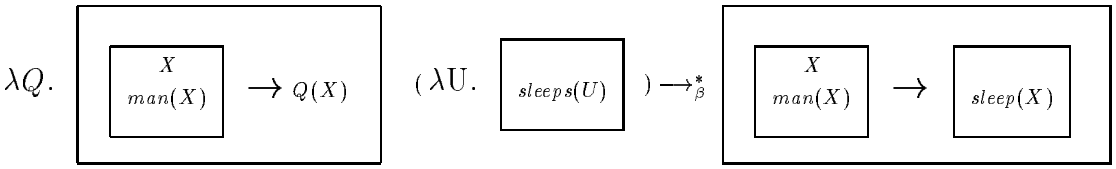
Towards a Dynamic Type Theory

Michael Kohlhase and Susanna Kuschert
Universität des Saarlandes

Over the past few years, there have been a series of attempts [Zee89, GS90, EK95, Mus94, KKP95] to combine the Montagovian type theoretic framework [Mon74] with dynamic approaches, such as DRT [Kam81]. The motivation for these developments is to obtain a general logical framework for discourse semantics that combines compositionality and dynamic binding.

Let us look at an example of compositional semantics construction in λ-DRT which is one of the above formalisms [KKP95]. By the use of β-reduction we arrive at a first-order DRT representation of the sentence `Every man sleeps.`



For the purposes of the logical analysis in this paper λ-DRT employs a linearised variant of this representation,

$$\delta\{\}.(\delta\{X\}.\mathsf{man}(X)) \rightarrow \delta\{\}.\mathsf{sleep}(X))$$

where discourse referents are introduced by a dynamic binding operator δ. Note that here β-reduction may lead to free variables to be captured, which in pure λ-calculus is *the* thing impossible. For example, if we want to construct `sleeps today`, as is done below, the free event variable E in the functor is captured by the discourse referent of the argument. In λ-DRT, $\otimes$ is the conjunction operator for DRSes that intuitively merges two DRSes by uniting the sets of discourse referents and that of conditions.

$$\begin{array}{l}(\lambda P.\lambda U.(P(U) \otimes \delta\{\}.\mathsf{time}(E) = \mathsf{today})(\lambda Y.\delta\{E\}.\mathsf{sleep}(E,Y))) \\ \longrightarrow^*_\beta \lambda U.((\delta\{E\}.\mathsf{sleep}(E,U)) \otimes \delta\{\}.\mathsf{time}(E) = \mathsf{today})\end{array}$$

Unfortunately, the above mentioned unified formalisms have failed so far to duplicate a key aspect of type theory that has lead to interesting linguistic analyses in computational linguistics. Type theory (or higher-order logic) is a two-layered formalism, where the algebraic content (the behaviour of higher-order functions) is neatly packaged into a formalism of its own, namely simply typed λ-calculus; while the logical content (the specific semantics of connectives and quantifiers) is built on top of it. Thus the use of type theory allows us to deal with the complexities of natural language semantics on two distinct levels: the simply typed λ-calculus provides the theory of β-reduction (which is the motor of compositionality) whereas the logical side of semantics is dealt with by a system that is rather like predicate logic. By focussing on known mechanisms for dealing with each of the two subsystems, it has proved possible to use type theoretic techniques for natural language processing systems.

- Higher-order unification [Hue75] solves equations in the simply typed λ-calculus and leads to analyses of Ellipsis [DSP91], and Focus [GK96, Pul94]. Note that these accounts are inadequate for the treatment of the logical structure, so they make insufficient predictions about quantifiers and connectives.
- First-order automated theorem proving [Fit90] is used to reason about the logical structure of natural language, presuppositions, and to integrate world knowledge into natural language semantics. Note that these approaches normally cannot capture higher-order aspects of the semantics like compositionality or underspecified (such as elliptic-) semantic elements.
- Only recently, logic formalisms for higher-order theorem proving [Koh95] have appeared that are generalizations of both higher-order unification and automated theorem proving. These can be used to integrate world knowledge into the unification-based approaches [GKvL96].

The goal of this paper is to lay the foundation for analyses like the above by providing an algebraic foundation of compositional formalisms for discourse semantics as an analogon to the simply typed λ-calculus. Just as that can be specialized to type theory by simply providing a special type o for truth values and postulating the quantifiers and connectives as constants with fixed semantics, the proposed *dynamic λ-calculus* $\mathcal{DLC}$ can be specialized to λ-DRT [KKP95] by essentially the same measures, yielding a much more principled and modular treatment of λ-DRT than before.

However, we expect the benefits from a clean separation of the structural and logic parts of compositional discourse semantics will not be restricted to this. In particular, $\mathcal{DLC}$ can serve as the formal basis for the development of higher-order unification algorithms for compositional formalisms for discourse semantics, which in turn can be expected to lead to dynamic analyses of ellipses, focus, corrections, ..., corresponding to those discussed above. First experiments with the formal system have shown that these will be more intuitive than those for the static case.

The proposed formalism $\mathcal{DLC}$ focuses on the interaction of dynamic binding (declaration of discourse referents), function abstraction and function application. Most of the burden of this is carried by an elaborate type system that takes into account structural properties of dynamic systems, such as the binding power and the accessibility relation.

Development of λ-DRT has shown that capturing of free variables by formulae containing dynamic binding constructs, i.e. the interaction between λs and δs, is a central theme. Thus the central idea for a type system for a dynamic λ-calculus is that types have to incorporate information about variables. In $\mathcal{DLC}$ this is represented by the fact that variable contexts (local functions that specify type information for variables) are contained in the types. So, if Γ is a variable context, and α is a type (which we call the *characteristic type*), then $\Gamma\#\alpha$ is also a type. In particular, types have to represent information about the free variables of a formula (those that can be captured) and about those that have dynamic binding power (that can capture free variables of other formulae). This distinction is made by annotating the variables with $-$ for the former and $+$ for the latter. If, for instance $\Gamma = [X^-\colon\alpha], [Y^+\colon\beta]$ and α is the base type o (for truth values) then $\Gamma\#o$ describes the set of propositions that contain a free variable X_α and dynamically binds a variable (introduces a discourse referent) Y_β. Types where the context Γ is empty are called *simple*. For non-empty contexts types are called *dynamic*, if Γ contains positive variables else *static*. Naturally, since $\mathcal{DLC}$ is a λ-calculus, the set of types is also closed under function types (i.e. $\alpha \to \beta$ is a type whenever α and β are).

$\mathcal{DLC}$-formulae are built up from a signature Σ by the set of inference rules (which we have slightly simplified for presentation) below, i.e. a formula $\mathbf{A}$ is a well-formed $\mathcal{DLC}$ formula of type α, iff the judgment $\mathbf{A}\colon\alpha$ is provable by these rules.

$$\frac{c \in \Sigma_\alpha}{c\colon\Gamma\#\alpha} \qquad \frac{X \notin \mathbf{Dom}(\Gamma)}{X\colon\Gamma, [X^-\colon\alpha]\#\alpha} \qquad \frac{\mathbf{A}\colon\Gamma\#\alpha \to \beta \quad \mathbf{B}\colon\Delta\#\alpha \quad \Gamma\big|\big|\Delta}{\mathbf{AB}\colon\Gamma,\Delta\#\beta}$$

$$\frac{\mathbf{A}\colon\Gamma, [X^-\colon\alpha]\#\beta}{(\lambda X_\alpha.\mathbf{A})\colon\Gamma\#(\alpha\to\beta)} \qquad \frac{\mathbf{A}\colon\Gamma\#\beta}{(\delta X_\alpha.\mathbf{A})\colon\Gamma, [X^+\colon\alpha]\#\beta}$$

In these rules we employ the convention that at the merging of contexts, positively signed variables overwrite negatively signed of the same type and name. Note that this overwriting is important for the last rule for *dynamic abstraction*, where $[X^-\colon\alpha]$ may have been present in Γ, indicating the presence of a free variable X of type α in $\mathbf{A}$. Also, by this overwriting effect we model variable capturing at function application, as shown in the example above. In the above rule for function application $\Gamma\big|\big|\Delta$ means that the two are equal on the common negatively signed variables and have no positive variables in common. The rule for *functional abstractions* discharges the free variable X^- from the context, since it is no longer free. The other rules do not change the context; that for variables only insists that the variable in question be declared in it.

In $\mathcal{DLC}$, the definitions of bound and free variables is straightforward; we distinguish functionally bound (by a λ) and dynamically bound (by a δ) variables. In the same way, we can reuse the standard λ-calculus reduction rules for β and η-conversion,

$$\frac{}{(\lambda X.\mathbf{A})\mathbf{B} \longrightarrow_\beta [\mathbf{B}/X]\mathbf{A}} \qquad \frac{X \notin \mathbf{Free}(\mathbf{A})}{(\lambda X.\mathbf{A}X) \longrightarrow_\eta \mathbf{A}}$$

and rephrase the α-conversion rule of the λ-calculus to define renaming of both functionally and dynamically bound variables thus

$$\frac{\mathbf{A}\colon\alpha \quad X \notin \mathcal{C}(\alpha) \quad Y\text{new}}{\mathbf{A} \longrightarrow_\alpha \mathbf{C}^X_Y(\mathbf{A})}$$

This new extended α-rule exploits the richer type system of $\mathcal{DLC}$ by using the following observation. A functionally bound variable is not free in $\mathbf{A}\colon\alpha$, if it does not occur in the type α. If a dynamic variable X does not occur on any level in $\mathbf{A}$'s type, this means that X occurs in a static subterm of $\mathbf{A}$ (i.e. one that has a static type) which does not contain a variable λ-abstracted in $\mathbf{A}$ through which X may still bind variables by variable capture — in other words, X cannot bind a free variable of the same name, neither through merging with another expression nor through β-conversion. Thus, in the above definition, $X \notin \mathcal{C}(\alpha)$ checks that X does not occur in the type of $\mathbf{A}$ and $\mathbf{C}^X_Y$ changes the bound variable X to Y. Note that for the existing systems for compositional discourse semantics mentioned above, α-renaming of dynamically bound variables constitutes a problem — in fact, it was from an attempt to understand full α-conversion that $\mathcal{DLC}$ evolved.

Finally, we introduce a logical constant ∂ which is a dynamification operator for binary relations $\mathbf{R}$. It is specified by the following reduction rules

$$\frac{}{\partial\mathbf{R}(\delta X.\mathbf{A})\mathbf{B} \longrightarrow_\delta \delta X.(\partial\mathbf{R}\mathbf{A}\mathbf{B})} \qquad \frac{\mathbf{A}\colon\alpha \quad \mathbf{B}\colon\beta \quad \alpha,\beta \text{ static}}{\partial\mathbf{R}\mathbf{A}\mathbf{B} \longrightarrow_\delta \mathbf{R}\mathbf{A}\mathbf{B}}$$

We have chosen ∂ as a primitive for $\mathcal{DLC}$ instead of the $\otimes$-operator, which can be derived from it, (see the discussion of λ-DRT below), since we hope that it will be possible to characterize the dynamic operators of other approaches with it in relation to known binary relations.

Given this, we arrive at λ-DRT by fixing the set of base types to $\mathcal{BT} = \{e, o\}$ (individuals and truth values) and defining a set of connectives such as conjunction, disjunction, negation and implication. In standard DRT, the binding properties of the discourse referents in the context must be defined in an accessibility relation. With the extended type system we are now able to express this relation directly within the syntax. For example, we define the negation operator to be of type $\neg\colon\Gamma\#o \to \Gamma^-$ where Γ^- is the set of negatively signed variables in Γ. Thus only those variables which are not bound dynamically in the argument of the negation — i.e. the free variables — are available outside the negated expression. In the same way we define a collection of implication operators of type $\Rightarrow_{\Gamma\Delta}\colon(\Gamma\#o) \to (\Delta\#o) \to ((\Gamma^-, (\Delta^- \setminus \Gamma^+))\#o)$. Note that (Δ^-/Γ^+) captures the accessibility relation between the two arguments: only those free variables of the second argument are available to the outside which are not bound by the discourse referents of the first argument. The dynamic conjunction operator $\otimes$ can be defined as $\partial\wedge$, and is thus generalized to merge both, DRSes and conditions.

Just as in standard λ-calculus, the type does not change through any of these reductions, under the type instantiations necessary for the well-formedness of the application which is to be β-reduced. In the above example, if Q gets the type $Q\colon(\Delta\#\alpha) \to o$, we can derive the type of the functor $\delta\{\}.(\delta\{X\}.\mathsf{man}(X)) \to Q(X)$ to be $(\Delta^- \setminus [X^+\colon\alpha])\#((\Delta\#\alpha) \to o) \to o$, meaning that the free variables of the expressions are those that come in by the argument Q less X. The argument has type $\lambda X.\delta\{\}.\mathsf{sleep}(X)\colon\emptyset\#\alpha \to o$. We need to instantiate Δ by $\emptyset$ to allow the application and get the correct type $\emptyset\#o$ for the result, meaning that in the representation of `Every man sleeps` there are no free variables and no dynamic binding power to the outside.

The semantics of $\mathcal{DLC}$ draws upon ideas and insights behind λ-DRT in [KKP95] and [Kus96]. There, intensionalization and dynamic denotations were found to complement each other in modelling the interaction of λs and δs which, recall, we want to understand. In λ-DRT intensionalisation

was used to guard variables by delaying evaluation of the current state wherever the current state was not available through a proper δ-abstraction. In generalization of this, dynamicity in $\mathcal{DLC}$'s semantics is achieved by an implicit intensionalization of *all* linguistic variables (i.e. not bound by λ-abstraction). Note that because of this we can do away with the $^\wedge$ and $^\vee$ of λ-DRT.

Given this, the domains $\mathcal{D}_\alpha$ of simple types α are just those known from simple type theory. Denotations of complex objects are functions from states (variable assignments) to values of simple types ($\mathcal{D}_{\Gamma\#\alpha} = \mathcal{F}(\mathcal{B}_\Gamma; \mathcal{D}_\alpha)$), where $\mathcal{B}_\Gamma = \bigcup \Delta \supseteq \Gamma\mathcal{F}(\mathbf{Dom}(\Delta); \mathcal{D})$ is the set of **Γ-states**. The core of $\mathcal{DLC}$'s semantics is the interpretation[1] of variables. $\mathcal{I}_\varphi(X)$ of a variable X is either $\varphi(X)$ in the case of λ-bound variables ($X \in \mathbf{Dom}(\varphi)$) or the function $\{t \mapsto t(X) \mid t \in \mathcal{B}_{[X:\alpha]}$ for linguistic variables, which represents the implicit intensionalization for a value of X. Furthermore, we have

$$\begin{aligned} \mathcal{I}_\varphi(\delta X.\mathbf{D}) &= \mathcal{I}_{\varphi - X}(\mathbf{D}) \\ \mathcal{I}_\varphi(\lambda X.\mathbf{D})@\mathcal{A} &= \mathcal{I}_{\varphi,[\mathcal{A}/X]}(\mathbf{D}) \\ \mathcal{I}_\varphi(\mathbf{AB}) &= \mathcal{I}_\varphi(\mathbf{A})@\mathcal{I}_\varphi(\mathbf{B}) \end{aligned}$$

Here, the interpretation of a dynamic abstraction needs no more than making sure that the variable X is not interpreted by φ due to the implicit intensionalization. With this, the interpretation of the functional parts is just as standard in the simple λ-calculus. The @-operator is the dynamic application operator which can be defined by means of the ∂-operator from the static application operator `appo` by @ = ∂`appo`. In this setup the dynamification operator ∂ raises to central importance; it is here that contexts are coordinated, as follows from the definition of its semantics below.

$$\mathcal{I}(\partial)@\mathcal{R}@\mathcal{A}@\mathcal{B} = \{a_1 \cup b_1 \mapsto \mathcal{R}(a_2, b_2) \mid a_1 \| b_1, a_1 \mapsto a_2 \in \mathcal{A}, b_1 \mapsto b_2 \in \mathcal{B}\}$$

We are convinced that $\mathcal{DLC}$ with its new typing system constitutes a powerful algebraic basis for systems combining functional and dynamic logics and that its further study may reveal more of the properties of the interplay of their features. In particular, we hope that in the same way as the types guide the higher order unification of standard λ-calculus, this type system may be useful for dynamic higher order unification.

References

[DSP91] M. Dalrymple, S. Shieber, and F. Pereira. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14:399 – 452, 1991.

[EK95] Jan van Eijck and Hans Kamp. Representing discourse in context. In J.F.A.K. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*. Elsevier Science B.V., 1995.

[Fit90] M. Fitting. *First-order Logic and Automated Theorem Proving*. Springer Verlag, Berlin, 1990.

[GK96] Claire Gardent and Michael Kohlhase. Focus and higher–order unification. In *The 16th International Conference on Computational Linguistics*, Copenhagen, Denmark, 1996. forthcoming.

[GKvL96] Claire Gardent, Michael Kohlhase, and Noor van Leusen. Corrections and higher–order unification. In *3. Konferenz zur Verarbeitung natuerlicher Sprache*, Bielefeld, Germany, 1996. forthcoming.

[GS90] J. Groenendijk and M. Stockhof. Dynamic Montague Grammar. In L. Kalman and L. Polos, editors, *Papers from the Second Symposium on Logic and Language*, pages 3 – 48. Budapest, Akademiai Kiadoo, 1990.

[Hue75] Gérard P. Huet. A unification algorithm for the typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[Kam81] H. Kamp. A theory of truth and semantic representation. In J. Groenendijk, Th. Janssen, and M. Stockhof, editors, *Formal Methods in the Study of Language*, pages 277 – 322. Mathematisch Centrum Tracts, Amsterdam, 1981.

[KKP95] M. Kohlhase, S. Kuschert, and M. Pinkal. A type-theoretic semantics for λ-DRT. *Proceedings of the Tenth Amsterdam Colloquium*, 1995.

[1] As usual, denotations of expressions are computed with respect to an assignment φ for λ-bound variables.

[Koh95] Michael Kohlhase. Higher-Order Tableaux. In R. Hähnle P. Baumgartner and J. Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods*, volume 918 of *Lecture Notes in Artificial Intelligence*, pages 294–309, 1995.

[Kus96] S. Kuschert. *Higher Order Dynamics: Relating operational and denotational semantics for λ-DRT.* CLAUS-Report 72, Universität des Saarlandes, 1996.

[Mon74] Richard Montague. The proper treatment of quantification in ordinary English. In R.H. Thomason, editor, *Formal Philosophy, Selected Papers of Richard Montague*, pages 247–270. New Haven and London, 1974.

[Mus94] R. Muskens. A compositional discourse representation theory. In P. Dekker and M. Stockhof, editors, *Proceedings of the 9th Amsterdam Colloquium*, pages 467 – 486. ILLC, Amsterdam, 1994.

[Pul94] S.G. Pulman. Higher order unification and the interpretation of focus. Technical Report CRC-049, SRI Cambridge, UK, 1994.

[Zee89] H. Zeevat. A compositional approach to DRT. *Linguistics and Philosophy*, 12:95–131, 1989.