A Logic for Easy Linking Semantics

Regine Eckardt

Göttingen University, Germany
regine.eckardt@phil.uni-goettingen.de

Abstract. Most semantic frameworks assume that the denotations of verbs expect their arguments in a certain specific order. In fixed word order languages, hence, we could say that order codes case marking. Moreover, all syntax-semantic mappings have to provide a solution for the fact that DPs can denote individual concepts of (extensional) type e as well as generalized quantifiers ($\langle\langle e,t\rangle,t\rangle$). The paper presents a new variant of type logic which offers a lean syntax-semantics interface for semantic representation in a Montagovian format. Specifically, the syntax-semantics mapping does not require obligatory quantifier raising (as Heim+Kratzer, 1998) and does not force the semanticist to make claims about a fixed underlying order of arguments of the verb. The latter feature will facilitate semantic research on free word order languages and semantic research on languages where no syntactic analysis in a Minimalist framework is as yet available.

1 Linking: Troubles and a Vision

Which syntax feeds semantics? In the present paper, I want to address the syntax-semantics interface from the back end, so to speak, and propose a new logical backbone for semantics, one that is better suited to host syntax. I should stress that this is a service article. I will not criticize, defend or propose any linguistic analysis but want to present a linking formalism that is easy to handle and can be adapted for a wide range of potential semantic analyses. Nevertheless, my work was inspired by linguistic questions which I will briefly review.

Type mismatch problem: It is a common assumption that verbs denote relations between entities. We can use names, indexicals or definite NPs to refer to entities. Moreover, we can use DPs that denote quantifiers over entities. In that case, a type mismatch between verb argument and DP denotation has to be resolved. While some theories endorse the assumption that verbs denote relations between generalized quantifiers, most people prefer to retain the original logical type of verbs. For these, Heim + Kratzer (1998) develop the by now standard way to resolve the type mismatch between verb and quantifiers. They propose an analysis where quantifier raising, coindexing and the interpretation of traces as variables serves, not only to settle matters of scope, but also as the standard way to enable semantic composition of verb projection and quantificational DP. Hence, the type mismatch problem is considered as solved by many semanticists. However, the semantic composition of even a simple sentence like *John likes most Fellini movies* requires quantifier raising, interpreted traces, coindexing, and lambda abstraction.

Order codes argument structure: Standard semantic treatments of English and other languages assume a fixed (underlying) order of arguments of the verb. Word order, rather than case marking, is the factor that ensures that each DP or PP instantiates the correct argument place of the verb. According to this standard analysis, free word order languages where argument structure is exclusively determined by case marking should not exist. If a language is suspected to be of that type (see Haug, 2009 on Ancient Greek), or if a language is not as yet sufficiently well understood to make claims about word order, semantic analysis requires to stipulate a basic order of verbal arguments. This common feature of truth conditional semantics in the Montagovian format can even lead scholars to adopt other semantic frameworks which allow for

a more direct impact of case marking in semantic interpretation. Hence, Montagovian semantics with interpreted case marking should be an attractive generalization of the standard framework.

The tacit argument problem: Many analyses propose that the verb has arguments that are not instantiated by overt phrases in the sentence. One example is provided in recent papers on tense by von Stechow (von Stechow et al., 2009). His analysis rests on a tense argument of the verb. In order to instantiate this argument in matrix clauses, he has to assume that there is a tacit temporal PRO, used as a dummy syntactic object that figures in quantifier raising. PRO leaves a trace which is interpreted as a time variable and instantiates the temporal argument of verbs. PRO is not a generalized quantifier, so it can not initiate lambda abstraction. In non-embedded sentences, von Stechow has to assume that PRO passes its index to an independent lambda operator and gets deleted afterwards. While Minimalist syntax allows to delete non-interpretable material, the entire process looks like an artifact of a specific kind of theory rather than an insight about the logical structure of language.

The event problem: In a standard Davidsonian analysis, event modifiers can apply to the event argument of the verb at many levels in syntax. In the standard fixed word order paradigm, we have to make a claim whether the event argument should be the first, or the second, etc. or the last argument of the verb. There is no agreed answer to this question and authors tend to avoid any principled position. I will discuss two possible options here.

Solution 1: We could claim that the event is an early argument of the verb such that, for instance, *love* denotes $\lambda e \lambda y \lambda x LOVE(x, y, e)$. λe gets instantiated by the trace x_e of an uninterpretable dummy E-PRO. E-PRO is co-indexed with x_e and has to be raised to all positions immediately below an event modifier MOD. In that position, it has to pass its index to an independent lambda operator that makes x_e accessible. After combination of MOD and verb projection, another trace of PRO instantiates the event argument of the verb, thereby making the argument inert until needed the next time. (Note: if there is more than one event modifier in a sentence, we will need a chain of traces of PRO).

Solution 2: We could alternatively claim that the event is a late argument of the verb, and our example verb *love* denotes $\lambda y \lambda x \lambda e LOVE(x, y, e)$. If an event modifier wants to combine with the verb before the verb has met all its DP arguments, the modifier has to use some standard procedure to instantiate the innermost argument of an n-place relation and to reopen all other arguments after modification. Such modes of combination can certainly be defined. Still, the resulting analysis again carries the flavor of repairing theory-internal problems rather than offering insights about the logical structure of language.

It should be pointed out that Kratzer (2002/unpublished) might offer a solution: She assumes that each quantificational DP binds the (currently open) event argument with an existential quantifier, and at the same time introduces a new, plural event argument that remains accessible and consists of the sum of all smaller events. Following this proposal, a sentence like *Sally fed all chicken in one hour* then means $\exists E \forall x(Chicken(x) \rightarrow \exists e Feed(Sally, x, e) \wedge e \subset E) \wedge \tau(E) = 1hour$) (ignoring further minimality requirements on events). Her analysis is motivated by the observation that different event modifiers can take scope below and above nominal quantifiers in one and the same sentence. Yet, the event problem originally is not a scope problem. If we want to generalize Kratzer's solution to a mechanism where the event parameter is accessible at each syntactic level, we'd have to claim that any DP (including definite noun phrases, proper names and other non-scope-taking DPs) existentially binds the event argument of the verb, combines with the verb, and afterwards introduces a new plural event that has the existentially bound first event as its part. Hence, a sentence like *Sally fed Prillan* will receive the following interpretation (again, leaving minimality conditions on E aside): $\exists E(\exists e(Feed(Sally, Prillan, e) \wedge e \subset E))$ Even though this may not be wrong in a strictly logical sense, it is at least redundant. Event semantics

would loose much of its original appeal: Events should make semantic representations elegant and perspicuous, and not redundant and unperspicuous.

In this paper, I will define Linking Logic, a type logic on finite variable assignments, and Easy Linking Logic which endorses variables that are indexed with abstract case labels. This will allow us to design Easy Linking Semantics, a format for semantic analysis and composition that is independent of any specific grammatical framework and yet draws on earlier Montagovian semantics in a maximally conservative manner.

2 Linking Logic

In this section, I want to define a type logic which operates on partial variable assignments.[1] All terms t and formula ϕ are interpreted relative to models M and variable assignments g. Unlike normal logics, however, the interpretation will only be defined for variable assignment functions which have the free variables of t or ϕ as their domain. No formula can be evaluated relative to an assignment which is too "rich". As a consequence, variable binding will not always lead to interpretable formula. E.g. $\exists x\phi$ will only be interpretable if x occurs free in ϕ. These properties are not desired or desirable in logics for mathematics and philosophy in general, perhaps. However, they reflect deep insights about natural language interpretation. For example, the ban on vacuous quantification has been proposed as a principle at LF. My analysis implements this ban at an even deeper level in the logical backbone of semantic analysis.

Following standard semantic practice, I will use the atomic types e, s, t in the sample system. Simpler and richer systems are possible.

Types:

- e, s, t are atomic types.
- If σ and τ are types, then $\langle\sigma, \tau\rangle$ is a type.
- Nothing else is a type.

A type logical syntax: A type logic language L on basis of these types consists of a set of constants for each type τ, and a set of variables for each type τ. In parallel, I will define the function fr that maps any term to the set of free variables that occur in that term. The terms in L are defined as follows:

- For each type τ, any constant c of type τ is a term of type τ. The set of free variables $fr(c) := \emptyset$.
- For each type τ, any variable v_i of type τ is a term of type τ. The set of free variables $fr(v_i) := \{v_i\}$.
- If A is a term of type $\langle\sigma, \tau\rangle$ and B is a term of type σ, then $A(B)$ is a term of type τ. The set of free variables $fr(A(B)) := fr(A) \cup fr(B)$.
- Logical connectives on type t: If ϕ and ψ are of type t, then $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \rightarrow \psi$ and $\neg\phi$ are terms of type t. The free variables are defined as follows: $fr(\neg\phi) := fr(\phi)$, and $fr(\phi \wedge \psi) = fr(\phi \vee \psi) = fr(\phi \rightarrow \psi) = fr(\phi) \cup fr(\psi)$-
- If ϕ is a term of type τ, and if $fr(\phi)$ contains variable v_i of type σ then $\lambda v_i.\phi$ is a term of type $\langle\sigma, \tau\rangle$. The set of free variables $fr(\lambda v_i.\phi) := fr(\phi) - \{v_i\}$.

[1] An extended version of the paper also includes predicate logic on partial variable assignments, which might offer an easier way into the format.

The present system does not introduce syncategorematic quantification as an operation on type t terms. Quantificational expressions can enter the system at the usual places: Determiners relate two sets and denote entities of type $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$; the denotations of determiner phrases have the type of generalized quantifiers $\langle\langle e,t\rangle,t\rangle$ and the normal universal and existential quantifiers $\forall,\exists$ will be defined as specific generalized quantifiers below. We will now turn to interpretation. In the following, I will use the notation $g|_A$ for the partial function $g*$ which arises by restricting g to domain A. Hence, $g|_{fr(\phi)}$ stands for g, restricted to the free variables in term ϕ.

Interpretation: Let D_e, D_s be domains of entities and worlds, and let $D_t := \{0,1\}$ as usual. Let $D_{\langle\sigma,\tau\rangle} := \{f|f : D_\sigma \rightarrow D_\tau\}$ the respective functional domains, and use $\mathbf{D}$ to refer to this hierarchy of sets. Let moreover I be a function which maps all constants of type τ into D_τ. The type logical language L is interpreted relative to the model $M = \langle\mathbf{D},\mathbf{I}\rangle$ and partial variable assignments g from **Var** into $\mathbf{D}$. Specifically, the interpretation of any term ϕ will only be defined for assignments g such that $dom(g) = fr(\phi)$. As before, $\emptyset$ is used for the empty variable assignment.

- Let c be a constant of type τ. $||c||^{M,\emptyset} := I(c)$.
- Let v_i be a variable of type τ. Let g be an assignment which is defined on $fr(v_i) := \{v_i\}$. Then $||v_i||^{M,g} := g(v_i)$.
- Let A be term of type $\langle\sigma,\tau\rangle$ and B a term of type σ. Let g be a variable assigment with $dom(g) = fr(A(B)) = fr(A) \cup fr(B)$. Then $||A(B)||^{M,g} := ||A||^{M,g_1}(||B||^{M,g_2})$ where $g_1 :=$ g restricted to $fr(A)$ and $g_2 = g$ restricted to $fr(B)$.
- Logical connectives on type t: Let ϕ and ψ be of type t. Let moreover g any assignment with $dom(g) = fr(\phi) \cup fr(\psi)$.
  $||\phi \wedge \psi||^{M,g} = 1$ iff $||\phi||^{M,g_1} = 1$ and $||\psi||^{M,g_2} = 1$.
  $||\phi \vee \psi||^{M,g} = 1$ iff $||\phi||^{M,g_1} = 1$ or $||\psi||^{M,g_2} = 1$.
  $||\phi \rightarrow \psi||^{M,g} = 1$ iff $||\phi||^{M,g_1} = 0$ or $||\psi||^{M,g_2} = 1$.
  $||\neg\phi||^{M,g_1} = 1$ iff $||\phi||^{M,g_1} = 0$
  In all cases, $g_1 := g|_{fr(\phi)}$ and $g_2 := g|_{fr(\psi)}$.
- If ϕ is a term of type τ, and if $fr(\phi)$ contains variable v_i of type σ then $\lambda v_i.\phi$ is a term of type $\langle\sigma,\tau\rangle$. Let g be an assignment with $dom(g) = fr(\phi) - \{v_i\}$. Then $||\lambda v_i.\phi||^{M,g} :=$ the function which maps all $m \in D_\sigma$ to $||\phi||^{M,g'}$ where $g' := g \cup \{\langle v_i, m\rangle\}$.

This concludes the definition of a type logical language with sparse assignments. Any term in L can exclusively be interpreted with respect to variable assignments that run exactly on the free variables of the term. While this may look like a restriction at first sight, the system covers all and exactly the functions served by variable assignments elsewhere. The mayor difference between sparse assignment logics and classical logics arises already in the definitions of well-formed terms. Whereas classical logics allow for vacuous binding, the use of λ-abstraction is restricted to terms where the bound variable actually occurs free in the term. Let ϕ be a term of type t and let the variable v_i be in $fr(\phi)$. Then we will use the following abbreviations:

$\exists v_i\phi := \neg(\lambda v_i.\phi = \lambda v.\neg(v = v))$
$\forall v_i\phi := \lambda v_i\phi = \lambda v.v = v$

The two quantifiers inherit the ban on vacuous binding from λ-abstraction. Apart from that, they have the usual truth conditions. Let us check this for the existential quantifier $\exists v_i\phi$. We know that $v_i \in fr(\phi)$ and $fr(\exists v_i\phi) = fr(\phi) - \{v_i\}$. Given a model M and assignment g which is defined on $fr(\phi) - \{v_i\}$, $||\exists v_i\phi||^{M,g} = 1$ iff there is an extension $g* = g \cup \{\langle v_i, m\rangle\}$ such that

$||\phi||^{M,g*} = 1$. Note that ϕ is defined for assignment $g*$ because we assumed that v_i is free in ϕ.

Another operator that will be used later is the subset relation $\subset$ of type $\langle\langle e,t\rangle,\langle\langle e,t\rangle,t\rangle\rangle$. If A, B are terms of type $\langle e,t\rangle$, then $||A \subset B||^{M,g}$ is defined for all g with $dom(g) = fr(A) \cup fr(B)$.

$||A \subset B||^{M,g} = 1$ iff $||A||^{M,g|_{fr(A)}}$ is the characteristic function of a set A' in M, $||B||^{M,g|_{fr(B)}}$ is the characteristic function of a set B' in M and $A' \subset B'$.

This might be a good place to illustrate that bound variables do not have any influence on the meaning of terms. Consider the terms $\lambda v_2.MAN(v_2)$ and $\lambda v_9.WALK(v_9)$.

$||\lambda v_2.MAN(v_2) \subset \lambda v_9.WALK(v_9)||^{M,g} = 1$ iff

$||\lambda v_2.MAN(v_2)||^{M,g} \subset ||\lambda v_9.WALK(v_9)||^{M,g}$, that is iff
the set MAN with the characteristic function $||\lambda v_2.MAN(v_2)||^{M,g}$ is a subset of the set $WALK$ with the characteristic function $||\lambda v_9.WALK(v_9)||^{M,g}$.

Although the computation of the two latter characteristic functions operates via v_2 and v_9, the same functions would result if we execute the computation via any other variable. Generally, bound variables can be renamed like in classical logics (i.e. taking care that the new variable isn't one bound by an operator inside the scope of the original binding operator). We can hence freely use renaming of variables, for instance in order to graphically distinguish saturated arguments from open arguments of the verb.

3 Easy Linking Semantics

In what follows, I will use an Easy Linking Logic L_{link} which deviates from the systems above in its variables of type e. Apart from ordinary variables, we will use variables with abstract case labels like *nom, acc, dat, gen.* These include labels for prepositional cases like *by, for, to, with.* We will also assume that if the same preposition can be used with different thematic roles, and combines with the same verb twice, it will count as two different labels. Hence, $with_1$ in *with great care* counts as a different abstract prepositional case than the $with_2$ in *with a hammer* in the following sentence.

(1) *With great care, Joan opened the box with a hammer.*

Finally, I propose to use the labels *t, pl, e* for times, place and events. Hence, $\mathbf{Var} = \{v_{nom}, v_{acc}, v_{dat}, ..., e, t, pl, v_1, v_2, v_3, ...\}$. The exact choice of labels can be adapted if necessary. Likewise, we can assume that the linking logic L_{link} has more abstract case indices than we actually want to use of some specific semantic analysis. As before, formulae in L_{link} will be interpreted in suitable models M relative to finite assignments g.

What is the meaning of a verb in Easy Linking Semantics? I assume that the "conceptual" content of verbs in English should be captured in a variable-independent way as an n-place relation between objects, events, and worlds as usual. Hence, we will use conceptual denotations of verbs like the following:

$[[stab]]^c = ||\lambda x\lambda y\lambda e\lambda w.STAB(x,y,e,w)||^M$
$[[buy]]^c = ||\lambda x\lambda y\lambda z\lambda e\lambda w.BUY(x,y,z,e,w)||^M$
$[[sell]]^c = ||\lambda x\lambda y\lambda z\lambda e\lambda w.SELL(x,y,z,e,w)||^M$
$[[kiss]]^c = ||\lambda x\lambda y\lambda e\lambda w.KISS(x,y,e,w)||^M$
$[[rain]]^c = ||\lambda e\lambda w.RAIN(e,w)||^M$

These denotations can be viewed as conceptual values of English as well as German, Dutch, Russian or Japanese verbs, and they are not committed to any syntax-semantics interface. For the sake of illustration, I decided to use the Davidsonian format with an event argument for the verb. This is *not* what Beaver & Condoravdi propose, but Easy Linking Semantics is particularly attractive if you want to use events.

When verbs enter into the composition of a sentence, they change to their linking semantics. Each verbal argument is instantiated with a variable which carries the abstract case label that corresponds to the phrase that realizes this argument in sentences. Event and world argument will likewise be instantiated by specific event- and world variables. The following examples illustrate the step. I use [[...]] for the linking semantics of words in English, whereas ||...|| evaluates terms in L_{link} in a model M.

$[[stab]] \longrightarrow ||STAB(v_{nom}, v_{acc}, e, w)||^M$
$[[buy]] \longrightarrow ||BUY(v_{nom}, v_{acc}, v_{from}, e, w)||^M$
$[[sell]] \longrightarrow ||SELL(v_{nom}, v_{acc}, v_{to}, e, w)||^M$
$[[kiss]] \longrightarrow ||KISS(v_{nom}, v_{acc}, e, w)||^M$
$[[rain]] \longrightarrow ||RAIN(e, w)||^M$

These L_{link} terms each denote a set of partial assignments from variables with case labels into the model domain M. In using variables, I make the syntax look as similar to traditional logic as possible. In using variables with case indices, I endorse Beaver & Condoravdi's proposal that linking should be part of the semantic value of verbs rather than part of a trace mechanism at the syntax-semantics interface.

3.1 Saturation of arguments

We will assume that DPs carry their abstract case as a syntactic feature. These cases will enter the semantic composition; hence the denotation of DP_{case} is a tuple which consists of generalized quantifier (the same as in classical semantics) and its case label. In a sentence like the following, the subject DP *Ann* hence is interpreted as $\langle \lambda P.P(ANN), nom \rangle$.

(2) *Ann coughed*

Generally, a DP combines with a sister constituent XP as follows:

[[DP_{case} XP]] = $\langle$[[DP]] , *case* $\rangle \oplus$ [[XP]]
= [[DP]] ($\lambda v_{case}.\psi$) where ψ is an L_{link} term that codes the denotation of XP: [[XP]] = $||\psi||^M$.

Note that this definition does not depend on any specific term that is used to represent the meaning of XP. It can be shown that for any two terms Ψ_1, Ψ_2 which both code the meaning of XP, the result of the above lambda-abstraction is identical for both terms. The crucial insight is that all ways to code the meaning of XP must coincide in their free variables, and these always have to contain v_{case}. Equivalent codings will then yield the same logical object for the same variable assignments; which is all that is needed to ensure identical results of lambda-abstraction over v_{case}. Hence, the result of semantic composition is well-defined. Let me show an example.

[[*Ann*]] = $\langle ||\lambda P.P(ANN)||^M, nom \rangle$

[[*coughed*]] = $||COUGH(v_{nom}, e, w)||^M$

$[[\ \textit{Ann coughed}\ ]] = ||\lambda P.P(ANN)||^M(||\lambda v_{nom}.COUGH(v_{nom}, e, w)||^M)$
$= ||\lambda v_{nom}.COUGH(v_{nom}, e, w)(ANN)||^M$
$= ||COUGH(ANN, e, w)||^M$

The next example shows object quantification. The procedure is very similar to a Heim-Kratzer treatment though without any need to raise the object DP.

(3) *Ann read every book.*

$[[read]] = ||READ(v_{nom}, v_{acc}, e, t)||^M$

$[[\ \textit{every book}\ ]] = \langle||\lambda Q_{<e,t>}\forall x(BOOK(x) \rightarrow Q(x)||^M, acc\rangle$

$[[\ \textit{read every book}\ ]] = ||\lambda Q_{<e,t>}\forall x(BOOK(x) \rightarrow Q(x)||^M(||\lambda v_{acc}.READ(v_{nom}, v_{acc}, e, t)||^M)$
$= ||\forall x(BOOK(x) \rightarrow \lambda v_{acc}.READ(v_{nom}, v_{acc}, e, t)(x)||^M$
$= ||\forall x(BOOK(x) \rightarrow READ(v_{nom}, x, e, t)||^M$

$[[\ \textit{Ann read every book.}\ ]]$
$= \langle||\lambda P.P(ANN)||^M, nom\rangle \oplus ||\forall x(BOOK(x) \rightarrow READ(v_{nom}, x, e, t)||^M$
$= ||\lambda P.P(ANN)||^M(||\lambda v_{nom}.\forall x(BOOK(x) \rightarrow READ(v_{nom}, x, e, t)||^M)$
$= ||\lambda v_{nom}.\forall x(BOOK(x) \rightarrow READ(v_{nom}, x, e, t)(ANN)||^M$
$= ||\forall x(BOOK(x) \rightarrow READ(ANN, x, e, t)||^M$

The derivation of subject quantifiers is exactly parallel. And, of course, two quantificational DPs can combine in one sentence. The order of application will determine scope relations; I leave it to the reader to compute more examples. [2]

So far, I have not specified how world and event variables should be bound. As for the world variable, I refer the reader to the treatment of intensionality as proposed in Fintel & Heim (2007). Actually, their use of partial assignments as part of their metalanguage is the same as our use of partial assignments as part of the underlying logic. Hence, the present account is fully compatible with their intensional apparatus. Unlike the world index, the event parameter should be bound at some place. We can do so by making use of an existential closure operator ECL for the variable e at any point. Let Φ be some L_{link} term that represents the meaning of XP where e occurs free in Φ. $[[$ ECL XP $]] = ||\lambda e.\Phi \neq \emptyset||^M = ||\exists e\Phi||^M$. As before, existential closure is only defined if e occurs free in Φ, and yields the same result for all equivalent terms that could represent the meaning of XP.

Unlike DP arguments, the Davidsonian event variable is often used in order to collect several event modifications before it undergoes existential closure. This can be implemented in the present system by assuming that event modifiers leave the event argument as an open variable. The event argument can be bound either by ECL or by an overt quantifying expression, but not by an event modifier.

(4) *Ann read every book carefully*

1. $[[read]] = ||READ(v_{nom}, v_{acc}, e, t)||^M$
2. $[[carefully]] = \langle||\lambda P(CAREFUL(e) \wedge P(e))||^M, e\rangle$

[2] Longer draft with more examples available on request.

3. [[*read carefully*]] $= ||\lambda P(CAREFUL(e) \wedge P(e))||^M(||\lambda e.READ(v_{nom}, v_{acc}, e, t)||^M)$
   $= ||(CAREFUL(e) \wedge \lambda e.READ(v_{nom}, v_{acc}, e, t)(e))||^M)$
   $= ||(CAREFUL(e) \wedge READ(v_{nom}, v_{acc}, e, t))||^M)$

4. [[*ECL read carefully*]] $= ||\exists e(CAREFUL(e) \wedge READ(v_{nom}, v_{acc}, e, t))||^M)$

5. [[*every book*]] $= \langle||\lambda Q_{<e,t>}\forall x(BOOK(x) \rightarrow Q(x)||^M, acc\rangle$

6. [[*[ECL read carefully] every book*]] =
   $||\lambda Q_{<e,t>}\forall x(BOOK(x) \rightarrow Q(x)||^M(||\lambda v_{acc}.\exists e(CAREFUL(e) \wedge READ(v_{nom}, v_{acc}, e, t))||^M)$
   $= ||\forall x(BOOK(x) \rightarrow \exists e(CAREFUL(e) \wedge READ(v_{nom}, x, e, t))||^M$

7. [[*Ann read every book carefully*]] =
   $\langle||\lambda P.P(ANN)||^M, nom\rangle \oplus ||\forall x(BOOK(x) \rightarrow \exists e(CAREFUL(e) \wedge READ(v_{nom}, x, e, t)))||^M$
   $= ||\forall x(BOOK(x) \rightarrow \exists e(CAREFUL(e) \wedge READ(ANN, x, e, t)))||^M$

Alternatively, we can apply ECL after combining verb and object DP and get the following.

$$||\exists e(\forall x(BOOK(x) \rightarrow CAREFUL(e) \wedge READ(ANN, x, e, t)))||^M$$

Finally, the following example can be treated similarly if we replace ECL by the event quantifier *twice*.

(5) *Ann twice read every book carefully.*

The quantifier *twice* contributes $\langle||\lambda P\exists e_1 \exists e_2(e_1 \neq e_2 \wedge P(e_1) \wedge P(e_2))||^M, e\rangle$. Combination with any XP proceeds by lambda-abstraction over the event argument in the semantics of XP, and functional application. We can derive the following two readings.

$$\begin{aligned} ||\exists e_1 \exists e_2(e_1 \neq e_2 \wedge \forall x(BOOK(x) \rightarrow CAREFUL(e_1) \wedge READ(ANN, x, e_1, t)) \wedge \\ \forall x(BOOK(x) \rightarrow CAREFUL(e_2) \wedge READ(ANN, x, e_2, t)))||^M \\ ||\forall x(BOOK(x) \rightarrow \exists e_1 \exists e_2(e_1 \neq e_2 \wedge CAREFUL(e_1) \wedge READ(ANN, x, e_1, t)) \\ \wedge CAREFUL(e_2) \wedge READ(ANN, x, e_2, t)))||^M \end{aligned}$$

I will leave it at these illustrations of possible ways to use Linking Logic and Easy Linking Semantics in designing a semantics for fragments of English. The linking mechanism rests on the idea that clauses are closed domains in which every argument of the verb occurs only once. In this preliminary version, I will leave it open whether we will combine Easy Linking Semantics with indices in those cases where parts of a clause undergo long distance movement (or scope). Likewise, I will not detail the analysis of passives here. Passivation requires a different instantiation in linking semantics value of the verb which reflects the shifted grammatical roles. So far, I have demonstrated how Easy Linking Semantics can implement quantification, argument saturation and argument modification without binding the argument. Beaver & Condoravdi propose that modification is particularly needed for the time argument of verbs, and develop a particular way of shifting the value of the time arguments, which is effected by temporal modifiers. I will not take a stand as to whether this is the best, the only, or just one way of treating temporal modification but I want to show that it can be implemented in Easy Linking Semantics, too.

3.2 Functional shifting of arguments

Beaver & Condoravdi make repeated use of operations that shift the value of variable assignments. For instance, they use functions which map each set of points of time onto the maximal subset which entirely consists of time points in July, in order to test what happened in *July*). These functions serve a special purpose in their overall tense semantics which I will not recapitulate here. However, let us see how values of the time argument of verbs can be shifted by means of a simple function, e.g. the function which maps a time point τ to $\tau + 1$. I will generally use t for the time argument (variable) and greek letters for time points.

For the sake of simplicity, I will ommit the Davidsonian event argument in the present section. This is not to say that the technique is restricted to non-Davidsonian semantics. Consider the following formula in L_{link} which states that Anne coughed in w at t.

$||COUGH(ANNE, w, t)||^{M,g}$

The formula is defined for all g with the domain $\{t, w\}$ on times and worlds in M which are such that their extension to v_{nom} which map v_{nom} to $ANNE$ is in $[[cough]]$. Assume that we want to modify this formula in a way that ensures that Anne coughed at the time point that follows on $g(t)$. If you need a linguistic counterpart of this modification, you could imagine that it is contributed by *one moment later*. We can achieve this modification by lambda-abstracting over t and applying the resulting function to $(t + 1)$. The computation proceeds as follows:

1. $||COUGH(ANNE, w, t)||^{M,g} = 1$ iff $\mathrm{dom}(g) = \{t, w\}$ and all extensions of g to v_{nom} which map v_{nom} to $ANNE$ are in the denotation of *cough*.
2. $||\lambda t.COUGH(ANNE, w, t)||^{M,g'}$ is defined for all assignments where $\mathrm{dom}(g') = \{w\}$. It denotes that function **F** from time points τ to $\{\ 0,\ 1\}$ which maps τ to 1 exactly if the extension $g" := g' + \langle t, \tau\rangle$ is such that $||COUGH(ANNE, w, t)||^{M,g"} = 1$.

We will now apply this function to the term $t + 1$.

1. $||\lambda t.COUGH(ANNE, w, t)(t + 1)||^{M,g}$ is defined for our old assignments g with $\mathrm{dom}(g) = \{t, w\}$. (Note that t is again free in the new formula, because it was free in the argument term.)
2. According to our definition of functional application in SALo, we get
   $||\lambda t.COUGH(ANNE, w, t)(t + 1)||^{M,g}$
   $= ||\lambda t.COUGH(ANNE, w, t)||^{M,g_1}||(t + 1)||^{M,g_2}$
   where $g_1 = g|_{\{w\}}$ and $g_2 = g|_{\{t\}}$. This latter combination is equal to:
3. $\mathbf{F}(g_2(t) + 1)$, where **F** is $||\lambda t.COUGH(ANNE, w, t)||^{M,g_1}$. Given that $g_2(t) = g(t)$, this application yields *true* exactly if $ANNE$ coughs at time $g(t) + 1$. The application yields $false$ else.

Generalizing this mechanism, we can apply a functional shift to the tense argument t in a given formula. Like in all other cases, a modifier that involves the argument will first effect lambda abstraction over that argument. Next, this lambda term is applied to a term of the form $F(t)$. The argument place remains open; the formula is still defined for partial variable assignments g which have the respective variable in their domain (the time variable t in our example).

Functional shifts can be combined. We could decide to apply a function $G(t) := 2t$ in addition to $F(t) = t + 1$ (whatever sense this may make on times). The order of semantic application determines the order in which F and G operate on the tense argument. Remember that, in the following formulae, λt binds only the open variable t in ϕ.

$||\lambda t.\phi(t)(F(t))||^M = ||\phi(t+1)||^M$
$||\lambda t.\phi(F(t))(G(t))||^M = ||\lambda t.\phi(t+1)(G(t))||^M = ||\phi(2t+1)||^M$
$||\lambda t.\phi(t)(G(t))||^M = ||\phi(2t)||^M$
$||\lambda t.\phi(G(t))(F(t))||^M = ||\lambda t.\phi(2t)(F(t))||^M = ||\phi(2(t+1))||^M = ||\phi(2t+2))||^M$

Beaver & Condoravdi (2007) use functional composition in order to model stacked temporal modifiers of the kind *in the morning on Saturday for three weeks in 2008*. They exploit the fact that the syntactic order of temporal modifiers determines the order of application in the semantic representation. In their framework, certain ungrammatical orders of modifiers can be explained by the fact that the respective composition of functions is undefined or yields empty results.

4 Summary

The present paper spells out a type logic on partial variable assignments which combines the expressive power of classical type logic with full control over the open variables of each term. Full control over free variables can be a convenient feature in many contexts in natural language semantics. In a next step, I proposed to use type logics that use variables which are indexed with abstract case labels. This type logic can serve as the backbone of semantic analysis, offering a convenient way to activate and inactivate parameters in the semantic computation. I proposed a specific example of Easy Linking Semantics to illustrate the potential of the linking mechanism. It allows to define semantic combination of argument and operator in much the same way as the QR-based mechanism proposed in Heim & Kratzer (1998), but without quantifier raising at LF. This is particularly advantageous for verb arguments which do not meet their modifying or saturating phrase at a fixed place in the sentence. Such verb arguments include the time argument, space argument, but also the event argument, if you chose to operate in a traditional Davidsonian event semantics (Parsons, 1990). Easy Linking Semantics is likewise an attractive alternative framework in modeling the semantics of free word order languages. It is also suited to formulate the semantic component for grammars that do not make use of movement operations in the same way as GB and Minimalist grammars. Easy Linking Semantics, finally, is closely related to Linking Semantics as in Beaver & Condoravdi (2007). It offers a near-type logic way to refer to denotations in their linking structures and can be generalized to accommodate their event-free semantic fragment of English (see extended version).

5 References

Beaver, D. and C. Condoravdi. 2007. On the Logic of Verbal Modification. In M. Aloni, P. Dekker, F. Roelofson (eds.): *Proceedings of the Amsterdam Colloquium 2007*: 6 - 12.
Davidson, D. 1980[1967]. The Logical Form of Action Sentences. In: *Essays on Actions and Events*, pp. 105-122. Clarendon Press, Oxford.
von Fintel, K. and I. Heim. 2007. *Intensional Semantics.*
http://semantics.uchicago.edu/kennedy/classes/s08/semantics2/vonfintel+heim07.pdf
Haug, Dag, H. Eckhoff, M. Majer and E. Welo. 2009. Breaking down and putting back together again. Analysis and Synthesis of New Testament Greek. J. of Greek Linguistics 9(1): 56 - 92.
Heim, I. and A. Kratzer. 1998. *Semantics in Generative Grammar.* Malden: Blackwell.
Kratzer, A. 2002/in progress. *The Event Argument of the Verb.* Manuscript, Semantics Archive.
Parsons, T. 1990. *Events in the semantics of English.* Boston: MIT Press.
von Stechow, A. and A. Grønn. 2009. The (Non-)Interpretation of Subordinate Tense. Manuscript presented at Oslo University, Göttingen University.