Type Theory for Natural Language Semantics

Stergios Chatzikyriakidis and Robin Cooper
Centre for Linguistic Theory and Studies in Probability (CLASP)
University of Gothenburg

Keywords: semantics, simple type theory, constructive type theory, dependent types, record types, type theory with records (TTR)

Summary

Type theory is a regime for classifying objects (including events) into categories called types. It was originally designed in order to overcome problems relating to the foundations of mathematics relating to Russell's paradox. It has made an immense contribution to the study of logic and computer science and has also played a central role in formal semantics for natural languages since the initial work of Richard Montague building on the typed λ-calculus. More recently, type theories following in the tradition created by Per Martin-Löf have presented an important alternative to Montague's type theory for semantic analysis. These more modern type theories yield a rich collection of types which take on a role of representing semantic content rather than simply structuring the universe in order to avoid paradoxes.

1 History

At the beginning[1] of the 20th century Bertrand Russell was trying to overcome a serious problem that he had discovered in Gottlob Frege's formalization of Georg Cantor's set theory. The problem has subsequently come to be known as Russell's paradox and concerns a contradiction that arises when sets are allowed to be members of themselves and any intuitive property can be used to characterize a set. Russell pointed out that a set X, defined as the set of all sets that are not members of themselves, leads to a contradiction when one asks the question of whether X is a member of itself. If $X \in X$, then by definition $X \notin X$ holds, and if $X \notin X$ holds, then again by definition $X \in X$. Russell proposed to solve this problem by imposing a regime on set theory which did not allow sets to be members of themselves (Russell, 1903; Whitehead and Russell, 1925). He introduced different types: individuals, sets of individuals, sets of sets of individuals, and so on. An object of a given type could not be an element of an object of the same type — thus no set could be a member of itself.[2]

This led to what has come to be known as the *simple theory of types*, first formulated by Alonzo Church (1940) based on the λ-calculus which Church had introduced earlier. This

[1]For an account of the development of type theory from a logical perspective see Coquand (2015).

[2]A similar paradox has been shown by Girard (1972) to be operative for the original formulation of constructive type theory by Martin-Löf (1971). The interested reader is directed to Coquand (1986) for an in-depth analysis of the issue. It is worth mentioning that the idea of a stratified universe used today to avoid Girard's paradox, is already present in Russell's work.

formulation of the *simply typed* λ*-calculus* has had wide applications in computer science and, most importantly for our purposes, linguistic semantics, given that Montague's Intensional Logic (IL) (Montague, 1973, 1974) is an extension of Church's simply typed λ-calculus. The simple theory of types as embodied in Montague's IL[3] has dominated the field of formal semantics since its beginning in the early seventies (despite significant alternatives such as property theory, Chierchia and Turner, 1988; Fox and Lappin, 2005, and situation semantics, Barwise and Perry, 1983; Cooper, 1996).[4]

Starting with the work of Sundholm and Ranta (Sundholm, 1989; Ranta, 1991, 1994), approaches to natural language semantics have been developed relating to a kind of type theory developed by Per Martin-Löf (Martin-Löf, 1984; Nordström *et al.*, 1990) and known as *constructive type theory* (CTT) or *Martin-Löf type theory* (MLTT). The initial goal of CTT was to provide an alternative foundation for mathematics based on constructivism (see the discussion on constructivism in Section 1). Similar ideas have been pursued using Girard's system F, a version of the λ-calculus more expressive than the simply typed lambda calculus because of its ability to quantify over types (Girard, 1971; Girard *et al.*, 1990a).

From the perspective of linguistic semantics, there are three important differences between CTT and simple type theory:

rich selection of types CTT is a *rich* type theory in that it does not just provide a way of dividing a domain of mathematical objects into major ontological categories such as individuals, sets, sets of sets and so on as one finds in simple type theory. It also provides types which can correspond to the content of linguistic expressions. For example, there may be a type *Dog*, the type to which all dogs might belong or, as suggested by Ranta (1994), *Flight-by-Amundsen-over-the-North-Pole*, a type to which any event of a flight by Amundsen over the North Pole belongs. The richness of modern type theories as compared with simple type theory significantly changes the potential contribution of type theory to natural language semantics.

proof theoretic approach to types Montague's use of simple type theory was firmly rooted in the model theoretic tradition. Type theory was seen as a way of organizing set-theoretic objects which could serve as the extensions (or intensions) of phrases of natural language. In contrast CTT comes from the proof theoretic tradition in logic which concentrates on inference rules defined on the syntax of logical expressions. If we are interested in linguistically relevant types like *Dog* and *flight-by-Amundsen-over-the-North-Pole* this might, for example, lead us to try to characterize what follows from something being of the type *Dog* (for example, also being of the type *Mammal*) or *flight-by-Amundsen-over-the-North-Pole* (for example, also being of the types *Event* and *has-agent-Amundsen*).

constructivism (intuitionism) Constructive mathematics claims that it is necessary to construct a mathematical object in order to prove that it exists. Intuitionism (which is often equated with constructivism) is a kind of constructivism where the construction of mathematical objects (like sets) is founded on the intuitions of mathematicians. CTT as its name might suggest provides an intuitionistic approach to type theory founded on the idea that to show the existence of an object, you have to construct, or at least provide a method of constructing, the object. Montague's use of simple type theory on the other

[3]Or Gallin's (1975) Ty2, which differs from IL in that it has expressions which denote objects of Montague's type s for pairs of possible worlds and moments of time.

[4]For a detailed but still short introduction to the history of type theory, the interested reader is directed to Coquand (2011). We do not give a detailed exposition of the history of type theory here.

hand is platonistic in that it countenances the existence of abstract objects and sets that we do not know how to construct. This contrast is related to issues concerning computation such as decidability and the related issue of whether semantics is mathematics or psychology (Partee, 1979, 2014). As we shall see, it is not a straightforward choice between intuitionism as a psychological view of semantics and platonism as providing a view of semantics as mathematics.

There is now a considerable body of work which applies this kind of type theory to natural language semantics, including Ranta (1994); Luo (2010, 2011); Chatzikyriakidis and Luo (2014b); Boldini (2000); Piwek and Krahmer (2000); Retoré (2013); Bekki (2014); Ranta (2015), as well as the Grammatical Framework (GF, Ranta, 2011), a computational grammar formalism based on type theory, showing that type-theoretic semantics generalizes to a multilingual setting where it can be used, for example, for defining translation interlinguas. Since the turn of the century a third strand of type theoretic approaches has arisen which tries to combine the best of the model theoretic approach of Montague with the type theoretical techniques from CTT. This includes work on Type Theory with Records (TTR, Cooper, 2005a,b, 2012; Cooper and Ginzburg, 2015) and work by Grudzinska and Zawadowski (2014).

2 Choices in Type Theory

In this section we discuss a number of basic choices made in the various versions of type theory that are relevant for linguistic semantics.

2.1 Basic Types

There is a fundamental difference between simple type theory with a very limited number of basic types and CTT which makes use of a multitude of basic types (or at least leaves the type theory open to adding new basic types). As formulated by Church (1940) the simply typed lambda calculus employed two basic types: type ι for individuals and o for propositions. Montague (1973) uses e (for entities or individuals), corresponding to Church's ι, and t (for truth values). Truth values serve as the extension of propositions which for Montague belong to a complex type (see Section 2.2).

The central idea in simple type theory is that you can build all the types you need from two basic types corresponding to individuals and propositions/truth-values and all possible functions which can be constructed from these two (see Section 2.3). Variants have been proposed in the semantics literature which have more than two basic types. Gallin (1975) proposed a type theory (*TY2*) that adds a third basic type, s (for world-time pairs).[5]

If you want to make subdivisions within these types, for example, if you want to talk about different kinds of individuals like dogs and mammals, then you do this by using function types of some kind, for example, functions from individuals to propositions (Church) or truth-values (Montague). Alternatively, one can introduce *sorts* which correspond to subsets of the elements of types (Kohlhase, 1992, 1994). For example, one can structure the domain of individuals to

[5]Actually Gallin's proposal does more than that. It reformulates Montague's IL in a simpler way via using classical higher-order logic and Henkin models. There are good reasons to use Gallin's *TY2* system instead of Montague's IL. The most important one is simplicity: Montague's IL is an extension of Church's logic which however is more complicated and also "destroys" some of the fundamental mathematical properties of Church's logic (e.g. the Church-Rosser property, see (Muskens, 1996)). Gallin's proposal does not have these unwanted side-effects and, most importantly, can express everything that can be expressed in IL.

partially ordered subsets and as such have sorts like *MAN*, *HUMAN*, ANIMAL etc. Thomason (1980) also introduced a type theory with three basic types: e (entities), t (truth-values) and p (propositions). In this way, it is a kind of merge of Church and Montague. (See Section 2.2 for more discussion.)

In rich type theories like CTT there is no restriction on the basic types one can use. In this way the types can do the work that sorts might do in a simple type theory. If one wants one can choose to have a basic type *Dog* to which any dog belongs or one can choose to define a complex type *Dog*, constructed from other type theoretic elements. While this freedom also exists in TTR in practice relatively few basic types have been used such as *Ind* (the type of individuals) and *RecType* (the type of record types). (Types are regarded as first class objects and can themselves belong to types. A technique called *stratification* is used in order to avoid this giving rise to the paradoxes (Turner, 2005).) Thus in TTR *Ind* does the work of Church's ι or Montague's e whereas in CTT there is normally not such a type but (in type theories developed for linguistic applications) more contentful basic types like *Dog* and *Boy*. In Section 2.2 we will discuss what corresponds to Church's o and Montague's t.

2.2 Propositions

While Church (1940) introduces the type o for propositions he does not make any commitment to the nature of propositions. He says, p. 57, "We purposely refrain from making more definite the nature of the types o and ι, the formal theory admitting of a variety of interpretations in this regard." For Montague (1973) the situation is different. For him propositions are constructed as functions from pairs of a possible world and a moment of time to truth-values (i.e. the characteristic function of a set of world-time pairs). The type of propositions for Montague is the complex type $\langle s, t\rangle$ (see the section on complex types, 2.3). Thus the type theory is being used here to encode a particular theory of what propositions are. While it is, of course, desirable for a semantic theory of natural language to include a theory of what propositions are, there is a well-known problem with the idea that propositions are sets of world-time pairs. It means that you cannot have two distinct logically equivalent propositions. If p and q are logically equivalent, then they are true at the same worlds and times, which according to this theory would mean that they are the same proposition. Thus it becomes puzzling to give an account of how somebody could learn about the logical equivalence of p and q or how it could be that somebody could believe p but not believe q. In the linguistic literature this problem is often referred to as the problem of *hyperintensionality*, "hyper-" in the sense of going beyond the kind of intensionality introduced by considering propositions as sets of world-time pairs. For an account of the problem and a survey of some attempts to solve the puzzle see Muskens (2007); Fox and Lappin (2005), Chapters 1–2. It was to address this problem that Thomason (1980) went back to Church's idea of having a basic type for propositions to avoid the commitment to propositions as sets of possible worlds and times. This strategy has the consequence, of course, that if you have a theory of propositions, it has to reside elsewhere that in the type theory you use.

Rich type theories like CTT follow a dictum which derives from the intuitionistic logic of the 1930's: "propositions as types" also known as the Curry-Howard Correspondence.[6] For an introduction to this idea from the perspective of computer science see Wadler (2015) and for a linguistic perspective see Ranta (1994). The idea is related to the richness of these type systems. If you have a type (of events or situations) *Flight-by-Amundsen-over-the-North-Pole*

[6]Or the Curry-Howard Isomorphism.

then you can use this type as the proposition represented by the sentence *Amundsen flew over the North Pole* (ignoring issues of tense). The proposition is true just in case there is an event of the type, that is, the type is non-empty or *inhabited* in Martin-Löf's terminology. If you have an *extensional* type theory (one where types are characterized by the sets of objects that belong to them, their *witnesses* or inhabitants, that is, you cannot have two distinct types with exactly the same witnesses), you will fall into a similar trap as Montague did. But if you have an *intensional* type theory (one where types are characterized independently of their witnesses, that is, you *can* have two distinct types with exactly the same witnesses), then you have an approach to hyperintensionality which is related to that proposed by Thomason (and indeed also to property theory as discussed by Fox and Lappin, 2005). If we use this strategy to treat natural language then it is natural to treat types as first-class citizens in the sense that they can be arguments to predicates. Thus in order to treat a sentence like *Sam believes that Amundsen flew over the North Pole* we could use the type corresponding to *Amundsen flew over the North Pole* as the second argument of the predicate 'believe'. This means that a rich type theory plays a role in a linguistic theory that is not played by simple type theory. Both kinds of type theory provide a regime for organizing the universe of semantic objects according to the types. Rich type theories in addition have the possibility of contributing the types themselves to the universe of semantic objects.

Since all types either have or do not have a witness (or, in the case of the probabilistic type theory introduced in Cooper *et al.*, 2015a, have a certain probability of having a witness), they all have the potential to be used as a proposition (true if there is a witness and false otherwise). Martin-Löf introduced the term *proof object* for witnesses of types used as propositions. In some versions of type theory only some of the types are identified as propositions. That is, they are witnesses of a type *Prop*, a type of types, of which only the types which count as propositions are witnesses. This distinction is not drawn in TTR, although in practice since the contents of declarative sentences are always record types (see Section 2.3), in linguistic semantics using TTR it is record types that correspond to propositions expressed by natural language. Since record types also can be thought of as modelling discourse representation structures (DRSs) this is a bit like a semantic theory where DRSs play the role of propositions, whereas in standard discourse representation theory propositions would be Montague style propositions which serve as the interpretation of a DRS.

2.3 Complex Types

In this section we will discuss various kinds of complex types which have been proposed.

2.3.1 Function types

In simple type theory the idea is to choose a small number of basic types and then close the set of types under function type formation. Thus Church's set of types could be given by the following recursive definition (using more modern notation for function types than Church's notation):

1. ι and o are types
2. If α and β are types, then $(\alpha \to \beta)$ is a type
3. Nothing else is a type

Here the type $(\alpha \rightarrow \beta)$ is the type of (total) functions from objects of type α to objects of type β. An example of a type in this system would be $(\iota \rightarrow (\iota \rightarrow o))$, the type corresponding to two-place predicates.

Montague's set of types is given by the following definition (again using the arrow-notation for function types rather than Montague's original):

1. e and t are types
2. If α and β are types, then $(\alpha \rightarrow \beta)$ is a type
3. If α is a type, then $(s \rightarrow \alpha)$ is a type
4. Nothing else is a type

Here the type $(s \rightarrow \alpha)$ is the type of (total) functions from world-time pairs to objects of type α. The status of s in Montague's system is a little strange. He presumably did not introduce it as a basic type (as Gallin did, as discussed in Section) because he did not envision expressions denoting world-time pairs and did not want functions with world-time pairs in their range.

In characterizing a type system there are two important things we have to do: (1) we have to say what types there are (as we did in the recursive definitions above) and (2) we have to say what kind of objects are witnesses for the types. In the case of basic types we may not have much to say about the nature of the witnesses, or even assume, as Church did in the quotation we gave in Section 2.2, that the type theory will be used in different domains and that the witnesses for the basic types will be given by something like a model or an interpretation for the type theory. In the case of the complex types, however, we normally require an exact characterization of the witnesses for the types, given an assumption about what witnesses are associated with the basic types. The two tasks (1) and (2) become particularly important in rich type theories like for example type theories within the tradition of Martin-Löf or Girard's system F or systems like TTR where in addition to function types there are many other kinds of complex types. The plethora of available complex types is another source of richness in these type theories. Here we will look at a few key examples of the additional complex types that have been introduced.

2.3.2 Dependent types

A dependent type is something that requires one or more arguments to yield a type (as opposed to an object belonging to a type). We can think of it as a function which takes object(s) of one or more types and yields a type as a result. For example, in a classical Montague semantics a verb-phrase corresponds to a function from individuals (objects of type e) to a truth-value (an object of type t). In TTR a verb-phrase instead returns a type (of situation in which the argument to the function has a certain property, that is, not a truth-value or a situation, but a type). Thus verb-phrase interpretations in TTR are dependent types. In TTR dependent types are encoded explicitly as functions. In the literature on CTT they are often referred to as a family of types that is indexed by objects of one or more types. In CTT a common use of dependent types of this kind is as n-ary predicates which take n arguments to form a type. For example, 'see' would be a dependent type which requires two arguments to form a type such as 'see(a,b)'. If we follow Ranta's suggestion which we discussed in Section 2.2, then a witness for this type would be an event, intuitively an event that proves that a sees b.

2.3.3 Dependent function types (Π-types)

The term "dependent type" is also (somewhat confusingly perhaps) used in the type theory literature to refer to complex types which are constructed using a dependent type in the sense of Section 2.3.2. Thus, for example, there are *dependent function types* in which the type of the range of the function depends on which particular object you choose in the domain of the function. Suppose, for example, we have a type *Girl* whose witnesses are girls and we want to define the type of mappings from girls, a, to an event where a runs. One kind of notation for this type corresponds to the notation for simple function types: $((x{:}\mathit{Girl}) \rightarrow \mathrm{run}(x))$. These dependent function types are also called Π-types, using the notation $\Pi x{:}\mathit{Girl}\,.\,\mathrm{run}(x)$. A Π type degenerates into a normal function type in the non-dependent case, when the second type does not depend on the value of the first. Somewhat in more detail and closer to the formulations one finds in CTT, when A is a type and P is a predicate over A, $\Pi x{:}A.P(x)$ is the dependent function type that, in the embedded logic, stands for the universally quantified proposition $\forall x{:}A.P(x)$.

According to the Curry-Howard Correspondence such Π-types correspond to universally quantified propositions like *every girl runs*. The intuition is this: if the type corresponds to a true proposition then there must be a witness for the type. Such a witness will be a (total) function from girls, a, to a witness for the type 'run(a)', an event where a runs. If there is such a function then every girl runs and if there is not such a function then it is not true that every girl runs.

Π types can also be useful for characterizing the compositional mechanisms of natural language semantics. Suppose that *Prop* is the type of types which can serve as propositions. We could characterize these as the types which can be the content of a declarative sentence. Thus $\Pi x{:}\mathit{Girl}\,.\,\mathrm{run}(x)$ would be of this type (in symbols: $\Pi x{:}\mathit{Girl}\,.\,\mathrm{run}(x) : \mathit{Prop}$). What then would be the type of the content of an intransitive verb like *runs*? If we have a type of individuals like Church's ι or Montague's e, a type which we will represent as *Ind*, following the convention in TTR, we could say that the type of intransitive verb contents is $(\mathit{Ind}\rightarrow\mathit{Prop})$, that is any intransitive verb content will be a function from individuals to propositions. However, in most if not all type theories following Martin-Löf (except for TTR) there is no type *Ind* as such, but rather a collection of types such as *Girl*, *Boy*, *Dog* which can serve as the content of common noun phrases. In case one wants to have the predicate over the whole universe of common nouns, a Π type can be used to do that. Let us follow Luo (2012); Chatzikyriakidis and Luo (2013) in calling this type CN.[7] Thus $\mathit{Girl} : \textsc{cn}$ and so on. We will in general represent the content of a phrase α as $[\![\alpha]\!]$. Thus $[\![\mathrm{girl}]\!] = \mathit{Girl}$ and $[\![\mathrm{girl}]\!] {:} \textsc{cn}$. Given such a treatment of common nouns, what should the type of $[\![\mathrm{runs}]\!]$ be? One option is to say that it is a function from types $A{:}\textsc{cn}$ to a function from objects of type A to propositions, that is $[\![\mathrm{runs}]\!] : \Pi A{:}\textsc{cn}\,.\,(A \rightarrow \mathit{Prop})$.[8] We can similarly treat the types for quantifiers and VP-adverb contents respectively as:[9]

$\Pi A : \textsc{cn}.\ (A \rightarrow \mathit{Prop}) \rightarrow (A \rightarrow \mathit{Prop})$

and

$\Pi A : \textsc{cn}.\ (A \rightarrow \mathit{Prop}) \rightarrow \mathit{Prop}$

[7]In this kind of type theory such a type of types is often called a *universe*, see Luo (2012); Chatzikyriakidis and Luo (2013). For a recent detailed defense of the common nouns as types view, see Chatzikyriakidis and Luo (2017c).

[8]An alternative treatment is to say that $[\![\mathrm{runs}]\!]$ is of type $(\mathit{Animal} \rightarrow \mathit{Prop})$ where for all $A{:}\textsc{cn}$ A is a subtype of *Animal*. See Section 2.4 for a discussion of subtyping.

[9]A similar use is found in the work of Retoré and colleagues Retoré (2013); Richard and Retoré (2012), though within Girard's system F Girard (1971); Girard *et al.* (1990a). For constructive approaches to generalized quantifiers see Sundholm (1989); Tanaka *et al.* (2013).

2.3.4 Σ-types

According to the Curry-Howard Correspondence existential quantification corresponds to products or disjoint unions. The constructor/operator Σ is a generalization of the Cartesian product of two sets that allows the second set to depend on values of the first. If A is a type and B is an A-indexed family of types (that is a function from objects of type A to types), then $\Sigma x{:}A \,.\, B(x)$, is a type, whose witnesses are pairs (a, b) such that a is of type A and b is of type $B(a)$. So, for example, the type which serves as the content for *a girl runs* could be $\Sigma x{:}\mathit{Girl} \,.\, \mathrm{run}(x)$. A witness for this type would be a pair consisting of a witness, a, for *Girl* and a witness for $\mathrm{run}(a)$, that is a proof that a runs, which in turn can be seen as an event where a runs.[10]

As the witnesses of Σ-types are ordered pairs, they can be used not only as the contents of existentially quantified sentences but also in other cases where such an ordered pair might seem appropriate. An example which goes back at least as far as Ranta (1994) is the interpretation of modified common nouns. (For recent developments of the idea see Luo, 2011; Chatzikyriakidis and Luo, 2013.) A modified common noun like *tall girl* can be interpreted as the following Σ type (where the type of $[\![\mathrm{tall}]\!]$ is $\Pi A{:}\textsc{cn} \,.\, (A \to \mathit{Prop})$, the same type as discussed for $[\![\mathrm{runs}]\!]$ in Section 2.3.3):

$[\![\text{tall girl}]\!] = \Sigma x : \mathit{Girl} \,.\, [\![\text{tall}]\!](x)$

Witnesses of this type are pairs consisting of a girl and a proof that the girl is tall for a girl. For intersective adjectives we can think of the content of the adjective as being a constant function which always returns the same function no matter what type we give it as argument. An example, might be *Swedish*.[11] The modification process is iterable. Thus $[\![\text{tall Swedish girl}]\!]$ could be the following: first we define Swedish girl:

$[\![\text{Swedish girl}]\!] = \Sigma x : \mathit{Girl} \,.\, [\![\text{Swedish}]\!](x)$

Then, we have the following:

$[\![\text{Tall Swedish girl}]\!] = \Sigma y : [\![\text{Swedish girl}]\!] \,.\, [\![\text{tall}]\!](y)$

A witness of this type would be again a pair, but this time the first projection is a Σ type. More concretely, the pair consists of a proof that y is a Swedish girl and a proof that y is tall (given that it is a Swedish girl (you might take this to mean that y is tall for a Swedish girl)).[12] For further discussion and the use of subtyping for adjectival modification see Chatzikyriakidis and Luo (2013).

We can use Π-types and Σ-types together in order to obtain an analysis of donkey anaphora (Sundholm, 1989; Ranta, 1994) which is equivalent to the original analysis in Discourse Representation Theory (Kamp, 1981). In order to do this we will need projection operations π_1 and π_2 on order pairs so that $\pi_1(a, b) = a$ and $\pi_2(a, b) = b$. Following Ranta's (1994) presentation, we use Σ-types both for existential quantification and for common nouns as discussed in Section 2.3.4. Thus the content of *man who owns a donkey* is the type:

[10]When A and B are types, we can also write $\Sigma(A, B)$ to represent the product type whose witness are pairs (a, b) where $a{:}A$ and $b{:}B$.

[11]Although $[\![\text{Swedish}]\!]$ for *Girl* might return a function which yields true propositions for individuals of Swedish nationality whereas for *Academic* it might return a function which yields true propositions for individuals working at a Swedish university.

[12]It is important to note here that this treatment does not have anything to say on the different and context dependent standards of measurement for every type. As such, nothing in this treatment can guarantee that the standard contextual degree parameter for type *girl* will be different than the standard contextual degree parameter for another type, say *boy*. What this treatment achieves is restricting the domain the adjective applies depending on the case. In this way, unwarranted inferences like skilful man ⇒ skilful surgeon are blocked. A welcome consequence of this is that this basic inferential property is handled via typing and not via a meaning postulate. Furthermore, nothing in this account precludes incorporating grades. One way to do that is presented in Chatzikyriakidis and Luo (2017a) using indexed types. The interested reader is directed there for more details.

$(\Sigma x{:}\mathit{Man})(\Sigma y{:}\mathit{Donkey})(\mathrm{own}(x, y))$

Note that a witness, z, for this type will be an ordered pair $(m, (d, s))$ where $m : \mathit{Man}$, $d : \mathit{Donkey}$ and $s : \mathrm{own}(m,d)$. Thus $\pi_1(z) = m$ and $\pi_1(\pi_2(z)) = d$. The content of *every man who owns a donkey beats it* is now the following Π-type:[13]

$\Pi z{:}(\Sigma x{:}\mathit{Man})(\Sigma y{:}\mathit{Donkey})(\mathrm{own}(x, y))(\mathrm{beat}(\pi_1(z), \pi_1(\pi_2(z))))$

2.3.5 Record types

TTR uses record types in place of Σ-types. We will make the relationship between the two clear below, after we have explained what a record type is. There is a large body of work on adding records to CTT, Tasistro (1997); Betarte (1998); Betarte and Tasistro (1998), among many others. Records and record types are feature structure-like objects which have been used in TTR for modelling discourse representation structures (Kamp and Reyle, 1993), frames (Fillmore, 1982), situation types (Barwise and Perry, 1983) and dialogue gameboards (Ginzburg, 2012). (See Cooper, 2005b, 2015; Ginzburg, 2012 for discussion.) In this discussion we will use the graphical notation for record types from TTR which displays their similarity to feature structures.

We first consider a record type corresponding to the Σ-type $\Sigma x{:}\mathit{Girl} \,.\, \mathrm{run}(x)$:

$$\begin{bmatrix} \mathrm{x} & : & \mathit{Girl} \\ \mathrm{e} & : & \mathrm{run(x)} \end{bmatrix}$$

A record type consists of a set of fields[14] each of which is a pair consisting of a label (such as 'x' and 'e' in this example) and a type, which may be a dependent type like 'run(x)' in this example. The label and the type are displayed with a ':' between them. A record will be of this type just in case it has two fields with the same labels as the type and objects of the types in the respective fields. Thus an example of a record of this type would be:

$$\begin{bmatrix} \mathrm{x} & = & a \\ \mathrm{e} & = & s \end{bmatrix}$$

where $a{:}\mathit{Girl}$ and $s{:}\mathrm{run}(a)$, that is, s is a proof that a runs, which can be understood as saying that s is a situation in which a runs. The fields in records are displayed with '=' between the label and the object. An important aspect of the theory of record types is that a record may have more fields than required by the record type and still be a witness for the type. Thus, for example, the following record would also be of this type:

$$\begin{bmatrix} \mathrm{x} & = & a \\ \mathrm{y} & = & b \\ \mathrm{e} & = & s \end{bmatrix}$$

where a and s are as before and b could be of any type. We will return to this in Section 2.4.

The correspondence between the Σ-type and the record type is that they both have witnesses which are a pair consisting of a girl and a proof that she runs. In the case of the Σ-type the pair is ordered and in the case of the record type the pair is indexed by the labels 'x' and 'e'. In place of the projections π_1 and π_2 which we associated with ordered pairs, we have projections based on labels for records. If r is a record containing the label ℓ, then $r.\ell$ is the object occurring in the field with label ℓ. Thus if r is either of the records above, $r.\mathrm{x}$ is a and $r.\mathrm{e}$ is s. This system of labelling together with the fact that we can have any finite number of fields in both records and record types yields the possibility of using objects with flatter structures than are provided by Σ-types. Thus a type corresponding to *a man owns a donkey* could be:

[13]Ranta's original account has been argued to suffer from compositional issues. For an approach that remedies these issues, see Bekki (2014).

[14]In TTR a record type is an unordered set of fields. In other approaches the fields are ordered.

$$\begin{bmatrix} \text{x} & : & \textit{Man} \\ \text{y} & : & \textit{Donkey} \\ \text{e} & : & \text{own(x,y)} \end{bmatrix}$$

Using this machinery we can have the following function type, using the notation for dependent function types introduced at the beginning of Section 2.3.3. It corresponds to the Π-type for donkey anaphora presented in Section 2.3.4:

$$(r{:}\begin{bmatrix} \text{x} & : & \textit{Man} \\ \text{y} & : & \textit{Donkey} \\ \text{e} & : & \text{own(x,y)} \end{bmatrix}) \rightarrow \text{beat}(r.\text{x},r.\text{y})$$

2.4 Subtyping

In simple type theory it is standardly assumed that there will be no overlap between the sets of objects of two distinct types. That is, no object may be of more than one type. This assumption is normally carried over to rich type theories like CTT. However, when we include the kind of types needed for linguistic semantics, this constraint seems untenable. For example, if we have types corresponding to common nouns it seems clear that intuitively a witness for *Man* is also a witness for *Person* and in turn a witness for *Person* is a witness for *AnimateBeing*, thus giving rise to the kind of type hierarchies that are familiar to linguists from feature based systems and also work on ontologies in artificial intelligence. If intuitively any witness for T_1 must also be a witness for T_2 then T_1 is said to be a subtype of T_2 (in symbols, $T_1 \leq T_2$ or $T_1 \sqsubseteq T_2$). Of course, an object can also intuitively belong to two types which do not stand in the subtype relation. For example, a single person can be both of type *Man* and type *Doctor* although neither type is a subtype of the other.

This intuitive notion of subtyping, which is what is found in systems such as feature structure logic is known as *subsumptive subtyping.* In a proof-theoretical approach subsumptive subtyping gives rise to the following inference rule:

$$\frac{a{:}A, A \leq B}{a{:}B}$$

Subsumptive subyping is reflexive and transitive and basically allows one to use a term of type A in a context where a term of type B is required instead just in case $A \leq B$.

An alternative to subsumptive subtyping is *coercive subtyping* (Luo, 2011, 2010, 2012). This allows us to maintain that each object has exactly one type and in particular allows for two types in the subtype relation to have different ways of constructing canonical objects.[15] In coercive subtyping when A is a subtype of B this is associated with a designated coercion function, c, from type A to type B. Thus, technically, an object, a of type A need not be identical with an object, b in type B but the function, c such that $c(a) = b$ shows that b is a “counterpart” of a in type B. In this way a can be a canonical object of type A and $c(a)$ a canonical object of type B even though the definitions of canonicity for the two types are distinct. Similar mechanisms of subtyping are also found in Retoré (2012); Richard and Retoré (2012); Retoré (2013); Asher and Luo (2012); Asher (2011); Callaghan and Luo (2000); Barras *et al.* (2000).

Record types can be used (as in, for example, Tasistro, 1997; Betarte, 1998; Betarte and Tasistro, 1998; Coquand *et al.*, 2004) can be used to introduce subsumptive subtyping in a

[15]For example, $11-4{:}Nat$ and $4+3{:}Nat$ compute to the same canonical object $7{:}Nat$. The notion of canonical object is important in a proof theoretic approach where you do not have a model which shows that differents terms have the same referent.

restrictive and controlled way. As pointed out in Section 2.4 a record, r, is judged to be of a record type, T, just in case there are fields in r with the same labels as the fields in T and the objects in those fields in r are of the types in the corresponding fields in T. There may be more fields in r with labels that are not mentioned in T. This means, for example, that any record of type

$$\begin{bmatrix} \mathrm{x} & : & \mathit{Man} \\ \mathrm{y} & : & \mathit{Donkey} \\ \mathrm{e} & : & \mathrm{own(x,y)} \end{bmatrix}$$

will also be of type

$$\begin{bmatrix} \mathrm{x} & : & \mathit{Man} \\ \mathrm{y} & : & \mathit{Donkey} \end{bmatrix}$$

and also of type

$$\begin{bmatrix} \mathrm{x} & : & \mathit{Man} \end{bmatrix}$$

That is, the first type is a subtype of the second, which in turn is a subtype of the third. If one construes these types as types of situations, as is done in TTR, this subtyping relationship can be glossed as saying that any situation in which a man owns a donkey is a situation in which there is a man and a donkey which in turn is a situation in which there is a man. In TTR much of the work on subtyping is done by record types, although non-record types are also allowed to enter into the subtype relation. (For example, you might want to treat *Man* and *Person* as basic types and say that the first is a subtype of the second.) There is no requirement that objects belong to exactly one type. Ending our discussion on subtyping, it is important to add that systems of subtyping exist for simple type theory as well as other systems like for example system F. The interested reader is encouraged to check Cardelli (1988) and Cardelli *et al.* (1991); Retoré (2013) respectively for more details.

2.5 Contexts

There is a notion of context in CTT which has been used to analyze various aspects of linguistic semantics such as text and discourse representation (Ranta, 1994), linguistic context (Boldini, 2000; Chatzikyriakidis and Luo, 2014c; Boldini, 2001; Piwek and Krahmer, 2000) and intensionality (Ranta, 1991, 1994; Chatzikyriakidis and Luo, 2013). In the case of intensionality, type theoretic contexts are used in place of possible worlds. They can be thought of as incomplete possible worlds or situations (Ranta, 1994; Boldini, 2000; Chatzikyriakidis and Luo, 2014c).

A context in CTT can be thought of in various ways: one way is as a list of variable declarations, where variables stand for proofs of the corresponding assumptions (Boldini, 2000) and another is a sequence of type judgements (Ranta, 1994). The standard way to write a context is as an expression of the form:

$$\Gamma = x_1 : A_1, x_2 : A_2(x_1), \ldots, x_n : A_n(x_1, \ldots, x_{n-1})$$

Here we have a series of types $A_1, A_2, \ldots, A_n$ and a series of variables $x_1, x_2, \ldots, x_n$ which are assumed to be witnesses (which are often referred to as "proofs") of these types. Each type may depend on the witnesses chosen for any of the previous types in the sequence which is represented in this notation by $A_2(x_1)$ (meaning that A_2 may depend on the value of x_1 and $A_n(x_1, \ldots, x_{n-1})$ meaning that A_n may depend on the values of any of the previous variables.

One way to gloss a context in CTT is as a "let"-statement: "let x_1 be an object of type A_1, x_2 an object of type $A_2(x_1)$,..., and x_n be an object of type $A_n(x_1, \ldots, x_{N-1})$".

In CTT such contexts can be used to express an assumption on a rule of inference. Ranta (1994) pointed out that they can be used to represent the content of previous discourse. Consider the discourse:

A farmer owns a donkey. He loves it.

At the end of the first sentence we can consider ourselves to be in the context:

$$x_1 : (\Sigma x : \mathit{Farmer})(\Sigma y : \mathit{Donkey})(\mathrm{own}(x, y))$$

In processing the second sentence, we can extend this context, picking up on the variables already declared, using the projection operators:

$$x_2 : (\mathrm{love}(\pi_1(x_1), \pi_1(\pi_2(x_1)))$$

As Ranta notes, there is an obvious correspondence between this use of contexts and the notion of discourse representation structure as used in Kamp and Reyle (1993).

While such contexts are an important notion in type theory they are not, in the standard formulation, objects which are themselves assigned a type. Rather they are assumptions expressed in a proof theoretic language. This means that, for example, we cannot use contexts as standardly characterized in type theory to talk about a function from a particular kind of context to a proposition (which we would construe as a type as discussed in Section 2.2). Such a function could be used, for example, to model Kaplan's notion of *character* (Kaplan, 1978). Introducing record types (as discussed in Section 2.3.5) allows us to treat contexts as objects which have a type. Thus the type of a context in which a farmer owns a donkey could be identical with the type that models the proposition for the sentence *a farmer owns a donkey*:

$$\begin{bmatrix} \mathrm{x} & : & \mathit{Farmer} \\ \mathrm{y} & : & \mathit{Donkey} \\ \mathrm{e} & : & \mathrm{own(x,y)} \end{bmatrix}$$

This corresponds in an obvious way to the context above (and also the corresponding discourse representation structure). Now, for example, we can define a function which takes contexts in which a farmer owns a donkey to a type of context in which that farmer loves that donkey:

$$\lambda r{:} \begin{bmatrix} \mathrm{x} & : & \mathit{Farmer} \\ \mathrm{y} & : & \mathit{Donkey} \\ \mathrm{e} & : & \mathrm{own(x,y)} \end{bmatrix} . \begin{bmatrix} \mathrm{e}' & : & \mathrm{love}(r.\mathrm{x},\ r.\mathrm{y}) \end{bmatrix}$$

The correspondence between records and contexts is exploited a great deal in the TTR literature (see for example Ginzburg, 2012; Cooper, in prep).

Ranta also proposes that contexts can be used to represent the belief state of an agent. An agent, a, believes a proposition p (that is, a type), just in case p is in a's belief context (denoted by Γ_a). Thus suppose that Kim's belief state is characterized by:

$$\Sigma x : \mathit{Donkey}.\mathrm{kick}(x)$$

That is, Kim believes that some donkey kicks. Then Kim's belief state, Γ_k could be:

$$\Gamma_k = y : (\Sigma x{:}\mathit{Donkey.kick}(x))$$

Ranta further uses a generalized belief operator $B\Gamma_a$ which is obtained by binding the variables of Γ_a with Π. The idea is that if A is a proposition in context Γ_k, then

$$B_k(A) = \Pi\Gamma_k.A = \Pi y : (\Sigma x : \mathit{Donkey}.\mathit{kick}(x)).A$$

is also a proposition. This proposition will be true just in case A follows from any instantiation of Kim's belief context $\Gamma{:}k$.

Cooper (in prep) has a similar proposal using record types. Recreating this example with the kind of example used there would have Kim's belief state as the record type corresponding to the context above:

$$\left[\begin{array}{lll} \text{y} & : & \left[\begin{array}{lll} \text{x} & : & \mathit{Donkey} \\ \text{e} & : & \text{kick(x)} \end{array}\right] \end{array}\right]$$

The object of Kim's belief would would be the record type corresponding to the Σ-type in Ranta's analysis (see Section 2.3.5):

$$\left[\begin{array}{lll} \text{x} & : & \mathit{Donkey} \\ \text{e} & : & \text{kick(x)} \end{array}\right]$$

The basic idea for checking whether Kim stands in the belief-relation to some type, T, is to check whether the type representing her belief state is a subtype of T', where T' involves a relabelling of the paths in T. The relabelling here would be x$\rightsquigarrow$y.x, e$\rightsquigarrow$y.e which would obtain in this case the exact same type as we have used to represent Kim's belief state.[16] Thus we are checking that any record of the belief state type would also be of type T, appropriately relabelled. Thus we can see that both these analyses involve a universal quantification, either in terms of an explicit Π-type or implicitly in terms of the definition of subsumptive subtyping.

A slightly more complicated example of this kind of analysis is the case of an intensional adjective like *alleged* in the analysis proposed by Chatzikyriakidis and Luo (2013). We let $A_N : \text{CN}$ be the interpretation of a common noun N. Then, we interpret

$$[\![\mathit{alleged}\ N]\!] = \Sigma a : \mathit{Human}.\ B(a, A_N)$$

where $B(a, A)$ signifies that a believes A according to Ranta's analysis. Thus *John is an alleged criminal* means that John is of the type

$$\Sigma a : Human\ .\ B(a, [\![criminal]\!])$$

Thus an alleged criminal is who is believed by somebody to be a criminal.[17,18]

[16]For detailed discussion of relabelling see Cooper (in prep).

[17]This is the analysis presented in Chatzikyriakidis and Luo (2012). Perhaps a better analysis of this would not involve belief contexts but contexts containing a sequence of allegations.

[18]A related extension of the type theoretic contexts is that of signatures. Signatures are a more elaborate context mechanism where subtyping entries as well as manifest entries (specifying particular objects rather than variables) can be introduced (Chatzikyriakidis and Luo, 2014c). Manifest entries roughly correspond to manifest fields in record types (Coquand *et al.*, 2004; Cooper and Ginzburg, 2015). For more information on the exact differences of the notion of context and signature the interested reader is directed to Chatzikyriakidis and Luo (2014c).

Further reading

A good introduction to the use of simple type theory in Montague Semantics is Dowty *et al.* (1981). See also Girard *et al.* (1990b); Krivine (1993) for discussions of simple type theory. The classic introduction to the use of CTT for natural language semantics is Ranta (1994). Luo (2010) is a good place to start reading about the use of coercive subtyping for semantics. The most recent short introduction to TTR is Cooper and Ginzburg (2015). A detailed introduction is in preparation and drafts are available for download (Cooper, in prep). For an introduction to the application of system F to natural language semantics, see Retoré (2013). For Asher's TCL, see Asher (2011). For a discussion of introducing probabilities within TTR, see Cooper *et al.* (2015b). For the use of proof assistants for NL semantics, see Ranta (2004); Chatzikyriakidis and Luo (2014a); Bernardy and Chatzikyriakidis (2017). For type theories within the tradition of Martin Löf, the standard reference is Martin-Löf (1984), but see also Luo (1994). For a recent collection of papers on the application of various kinds of type theory to natural language see Chatzikyriakidis and Luo (2017b). There is also a forthcoming special issue of the *Journal of Language Modelling* devoted to the application of type theories to lexical semantics (Retoré and Cooper, forth).

References

Asher, Nicholas (2011) *Lexical Meaning in Context: a Web of Words*, Cambridge University Press.

Asher, Nicholas and Zhaohui Luo (2012) Formalisation of coercions in lexical semantics, *Sinn und Bedeutung 17, Paris*, Vol. 223, .

Barras, Bruno *et al.* (2000) The Coq Proof Assistant Reference Manual (Version 6.3.1). INRIA-Rocquencourt.

Barwise, Jon and John Perry (1983) *Situations and Attitudes*, Bradford Books, MIT Press, Cambridge, Mass.

Bekki, Daisuke (2014) Representing anaphora with dependent types, in *International Conference on Logical Aspects of Computational Linguistics*, pp. 14–29.

Bernardy, Jean-Philippe and Stergios Chatzikyriakidis (2017) A Type-Theoretical system for the FraCaS test suite: Grammatical Framework meets Coq. Ms, University of Gothenburg. `http://www.stergioschatzikyriakidis.com/uploads/1/0/3/6/10363759/iwcs_bercha.pdf`.

Betarte, Gustavo (1998) *Dependent Record Types and Algebraic Structures in Type Theory*, PhD dissertation, Department of Computing Science, University of Gothenburg and Chalmers University of Technology.

Betarte, Gustavo and Alvaro Tasistro (1998) Extension of Martin-Löf's type theory with record types and subtyping, in G. Sambin and J. Smith (eds.), *Twenty-Five Years of Constructive Type Theory*, Oxford Logic Guides *36*, Oxford University Press, Oxford.

Boldini, P. (2000) Formalizing context in intuitionistic type theory., *Fundamenta Informaticae*, Vol. 42, No. 2, pp. 1–23.

Boldini, P. (2001) The Reference of Mass Terms from a Type-Theoretical Point of View, Paper from the 4th International Workshop on Computational Semantics.

Callaghan, Paul and Zhaohui Luo (2000) Plastic: an Implementation of Typed LF with Coercive Subtyping and Universes, Submitted to J. of Automated Reasoning, special issue on Logical Frameworks.

Cardelli, Luca (1988) Structural subtyping and the notion of power type, in *Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 70–79.

Cardelli, Luca, Simone Martini, John C Mitchell and Andre Scedrov (1991) An extension of system F with subtyping, in *International Symposium on Theoretical Aspects of Computer Software*, pp. 750–770.

Chatzikyriakidis, Stergios and Zhaohui Luo (2012) An Account of Natural Language Coordination in Type Theory with Coercive Subtyping, in Y. Parmentier and D. Duchier (eds.), *Proc. of Constraint Solving and Language Processing (CSLP12). LNCS 8114*, pp. 31–51, Orleans.

Chatzikyriakidis, Stergios and Zhaohui Luo (2013) Adjectives in a modern type-theoretical setting, in G. Morrill and J. Nederhof (eds.), *Proceedings of Formal Grammar 2013. LNCS 8036*, pp. 159–174.

Chatzikyriakidis, Stergios and Zhaohui Luo (2014a) Natural Language Inference in Coq, *J. of Logic, Language and Information.*, Vol. 23(4), pp. 441–480.

Chatzikyriakidis, Stergios and Zhaohui Luo (2014b) Natural Language Reasoning Using proof-assistant technology: Rich Typing and beyond, in *Proceedings of EACL2014*.

Chatzikyriakidis, Stergios and Zhaohui Luo (2014c) Using signatures in type theory to represent situations, in *JSAI International Symposium on Artificial Intelligence*, pp. 172–183.

Chatzikyriakidis, Stergios and Zhaohui Luo (2017a) Adjectival and Adverbial Modification: The View from Modern Type Theories, *Journal of Logic, Language and Information*, Vol. 26, No. 1, pp. 45–88.

Chatzikyriakidis, Stergios and Zhaohui Luo, eds. (2017b) *Modern Perspectives in Type-Theoretical Semantics*, Springer.

Chatzikyriakidis, Stergios and Zhaohui Luo (2017c) On the Interpretation of Common Nouns: Types v.s. Predicates, in S. Chatzikyriakidis and Z. Luo (eds.), *Modern Perspectives in Type Theoretical Semantics*, Springer.

Chierchia, Gennaro and Raymond Turner (1988) Semantics and property theory, *Linguistics and Philosophy*, Vol. 11, No. 3, pp. 261–302.

Church, Alonzo (1940) A formulation of the simple theory of types, *Journal of Symbolic Logic*, Vol. 5, No. 1, pp. 56–68.

Cooper, Robin (1996) The Role of Situations in Generalized Quantifiers, in S. Lappin (ed.), *The Handbook of Contemporary Semantic Theory*, Blackwell, Oxford.

Cooper, Robin (2005a) Austinian truth, attitudes and type theory, *Research on Language and Computation*, Vol. 3, pp. 333–362.

Cooper, Robin (2005b) Records and Record Types in Semantic Theory, *Journal of Logic and Computation*, Vol. 15, No. 2, pp. 99–112.

Cooper, Robin (2012) Type Theory and Semantics in Flux, in R. Kempson, N. Asher and T. Fernando (eds.), *Handbook of the Philosophy of Science*, Vol. 14: Philosophy of Linguistics, pp. 271–323, Elsevier BV. General editors: Dov M. Gabbay, Paul Thagard and John Woods.

Cooper, Robin (2015) Frames as Records, in A. Foret, G. Morrill, R. Muskens and R. Osswald (eds.), *Preproceedings of the 20th Conference on Formal Grammar*. `http://fg.phil.hhu.de/2015/Preproceedings-FG-2015.pdf`.

Cooper, Robin (in prep) Type theory and language: from perception to linguistic communication. Draft of book chapters available from `https://sites.google.com/site/typetheorywithrecords/drafts`.

Cooper, Robin, Simon Dobnik, Shalom Lappin and Staffan Larsson (2015a) Probabilistic Type Theory and Natural Language Semantics, *Linguistic Issues in Language Technology*, Vol. 10, No. 4, pp. 1–45.

Cooper, Robin, Simon Dobnik, Staffan Larsson and Shalom Lappin (2015b) Probabilistic type theory and natural language semantics, *LiLT (Linguistic Issues in Language Technology)*, Vol. 10, pp. 1–43.

Cooper, Robin and Jonathan Ginzburg (2015) Type Theory with Records for Natural Language Semantics, in Lappin and Fox (2015), pp. 375–407.

Coquand, Thierry (1986) Myths about the mutual exclusion problem, in *Proceedings of the IEEE Symposium on Logic in Computer Science*, pp. 227–236.

Coquand, Thierry (2011) Introduction to dependent type theory. Presentation slides of a talk at CIRM, June 2011.

Coquand, Thierry (2015) Type Theory, in E. N. Zalta (ed.), *The Stanford Encyclopedia of Philosophy*, summer 2015 edition.

Coquand, Thierry, Randy Pollack and Makoto Takeyama (2004) A Logical Framework with Dependently Typed Records, *Fundamenta Informaticae*, Vol. XX, pp. 1–22.

Dowty, David, Robert Wall and Stanley Peters (1981) *Introduction to Montague Semantics*, Reidel (Springer).

Fillmore, Charles J. (1982) Frame semantics, in *Linguistics in the Morning Calm*, pp. 111–137, Hanshin Publishing Co.

Fox, Chris and Shalom Lappin (2005) *Foundations of Intensional Semantics*, Blackwell Publishing.

Gallin, Daniel (1975) *Intensional and Higher-Order Modal Logic*, North Holland American Elsevier.

Ginzburg, Jonathan (2012) *The Interactive Stance: Meaning for Conversation*, Oxford University Press, Oxford.

Girard, Jean-Yves (1971) Une extension de l'interpretation fonctionelle de Gödel à l'analyse et son application à l'élimination des coupures dans et la thèorie des types, in *Proc. 2nd Scandinavian Logic Symposium.* North-Holland.

Girard, J.-Y. (1972) *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur*, PhD dissertation, Université Paris VII.

Girard, Jean-Yves, Yves Lafont and Paul Taylor (1990a) *Proofs and Types*, Cambridge University Press.

Girard, Jean-Yves, Paul Taylor and Yves Lafont (1990b) *Proofs and Types*, Cambridge University Press.

Grudzinska, J. and M. Zawadowski (2014) System with Generalized Quantifiers on Dependent Types for Anaphora, in *Proc. of EACL 2014.*

Kamp, Hans (1981) A Theory of Truth and Discourse Representation, in J. Groenendijk, T. Janssen and M. Stokhof (eds.), *Formal Methods in the Study of Language*, Mathematical Centre Tracts *135*, Mathematisch Centrum, Amsterdam.

Kamp, Hans and Uwe Reyle (1993) *From Discourse to Logic*, Kluwer, Dordrecht.

Kaplan, David (1978) On the Logic of Demonstratives, *Journal of Philosophical Logic*, Vol. 8, pp. 81–98.

Kohlhase, Michael (1992) Unification in order-sorted type theory, in A. Voronkov (ed.), *Proceedings of the International Conference on Logic Programming and Automated Reasoning LPAR'92* (*LNAI* 624), pp. 421–432, Springer Verlag, St. Petersburg, Russia.

Kohlhase, Michael (1994) *A Mechanization of Sorted Higher-Order Logic Based on the Resolution Principle*, PhD dissertation, Universität des Saarlandes, Germany.

Krivine, Jean Louis (1993) *Lambda-calculus, types and models*, Ellis Horwood.

Lappin, Shalom and Chris Fox, eds. (2015) *The Handbook of Contemporary Semantic Theory*, second edition, Wiley-Blackwell.

Luo, Zhaohui (1994) *Computation and Reasoning: A Type Theory for Computer Science*, Oxford University Press.

Luo, Zhaohui (2010) Type-Theoretical Semantics with Coercive Subtyping, in *Semantics and Linguistic Theory 20 (SALT20), Vancouver*, pp. 151–183.

Luo, Zhaohui (2011) Contextual analysis of word meanings in type-theoretical semantics, in *Logical Aspects of Computational Linguistics (LACL'2011). LNAI 6736*, pp. 159–174.

Luo, Zhaohui (2012) Common Nouns as Types, in D. Bechet and A. Dikovsky (eds.), *Logical Aspects of Computational Linguistics (LACL'2012). LNCS 7351.*

Martin-Löf, P. (1971) An Intuitionistic Theory of Types. manuscript.

Martin-Löf, Per (1984) *Intuitionistic Type Theory*, Bibliopolis.

Montague, Richard (1973) The proper treatment of quantification in ordinary English, in J. Hintikka, J. Moravcsik and P. Suppes (eds.), *Approaches to Natural Languages*, pp. 247–270, D. Reidel Publishing Company.

Montague, Richard (1974) *Formal Philosophy*, Yale University Press, Collected papers edited by R. Thomason.

Muskens, Reinhard (1996) Combining Montague semantics and discourse representation, *Linguistics and philosophy*, Vol. 19, No. 2, pp. 143–186.

Muskens, Reinhard (2007) Intensional models for the theory of types, *Journal of Symbolic Logic*, Vol. 72, No. 1, pp. 98–118.

Nordström, Bengt, Kent Petersson and Jan Smith (1990) *Programming in Martin-Löf's Type Theory: An Introduction*, Oxford University Press.

Partee, Barbara H. (1979) Semantics – Mathematics or Psychology?, in R. Bäuerle, U. Egli and A. v. Stechow (eds.), *Semantics from Different Points of View* (*Springer Series in Language and Communication* 6), Springer.

Partee, Barbara H. (2014) The History of Formal Semantics: Changing Notions of Linguistic Competence. 9th Annual Joshua and Verona Whatmough Lecture, Harvard, `https://udrive.oit.umass.edu/partee/Partee2014Harvard.pdf`, `https://www.youtube.com/watch?v=0VV-1NDKmEc`.

Piwek, Paul and Emiel Krahmer (2000) Presuppositions in context: Constructing bridges, in *Formal aspects of context*, pp. 85–106, Springer.

Ranta, Aarne (1991) Constructing possible worlds*, *Theoria*, Vol. 57, No. 1-2, pp. 77–99.

Ranta, Aarne (1994) *Type-Theoretical Grammar*, Oxford University Press.

Ranta, A. (2004) Dialogue systems as proof editors, *Journal of Logic, Language and Information*, Vol. 225-240, No. 13, .

Ranta, Aarne (2011) *Grammatical Framework: Programming with Multilingual Grammars*, CSLI Publications.

Ranta, Aarne (2015) Constructive Type Theory, in Lappin and Fox (2015), pp. 345–374.

Retoré, Christian (2012) Variable Types for Meaning Assembly: a Logical Syntax for Generic Noun Phrases Introduced by 'most', *Recherches linguistiques de Vincennes*, Vol. 41, pp. 83–102.

Retoré, Christian (2013) The Montagovian Generative Lexicon Tyn: a Type Theoretical Framework for Natural Language Semantics, in R. Matthes and A. Schubert (eds.), *Proceedings of TYPES2013*.

Retoré, Christian and Robin Cooper, eds. (forth) *Type theory and lexical semantics*, Special Issue of *Journal of Language Modelling*.

Richard, Moot and Christian Retoré (2012) *The Logic of Categorial Grammars*, Springer.

Russell, Bertrand (1903) *The Principles of Mathematics*, Routledge, paperback edition, 1992.

Sundholm, Gőran (1989) Constructive Generalized Quantifiers, *Synthese*, Vol. 79, No. 1, pp. 1–12.

Tanaka, Ribeka, Yuki Nakano and Daisuke Bekki (2013) Constructive Generalized Quantifiers Revisited, in *JSAI International Symposium on Artificial Intelligence*, pp. 115–124.

Tasistro, Alvaro (1997) *Substitution, record types and subtyping in type theory, with applications to the theory of programming*, PhD dissertation, Department of Computing Science, University of Gothenburg and Chalmers University of Technology.

Thomason, Richmond H. (1980) A model theory for propositional attitudes, *Linguistics and Philosophy*, Vol. 4, pp. 47–70.

Turner, Raymond (2005) Semantics and Stratification, *Journal of Logic and Computation*, Vol. 15, No. 2, pp. 145–158.

Wadler, Philip (2015) Propositions as Types, *Communications of the ACM*, Vol. 58, No. 12, pp. 75–84.

Whitehead, Alfred and Bertrand Russell (1925) *Principia Mathematica*, 2nd edition, Cambridge University Press.