) – if the *negation* of the first disjunct entails the presupposition of the second disjunct, then it fails to project.

(4) *Karttunen's generalization*
Let S be a sentence of the form "A or B".

a. If ⟦A⟧ presupposes π then ⟦S⟧ presupposes π.

b. If ⟦B⟧ presupposes π then ⟦S⟧ presupposes π, unless $\neg$ ⟦A⟧ entails π.

- The problem: Karttunen's generalization is too weak. In each case below, the negation of the first disjunct is too weak to entail the presupposition of the second.

(5) Either Paul never vaped and he jogged every day, or he stopped vaping. *no presupposition*
$\neg(\neg p \wedge q) \not\models p$

(6) Either there is no King of France and the country is in chaos or the King Of France is in exile. *no presupposition*
$\neg(\neg p \wedge q) \not\models p$

(7) Either nobody left early or only Josie left early. *no presupposition*
$\neg(\neg\exists x[P\ x]) \not\models P\ j$

- In order to account for this and similar data, we suggest the refinement to Karttunen's generalization in (8).
- The difference lies in clause (8b), which says that the presupposition of the second disjunct doesn't project, just in case it is entailed by grand conjunction of the negations of each excludable *alternative* to the second disjunct.[1]

[1] We take the *excludable* alternatives to a sentence ϕ to be those that are logically non-weaker than ⟦ϕ⟧, following e.g., Magri (2009).

(8) Let S be a sentence of the form "A or B".

a. If ⟦A⟧ presupposes π then ⟦S⟧ presupposes π.

b. If ⟦B⟧ presupposes π then ⟦S⟧ presupposes π, unless $\bigwedge_{\psi \in \text{excl B}} [\neg\psi]$ entails π.

- We assume that in a complex sentence such as "A or B", A is an alternative to B, and every sub-constituent of A is an alternative to B. This falls out straightfowardly from, e.g., Fox & Katzir's (2011) structural theory of alternatives.
- The refined generalization in (8) now encompasses the problematic datapoint in (5), since (a) the first disjunct raises the alternatives *Josie never smoked* and *Josie jogged every day*, and (b) negating both alternatives entails the presupposition of the second disjunct: *that Josie used to smoke*.
- In the very last section of the talk, we'll sketch an analysis that derives the presupposition projection associated with disjunction (so-called "dynamic disjunction"), from local exhaustification, while improving on the explanatory power of classical dynamic semantics.
- First we'll sketch Heim's dynamic semantics, a theory tailored to capture the Karttunen's generalizations for presupposition projection.

3. Heim's update semantics

- Here I'll give a (non-standard) presentation of Heim's (1983) propositional dynamic semantics, which aims to give an account of Karttunen's generalizations regarding presupposition projection.
- The generalizations pertaining to each of the logical connectives are given below:

(9) The Heim-Karttunen generalizations

a. If A presupposes π, then a sentence of the form "not A" presupposes π.

b. If A presupposes π and B presupposes ρ, then a sentence of the form "A and B" presupposes π, and unless A entails ρ, also presupposes ρ.

c. If A presupposes π and B presupposes ρ, then a sentence of the form "if A then B" presupposes π, and unless A entails ρ, also presupposes ρ.

d. If A presupposes π and B presupposes ρ, then a sentence of the form "A or B" presupposes π, and unless "not A" entails ρ, also presupposes ρ.

- Let's briefly illustrate some predictions.

(10) Paul didn't stop vaping. *presupposes that Paul vaped*

(11) Paul vaped and Paul didn't stop vaping. *presuppositionless*

(12) Paul didn't stop vaping and Paul vaped.
presupposes that Paul vaped

(13) If Paul and Sophie vaped, then Paul would never stop vaping.
presuppositionless

(14) Either Paul never vaped, or Paul stopped vaping.
presuppositionless

- I'll take the Heim-Karttunen generalizations to be essentially correct, modulo the refinement to the disjunction generalization discussed in the previous section. Let's now see how Heim's dynamic semantics aims to derive these generalizations.
- Sentences express *updates* of the common ground. The common ground is simply modelled as a set of possible worlds, i.e., a proposition. This reifies, in the semantics, Stalnaker's (1976) treatment of assertion.
- I'll define an operator $\mathbb{A}$ (*assert*) which takes us from propositions (of type $\mathsf{s} \to \mathsf{t}$) to updates (type $\{\mathsf{s}\} \to \{\mathsf{s}\}$).[2]
- Asserting a proposition ϕ is a function from the context c to an updated context, resulting from intersecting c and the set characterized by ϕ.

(15) Total assertion operator (def.)
$\mathbb{A}\ \phi \coloneqq \lambda c\ .\ c \cap \{\, w \mid \phi\ w \,\}$ $\quad (\mathsf{s} \to \mathsf{t}) \to \{\mathsf{s}\} \to \{\mathsf{s}\}$

(16) $\mathbb{A}\ [\![\text{Paul vapes}]\!] = \lambda c\ .\ c \cap \{\, w \mid \mathsf{vapes}_w\ \mathsf{p} \,\}$ $\quad \{\mathsf{s}\} \to \{\mathsf{s}\}$

- Within a Heimian framework, *presuppositions* place requirements on the common ground – concretely, presuppositions must be *redundant* (i.e., entailed by) the common ground. If they are not, the update is undefined.

[2] I'll flip-flop between talking about propositions as functions from worlds to truth values (type $\mathsf{s} \to \mathsf{t}$), and sets of worlds (type $\{\mathsf{s}\}$). Most of the time, we can think of sets as being equivalent to their characteristic functions, but there are a couple of places in which the difference will become important, e.g., when we talk about presuppositional propositions.

- We can redefine our assertion operator as a function from a *partial proposition* to a *partial update* – this makes the relation between a trivalent theory of presuppositions and a dynamic treatment especially transparent.

(17) Partial assertion operator (def.)
$\mathbb{A}\ \phi \coloneqq \lambda c\ :\ c \subseteq \mathsf{dom}\ \phi\ .\ c \cap \{\, w \mid \phi\ w \,\}$

(18) $[\![\text{Paul stopped vaping}]\!] = \lambda w\ :\ \mathsf{vaped}_w\ \mathsf{p}\ .\ \neg\mathsf{vapes}_w\ \mathsf{p}$

(19) $\mathsf{dom}\ [\![\text{Paul stopped vaping}]\!] = \{\, w \mid \mathsf{vaped}_w\ \mathsf{p} \,\}$

(20) $\mathbb{A}\ [\![\text{Paul stopped vaping}]\!] = \lambda c\ :\ c \subseteq \{\, w \mid \mathsf{vaped}_w\ \mathsf{p} \,\}$
$\quad . \ c \cap \{\, w \mid \neg\mathsf{vapes}_w\ \mathsf{p} \,\}$

- So, a sentence with a presupposition π is associated with an update on the common ground c, which is defined just in case π *is redundant in* (i.e., is entailed by) c.
- The logical connectives are treated as *functions from updates to updates*. In each case we give the formal definition of the function, and an informal algorithmic implementation of the function.

(21) *Heimian negation*

a. $\neg\ u \coloneqq \lambda c\ .\ c \setminus (u\ c)$

b. Take the result of updating c with u, and subtract the result from c.

(22) *Heimian conjunction*

a. $u \wedge v \coloneqq \lambda c\,.\,(v \circ u)\,c$

b. First update c with u, then update the result with v.

(23) *Heimian implication*

a. if u then $v \coloneqq (\neg\, u\, c) \cup (u \wedge v)\, c$

b. First, update c with u, and subtract the result from c, and store the result as c'. Next, update c with u, and then update the result with v, storing the result as c''. Now, take the union of c' and c''.

(24) *Heimian disjunction*

a. $u \vee v \coloneqq \lambda c\,.\,u\,c \cup v\,(\neg\, u\, c)$

b. First, update c with u, retaining the result c'. Then, update c with u, subtract the result from c, and update *this* with v, retaining the result c''. Finally, take the union of c' and c''.

- The way the connectives are defined is tailored to capture the Karttunen/Heim rules presupposition projection. Let's work through some examples in more detail.[3]
- Presuppositions project out of negation:

(25) Paul didn't stop vaping. *presupposes Paul vaped*

[3] Crucially, we assume that, if at any stage the result of some sub-update is *undefined*, the entire update is undefined.

(26) a. $\mathbb{A}\ [\![\text{Paul stopped vaping}]\!] = \lambda c : c \subseteq \{w \mid \text{vaped}_w\ \text{p}\}$
$. c \cap \{w \mid \neg\ \text{vapes}_w\ \text{p}\}$

b. $\neg\,(26a) = \lambda c : c \subseteq \{w \mid \text{vaped}_w\ p\}\ .\ c \smallsetminus (c \cap \{w \mid \text{vapes}_w\ \text{p}\})$

- A conjunctive LF $u \wedge v$ is predicted to be defined iff the common ground c updated with u entails the presupposition of v.

(27) Paul vaped and Paul stopped vaping. *no presupposition*

(28) a. $\mathbb{A}\ [\![\text{Paul vaped}]\!] = \lambda c\,.\,c \cap \{w \mid \text{vaped}_w\ \text{p}\}$

b. $\mathbb{A}\ [\![\text{Paul stopped vaping}]\!] = \lambda c : c \subseteq \{w \mid \text{vaped}_w\ \text{p}\}$
$. c \cap \{w \mid \neg\ \text{vapes}_w\ \text{p}\}$

c. $(28a) \wedge (28b) = \lambda c\ : (c \cap \{w \mid \text{vaped}_w\ \text{p}\}) \subseteq \{w \mid \text{vaped}_w\ p\}$
$. (c \cap \{w \mid \text{vaped}_w\ \text{p}\}) \cap \{w \mid \neg\ \text{vapes}_w\ \text{p}\}$

(29) Paul stopped vaping and Paul vaped. *presupposes Paul vaped*

(30) $(28b) \wedge (28a) = \lambda c\ : c \subseteq \{w \mid \text{vaped}_w\ p\}$
$. (c \cap \{w \mid \neg\ \text{vapes}_w\ \text{p}\}) \cap \{w \mid \text{vaped}_w\ \text{p}\}$

- Moving on to disjunction, a disjunctive LF is predicted to be defined iff the common ground entails the presupposition of the first disjunct *and* the common ground updated with the *negation* of the first disjunct entails the presupposition of the second disjunct.

(31) Paul never vaped or Paul stopped vaping.
no presupposition

(32) a. $\neg\, \mathbb{A}\ [\![\text{Paul vaped}]\!] = \lambda c\,.\, c \smallsetminus (c \cap \{\, w \mid \text{vaped}_w\ \text{p}\,\})$

b. $\mathbb{A}\ [\![\text{Paul stopped vaping}]\!] = \lambda c : c \subseteq \{\, w \mid \text{vaped}_w\ \text{p}\,\}$
$. c \cap \{\, w \mid \neg\ \text{vapes}_w\ \text{p}\,\}$

c. (32*a*) $\vee$ (32*b*)
$= \lambda c : (c \cap \{\, w \mid \text{vaped}_w\ \text{p}\,\}) \subseteq \{\, w \mid \text{vaped}_w\ \text{p}\,\}$
$. (c \smallsetminus (c \cap \{\, w \mid \text{vaped}_w\ \text{p}\,\}))$
$\cup\, ((c \cap \{\, w \mid \text{vaped}_w\ \text{p}\,\}) \cap \{\, w \mid \neg\ \text{vapes}_w\ \text{p}\,\})$

3.1. Criticism of Dynamic Semantics

- It's been repeatedly pointed out (see, e.g., Schlenker 2009, 2010) that it's possible to define "deviant" dynamic connectives which are truth-conditionally adequate, but nonetheless get the presupposition projection facts wrong.[4]
- An extremely simple example is *reverse dynamic conjunction*.

(33) $u \vartriangle v \coloneqq \lambda c\,.\, (u \circ v)\, c$

- Since the order in which the common ground c gets updated is reversed, this entry predicts that *Paul stopped vaping and Paul vaped* should be presuppositionless, whereas *Paul vaped and Paul stopped vaping* should presuppose that Paul vaped.
- In the next section I'll lay out a compositional update semantics that addresses this criticism by avoiding building the update logic into the meaning of the connectives.

[4] See Rothschild (2011) for detailed discussion of this point.

4. A state-monadic update semantics

- In this section I'll attempt to address criticisms of dynamic semantics by constructing a compositional update semantics, where the connectives receive their familiar, logical entries.
- We'll do this by constucting a state-monadic fragment, building on Charlow's (2014) dynamic semantics for anaphora.
- We'll factor the update logic out into a minimal inventory of type shifters. This will make the relationship between the update logic and the underlying fragment especially transparent.

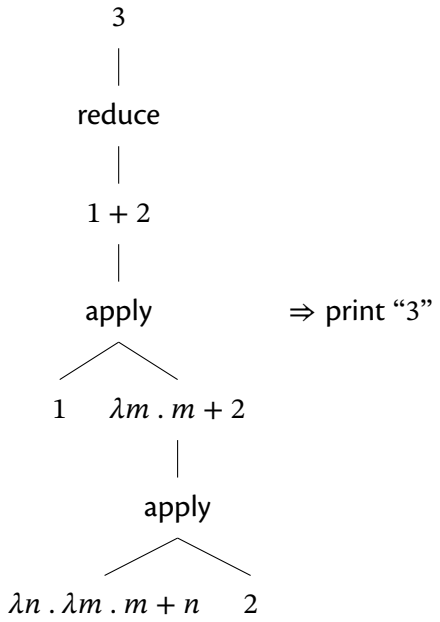
4.1. An aside on side-effects and monads

- Here, I'll be following existing work by, e.g., Shan (2002), Asudeh & Giorgolo (2016), and especially Charlow (2014), by using *monads* to extend a pure, Montagovian fragment.[5]
- Monads are a tool developed by computer scientists to model so-called *side-effects* in pure functional programming languages such as Haskell.
- What exactly is a pure functional programming language? As formal semanticists, we're already familiar with one such language – the simply typed lambda calculus.
- The *functional* part comes from the fact that programs are written as mathematical functions. *Purity* refers to the idea that computation can be modelled simply as function-argument application.
- *Side-effects* refer to any result of a computation that goes beyond function argument application.

[5] Note, that's it's not crucial to understand what a monad is in order to understand, the formal fragment, but the intuition behind monads will be helpful.

- Imagine a program that takes two integers, computes their sum, and prints the result as a string. The program returns a numerical value as the result of applying the *addition* function to its two arguments, but it returns an additional result which can't be reduced to function-argument application – namely, a printed string. This is illustrated in (34).

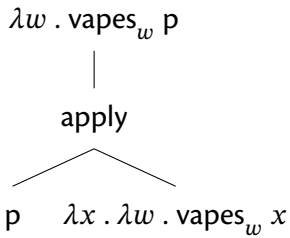
(34)



⇒ print "3"

- The intuition we're going to pursue is that an update to the common ground is a *side-effect* of pure-linguistic computation, and we're going to use monads (specifically, the state monad) to model this in a fully compositional way. The intuitive idea is illustrated in (35)

(35)



⇒ $c \cap \{ w \mid \mathsf{vapes}_w\ \mathsf{p} \}$

4.2. The formal fragment

- First, I'll define a type-constructor for *updates*; an update is a function from a context set, to a pair consisting of an ordinary value and a potentially updated context set.[6]

(36) $\mathsf{U\ a} := \{ \mathsf{s} \} \to (\mathsf{a} * \{ \mathsf{s} \})$

- We now define an injection function ρ to lift ordinary values into the update-semantic space. We furthermore define a binary function ($\circledast$), which performs *application* in the update-semantic space. The type constructor U, together with these two functions, constitutes an instantiation of the State monad.

(37) a. $a^{\rho} := \lambda c\ .\ \langle a, c \rangle$ $\qquad \mathsf{a} \to \mathsf{U\ a}$

b. $m \circledast n$
$:= \lambda c\ .\ \langle \mathsf{A}\ x\ y, c'' \rangle$
$\langle x, c' \rangle := m\ c$
$\langle y, c'' \rangle := n\ c'$

$\left.\begin{matrix} \mathsf{U\ (a \to b) \to U\ a} \\ \mathsf{U\ a \to U\ (a \to b)} \end{matrix}\right\} \to \mathsf{U\ b}$

[6] I use $(*)$ here as the type constructor for pairs, and later I use $\langle . \rangle$ as the corresponding data constructor for pairs.

- Let's dwell for a moment on the definition of (⊛):
  - It takes two updates m and n as its inputs.
  - The input context set c is first fed into m, returning a potentially updated context c'.
  - c' then gets fed into n, finally returning a potentially updated context c''.
  - The ordinary values contained within the two updates simply undergo function application.
- We now have everything we need to compose updates while keeping track of a potentially updated context, but as it stands, nothing interesting ever happens. Applying ρ to ordinary semantic values results in *vacuous updates*, e.g. for *Paul vapes*:

(38) $(\text{Paul vapes})^{\rho} = \lambda c \,.\, \langle \lambda w \,.\, \text{vapes}_w\ \text{p}, c \rangle$ U (S t)

- In order to insert some dynamic action into our fragment we'll (re-)define an *assert* operator.

(39) $\mathbb{A}\, m \coloneqq \lambda c \,.\, \langle p, c' \cap p \rangle$ U (S t) → U (S t)
$\quad$ for $\langle p, c' \rangle \coloneqq m\, c$

(40) $\mathbb{A}\, (\text{Paul vapes})^{\rho} = \lambda c \,.\, \left\langle \begin{matrix} \lambda w \,.\, \text{vapes}_w\ \text{p}, \\ c \cap \{ w \mid \text{vapes}_w\ \text{p} \} \end{matrix} \right\rangle$

- Now we can illustrate how state-monadic composition updates the common ground from left-to-right (note that we're just lifting static conjunction here).

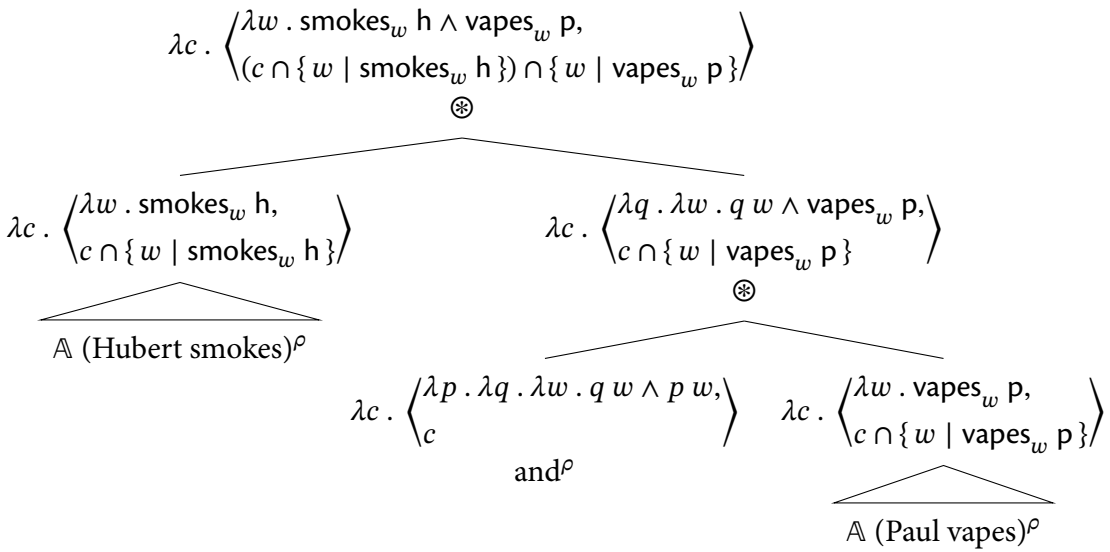
(41) Hubert smokes and Paul vapes. *LF in (42)*

- In our fragment, ordinary one-place predicates are of type e → S t, i.e., functions from individuals to propositions.
- We take presuppositional one-place predicates on the other hand to be of type e → U (S t), i.e., functions from individuals to (partial) propositional updates.
- Unlike ordinary predicates, presuppositional predicates place requirements on the input context.
- The entry for *stop smoking* in (43) for example, takes an individual x and an input context c, and returns a proposition plus an output context just in case the input context entails that *x smoked*.

(43) $\text{stopSmoking} = \lambda x \,.\, \lambda c : c \subseteq \{ w \mid \text{smoked}_w\ x \}$ e → U (S t)
$\quad . \langle \lambda w \,.\, \neg\, \text{smoked}_w\ x, c \rangle$

- It's of course possible to state the meanings of the Heimian dynamic connectives directly in our fragment, but this hardly constitutes a conceptual improvement over vanilla dynamic semantics.
- Instead, we're going define a function that systematically lifts static, propositional connectives to their dynamic counterparts. As we'll see, this will get the right results for negation, conjunction, and implication, but *not* for disjunction.
- For readability, we're going to provide a set-theoretic formulation of the propositional connectives.
- For technical reasons, each connective comes with an additional contextual *domain* parameter – in a familiar static setting, this will always just be saturated by the set of all possible worlds.

(42)



(44)

a. $\mathsf{not}_p \coloneqq \lambda p\,.\,\lambda q\,.\,q \setminus p$ $\quad \{\mathsf{s}\} \to \{\mathsf{s}\} \to \{\mathsf{s}\}$

b. $\mathsf{and}_p \coloneqq \lambda q\,.\,\lambda p\,.\,\lambda r\,.\,r \cap p \cap q$ $\quad \{\mathsf{s}\} \to \{\mathsf{s}\} \to \{\mathsf{s}\} \to \{\mathsf{s}\}$

c. $\mathsf{if...then}_p \coloneqq \lambda q\,.\,\lambda p\,.\,\lambda r\,.(r \setminus p) \cup q$ $\quad \{\mathsf{s}\} \to \{\mathsf{s}\} \to \{\mathsf{s}\} \to \{\mathsf{s}\}$

d. $\mathsf{or}_p \coloneqq \lambda q\,.\,\lambda p\,.\,\lambda r\,.\,r \cap (p \cup q)$ $\quad \{\mathsf{s}\} \to \{\mathsf{s}\} \to \{\mathsf{s}\} \to \{\mathsf{s}\}$

- Now we can define our dynamic lifter function. Again, for technical reasons I'll give two functions – one to lift a unary propositional connective and one to lift a binary propositional connective. It's possible to give a uniform definition, but it's simpler to write this way.

(45) $\mathsf{d\text{-}lift}_1\ f m \coloneqq \lambda c\,.\,\left\langle \begin{matrix} f\ p\ \{\,w \mid w \in D_\mathsf{s}\,\},\\ f\ c' c\end{matrix}\right\rangle$
for $\langle p, c'\rangle \coloneqq m\ c$

(46) $m\ (\mathsf{d\text{-}lift}_2\ f)\ n \coloneqq \lambda c\,.\,\left\langle \begin{matrix} f\ q\ p\ \{\,w \mid w \in D_\mathsf{s}\,\}\\ f\ c''\ c'\ c\end{matrix}\right\rangle$
$\langle p, c'\rangle \coloneqq m\ c$
$\langle q, c''\rangle \coloneqq n\ c'$

- In informal terms, the propositional connective f is applied to the ordinary value, in which case its inner-argument is simply the set of all possible worlds, and it is also applied to the updated common ground, in which case its inner-argument is the input context c.

- Applying d-lift to each of the propositional connectives (except disjunction) gives us...the Heimian dynamic connectives! (You can verify this for yourselves).[7]

(47) a. $\text{d-lift}_1\ \text{not}_p = \lambda m\,.\,\lambda c\,.\,\left\langle \begin{matrix}\{w \mid w \in D_s\} \smallsetminus p, \\ c \smallsetminus c'\end{matrix}\right\rangle$

for $\langle p, c'\rangle \coloneqq m\ c$

b. $\text{d-lift}_2\ \text{and}_p = \lambda n\,.\,\lambda m\,.\,\lambda c\,.\,\left\langle \begin{matrix}\{w \mid w \in D_s\} \cap p \cap q, \\ (c \cap c') \cap c''\end{matrix}\right\rangle$

$\langle p, c'\rangle \coloneqq m\ c$

$\langle q, c''\rangle \coloneqq nc'$

c. $\text{d-lift}_2\ \text{if..then}_p = \lambda n\,.\,\lambda m\,.\,\lambda c\,.\,\left\langle \begin{matrix}\{w \mid w \in D_s\} \smallsetminus p \cup q, \\ (c \smallsetminus c') \cup c''\end{matrix}\right\rangle$

$\langle p, c'\rangle \coloneqq m\ c$

$\langle q, c''\rangle \coloneqq nc'$

- Let's briefly illustrate the predictions we make for presupposition projection – see (49).

(48) If Paul and Sophie vaped, then Paul stopped vaping.

no presupposition

- The predictions we make for disjunction are deviant however – let's first consider the Heim rule for disjunction recast within our new framework.

[7] A small proviso is necessary here – we must assume that every propositional node is first subjected to the Stalnakerian assert operator $\mathbb{A}$.

(50) $\text{or}_d = \lambda n\,.\,\lambda m\,.\,\lambda c\,.\,\left\langle \begin{matrix}\{w \mid w \in D_s\} \cap (p \cup q), \\ c \cap (c' \cup c'')\end{matrix}\right\rangle$

$\langle p, c'\rangle \coloneqq m\ c$

$\langle q, c''\rangle \coloneqq n\ (c \smallsetminus c')$

- Our d-lift rule, however, predicts that the local context of the second disjunct should just be the first disjunct.

(51) $\text{d-lift}_2\ \text{or}_p = \lambda n\,.\,\lambda m\,.\,\lambda c\,.\,\left\langle \begin{matrix}\{w \mid w \in D_s\} \cap (p \cup q), \\ c \cap (c' \cup c'')\end{matrix}\right\rangle$

$\langle p, c'\rangle \coloneqq m\ c$

$\langle q, c''\rangle \coloneqq n\ c'$

- Lifted propositional *or* therefore makes deviant predictions for presupposition projection.

(52) #Either Paul vaped or Paul stopped vaping.

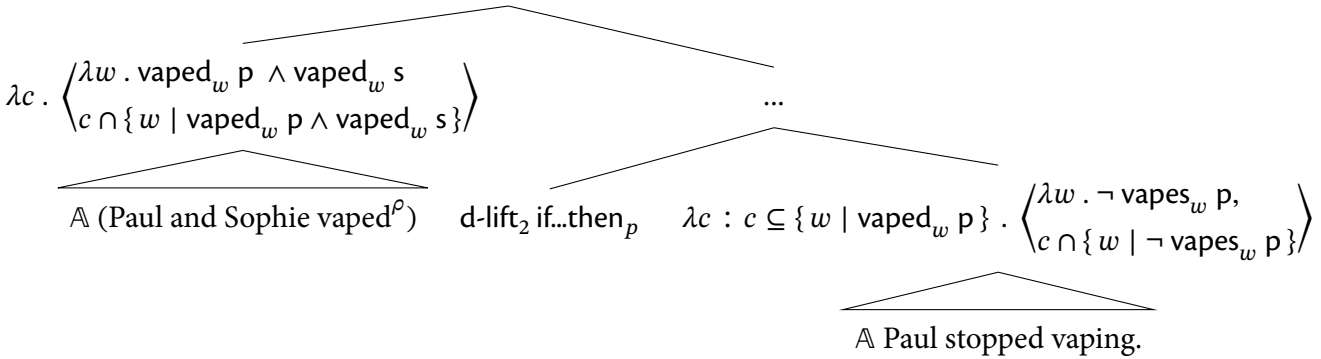
predicted to be presuppositionless

4.3. Enter exh

- I'd like to suggest that the predictions we make for *disjunction* aren't as bad as they seem – the odd projection behaviour we observe with disjunction can be blamed on the presence of an *exhaustification operator* exh.

- How should we define exh in a semantics with dynamic updates? I'd like to suggest the entry below:

(49) $\lambda c : (c \cap \{ w \mid \text{vaped}_w\ \text{p} \wedge \text{vaped}_w\ \text{s} \}) \subseteq \{ w \mid \text{vaped}_w\ \text{p} \}\ .\ \left\langle \begin{array}{l} \{ w \mid w \in D_s \} \setminus \{ w \mid \text{vaped}_w\ \text{p} \wedge \text{vaped}_w\ \text{s} \} \cup \{ w \mid \neg\ \text{vapes}_w\ \text{p} \}, \\ c \setminus (c \cap \{ w \mid \text{vaped}_w\ \text{p} \wedge \text{vaped}_w\ \text{s} \}) \\ \cup\ (c \cap \{ w \mid \text{vaped}_w\ \text{p} \wedge \text{vaped}_w\ \text{s} \} \cap \{ w \mid \neg\ \text{vapes}_w\ \text{p} \}) \end{array} \right\rangle$



(53) $\textsf{exh}\ m \coloneqq \lambda c\ .\ \langle p, c' \rangle$

$$\langle p, c' \rangle` \coloneqq m\ (c \cap \bigwedge_{q \in \text{excl}\ p} \{ \neg q \})$$

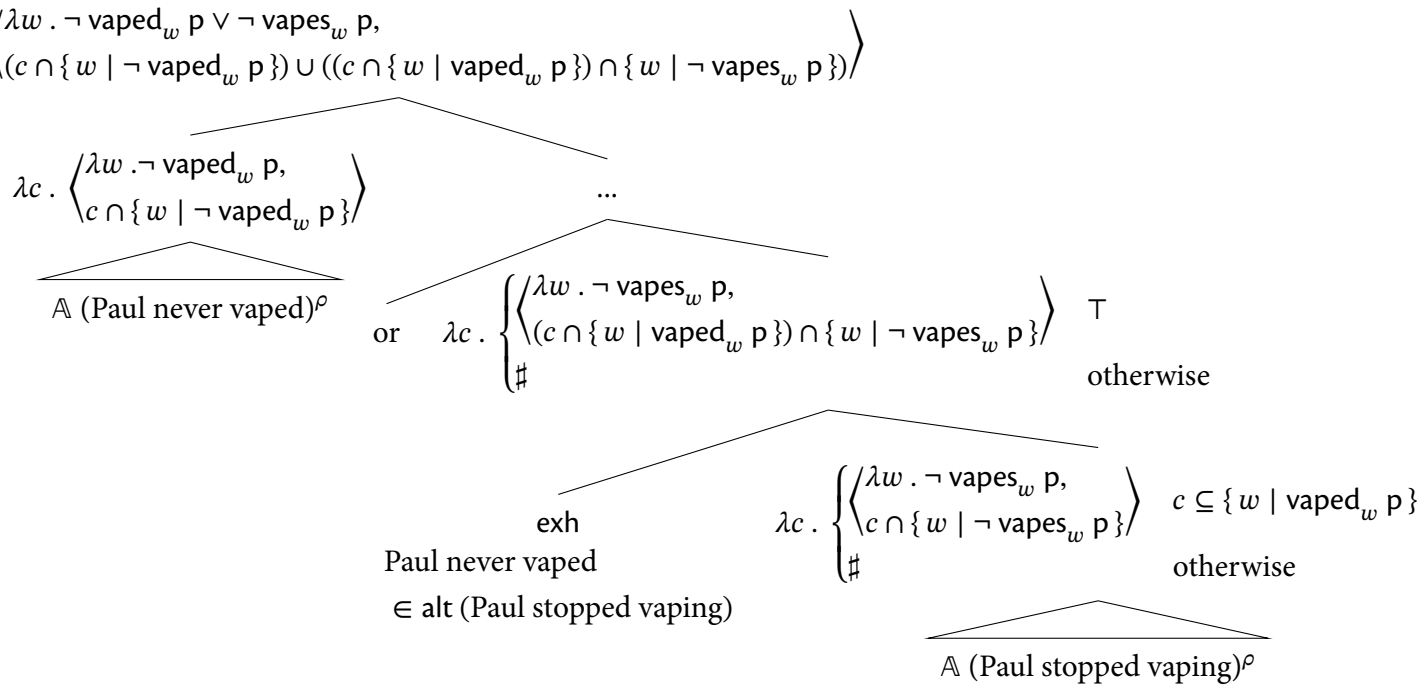
- Quasi-formally:
  - exh takes an update m as its prejacent, and updates the input context c with the implicatures of its prejacent, resulting in an updated context c'.[8]
  - exh updates the context c' with its prejacent.
  - In the ordinary dimension, exh just returns its prejacent.
- Since the implicatures of the prejacent update the context *before* the resulting context is updated with the prejacent, the expectation is that the implicatures associated with m can locally satisfy the presupposition associated with m.
- We claim that this is exactly what happens with disjunction. Regarding alternatives, we must simply assume that in a sentence of the form "P or Q", P is an alternative to Q. This follows from, e.g., Fox & Katzir's (2011) algorithm for computing alternatives.

[8] I ignore innocent exclusion here for convenience, and assuming that exh simply negates logically non-weaker alternatives.

5. Conclusion

- We've argued that the Heim/Karttunen projection rule for disjunction is *too weak*.
- Conceptually, we pointed out that the relation between disjunction in a Gricean world and disjunction in a Heimian world is unclear.
- Finally, we argued that once we incorporate exhaustification into a compositional dynamic framework, we end up with (a) superior

(54) Paul never vaped or Paul stopped vaping.

(55)


empirical results, and (b) a more explanatory account of presupposition projection than classical dynamic semantics.

References

Asudeh, Ash & Gianluca Giorgolo. 2016. Perspectives. *Semantics and Pragmatics* 9.

Beaver, David I. 2001. *Presupposition and assertion in dynamic semantics* (Studies in logic, language, and information). Stanford, California: CSLI. 314 pp.

Charlow, Simon. 2014. *On the semantics of exceptional scope*.

Fox, Danny & Roni Katzir. 2011. On the characterization of alternatives. *Natural Language Semantics* 19(1). 87–107.

Heim, Irene. 1983. On the projection problem for presuppositions. In *Proceedings of WCCFL 2*, 114–125. Stanford University.

Karttunen, Lauri. 1973. Presuppositions of compound sentences. *Linguistic Inquiry* 4(2). 169–193.

Magri, Giorgio. 2009. *A theory of individual-level predicates based on blind mandatory implicatures. constraint promotion for optimality theory*. Massachussetts Institute of Technology dissertation.

Rothschild, Daniel. 2011. Explaining presupposition projection with dynamic semantics. *Semantics and Pragmatics* 4(0). 1–43.
Sauerland, Uli. 2004. Scalar implicatures in complex sentences. *Linguistics and Philosophy* 27(3). 367–391.
Schlenker, Philippe. 2009. Local contexts. *Semantics and Pragmatics* 2.
Schlenker, Philippe. 2010. Local contexts and local meanings. *Philosophical Studies: An International Journal for Philosophy in the Analytic Tradition* 151(1). 115–142.
Shan, Chung-chieh. 2002. Monads for natural language semantics. *arXiv:cs/0205026*.
Stalnaker, Robert. 1976. Propositions. In A. F. MacKay & D. D. Merrill (eds.), 79–91. New Haven: Yale University Press.
Walsh, Clare R. & P. N. Johnson-Laird. 2004. Co-reference and reasoning. *Memory & Cognition* 32(1). 96–106.

A. Illusory inferences

Walsh & Johnson-Laird (2004) found that inference patterns such as those exemplified in (56) are overwhelming accepted, despite not being classically valid.

(56) a. P_1: Either Nathan is drinking coffee and he is listening to the radio, or Henning is watching TV.

b. P_2: Nathan is drinking coffee

c. C: Nathan is listening to the radio.

We schematize the general inference pattern in (57):

(57) a. $P_1 : (\phi \wedge \psi) \vee \pi$

b. $P_2 : \phi$

c. $C : \psi$

If the second disjunct is enriched with the negation of each alternative to the first disjunct, the resulting stengthened meaning is $(\phi \wedge \psi) \vee (\pi \wedge \neg\phi \wedge \neg\psi)$. Upon hearing ϕ therefore, the hearer can conclude that they are in a world in which ψ is also true.

B. Implementation

- You can find an implementation of the formal fragment in Haskell at `https://github.com/patrl/monadicHeim`. It makes use of the `mtl` and `containers` libraries.
- The type of an *update* makes use of the `StateT` monad transformer from the `mtl` library. The `Maybe` monad is used to explicitly handle partiality.

```
type U = StateT (Set S) Maybe
```