Binding alongside Hamblin alternatives calls for variable-free semantics

Chung-chieh Shan
Harvard University

The compositional, bottom-up computation of alternative sets was first introduced by Hamblin (1973) into Montague grammar to treat in-situ *wh*-questions. In the thirty years since then, alternative sets have found their way into theories of focus (Rooth 1985), indeterminate pronouns (Shimoyama 2001), and free-choice indefinites (Kratzer and Shimoyama 2002). These theories often position alternatives as a scope-taking mechanism that operates separately from Quantifier Raising (May 1977), Quantifying In (Montague 1974), or some other scope-taking mechanism for "genuine" quantifiers like *most*. On these theories, then, it is not surprising that (say) in-situ *who* takes scope differently from *most*, as is empirically observed. In particular, if "genuine" scope requires syntactic movement but alternative scope does not, then constraints on movement apply only to the former, and we predict—correctly—that the scope of *most* is more restricted than the scope of in-situ *who*.

(1) Who denied that who left?
'Which x and y are such that x denied that y left?'

(2) Who denied that most people left?
*'Which x is such that, for most y, x denied that y left?'

Many theories of quantification, including Quantifier Raising and Quantifying In, make essential use of variables for binding. In the first half of this paper, I show that using variables for binding is incompatible with computing alternatives bottom-up. For example, a theory on which *who* denotes an alternative set and *most books* binds a variable cannot account for *who read most books*. To fix this problem, we can either perform binding without variables (Jacobson 1999, 2000) or compute alternatives non-compositionally. Since Karttunen (1977) has already explored the latter option, I consider the former here: in the second half of this paper, I spell out how to compute alternatives compositionally in a variable-free theory of binding and quantification. Both options fix the problem.

Karttunen performs all scope-taking in syntax, whereas I perform all scope-taking in semantics. Therefore, whichever option one takes, it is no longer tenable to hypothesize that "genuine" scope is more restricted because it and only it requires syntactic movement (unless, perhaps, we turn from alternative sets to choice functions (Reinhart 1992, 1997) for non-"genuine" scope-taking).

Alternatives and variables are two "linguistic side effects": add-on layers that enrich a semantics. My main message is that these two layers of enrichment do not mix in a compositional semantics. I now introduce separate compositional semantics for alternatives and variables, then show what goes wrong when they are combined.

R. Young (ed), *SALT XIV* 289-304, Ithaca, NY: Cornell University.

1. The problem

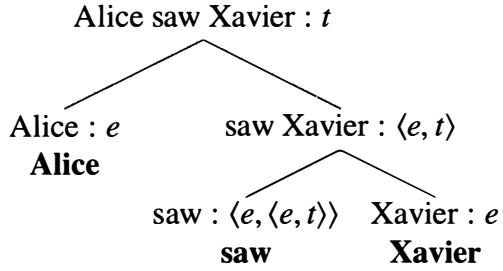
In the simplest version of Montague grammar, denotations are always combined using function application. The following schema shows the value as well as type produced by function application, combining two constituents β and γ.

(3) Plain-vanilla function application:

$$\begin{array}{c} [\![\beta]\!]([\![\gamma]\!]) : B \\ \diagup \quad \diagdown \\ [\![\beta]\!] : \langle A, B\rangle \quad [\![\gamma]\!] : A \end{array}$$

Plain-vanilla function application generates plain-vanilla sentences. In the trees below, bold text shows the words that make up a sentence.

(4) Alice saw Xavier.



1.1. *Alternatives*

To add alternatives to Montague grammar, we replace each type A with the type $\langle A, t\rangle$, in other words, the type of A-sets. For example, Hamblin takes a *wh*-constituent to denote a set of alternatives. A *wh*-NP denotes a set of individuals, and a *wh*-clause denotes a set of propositions, which an answer to the question would select from. A non-*wh* constituent denotes a singleton set, which can be thought of as a trivial set of alternatives; for example, *saw* denotes a singleton set containing the relation of seeing.

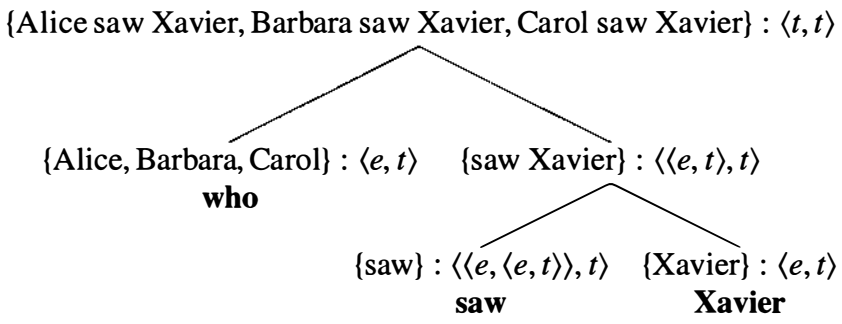
Intuitively, we can think of a set of A-alternatives as a nondeterministic A-value. Guided by this intuition, semantic combination of alternative sets proceeds pointwise: an A-set can be combined with an $\langle A, B\rangle$-set to produce a B-set. That is, each use of function application harbors two potential sources of nondeterminism: first, we may apply any of a set of functions; second, we may select any of a set of arguments. To implement this, we change the plain-vanilla function application rule (3) to:

(5) Alternative-friendly function application:

$$\begin{array}{c} \{\, f(x) \mid f \in [\![\beta]\!] \wedge x \in [\![\gamma]\!] \,\} : \langle B, t\rangle \\ \diagup \quad \diagdown \\ [\![\beta]\!] : \langle\langle A, B\rangle, t\rangle \quad [\![\gamma]\!] : \langle A, t\rangle \end{array}$$

If *who* denotes a set of three people, then *who saw Xavier* denotes a set of three propositions, as the following derivation shows. (To keep the example sentences simple in this paper, I pretend that *who* is an in-situ *wh*-phrase. Also, I stick to *wh*-phrases as examples of non-trivial alternative sets, but I could have picked any of the other applications of alternatives mentioned in the introduction.)

(6) Who saw Xavier? (Alice did.)

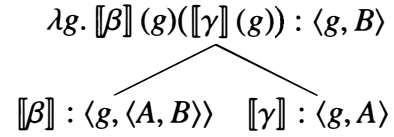


1.2. *Variables*

To add variables to Montague grammar, we replace each type A with the type $\langle g, A\rangle$, where g is the type of assignments. In other words, we hypothesize that denotations are functions from assignments, which are in turn functions from a countably infinite set of variables to the domain of individuals. For example, a pronoun or trace indexed by the variable name i denotes the function of type $\langle g, e\rangle$ that maps each assignment g to the individual $g(i)$. A constituent with nothing to bind denotes a constant function; for example, *saw* denotes a constant function mapping every assignment to the relation of seeing.

Semantic combination of functions from assignments proceeds pointwise: an assignment-to-A function can be combined with an assignment-to-$\langle A, B\rangle$ function to produce an assignment-to-B function. To implement this, we change the plain-vanilla function application rule (3) to:

(7) Variable-friendly function application:



As mentioned in the introduction, a primary application of variables is to interpret quantificational sentences, such as *Alice saw nobody*. For this and other applications, the grammar must provide some way to bind variables. A standard way to bind variables is Heim and Kratzer's predicate abstraction rule (1998), and it is what I consider in this paper (though other ways, like Partee's Derived VP rule (1973), do not mix with alternatives either).

(8) Predicate abstraction:

$$\lambda g.\, \lambda x.\, [\![\beta]\!]\,(g[x/i]) : \langle g, \langle e, A\rangle\rangle$$

$$\lambda_i \qquad [\![\beta]\!] : \langle g, A\rangle$$

The predicate abstraction rule is triggered by Quantifier Raising. It is easiest to see how the rule works by seeing it in action in a quantificational sentence. For example, (9) below derives *Alice saw nobody*. (I ignore all variables except i.)

(9) Alice saw nobody.

{Xavier ↦ Alice saw nobody,
Yves ↦ Alice saw nobody,
Zack ↦ Alice saw nobody} : $\langle g, t\rangle$

nobody {Xavier ↦ {Xavier ↦ Alice saw Xavier,
Yves ↦ Alice saw Yves,
Zack ↦ Alice saw Zack},
Yves ↦ {Xavier ↦ Alice saw Xavier,
Yves ↦ Alice saw Yves,
Zack ↦ Alice saw Zack},
Zack ↦ {Xavier ↦ Alice saw Xavier,
Yves ↦ Alice saw Yves,
Zack ↦ Alice saw Zack}} : $\langle g, \langle e, t\rangle\rangle$

λ_i {Xavier ↦ Alice saw Xavier,
Yves ↦ Alice saw Yves,
Zack ↦ Alice saw Zack} : $\langle g, t\rangle$

{Xavier ↦ Alice,
Yves ↦ Alice,
Zack ↦ Alice} : $\langle g, e\rangle$
Alice

{Xavier ↦ saw Xavier,
Yves ↦ saw Yves,
Zack ↦ saw Zack} : $\langle g, \langle e, t\rangle\rangle$

{Xavier ↦ saw,
Yves ↦ saw,
Zack ↦ saw} : $\langle g, \langle e, \langle e, t\rangle\rangle\rangle$
saw

{Xavier ↦ Xavier,
Yves ↦ Yves,
Zack ↦ Zack} : $\langle g, e\rangle$
t_i

The quantifier *nobody* raises to c-command its scope *Alice saw* t_i, leaving behind the trace t_i. Predicate abstraction applies to this scope to provide *nobody* with its semantic argument. When *nobody* applies this scope to Xavier, the trace receives an assignment mapping i to Xavier, so the trace returns Xavier. So far, so good.

1.3. *Alternatives and variables*

What goes wrong when we try to mix alternatives with variables? To do so, Kratzer and Shimoyama (2002) replace each type A with the type $\langle g, \langle A, t\rangle\rangle$. For example, *saw* now denotes a constant function mapping every assignment to the singleton set containing the relation of seeing. The function application rule is easy to update:

(10) Alternative-friendly, variable-friendly function application:

$$\lambda g.\{ f(x) \mid f \in [\![\beta]\!](g) \wedge x \in [\![\gamma]\!](g) \} : \langle g, \langle B, t\rangle\rangle$$

$$[\![\beta]\!] : \langle g, \langle\langle A, B\rangle, t\rangle\rangle \quad [\![\gamma]\!] : \langle g, \langle A, t\rangle\rangle$$

However, it is not clear how to update the predicate abstraction rule (8). From the last paragraph, we know that the types of the new rule must be as follows.

(11) Alternative-friendly predicate abstraction?

$$\textbf{???} : \langle g, \langle\langle e, A\rangle, t\rangle\rangle$$

$$\lambda_i \qquad [\![\beta]\!] : \langle g, \langle A, t\rangle\rangle$$

It turns out there is no rule with these types that produces the correct denotation. To see this, consider a sentence like *who saw nobody*, with a *wh*-phrase and a quantifier.

(12) Who saw nobody? (Alice did.)

$$\textbf{???} : \langle g, \langle\langle e, t\rangle, t\rangle\rangle$$

λ_i {Xavier ↦ {Alice saw Xavier, Barbara saw Xavier, Carol saw Xavier},
Yves ↦ {Alice saw Yves, Barbara saw Yves, Carol saw Yves},
Zack ↦ {Alice saw Zack, Barbara saw Zack, Carol saw Zack}}
$: \langle g, \langle t, t\rangle\rangle$

{Xavier ↦ {Alice, Barbara, Carol}, Yves ↦ {Alice, Barbara, Carol}, Zack ↦ {Alice, Barbara, Carol}}
$: \langle g, \langle e, t\rangle\rangle$

who

{Xavier ↦ {saw Xavier}, Yves ↦ {saw Yves}, Zack ↦ {saw Zack}}
$: \langle g, \langle\langle e, t\rangle, t\rangle\rangle$

{Xavier ↦ {saw}, Yves ↦ {saw}, Zack ↦ {saw}}
$: \langle g, \langle\langle e, \langle e, t\rangle\rangle, t\rangle\rangle$

saw

{Xavier ↦ {Xavier}, Yves ↦ {Yves}, Zack ↦ {Zack}}
$: \langle g, \langle e, t\rangle\rangle$

t_i

(As before, I pretend that *who* is an in-situ *wh*-phrase to keep the example simple. I also use different domains for *who*—Alice, Barbara, Carol—and for *nobody*—Xavier, Yves, Zack.) The quantifier *nobody* raises to c-command its scope *who saw* t_i, leaving behind the trace t_i. Our new predicate abstraction rule must then apply to this scope to provide *nobody* with its semantic argument. Shown near the top of (12), next to the λ_i, is the denotation of the scope of *nobody*, computed using the function application rule (10). From this denotation, we want to produce a constant function that maps every assignment to the following set of functions.

(13) $\{\{\mathrm{X} \mapsto \text{Alice saw X}, \mathrm{Y} \mapsto \text{Alice saw Y}, \mathrm{Z} \mapsto \text{Alice saw Z}\},$
$\{\mathrm{X} \mapsto \text{Barbara saw X}, \mathrm{Y} \mapsto \text{Barbara saw Y}, \mathrm{Z} \mapsto \text{Barbara saw Z}\},$
$\{\mathrm{X} \mapsto \text{Carol saw X}, \mathrm{Y} \mapsto \text{Carol saw Y}, \mathrm{Z} \mapsto \text{Carol saw Z}\}\} : \langle\langle e, t\rangle, t\rangle.$

Comparing the computed denotation in (12) against the desired denotation in (13), we see that predicate abstraction must "transpose" a value of type $\langle e, \langle A, t\rangle\rangle$ to a value of type $\langle\langle e, A\rangle, t\rangle$, where the type A is t (as usual). In other words, we want to turn a function to sets into a set of functions. But the denotation of the scope may as well be written

(14) $\{\text{Xavier} \mapsto \{\text{Barbara saw Xavier, Alice saw Xavier, Carol saw Xavier}\},$
$\text{Yves} \mapsto \{\text{Alice saw Yves, Carol saw Yves, Barbara saw Yves}\},$
$\text{Zack} \mapsto \{\text{Carol saw Zack, Alice saw Zack, Barbara saw Zack}\}\} : \langle g, \langle t, t\rangle\rangle,$

whose "transpose"

(15) $\{\{\mathrm{X} \mapsto \text{Barbara saw X}, \mathrm{Y} \mapsto \text{Alice saw Y}, \mathrm{Z} \mapsto \text{Carol saw Z}\},$
$\{\mathrm{X} \mapsto \text{Alice saw X}, \mathrm{Y} \mapsto \text{Carol saw Y}, \mathrm{Z} \mapsto \text{Alice saw Z}\},$
$\{\mathrm{X} \mapsto \text{Carol saw X}, \mathrm{Y} \mapsto \text{Barbara saw Y}, \mathrm{Z} \mapsto \text{Barbara saw Z}\}\} : \langle\langle e, t\rangle, t\rangle$

is clearly not what we want predicate abstraction to produce. Given that sets are not ordered, any predicate abstraction rule we can write has no way to tell the correct "transpose" from the incorrect ones.[1]

Kratzer and Shimoyama (2002) try to get around the problem by including every "transpose" in the output set. Their predicate abstraction rule produces a set of 27 alternative functions rather than 3 in the case (12) above.

(16) Kratzer and Shimoyama's alternative-friendly predicate abstraction:[2]

$$\lambda g.\{ f_{\langle e,A\rangle} \mid \forall x_e.\, f(x) \in [\![\beta]\!]\,(g[x/i])\} : \langle g, \langle\langle e, A\rangle, t\rangle\rangle$$

λ_i $\qquad$ $[\![\beta]\!] : \langle g, \langle A, t\rangle\rangle$

In a footnote, they write:

> There is a question about the correctness of [(16)]. It does not quite deliver the expected set of functions. As far as we can see, however, no wrong predictions are actually made, as long as we only use the definition for generating propositional alternatives.

This attempt works for simple cases like *who saw who*, but not when a quantifier binds into Hamblin alternatives, as in (12). In the latter cases, the rule (16) generates a nonexistent quasi-functional reading: Kratzer and Shimoyama predict that the answers in (17a) and (17b) should be felicitous, but they are not.

(17) Who saw nobody?
   a. *His$_i$ mother saw nobody$_i$.
   b. *Barbara didn't see Xavier, Alice didn't see Yves, and Carol didn't see Zack.

Thus a wrong prediction is actually made. Moreover, there is no correct predicate abstraction rule with the types in (11).

By the way, another problem with the rule (16) occurs when one *wh*-phrase binds into another, as in (18).

(18) Which man$_i$ sold which of his$_i$ paintings?

If any individual—even one who is not a man under discussion[3]—has no painting, then Kratzer and Shimoyama predict that no answer is felicitous. However, if Munch is under discussion, then *Munch sold "The Scream"* is a perfectly felicitous answer. This problem is easy to fix: just change the rule to produce

(19) $\lambda g.\{ f_{\langle e,A\rangle} \mid \forall x_e.\,(f(x) \in [\![\beta]\!]\,(g[x/i])) \vee (f(x) \text{ is undefined and } [\![\beta]\!]\,(g[x/i]) \text{ is empty})\}$

instead.

2. The diagnosis

The sentence (12) above is problematic for Kratzer and Shimoyama's merger of alternatives and variables because a function to sets loses information with respect to a set of functions. Predicate abstraction is not to blame for the empirical failure noted in (17). Rather, the grammar should not have fed predicate abstraction a function to sets in the first place. To capture the correct set of answer propositions to (12), the grammar must generate a set of functions to start with.

It may seem, then, that the proper merger of alternatives and variables should replace each type A not with the type $\langle g, \langle A, t\rangle\rangle$ but with the type $\langle\langle g, A\rangle, t\rangle$. We can think of $\langle g, \ldots\rangle$ as a "binding layer" in a type, and $\langle \ldots, t\rangle$ as an "alternative layer" in a type, so that the alternative layer must be outside the binding layer for the sentence (12).

Unfortunately, we cannot just switch the two layers around permanently: even though (12) calls for sets of functions, (18) calls for functions to sets, because every man is unlikely to have the same number of paintings. Thus the required order between the alternative and binding layers depends on the scopes and bindings in the sentence.[4] Worse, alternative and binding layers need to be arbitrarily interposed

in general. For example, (20) calls for an alternative layer that lies between two binding layers.

(20) Which man$_i$ told nobody$_j$ about which of his$_i$ paintings?

In order to capture the correct set of answer propositions to (20), the VP *told* t_j *about which of his*$_i$ *paintings* (after the quantifier *nobody* raises) must mean

(21) $\lambda x.\,\{\,\lambda y.$ 'told y about p' | 'p is one of x's paintings' $\}$,

of type $\langle e, \langle\langle e, et\rangle, t\rangle\rangle$: a function to sets of functions to properties. The outer binding layer "$\langle e, \ldots\rangle$" in this type is for the index i, and the inner binding layer "$\langle e, \ldots\rangle$" is for the index j. In between the two binding layers is an alternative layer "$\langle \ldots, t\rangle$".

To analyze (20) properly, binding by i must take place outside—yet binding by j must take place inside—the alternative layer. The standard theory of binding cannot accommodate this alternative layer in the middle, because it uses a single assignment to bind all variables wholesale. Kratzer and Shimoyama can hardly be faulted for mixing alternatives and variables the wrong way, when there is no right way to compute alternatives compositionally while binding variables using assignments![5] Instead, we need to handle each binding separately. That is precisely what is accomplished in variable-free semantics, which I now turn to.

3. A solution

In the rest of this paper, I extend Jacobson's variable-free semantics (1999, 2000) with alternative sets in a straightforward way, and show how alternatives and binding interact smoothly in the resulting system to interpret sentences like (12), (18), and (20) correctly.

3.1. *Preliminaries*

Actually, I cannot just start with Jacobson's theory, because it only addresses binding, not the quantification involved in the example sentences above. Fortunately, Barker (2002) has integrated Jacobson's theory with Hendriks's Flexible Types for quantification (1988, 1993) into a single system, so I take Barker's integration as my starting point.

For the purposes of this paper, we only need to say a few things about this starting point. It is a combinatory categorial grammar: the only mode of semantic combination is function application (as shown in (3)), but some unary rules (type-shifting operators) may apply to any denotation. Three type-shifting operators are relevant to us: Jacobson's **g** and **z**, and Hendriks's **AR**.

To deal with binding, Jacobson introduces two type-shifting operators, **g** and **z**. The **g** ("Geach") rule performs function composition; it combines one constituent that is waiting to be bound by a C with another constituent β to form a larger constituent that is again waiting to be bound by a C.

(22) Jacobson's **g**:

$$\lambda gz.\, [\![\beta]\!]\,(g(z)) : \langle\langle C, A\rangle, \langle C, B\rangle\rangle$$
$$\mathbf{g}$$
$$|$$
$$[\![\beta]\!] : \langle A, B\rangle$$

The **z** rule performs binding; if z is an individual argument to a constituent β, and g is another argument to β that is waiting to be bound by an individual, then **z** makes z bind into g. We need the second version of **z** below because, in the more complicated example (20), the subject *which man*$_i$ binds across *nobody*$_j$ into *which of his*$_i$ *paintings*.

(23) Jacobson's **z**:

$$\lambda gz.\, [\![\beta]\!]\,(g(z))(z) : \langle\langle e, A\rangle, \langle e, B\rangle\rangle$$
$$\mathbf{z}$$
$$|$$
$$[\![\beta]\!] : \langle A, \langle e, B\rangle\rangle$$
$$\lambda gyz.\, [\![\beta]\!]\,(g(z))(y)(z) : \langle\langle e, A\rangle, \langle B, \langle e, C\rangle\rangle\rangle$$
$$\mathbf{z}$$
$$|$$
$$[\![\beta]\!] : \langle A, \langle B, \langle e, C\rangle\rangle\rangle$$

To deal with quantification, Hendriks introduces a type-shifting operator **AR** (among others). **AR** is so named because it performs argument raising; it makes a predicate that takes an individual argument take a generalized quantifier argument instead. We need the second version of **AR** below because, in the more complicated example (20), the quantifier *nobody*$_j$ takes scope under both *which man*$_i$ and *which of his*$_i$ *paintings*.

(24) Hendriks's **AR**:

$$\lambda Xy.\, X(\lambda x.\, [\![\beta]\!]\,(x)(y)) : \langle\langle\langle e, t\rangle, t\rangle, \langle A, t\rangle\rangle$$
$$\mathbf{AR}$$
$$|$$
$$[\![\beta]\!] : \langle e, \langle A, t\rangle\rangle$$
$$\lambda xYz.\, Y(\lambda y.\, [\![\beta]\!]\,(x)(y)(z)) : \langle A, \langle\langle\langle e, t\rangle, t\rangle, \langle B, t\rangle\rangle\rangle$$
$$\mathbf{AR}$$
$$|$$
$$[\![\beta]\!] : \langle A, \langle e, \langle B, t\rangle\rangle\rangle$$

Using these type-shifting operators, we can generate sentences like *Every man*$_i$ *told nobody*$_j$ *about his*$_i$ *mother*. We are now ready to accommodate in-situ *wh*-phrases in the mix.

3.2. *Pervasive alternatives*

Given that variable-free semantics expresses binding with functions, the diagnosis in §2 that each binding layer may need to enclose an alternative layer indicates that

the range of each function may need to be a set of alternatives. Hence, generalizing to the worst case, I just let *all* functions return alternative sets. Whereas, as explained in §1.1, Hamblin's original alternative semantics replaces each type A with the type $\langle A, t\rangle$, I replace each type A with the type $\langle A', t\rangle$, where A' is defined recursively by

(25) $\quad e' = e, \qquad t' = t, \qquad \langle A, B\rangle' = \langle A', \langle B', t\rangle\rangle.$

Because $\langle A', \langle B', t\rangle\rangle$ is the type of A'-to-B' relations, what I am doing is essentially replacing functions with relations throughout each type. In particular,

(26) Intransitive verbs change type from $\langle e, t\rangle$ to $\langle\langle e, \langle t, t\rangle\rangle, t\rangle$.

(27) Transitive verbs change type from $\langle e, \langle e, t\rangle\rangle$ to $\langle\langle e, \langle\langle e, \langle t, t\rangle\rangle, t\rangle\rangle, t\rangle$.

(28) Generalized quantifiers change type from $\langle\langle e, t\rangle, t\rangle$ to $\langle\langle\langle e, \langle t, t\rangle\rangle, \langle t, t\rangle\rangle, t\rangle$.[6]

For example, the transitive verb *saw* does not involve interrogatives at all, so it denotes and returns singleton sets all around:

(29) $\quad [\![saw]\!] = \{\lambda x.\, \{\lambda y.\, \{\mathrm{saw}(y)(x)\}\}\}.$

It may be disconcerting how alternatives pervade every type in this revised alternative semantics. Nevertheless, it does not involve dynamic type-shifting (beyond what Jacobson and Hendriks already introduced). In this regard, my use of relations is like Hamblin's original alternative semantics and unlike Jacobson's (2002) and Winter's (2003) use of relations.

A relation can be thought of as a function that nondeterministically returns zero or more values, like how propositional denotations in dynamic semantics relate input and output discourse states (Groenendijk and Stokhof 1991).[7] With this intuition in mind, it is easy to see how to change the function application rule (3) to cope with pervasive alternatives. Each use of function application now harbors *three* potential sources of nondeterminism: first, we may apply any of a set of functions; second, we may select any of a set of arguments; third (and this is new compared to §1.1), the function may return any of a set of results. These three nondeterministic steps are formalized in the new function application rule below, now more of a relation application rule.

(30) Pervasive-alternative-friendly function application:

$$\{\, y \mid f \in [\![\beta]\!] \wedge x \in [\![\gamma]\!] \wedge y \in f(x) \,\}$$

$$[\![\beta]\!] : \langle\langle A, \langle B, t\rangle\rangle, t\rangle \qquad [\![\gamma]\!] : \langle A, t\rangle$$

To strengthen this intuition of nondeterminism, I now change the notation for the remainder of this paper: Henceforth, I write $\langle A, B\rangle$ to mean not the type of A-to-B functions, but the type of A-to-B relations. I write "$: A$" after a semantic value to mean not that the value is an element of the type A, but that the value is a *subset* of the type A.

I also change the notation for semantic values to suit pervasive alternatives: I use the abstraction syntax $\lambda x. \ldots$ to create not a function, but a singleton set containing a relation. I use the application syntax $\ldots(\ldots)$ to compute not function application, but the image of a set of arguments under a set of relations. Thus, for example, the λ-term $\lambda z.\, f(g(z))$ means no longer to compose the functions f and g, but to compose the relations f and g. Happily, this new term syntax fits the new type syntax introduced in the previous paragraph: if $f : \langle A, B\rangle$ and $g : \langle C, A\rangle$, then $\lambda z.\, f(g(z)) : \langle C, B\rangle$ still. Note that a λ-bound variable, like z in $\lambda z.\, f(g(z))$, is always a singleton.[8]

Because the new term syntax fits the new type syntax, and the new term and type syntax resembles the old term and type syntax, we can reinterpret Jacobson and Hendriks's λ-terms for their type-shifting operators to take pervasive alternatives into account. That is, we now read **g** in (22) as not function composition but relation composition, and also reinterpret (23) and (24) to redefine **z** and **AR**. This reinterpretation is the main payoff of the new notation.

One final piece of new notation: I write $\cdots + \cdots$ to denote the union of the two alternative sets denoted by the two subexpressions. Analogously, I write $\Sigma x. \ldots$ to denote the union of a family of alternative sets, obtained by varying x as a bound variable in the subexpression. These are the only ways to create non-singleton alternative sets in the new notation. Intuitively, + means to flip a coin to decide which branch to take. For instance, $f(x + y)$ is equivalent to $f(x) + f(y)$.

Hamblin's original insight was to model a *wh*-question as a nondeterministic set of propositions. To reimplement this insight in the new system, I now assign denotations to *who* and *which*. On one hand, I take *who* to be an unrestricted *wh*-NP, thus denoting the set of individuals.

(31) $$[\![who]\!] = \Sigma y.\, y : e.$$

On the other hand, *which* relates a property to individuals satisfying it.

(32) $$[\![which]\!] = \lambda P.\, \Sigma y \in P.\, y : \langle\langle e, t\rangle, e\rangle.$$

In other words, *which* maps each property to the set of individuals satisfying it. In yet other words, *which* simply denotes the singleton set containing the identity function on properties. This denotation makes crucial use of the pervasive nature of alternatives in the new system: it is a singleton set, but containing a function that returns non-singleton sets.

With implicit alternatives in the notation, λ-conversion is only allowable when the argument is a singleton. For example, we must be careful not to λ-convert

(33) $$(\lambda x.\, \mathrm{saw}(x)(x))(\mathrm{Alice} + \mathrm{Barbara}) : t$$

to

(34) $$\mathrm{saw}(\mathrm{Alice} + \mathrm{Barbara})(\mathrm{Alice} + \mathrm{Barbara}) : t.$$

These two λ-terms do not denote the same value—(33) is a set of two propositions

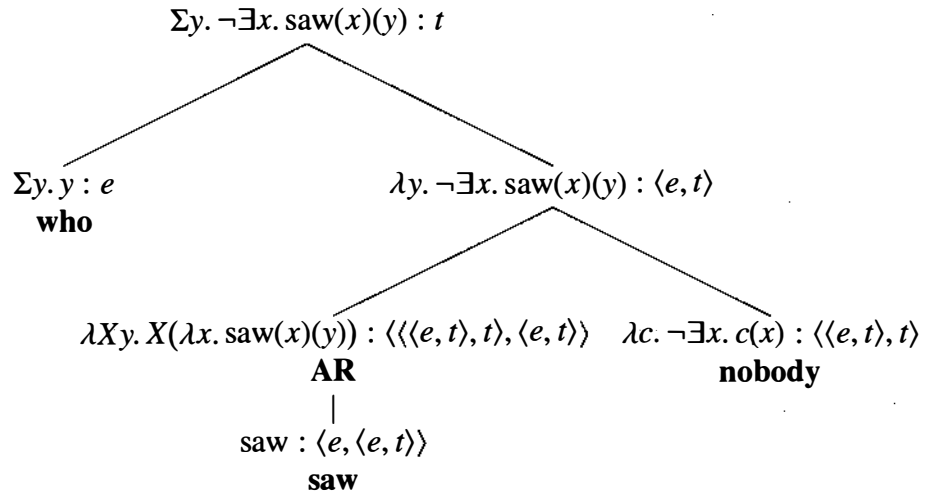
(35) $$\mathrm{saw}(\mathrm{Alice})(\mathrm{Alice}) + \mathrm{saw}(\mathrm{Barbara})(\mathrm{Barbara}),$$

whereas (34) is a set of four propositions

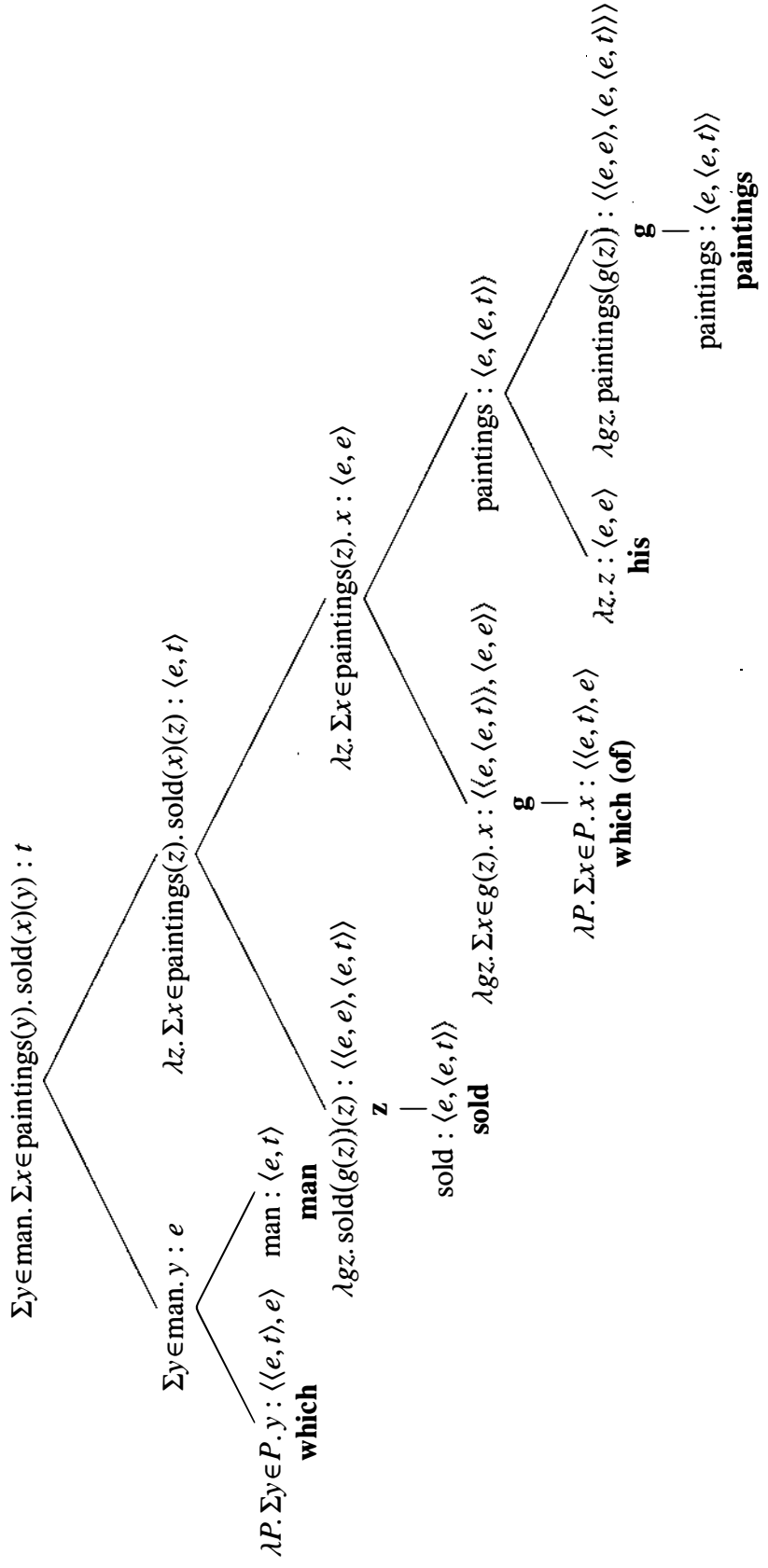
(36) saw(Alice)(Alice) + saw(Alice)(Barbara)
+ saw(Barbara)(Alice) + saw(Barbara)(Barbara).

That is all! We can finally return to the problematic sentences (12), (18), and (20), and derive the correct set of answer propositions for each of them.

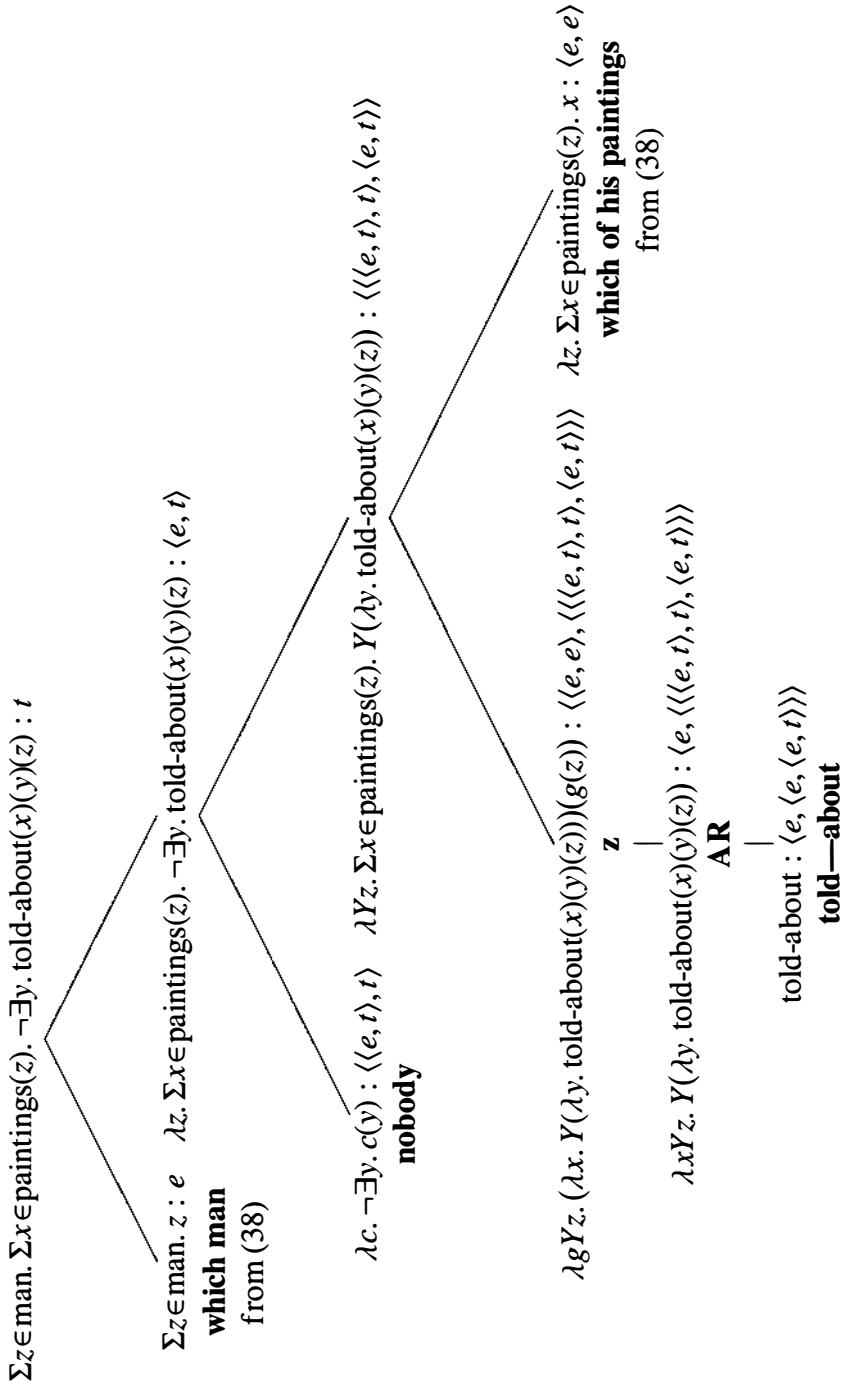
(37) Who saw nobody?



(38) Which man$_i$ sold which of his$_i$ paintings?



(39) Which man$_i$ told nobody$_j$ about which of his$_i$ paintings?



Endnotes

* Thanks to Stuart Shieber, Chris Barker, Chris Potts, the MIT syntax-semantics reading group, and the reviewers of SALT 14 and NELS 35. This work is

supported by the United States National Science Foundation Grant BCS-0236592.

1. This "inability to tell two values apart" can be formalized using *logical relations* and *parametricity* (Reynolds 1983) as for programming languages.

2. This rule is essentially Jacobson's (2002) type-shift operation m, which is in some sense the inverse of Winter's (2003) type-shift operation RG. The former converts a function to sets into a set of functions, and preserves information. The latter converts a set of functions to a function to sets, and loses information.

3. It is not clear empirically whether the question (18) is felicitous if some man under discussion has no painting. Our prediction in this case depends on how the function application rule (10) deals with $f(x)$ being undefined. This issue is independent of the rest of the paper.

4. In her paper in this volume, Jacobson also considers whether a constituent that uses both alternatives (for focus, in her case) and binding should denote a function to sets or a set of functions (§5.3). She decides on the former—a function to sets—for examples in which binding takes wider scope than focus, like *Every third-grade boy voted for Bill's mother, but every* FOURTH-*grade boy*$_i$ *voted for* HIS$_i$ *mother* (as opposed to Bill's). As she explains, "the basic intuition here to keep in mind is that only pronouns and gaps contribute an argument position [that is, a binding layer outside the alternative layer] which has widest scope over the focus value."

5. One could salvage assignments by redefining them to be functions from a finite number of variable names to individuals. For such a semantics to properly mix alternatives with binding, it must be capable of juggling multiple assignments, each of which may be a function from just one variable name to an individual. In other words, it must be variable-free in spirit.

6. A generalized quantifier returns an undefined value if invoked on a "nondeterministic" property, that is, a property that is not of the form $\lambda x.\{P(x)\}$ for some P.

7. In fact, the kind of pervasive nondeterminism introduced here can be used to reformulate dynamic semantics in a variable-free framework (Shan 2001).

8. In programming language terms, I am using a λ-calculus that is *impure* because it incorporates call-by-value nondeterminism (Wadler 1992; §8).

References

Barker, Chris. 2002. Remark on Jacobson 1999: Crossover as a local constraint. *Linguistics and Philosophy.* To appear.

Groenendijk, Jeroen, and Martin Stokhof. 1991. Dynamic predicate logic. *Linguistics and Philosophy* 14(1):39–100.

Hamblin, Charles Leonard. 1973. Questions in Montague English. *Foundations of Language* 10:41–53.

Heim, Irene, and Angelika Kratzer. 1998. *Semantics in generative grammar.* Oxford: Blackwell.

Hendriks, Herman. 1988. Type change in semantics: The scope of quantification and coordination. In *Categories, polymorphism and unification*, ed. Ewan Klein and Johan van Benthem, 96–119. Centre for Cognitive Science, Uni-

versity of Edinburgh.

———. 1993. Studied flexibility: Categories and types in syntax and semantics. Ph.D. thesis, Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Jacobson, Pauline. 1999. Towards a variable-free semantics. *Linguistics and Philosophy* 22(2):117–184.

———. 2000. Paycheck pronouns, Bach-Peters sentences, and variable-free semantics. *Natural Language Semantics* 8(2):77–155.

———. 2002. Direct compositionality and variable-free semantics: The case of binding into heads. In *SALT XII: Semantics and linguistic theory*, ed. Brendan Jackson. Ithaca: Cornell University Press.

Karttunen, Lauri. 1977. Syntax and semantics of questions. *Linguistics and Philosophy* 1(1):3–44.

Kratzer, Angelika, and Junko Shimoyama. 2002. Indeterminate pronouns: The view from Japanese. In *Proceedings of the 3rd Tokyo conference on psycholinguistics*, ed. Yukio Otsu, 1–25. Tokyo: Hituzi Syobo.

May, Robert C. 1977. The grammar of quantification. Ph.D. thesis, Department of Linguistics and Philosophy, Massachusetts Institute of Technology. Reprinted by New York: Garland, 1991.

Montague, Richard. 1974. The proper treatment of quantification in ordinary English. In *Formal philosophy: Selected papers of Richard Montague*, ed. Richmond Thomason, 247–270. New Haven: Yale University Press.

Partee, Barbara H. 1973. Some transformational extensions of Montague grammar. *Journal of Philosophical Logic* 2(4):509–534.

Reinhart, Tanya. 1992. Wh-in-situ: An apparent paradox. In *Proceedings of the 8th Amsterdam Colloquium*, ed. Paul Dekker and Martin Stokhof, 483–492. Institute for Logic, Language and Computation, Universiteit van Amsterdam.

———. 1997. Quantifier scope: How labor is divided between QR and choice functions. *Linguistics and Philosophy* 20(4):335–397.

Reynolds, John C. 1983. Types, abstraction and parametric polymorphism. In *Information processing 83: Proceedings of the IFIP 9th world computer congress*, ed. R. E. A. Mason, 513–523. Amsterdam: Elsevier Science.

Rooth, Mats Edward. 1985. Association with focus. Ph.D. thesis, Department of Linguistics, University of Massachusetts.

Shan, Chung-chieh. 2001. A variable-free dynamic semantics. In *Proceedings of the 13th Amsterdam Colloquium*, ed. Robert van Rooy and Martin Stokhof, 204–209. Institute for Logic, Language and Computation, Universiteit van Amsterdam.

Shimoyama, Junko. 2001. *Wh*-constructions in Japanese. Ph.D. thesis, Department of Linguistics, University of Massachusetts.

Wadler, Philip L. 1992. Comprehending monads. *Mathematical Structures in Computer Science* 2(4):461–493.

Winter, Yoad. 2003. Functional quantification. *Research on Language and Computation*. To appear.