Type-Logical Semantics*

Reinhard Muskens

Type-logical semantics studies linguistic meaning with the help of the theory of types. The latter originated with Russell as an answer to the paradoxes, but has the additional virtue that it is very close to ordinary language. In fact, type theory is so much more similar to language than predicate logic is, that adopting it as a vehicle of representation can overcome the mismatches between grammatical form and predicate logical form that were observed by Frege and Russell. The grammatical forms of ordinary language sentences consequently may be taken to be much less misleading than logicians in the first half of the 20th century often thought them to be. This was realized by Richard Montague, who used the theory of types to translate fragments of ordinary language into a logical language.

Semantics is commonly divided into *lexical* semantics, which studies the meaning of words, and *compositional* semantics, which studies the way in which complex phrases obtain a meaning from their constituents. The strength of type-logical semantics lies with the latter, but type-logical theories can be combined with many competing hypotheses about lexical meaning, provided these hypotheses are expressed using the language of type theory.

1 Typing Words and Objects

Type-logical semantics is usually based on some variant of Alonzo Church's formulation of the simple theory of types (Church 1940) and it is impossible to explain the application without also explaining the underlying theory. In this entry we will start with giving a more or less informal account of the formal underpinnings of the theory, moving to their linguistic application as we proceed. For an exposition of the relation between the *simple* theory of types and Russell's so-called *ramified* theory, see the entry on Theory of Types.

*From: E. Craig, editor, *Routledge Encyclopedia of Philosophy Online*. Routledge, 2011. http://www.rep.routledge.com/article/U061

Typing is allotting the objects in one's ontology to separate categories. Expressions are also categorized and receive the same types as the objects they refer to. Truth-values, for example, should be distinguished from entities of the cabbages and kings variety and these in their turn from, say, possible worlds and other kinds of objects. Predicates should be distinguished from individuals; two-place predicates from one-place predicates; and predicates of predicates (*no student*, for example, is often classified as a predicate of predicates, as it combines with the predicate *smokes* to form a sentence) should be distinguished from predicates of predicates of predicates.

Type theory provides the book-keeping system that allows one to keep track of all these different categories. The idea is to first choose types for the most basic ontological categories one wants to adopt—say t for truth-values, e for entities, and s for worlds—and to then stipulate that if α and β are types, $(\alpha\beta)$ is also a type. The latter is meant to be the type of functions that take objects of type α as their arguments and return a value of type β. For example, (et) is the type of functions from entities to truth values, the type of one-place predicates in a set-up of the theory that abstracts from possible worlds. Other examples of types are (st), $(e(st))$, and $((e(st))((e(st))(st)))$, but in order to facilitate reading two conventions will be introduced. The first is that outer parentheses will always be omitted and the second is that $\alpha\beta\gamma$ will abbreviate $\alpha(\beta\gamma)$. Thus (st) is written as st, $(e(st))$ as est, and $((e(st))((e(st))(st)))$ as $(est)(est)st$. Association is to the right when parentheses are restored.

This typing scheme gives a hierarchy of functions, not relations, but the predicates and predicates of predicates just considered can be modeled by functions. In fact, there are various ways to do this, depending on the ontology one wishes to adopt. The example that will be worked out here is based on an ontology of possible worlds, but it should be kept in mind that this choice is by no means forced upon us by type-theoretical considerations. Type-theoretical systems also square well with completely different assumptions about the ontological underpinnings of natural language.

A basic idea of possible worlds semantics is that the truth of a declarative sentence should be evaluated relative to a possible world. The meaning of a sentence therefore determines an object of type st, a function from worlds to truth values; each choice of a world corresponding to a truth value, namely, the truth value of the sentence at the world. Note that functions of this type are identifiable with the set of those possible worlds for which they return the value *true*, so that an object of type st can alternatively be taken to be a set of possible worlds. Such sets are often called *propositions* and this usage will be followed here.

CONSTANT	TYPE
walk, talk, . . .	est
man, woman, unicorn, . . .	est
happy, bald, . . .	est
love, hate, . . .	$eest$
every, a, no, the	$(est)(est)st$
john, mary, . . .	$(est)st$
necessarily, possibly	$(st)st$
believe, know, . . .	$(st)est$
not	$(st)st$
and, or	$\alpha\alpha\alpha$

Table 1: Some constants and their possible types.

Let us consider predicates such as **walk**, **man**, or **unicorn**. These can be taken to combine with terms for entities to form sentences (terms denoting propositions) and are therefore of type est. An example: **walk** of type est combines with **m** (for *Mary*) of type e to produce **walk m**, the sentence that Mary is walking, of type st. Functions of type est can also be viewed as relations between entities and possible worlds. If the function denoted by **walk** is applied to an entity, it returns a function that, when applied to a world, returns a truth value. So any combination of an entity and a world returns a truth value and the function denoted by **walk** can be identified with the set of pairs $\langle x, i\rangle$ (with x an entity and i a world) for which the function returns *true*. More generally, any function of a type $\alpha_1\alpha_2 \ldots \alpha_n t$ can be identified with an n-ary relation taking objects of type α_k as its k-th argument.

Other words of English can also be typed. Transitive verbs such as **love** can be modeled as functions of type $eest$, as they return a sentence when fed two arguments. Determiners such as **every**, **some**, **no** and **the** can be taken to be of the type $(est)(est)st$ we met above, as they combine with two terms of type est, such as **man** and **walk** or **woman** and **talk**, to form a sentence, e.g. **every man walk**. Table 1 gives more words that have been typed according to the principles used here: Proper names combine with a predicate to form a sentence. They can therefore be viewed as predicates of predicates (type $(est)st$). (Clearly, they can also be viewed as individual constants. The two perspectives are equally valid and both will be maintained here.) Sentential modifiers such as **necessarily** return a sentence when applied to a sentence and therefore denote functions that

return a proposition when applied to a proposition. Propositional attitude verbs combine with a sentence and an entity (an agent), returning a sentence. Negation goes from propositions to propositions (and from sentences to sentences on the syntactic level). The coordinating elements **and** and **or** can be associated with a whole family of types of the form $\alpha\alpha\alpha$ (where α must 'end in st'). $(st)(st)st$ is of this form and this allows for the conjunction and disjunction of propositions, but the more general typing will also make direct translations of other coordinations possible: *sing and dance* or *some or all*, for example, where in the former *and* has type $(est)(est)est$ and in the latter it has the rather complex type $((est)(est)st)((est)(est)st)(est)(est)st$, the type of functions that take two objects of type $(est)(est)st$ and return a third.

2 Terms

Type theory sports two term building operations: *application* and *lambda abstraction*. They are defined as follows.

(Application) If M is a term of type $\alpha\beta$ and N is a term of type α, then (MN) is a term of type β.

(Lambda Abstraction) If M is a term of type β and X is a variable of type α, then $(\lambda X.M)$ is a term of type $\alpha\beta$.

Standard notational conventions say that outer parentheses need not be written, that MN_1N_2 may abbreviate $(MN_1)N_2$, and that $\lambda X\lambda Y.M$, or even just $\lambda XY.M$, is short for $\lambda X.(\lambda Y.M)$. A similar convention holds for longer sequences of lambda binders. We will not always use these conventions here, as we want to emphasize the similarities between the structures of certain lambda terms and linguistic structures. The standard abbreviatory conventions sometimes obscure these similarities.

Let us illustrate the application and abstraction rules by forming some terms. Given the typing in Table 1, the application rule allows the formation of (**every man**) of type $(est)st$. A second use of the rule will give that ((**every man**)**walk**) is of type st. Another example, involving variables: Take the constant **love** of type $eest$, and let x and y be variables of type e. Then ((**love** x)y), or **love** xy for short, is a term of type st by the application rule (the distinction between constants and variables is irrelevant for typing). We will interpret it as expressing that y loves x, so that its structure reflects the structure found in natural language, where a transitive verb combines with its direct object before combining with its

subject. The abstraction rule now allows the formation of $(\lambda x.\mathbf{love}\, xy)$ of type *est*, so that $(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, xy)$ is of type *st*, from which it follows that $(\lambda y.(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, xy))$ is of type *est* again. Reasoning further along these lines, we find that $(\mathbf{every\ man})(\lambda y.(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, xy))$ is of type *st*. The latter may be taken to formalize that reading of the English sentence *every man loves a woman* in which the indefinite noun phrase *a woman* is in the scope of *every man*.

The semantic operation corresponding to the rule named application is indeed function application. The denotation of a term (MN) is the result of applying the function denoted by M to the object denoted by N. Lambda abstraction provides a dual. If the type of X is α then $(\lambda X.M)$ denotes a function with the set of all objects of type α as its domain such that, if the function is applied to an object d, the value that is returned is that of M with the variable X interpreted as d. For example, $\lambda x.\mathbf{love}\, x\mathbf{r}$, *being loved by Romeo*, denotes the function that, when applied to Mary, returns the proposition that Romeo loves Mary and, when applied to Juliet, returns the proposition that Romeo loves Juliet.

Certain principles will become valid given these interpretations of the term-building operations. Among these is *renaming of bound variables* (or *α-conversion*), familiar from predicate logic. $\lambda y.(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, xy)$ and $\lambda z.(\mathbf{a\ woman})(\lambda u.\mathbf{love}\, uz)$, for example, will denote exactly the same function (the function that, when applied to a person a, returns the proposition that a loves a woman). Another rule which becomes valid, known as the *β rule*, is formulated as follows.

(β) $(\lambda X.M)N = M[X := N]$, provided N is substitutable for X in M.

Here $M[X := N]$ should be read as 'the result of substituting N for all free occurrences of X in M.' The requirement that N be substitutable basically says that variables free in N should remain free in $M[X := N]$. An example: Let $\mathbf{j}$ be a term of type e, denoting John perhaps. The β rule allows one to conclude that $(\lambda y.(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, xy))\mathbf{j}$ equals $(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, x\mathbf{j})$, i.e. the property of loving a woman can be predicated of John if and only if John indeed loves a woman.

Let us spell out in detail how the β rule was applied in this last example. First we deleted the initial λy from $\lambda y.(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, xy)$, obtaining the term $(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, xy)$, in which y now occurs free. We then replaced this free occurrence of y by $\mathbf{j}$, with $(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, x\mathbf{j})$ as a result. In general, conversion of $(\lambda X.M)N$ proceeds by stripping off the initial binder λX from $\lambda X.M$ and by then substituting N for each free

occurrence of X in the resulting M. More examples of the use of the β rule will follow shortly. It plays a pivotal role in type-theoretic semantics.

3 Logical Form and Grammatical Form

The following are some terms of type st that can be formed with the help of the type assignment in Table 1 and the term-building rules application and abstraction.

(1) a. $(\mathbf{every\ man})(\lambda y.(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, xy))$

b. $(\mathbf{a\ woman})(\lambda x.(\mathbf{every\ man})(\lambda y.\mathbf{love}\, xy))$

c. $\mathbf{not}((\mathbf{the\ king})\mathbf{bald})$

d. $(\mathbf{the\ king})(\lambda x.(\mathbf{not}(\mathbf{bald}\, x)))$

e. $(\mathbf{mary})(\mathbf{believe}((\mathbf{a\ unicorn})(\mathbf{and\ walk\ talk})))$

f. $(\mathbf{a\ unicorn})(\lambda x.((\mathbf{mary})(\mathbf{believe}((\mathbf{and\ walk\ talk})x))))$

The terms in (1a) and (1b) are both meant to formalize the English sentence *every man loves a woman*, but while (1a) gives it its universal-existential reading, as discussed above, (1b) says that some woman has the property of being loved by every man, a second possible way to interpret the sentence. (1c) is a formalization of *the king is not bald* in the reading where *the king is bald* is denied to be true, while (1d) ascribes the property of not being bald to the king. The terms (1e) and (1f), lastly, both render the sentence *Mary believes some unicorn is walking and talking*, but while (1e) gives the *de dicto* reading of this statement, (1f) formalizes the *de re* reading, which can be paraphrased as *there is a unicorn Mary believes to be walking and talking*.

Note how close each of the terms in (1) is to the sentence it represents. (1e), for example, is essentially isomorphic to the hierarchical structure that most linguists would assign to the sentence it renders. And while other terms are not directly akin to any *surface* syntactic structure, they can still be regarded as syntactic structures in which certain noun phrases (such as *a woman* or *the king*) have moved from their original positions. Word order is ignored, though. The view that a level of structures very similar to the ones described here are part of the human grammar is widely shared among syntacticians (May 1985). This level is commonly called that of *Logical Form* or *LF*, but from our perspective it might as well have been called the

$$
\begin{aligned}
\textbf{every} &= \lambda P' P \lambda i. \forall x (P' x i \rightarrow P x i) \\
\textbf{a} &= \lambda P' P \lambda i. \exists x (P' x i \wedge P x i) \\
\textbf{no} &= \lambda P' P \lambda i. \forall x (P' x i \rightarrow \neg P x i) \\
\textbf{the} &= \lambda P' P \lambda i. \exists x (\forall y (P' y i \leftrightarrow x = y) \wedge P x i) \\
\textbf{john} &= \lambda P. P \mathbf{j} \\
\textbf{mary} &= \lambda P. P \mathbf{m} \\
\textbf{necessarily} &= \lambda p \lambda i. \forall j (\mathbf{R} i j \rightarrow p j) \\
\textbf{possibly} &= \lambda p \lambda i. \exists j (\mathbf{R} i j \wedge p j) \\
\textbf{believe} &= \lambda p \lambda x \lambda i. \forall j (\mathbf{B} x i j \rightarrow p j) \\
\textbf{know} &= \lambda p \lambda x \lambda i. \forall j (\mathbf{K} x i j \rightarrow p j) \\
\textbf{not} &= \lambda p \lambda i. \neg (p i) \\
\textbf{and} &= \lambda R R' \lambda \vec{x} \lambda i. (R \vec{x} i \wedge R' \vec{x} i) \\
\textbf{or} &= \lambda R R' \lambda \vec{x} \lambda i. (R \vec{x} i \vee R' \vec{x} i)
\end{aligned}
$$

Table 2: Meaning postulates.

level of *Grammatical Form*, as it is the proximity to natural language that is striking.

By themselves the forms in (1) are not adequate as logical formalizations of the English sentences they are rendering, as they do not give their correct truth conditions or adequately capture entailments among them. What is needed is a systematic association of such terms with more standard logical forms and we shall proceed to define one.

We have described Church's simple type theory as a pure lambda calculus, i.e. a lambda calculus without logical constants, and this is a perspective on the theory that is often taken (for example in Hindley 1997), but Church meant his theory to be a formalization of a version of Russell's (1908) theory of types, which is a higher order *logic*. He therefore added logical constants to the theory, and we shall follow him in this by stipulating that $\neg$ is a logical constant of type tt, that $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$ are of type ttt, and that logical constants $\exists$ and $\forall$ in any type $(\alpha t)t$ are present, as are symbols $=$ in any type $\alpha\alpha t$. This will give that $\wedge\varphi\psi$ is a term of type t if φ and ψ are, but such terms are more conveniently written as $\varphi \wedge \psi$ and a similar convention will hold for other connectives. We will also write $\exists X \varphi$ instead of $\exists(\lambda X.\varphi)$, $\forall X \varphi$ instead of $\forall(\lambda X.\varphi)$, $M = N$ instead of $=MN$, and conform to logical

usage generally. The resulting higher order logic is provided with a model theory in Henkin (1950).

The equations in Table 2 provide the systematic correlation between grammatical forms and truly logical forms we are after. The 'logical' words of Table 1 are each equated with a term interpreting them and there are also postulates that systematically relate proper names viewed as predicates of predicates and proper names viewed as individual constants (e.g. **john** can be predicated of a predicate just if that predicate can be predicated of **j**). A typographic convention will rule the use of variables in the terms employed here: P, with or without primes, is of type *est* (predicates), p of type *st* (propositions), i and j are of type s (worlds or indices), and x, y and z are of type e (entities).

Let us see how Table 2 can be used to systematically translate terms such as those in (1) to the kind of logical forms that logicians know and love; (1a) will be used as an example. Table 2 states that $\mathbf{a} = \lambda P' P \lambda i.\exists x(P'xi \wedge Pxi)$, a term that expects two predicates in order to form a proposition with them, and we can conclude that **a woman** equals $(\lambda P' P \lambda i.\exists x(P'xi \wedge Pxi))\mathbf{woman}$, which reduces to $\lambda P \lambda i.\exists x(\mathbf{woman}\, xi \wedge Pxi)$ by the β rule. (First strip off the initial $\lambda P'$ from $\lambda P' P \lambda i.\exists x(P'xi \wedge Pxi)$; then replace the P' in the resulting $\lambda P \lambda i.\exists x(P'xi \wedge Pxi)$, which has now become free, with **woman**.) We now reason further with the subterm $(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, xy)$ of (1a).

$$\begin{aligned}(\mathbf{a\ woman})(\lambda x.\mathbf{love}\, xy) &= (\lambda P \lambda i.\exists x(\mathbf{woman}\, xi \wedge Pxi))(\lambda x.\mathbf{love}\, xy) \\ &= \lambda i.\exists x(\mathbf{woman}\, xi \wedge (\lambda x.\mathbf{love}\, xy)xi) \\ &= \lambda i.\exists x(\mathbf{woman}\, xi \wedge \mathbf{love}\, xyi)\end{aligned}$$

Each of the last two reductions is obtained by applications of the β rule, the last one leading to replacement of the subterm $(\lambda x.\mathbf{love}\, xy)x$ by $\mathbf{love}\, xy$. In general, because terms that are related by the β rule can be identified, β reductions can be applied to subterms of any term under consideration.

Further reasoning gives the following equations.

$$\begin{aligned}(1a) &= (\lambda P \lambda i.\forall x(\mathbf{man}\, xi \rightarrow Pxi))(\lambda y \lambda i.\exists x(\mathbf{woman}\, xi \wedge \mathbf{love}\, xyi)) \\ &= \lambda i.\forall x(\mathbf{man}\, xi \rightarrow (\lambda y \lambda i.\exists x(\mathbf{woman}\, xi \wedge \mathbf{love}\, xyi))xi) \\ &= \lambda i.\forall x(\mathbf{man}\, xi \rightarrow (\lambda y \lambda i.\exists z(\mathbf{woman}\, zi \wedge \mathbf{love}\, zyi))xi) \\ &= \lambda i.\forall x(\mathbf{man}\, xi \rightarrow \exists z(\mathbf{woman}\, zi \wedge \mathbf{love}\, zxi))\end{aligned}$$

The first of these equations is obtained by observing that **every man** can be replaced with $\lambda P \lambda i.\forall x(\mathbf{man}\, xi \rightarrow Pxi)$ on the basis of the entry for

every in Table 2. The second is a β reduction. The third is an α conversion and the fourth the result of two more β reductions. (Note that the α conversion step was necessary to make the first of these two last β reductions possible. Replacing $(\lambda y \lambda i.\exists x(\mathbf{woman}\, xi \wedge \mathbf{love}\, xyi))x$ with $(\lambda i.\exists x(\mathbf{woman}\, xi \wedge \mathbf{love}\, xxi))$ would have been illegal because the free occurrence of x in the first term leads to a bound occurrence in the second.)

We have found that, given the meaning postulates in Table 2, (1a) is identical to (2a), a proposition which is true in the actual world @ if and only if $\forall x(\mathbf{man}\, x@ \rightarrow \exists z(\mathbf{woman}\, z@ \wedge \mathbf{love}\, zx@))$ is true. Very similar considerations lead to the conclusion that each of the other items in (1) equals the corresponding item in (2).

(2) a. $\lambda i.\forall x(\mathbf{man}\, xi \rightarrow \exists y(\mathbf{woman}\, yi \wedge \mathbf{love}\, yxi))$

b. $\lambda i.\exists x(\mathbf{woman}\, xi \wedge \forall z(\mathbf{man}\, zi \rightarrow \mathbf{love}\, xzi))$

c. $\lambda i.\neg\exists x(\forall y(\mathbf{king}\, yi \leftrightarrow x = y) \wedge \mathbf{bald}\, xi)$

d. $\lambda i.\exists x(\forall y(\mathbf{king}\, yi \leftrightarrow x = y) \wedge \neg\mathbf{bald}\, xi)$

e. $\lambda i.\forall j(\mathbf{Bm}ij \rightarrow \exists x(\mathbf{unicorn}\, xj \wedge \mathbf{walk}\, xj \wedge \mathbf{talk}\, xj))$

f. $\lambda i.\exists x(\mathbf{unicorn}\, xi \wedge \forall j(\mathbf{Bm}ij \rightarrow (\mathbf{walk}\, xj \wedge \mathbf{talk}\, xj)))$

In (2e) and (2f), read $\mathbf{Bm}ij$ as 'in world i world j is a doxastic alternative of **m**', so that (2f) comes to express that there is a unicorn who walks and talks in each of Mary's doxastic alternatives (Hintikka 1962). A technical detail: The **and** of (1e) is of type $(est)(est)est$, and so will translate as $\lambda PP'\lambda x\lambda i.(Pxi \wedge P'xi)$. The interested reader is invited to check some of the equivalences here and to experiment with some others.

The exposition thus far has been somewhat technical, but the conclusion is not technical at all. It is that the perceived gap between the grammatical form and the logical form of a sentence can be bridged. Philosophers during the first half of the 20th century adhered to the view that grammatical form was misleading and essentially different from logical form. Only the latter was correct. The *locus classicus* of this view is Russell's 'On Denoting' (Russell 1905), but the idea has found many able proponents in addition to Russell. Considerations such as the ones above, however, pioneered in the work of Richard Montague (Montague 1970a; Montague 1970b; Montague 1973) show that it is possible to have one's cake and eat it. The grammatical forms in (1) and the logical forms in (2) may look very different, but given suitable meaning postulates, are extensionally identical. This means that

the linguistic and the logical perspectives on form turn out to be compatible and equally valid. It may be thought ironic that the essential ingredient in this vindication of grammatical form and rejection of Russell's 'misleading form thesis' has been a version of Russell's own Theory of Types.

Reinhard Muskens

References and further reading

Barwise, J. and R. Cooper (1981). Generalized Quantifiers and Natural Language. *Linguistics and Philosophy 4*, 159–219. (This very readable article connects Montague's treatment of noun phrases with the logical theory of generalized quantifiers).

van Benthem, J. (1991). *Language in Action*. Amsterdam: North-Holland. (This book offers a range of perspectives on type-theoretic syntax and type-theoretic semantics).

Blackburn, P. and J. Bos (2005). *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI. (Automated translation of natural language into the language of predicate logic on a type-theoretical basis. The results are then sent to theorem provers to check consistency or entailments. Readers with knowledge of the Prolog programming language will find this text accessible.).

Carpenter, B. (1998). *Type-logical Semantics*. MIT Press. (An excellent introduction to the subject. The book is based on type-logical *syntax*, a form of categorial grammar, pioneered by Joachim Lambek, which is very close to type-logical *semantics*.).

Church, A. (1940). A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic 5*, 56–68. (A formulation of the theory that conforms to modern standards of rigour.).

Dowty, D., R. Wall, and S. Peters (1981). *Introduction to Montague Semantics*. Dordrecht: Reidel. (One of the two standard introductions to the theory in Richard Montague's original formulation. Montague's papers are terse, but this book succeeds admirably in making the theory accessible.).

Gallin, D. (1975). *Intensional and Higher-Order Modal Logic*. Amsterdam: North-Holland. (Type-theoretic semantics in Richard Montague's original formulation was based on IL, a variant of type theory containing all kinds of modal operators. Gallin, who was a student of Montague's,

showed that this logic can be embedded in simple type theory. This makes for a considerable streamlining of the theory, which was used in this entry.).

Gamut, L. T. F. (1991). *Logic, Language and Meaning, Volume 2*. Chicago: University of Chicago Press. (The other standard text on Montague semantics. Chapter 4 contains an excellent introduction to type theory and part of chapter 7 is devoted to a careful explanation of generalized quantifiers in natural language.).

Gazdar, G. (1980). A Cross-Categorial Semantics for Coordination. *Linguistics and Philosophy 3*, 407–409. (This entry's treatment of the coordinating elements *and* and *or* is based on Gazdar's short paper, as are virtually all other treatments of these items nowadays.).

Henkin, L. (1950). Completeness in the Theory of Types. *Journal of Symbolic Logic 15*, 81–91. (An elegant paper providing simple type theory with a semantics and showing that a complete axiomatization of semantic consequence can be obtained, provided one is willing to generalize the notion of a model.).

Hindley, R. (1997). *Basic Simple Type Theory*. Cambridge University Press. (A beautiful but forbiddingly technical development of the mathematics of simple type theory viewed as a pure lambda calculus.).

Hintikka, J. (1962). *Knowledge and Belief*. Cornell University Press. (The origin of the idea that an agent believes (knows) a proposition if and only if the proposition holds in all of the agent's doxastic (epistemic) alternatives.).

Lewis, D. (1972). General Semantics. In D. Davidson and G. Harman (Eds.), *Semantics of Natural Language*, pp. 169–218. Dordrecht: Reidel. (While the assumptions about syntactic theory that Lewis makes here may by now be somewhat long in the tooth, the article itself is still highly readable. Recommended.).

May, R. (1985). *Logical Form: Its Structure and Derivation*. Cambridge, MA: MIT Press. (A theory of the syntactic notion of Logical Form.).

Montague, R. (1970a). English as a Formal Language. In B. Visenti (Ed.), *Linguaggi nella Società e nella Tecnica*, pp. 189–224. Milan: Edizioni di Comunità. Reprinted in (Thomason 1974). (This is the first of Montague's three papers on natural language. It is not in a strict sense based on a type theory, but typing is in fact present throughout.).

Montague, R. (1970b). Universal Grammar. *Theoria 36*, 373–398. Reprinted in (Thomason 1974). (In his second paper on language Montague gives a very general outline of his theory.).

Montague, R. (1973). The Proper Treatment of Quantification in Ordinary English. In J. Hintikka, J. Moravcsik, and P. Suppes (Eds.), *Approaches to Natural Language*, pp. 221–242. Dordrecht: Reidel. Reprinted in (Thomason 1974). (This is the paper that is mostly referred to when people speak of Montague grammar. No easy read and with hindsight it is possible to say that most of its technicalities only serve to obscure its central idea. That idea, however, has shaped a whole new field of research.).

Partee, B. (1997). Montague Grammar. In J. van Benthem and A. ter Meulen (Eds.), *Handbook of Logic and Language*, pp. 5–91. Amsterdam: Elsevier. with H. Hendriks. (Barbara Partee has played a pivotal role in introducing Montague's work to the linguistic community. This is a survey paper with an excellent bibliography. The handbook it appeared in contains many other articles on the connections between logic and linguistic theory.).

Russell, B. (1905). On Denoting. *Mind 14* (56), 479–493. (Russell's famous article on denoting expressions. This entry's treatment of the words *all*, *a*, *no* and *the* is based on it, but Russell would have frowned upon the use of possible worlds.).

Russell, B. (1908). Mathematical Logic as Based on the Theory of Types. *American Journal of Mathematics 30*, 222–262. (The first formulation of (ramified) type theory.).

Thomason, R. (Ed.) (1974). *Formal Philosophy, Selected Papers of Richard Montague*. Yale University Press. (A selection of Montague's papers, collected after his untimely death.).