Natural Language Semantics

Reinhard Muskens

ESSLLI 2005, Edinburgh

About this Course

- This course gives an introduction to natural language semantics in the logical tradition.
- The course is foundational, but a good working knowledge of classical logic will be assumed.
- You are advised to take a course on logic or to work through a good introduction to predicate logic, should that knowledge be lacking. My favourite introductory texts are (Hodges 1977) and volume I of (Gamut 1991).
- Previous acquaintance with classical type theory and some knowledge of linguistic syntax will be helpful, but these will be introduced.

Legal Stuff

Overview of the Course

Meaning

- Natural Language Semantics is the study of meaning in ordinary language.
- How do linguistic expressions manage to convey meaning?
- A difficult question immediately presents itself: what is meaning?
- Trying to answer this question straightaway may land us in endless speculation. Interesting, but not really fruitful.
- Instead, let us ask ourselves what linguistic data are related to the question of meaning in natural language.

The Nature of Linguistic Data

- What are the data that linguistic theory has to deal with?
- Many linguists will follow Chomsky in answering that native speaker's intuitions are the crucial data.
- According to the theory, people have a linguistic competence that is innate. This innate competence is also called Universal Grammar.
- Universal grammar is not some metaphysical entity but should be thought of as a biologically conditioned state of the human brain that is characteristic for our species.
- In childhood humans can acquire one or more languages by setting a set of pre-given parameters. They then are in the possession of a native grammar.

The Nature of Linguistic Data (continued)

- Native speaker's intuitions are a direct reflection of native grammar. Observed behaviour is secondary.
- By studying native speaker's intuitions the linguist tries to give an account of native grammars. Cross-linguistic comparison will hopefully give insight into universal grammar.
- In practice, if you want to know whether a certain form is correct in, say, Warlpiri, the thing to do is ask a native speaker.
- A theory of Warlpiri will give certain predictions about Warlpiri that we can check with the native speakers. In this way linguistic theories become capable of refutation.
- Are there intuitions that are relevant for semantics?

Semantic Data

- Speakers of English (and of other languages) have the following kinds of intuitions.
- When presented with a situation in which there clearly is a glass on a table, they will judge the sentence there is a glass on the table to be true. They will judge its negation false.
- They will judge that Socrates must be mortal, if given the information that every man is mortal and that Socrates is a man. That is, they have pre-theoretic intuitions about entailment. (Warning: these intuitions are very imperfect.)
- There are similar intuitions about consistency, presupposition, anaphoric relations, etc.

Language and Logic

- At least some of the data on the previous slide clearly had to do with the logic of natural language.
- Is it possible to provide a language like English with a logic? If that could be done the resulting theory may be able to account for the kind of phenomena just described.
- A translation of English into logic would in fact also do the trick.
- But that translation should then be completely formal. It should be an algorithm that inputs an English expression and outputs a logical expression.
- The algorithm should not make use of speaker's competence. Speaker's competence ultimately is the thing that needs to be explained.

Dark Clouds on the Horizon

- But is the idea of providing language with an explicit logic viable at all? Here are some of Frege's views about natural language:
- Natural language is inexact. Logical relations are just hinted at but are not expressed in any precise way.
- Natural language is ambiguous.
- In natural languages the Logical and Psychological are mixed.
- A characteristic remark:

  > *Language is not ruled by logical laws in such a way that following grammar already guarantees formal correctness of the movements of thought. ((Frege 1882), my translation.)*

The Traditional View: Subject-predicate Form

Simple sentences can often be divided into a logical subject and a logical predicate:

(1) a. Socrates is white

b. $white(socrates)$

c. Every man is white

d. $\{x \mid man(x)\} \subseteq \{x \mid white(x)\}$

Unfortunately, slightly more complex sentences do not naturally divide in this way (although they still divide into a grammatical subject and predicate).

(2) a. [[Plato][is taller than Socrates]]

b. $taller(plato, socrates)$

c. [[Every boy][admires a girl]]

d. $\forall x[boy(x) \rightarrow \exists y[girl(y) \wedge admires(x, y)]]$

Mismatch

- There is a mismatch between grammatical form and logical form.
- [[every boy][admires[a girl]]]
  $\forall x\,(boy\,x \rightarrow \exists y\,(girl\,y \wedge admires\,xy))$
- Some elements of the grammatical form correspond nicely with contiguous parts of the logical form.
- But others do not.

Russell: Rejection of subject-predicate form

Whether any valid inferences are possible from language to non-linguistic facts is a question as to which I do not care to dogmatize; but certainly the inferences found in Leibniz and other a priori *philosophers are not valid, since all are due to a defective logic. The subject-predicate logic, which all such philosophers in the past assumed, either ignores relations altogether, or produces fallacious arguments to prove that relations are unreal. (Russell 1946)*

Misleading Form

(3) a. [[I [met [a unicorn]]]
b. [[Zeus][doesn't exist]]
c. [[The present King of France][is not bald]]

- The grammatical form of these sentences suggests that a unicorn, Zeus and the present King of France are terms. But if they are terms, what do they denote? If they do not denote, how come they obviously contribute to the meaning of expressions they are parts of?
- About the view that some form of existence must be ascribed to a unicorn etc. (a position which was in fact held by the philosopher Meinong) Russell remarked that [l]ogic [...] must no more admit a unicorn than zoology can [...].

Some Possible Formalisations

(4) a. [[I met][a unicorn]]
b. [[Zeus][doesn't exist]]
c. [[The present King of France][is not bald]]

(5) a. $\exists x[unicorn(x) \wedge met(I, x)]$
b. $\neg\exists x\, zeus(x)$
c. $\exists x[\forall y[king(y) \leftrightarrow x = y] \wedge \neg bald(x)]$
$\neg\exists x[\forall y[king(y) \leftrightarrow x = y] \wedge bald(x)]$

- The problems with non-denoting terms now have gone.
- But the form of the sentences in (4) radically differs from the form of their translations in (5).
- Philosophers like Russell and Quine held that the forms in (5) are correct, those in (4) misleading.

Montague's View: English as a Formal Language

Richard Montague

I reject the contention that an important theoretical difference exists between formal and natural languages. (Montague 1970a)

The Essence of Montague's Solution: Lambdas

- Move to a logic which contains predicate logic but also λ-abstraction over objects of any order.
- More often than not the forms of a sentence S and its standard formalization in predicate logic φ will not match. But we can often find a φ' such that:
  - φ is the normal form of φ' and hence φ and φ' are logically equivalent.
  - The form of S does match with that of φ'.

Application and Abstraction

- (Lambda Abstraction) $\lambda x.A$ denotes the function that, applied to some argument d, returns the value of A with x interpreted as d.
- (Application) AB denotes the result of applying the denotation of A to the denotation of B.
- $[\![\lambda x.A]\!]^a = \{\langle d, [\![A]\!]^{a[d/x]}\rangle \mid d \in D_\beta\}$, where D_β is the range of values that x can take (more on this later).
- $[\![AB]\!]^a = [\![A]\!]^a([\![B]\!]^a)$

Beta Conversion

- An essential rule is that of beta conversion:
  (β) $(\lambda x.A)B = A\{x := B\}$
- $A\{x := B\}$ stands for the result of substitution of B for any x that is free in A.
- Side condition on (β): no occurrence of a variable that is free in B may become bound in $A\{x := B\}$.
- x and B must be of the same type (more info on types will follow).
- Examples:
  - $(\lambda x.sing\, x \wedge dance\, x)bill = sing\, bill \wedge dance\, bill$
  - $(\lambda P' \lambda P \forall x\, [P'x \rightarrow Px])boy = \lambda P \forall x\, [boy\, x \rightarrow Px]$

The Essence of Montague's Solution: Example

- Here is an example that shows that Every boy admires a girl can have subject predicate form:

  S: [[Every boy][admires a girl]]
  φ': $(\lambda P \forall x\,[boy\,x \rightarrow Px])(\lambda z.\exists y[girl\,y \wedge admires\,zy])$
  φ: $\forall x[boy\,x \rightarrow \exists y[girl\,y \wedge admires\,xy]]$

- The trick can be repeated recursively; e.g. it is also possible to give a term that has the structure of [admires[a girl]] but is equivalent with $\lambda z.\exists y[girl\,y \wedge admires\,zy]$.
- The equivalence of φ and φ' is shown with the help of beta conversion:
- $(\lambda P \forall x\,[boy\,x \rightarrow Px])(\lambda z.\exists y[girl\,y \wedge admires\,zy])$
  $\forall x\,[boy\,x \rightarrow (\lambda z.\exists y[girl\,y \wedge admires\,zy])x]$
  $\forall x[boy\,x \rightarrow \exists y[girl\,y \wedge admires\,xy]]$

Translating Natural Language Expressions 1

Consider the following basic translations:

$$\begin{aligned}
\texttt{every}^\circ &= \lambda P' \lambda P \forall x\, [P'x \rightarrow Px] \\
\texttt{a}^\circ &= \lambda P' \lambda P \exists x\, [P'x \wedge Px] \\
\texttt{saw}^\circ &= \lambda Q \lambda x.Q(\lambda y.saw\, xy) \\
\texttt{man}^\circ &= man \\
\texttt{girl}^\circ &= girl
\end{aligned}$$

Add the rule $[AB]^\circ = A^\circ B^\circ$.
We shall translate `[[every man][saw[a girl]]]`.

Translating Natural Language Expressions 2

- $\texttt{[every man]}^\circ = \texttt{every}^\circ\,\texttt{man}^\circ =$
  $(\lambda P'\lambda P.\forall x\,[P'x \rightarrow Px])man = \lambda P\forall x\,[man\,x \rightarrow Px]$
- $\texttt{[a girl]}^\circ = \texttt{a}^\circ\,\texttt{girl}^\circ = (\lambda P'\lambda P.\exists x\,[P'x \wedge Px])girl =$
  $\lambda P\exists x\,[girl\,x \wedge Px]$
- $\texttt{[saw[a girl]]}^\circ = \texttt{saw}^\circ\,\texttt{[a girl]}^\circ) =$
  $(\lambda Q\lambda x.Q\lambda y.saw\,xy)\lambda P\exists x\,[girl\,x \wedge Px] =$
  $\lambda x.(\lambda P\exists x\,[girl\,x \wedge Px])\lambda y.saw\,xy =$
  $\lambda x.(\lambda P\exists z\,[girl\,z \wedge Pz])\lambda y.saw\,xy =$
  $\lambda x.\exists z\,[girl\,z \wedge (\lambda y.saw\,xy)z] = \lambda x.\exists z\,[girl\,z \wedge saw\,xz]$
- $\texttt{[[every man][saw[a girl]]]}^\circ =$
  $\texttt{[every man]}^\circ\,\texttt{[saw[a girl]]}^\circ =$
  $(\lambda P\forall x\,[man\,x \rightarrow Px])\lambda x.\exists z\,[girl\,z \wedge saw\,xz] =$
  $\forall x\,[man\,x \rightarrow (\lambda x.\exists z\,[girl\,z \wedge saw\,xz])x] =$
  $\forall x\,[man\,x \rightarrow \exists z\,[girl\,z \wedge saw\,xz]]$

Let's Evaluate

Evaluation

- This works quite well.
- We had to translate `saw` as $\lambda Q \lambda x.Q(\lambda y.saw\, xy)$.
- Translating `saw` simply as the binary relation *saw* would not have worked, as the latter does not combine with the quantifier $\lambda P \exists x\, [girl\, x \wedge Px]$ using application only. There is a type mismatch.
- It may be felt, however, that the translation of `saw` as $\lambda Q \lambda x.Q(\lambda y.saw\, xy)$ is not very intuitive. Later, we shall look into possibilities of translating transitive verbs as binary relations.

Scaling Up

In order to play this game with a larger portion of English we need to do three things.

1. Say something about syntax. This will provide us with the source of our translations.
   There are many possibilities. We will borrow ideas from generative grammar.
2. Develop the interpreting logic.
   Throughout the course we'll stick to (a many-sorted variant of) Church's classical type logic.
3. Give a set of translation rules sending syntactic structures to terms in the interpreting logic.

We'll do 2 first; then 1; then 3.

Types

- We start with a finite set of basic types, among which must be t (truth values). The set of basic types may also include e (entities), s (worlds), etc.
- If there are m basic types apart from t, the logic is called TY_m.
- The set of types is the smallest set such that:
  1. all basic types are types;
  2. if α and β are types then $(\alpha \rightarrow \beta)$ is a type;
- $\alpha \rightarrow \beta$ will very often be written $\alpha\beta$.

Frames

A frame is a set of non-empty sets $\{D_\alpha \mid \alpha \text{ is a type}\}$ such that

$$\begin{aligned} D_t &= \{0,1\}; \\ D_{\alpha\to\beta} &\subseteq \{f \mid f : D_\alpha \to D_\beta\}. \end{aligned}$$

A frame is standard if $D_{\alpha\to\beta} = \{f \mid f : D_\alpha \to D_\beta\}$ for all α, β.
We take no special interest in standard frames.

Relations as Functions

- Our types are functional and our frames are hierarchies of functions.
- What if we need sets or relations?
- Sets can be identified with their characteristic functions: if α is some type αt is the type of 'sets of αs'.
- Relations can be curried: A relation taking objects of type α_i $(1 \leq i \leq n)$ in its i-th argument is of type $(\alpha_1(\cdots(\alpha_n t)\cdots)$.
- For example, if e is the type of entities, $(et)((et)t)$ will be the type of binary relations between sets of entities.
- There are also type theories based on relations.

Terms

For each type α, assume the existence of a denumerably infinite set of variables VAR_α and a countable set of constants CON_α.

Definition (Terms)

Define, for each α, the set $\mathcal{T}_\alpha$ of terms of type α by the following induction.

1. $\mathrm{CON}_\alpha \subseteq \mathcal{T}_\alpha$, $\mathrm{VAR}_\alpha \subseteq \mathcal{T}_\alpha$;
2. $A \in \mathcal{T}_{\alpha\to\beta}$, $B \in \mathcal{T}_\alpha \Rightarrow (AB) \in \mathcal{T}_\beta$;
3. $A \in \mathcal{T}_\beta$, $x \in \mathrm{VAR}_\alpha \Rightarrow (\lambda x.\, A) \in \mathcal{T}_{\alpha\to\beta}$;
4. $A \in \mathcal{T}_\alpha$, $B \in \mathcal{T}_\alpha \Rightarrow (A = B) \in \mathcal{T}_t$.

We will write ABC for $((AB)C)$ etc. If $A \in \mathcal{T}_\alpha$ this may be indicated by writing A_α.

Abbreviations

$$
\begin{array}{rll}
\forall x_\alpha.\varphi & \text{is short for} & (\lambda x_\alpha.\varphi) = (\lambda x_\alpha.x = x) \\
\bot & \text{is short for} & \forall x_t.x_t \\
\top & \text{is short for} & \bot = \bot \\
\neg\varphi_t & \text{is short for} & \varphi = \bot \\
\varphi \wedge \psi & \text{is short for} & (\lambda f_{tt}.f\varphi = \psi) = (\lambda f_{tt}.f\top)
\end{array}
$$

- These abbreviations are essentially due to (Henkin 1950).
- Further abbreviations for $\vee$, $\rightarrow$, $\exists$, etc. are obtained in the standard way.

Interpretation: Preliminaries

- An interpretation function I for a frame $F = \{D_\alpha \mid \alpha \text{ is a type}\}$ is a function with the set of all constants as its domain such that $I(c_\alpha) \in D_\alpha$ for all c_α.
- Similarly, an assignment a for $\{D_\alpha \mid \alpha \text{ is a type}\}$ is a function taking variables as its arguments such that $a(x_\alpha) \in D_\alpha$ for all x_α.
- We write $a[d/x]$ for the assignment a' such that $a'(x) = d$ and $a'(y) = a(y)$ if $y \neq x$.
- The set of all assignments for F may be written $\mathcal{A}_F$.

General Models (Henkin 1950)

A general model is a triple $\langle F, I, V\rangle$ such that F is a frame $\{D_\alpha \mid \alpha \text{ is a type}\}$, I is an interpretation function for F and $V : \mathcal{A}_F \times \bigcup_\alpha \mathcal{T}_\alpha \to \bigcup F$ is a function such that $V(a, A_\alpha) \in D_\alpha$ for each a and A_α, while moreover

1. $V(a, c) = I(c)$, if c is a constant;
   $V(a, x) = a(x)$, if x is a variable
2. $V(a, AB) = V(a, A)(V(a, B))$;
3. $V(a, \lambda x_\beta.A) = \{\langle d, V(a[d/x], A)\rangle \mid d \in D_\beta\}$;
4. $V(a, A = B) = 1$ iff $V(a, A) = V(a, B)$.

We will write $[\![A]\!]^a$ for $V(a, A)$.

Values of Abbreviations

The values of abbreviated terms are as expected, e.g.

- $[\![\forall x_\alpha\, \varphi]\!]^a = 1$ iff $[\![\varphi]\!]^{a[d/x]} = 1$, for all $d \in D_\alpha$;
- $[\![\bot]\!]^a = 0$;
- $[\![\top]\!]^a = 1$;
- $[\![\neg\varphi]\!]^a = 1 - [\![\varphi]\!]^a$;
- $[\![\varphi \wedge \psi]\!]^a = 1$ iff $[\![\varphi]\!]^a = [\![\psi]\!]^a = 1$.

Entailment

- Let $\Gamma \cup \{\varphi\}$ be a set of formulas. Γ g-entails φ, $\Gamma \models_g \varphi$, iff, for each general model $\langle F, I, V \rangle$, and each assignment a for F, $V(a, \varphi) = 1$ if $V(a, \psi) = 1$ for all $\psi \in \Gamma$.
- $\models_g$ is recursively axiomatizable (Henkin 1950).
- Reasoning is classical with rules as expected.
- Extensionality is g-valid:

$$\models_g \forall XY(\forall \vec{x}(X\vec{x} \leftrightarrow Y\vec{x}) \rightarrow \forall Z(ZX \rightarrow ZY))$$

- We will come back to this last point. Having Extensionality is less than desirable in the context of natural language semantics.

The Plan

In one of the previous slides we the following plan was made.

1. Say something about syntax. This will provide us with the source of our translations.
   There are many possibilities. We will borrow ideas from generative grammar.
2. Develop the interpreting logic.
   Throughout the course we'll stick to (a many-sorted variant of) Church's classical type logic. Done
3. Give a set of translation rules sending syntactic structures to terms in the interpreting logic.

We will now look at syntax.

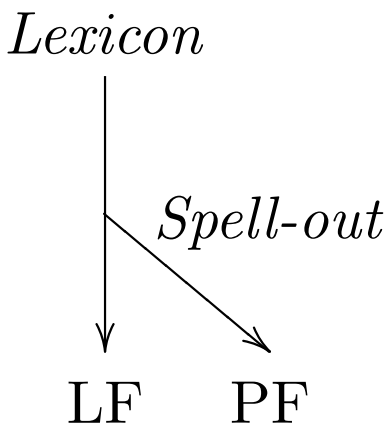
Our Choice of Syntax

- Syntax is no central issue in this course. On the other hand, we need syntactic structures to inductively define our semantics on.
- There are many competing syntactic theories. Clearly, semantic theory should remain agnostic about the question what constitutes a good syntactic theory.
- Fortunately, many syntactic theories lend themselves well to hooking up syntax and semantics and many decisions that must be made in semantic theory seem independent from decisions made in syntax.
- In this course we will assume that generative grammar provides us with syntactic input. Generative grammar (Chomsky's paradigm) is the syntactic theory that most linguists are familiar with.

Learning about Syntax

An excellent first course in syntax is (Santorini and Kroch 2000), which is readable online. In this course we have tried to work with syntactic analyses that are compatible with those provided in (Santorini and Kroch 2000).

Syntax: Overall Architecture



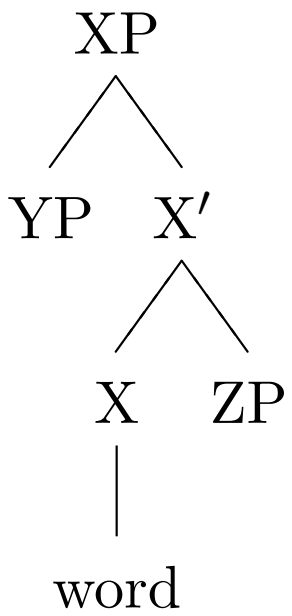
- Items are taken from a lexicon and enter into a derivation.
- The derivation essentially merges items into larger items.
- There are two kinds of merge: external merge and internal merge (= move).
- At spell-out the derivation splits. One part goes to Logical Form (LF), another to Phonological Form (PF).
- PF is input for articulation, LF for interpretation.

Merge and Move

- The merge operation takes two elements (words or trees).
  $[_{\mathrm{D}}$ the$]$ $[_{\mathrm{NP}}$ man$]$
- And merges them.
  $[_{\mathrm{D}'}[_{\mathrm{D}}$ the$]$ $[_{\mathrm{NP}}$ man$]]$
- Move takes an element that is internal to a tree:
  $[_{\mathrm{CP}}[_{\mathrm{C}'}$ was he reading $[_{\mathrm{DP}}$ what$]]]$
- And moves it to a higher position, leaving a trace:
  $[_{\mathrm{CP}}[_{\mathrm{DP}}$ what$]^{3}$ $[_{\mathrm{C}'}$ was he reading $\mathrm{t}_3]]$
- The element that was moved and its trace are coindexed.
- (A move can be explained as the merge of a tree with one of its internal elements, followed by erasure of the element that was copied. We will not pursue this.)

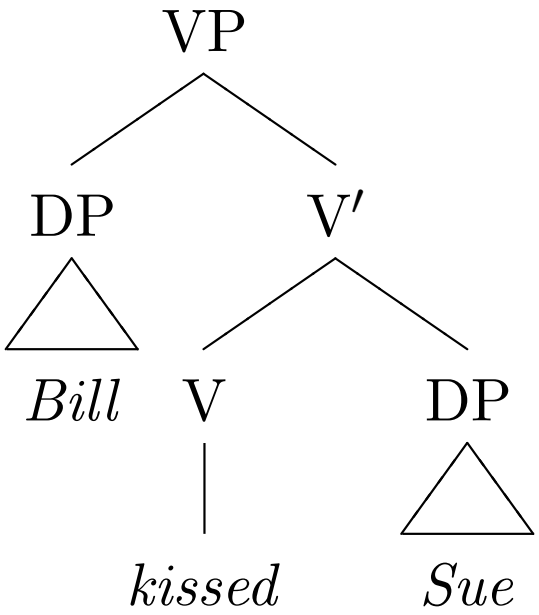
X' Theory

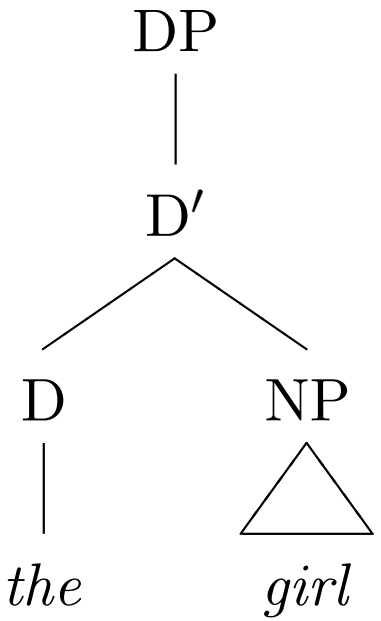
The X' theory holds that the basic units of phrase structure are built up according to the pattern on the left.



- X, Y and Z range over category labels.
- The word projects three levels of structure, X, X′ (pronounced *X-bar*), and XP.
- YP is called the specifier of XP; ZP the complement.
- Neither the specifier nor the complement need be present.

Examples



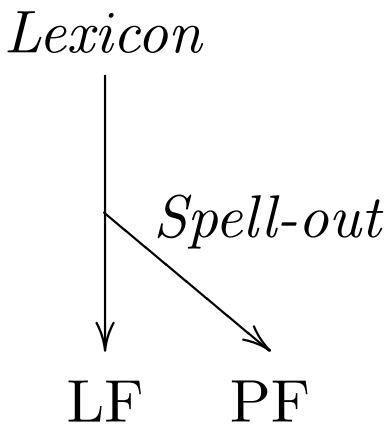


- Triangles hide structure.
- We will often use labeled bracketings instead of trees: $[_{\text{VP}}[_{\text{DP}}$ Bill$][_{\text{V}'}$ $[_{\text{V}}$ kissed$][_{\text{DP}}$ Sue$]]]$
- Brackets and labels that are irrelevant to an argument will often be suppressed.

VP Internal Subjects

- We will assume that subjects start within the Verb Phrase:
  $[_{\text{IP}}[_{\text{I}'}[_{\text{I}}\ \text{didn't}][_{\text{VP}}[_{\text{DP}}\ \text{every girl}][_{\text{V}'}\ \text{laugh}]]]]$
- And then move to the specifier of the Inflectional Phrase:
  $[_{\text{IP}}[_{\text{DP}}\ \text{every girl}]^{6}[_{\text{I}'}[_{\text{I}}\ \text{didn't}][_{\text{VP}}\ t_{6}[_{\text{V}'}\ \text{laugh}]]]]$
- This assumption is widely shared among syntacticians. What is still under discussion is where in the derivation this movement takes place.

Subject Raising as PF Movement



- We follow (Sauerland and Elbourne 2002) in assuming that this raising of subjects is PF movement, i.e. takes place between Spell-out and the PF interface.
- That means that
  $[_{\text{IP}}[_{\text{I}'}[_{\text{I}}\ \text{didn't}][_{\text{VP}}[_{\text{DP}}\ \text{every girl}][_{\text{V}'}\ \text{laugh}]]]]$
  can be used for interpretation.
- This is good as *every girl didn't laugh* has a reading in which the negation takes scope over the universal quantifier: $\neg\forall x\,(\mathit{girl}\,x \rightarrow \mathit{laugh}\,x)$.
- Note that there is also a $\forall x\,(\mathit{girl}\,x \rightarrow \neg\mathit{laugh}\,x)$ reading. We must provide for that reading too.

The Plan Again

We have now taken care of two of the three items on our list.

1. Say something about syntax. This will provide us with the source of our translations.
   There are many possibilities. We will borrow ideas from generative grammar. Done
2. Develop the interpreting logic.
   Throughout the course we'll stick to (a many-sorted variant of) Church's classical type logic. Done
3. Give a set of translation rules sending syntactic structures to terms in the interpreting logic.

So let's move on to the translation rules.

Less Structure

- The trees or labeled bracketings of syntactic theory provide more structure than we actually need.
- Category labels will not be needed for interpretation, so we will drop them from our representation.
- Nodes in linguistic trees can be in a relation of precedence, but this ordering is irrelevant to semantics (i.e. we will not distinguish between [XY] and [YX]).
- We will need the superscripts and subscripts that result from movement operations however: Traces t_n will be translated as free variables x_n and superscripts will correspond to an abstraction over such variables.

Proto Logical Forms

- Our characterization of the class of Logical Forms has not been entirely precise.
- Fortunately, no such precision is needed for semantics. We will use a class of structures that we may call proto-LFs. Starting with some set of words, it can be defined as follows.
  - Each word is a proto-LF;
  - If X is a proto-LF, then [X] is a proto-LF;
  - If X and Y are proto-LFs, then [XY] is a proto-LF;
  - If X and Y are proto-LFs, then [X^n Y] is a proto-LF, for each numeral n.
- There is a lot of uninterpretable gibberish among the proto-LFs, but the set also includes the LFs that we are interested in.

Type Driven Translation

- We will give rules for a type driven translation of syntactic structures (Klein and Sag 1985).
- Words are assigned one or more meanings by the lexicon.
- Rules will assign meanings to complex structures on the basis of the meanings of their constituents.
- Type driven translation is not based on a lot of explicit rules for particular structures as Montague's system was. Instead a minimal set of rules is given and translations almost seem to take care of themselves.
- This is why Emmon Bach coined the term shake 'n bake semantics for type driven translation.

Type Driven Translation, continued

- The rules on the next slide will have as side-conditions that certain logical expressions must be well-formed. This is were the type driven aspect comes in. If types don't match there is no translation!
- The translation is neither functional nor total. An expression may have several (non-equivalent) meanings, in which case it is ambiguous, or none, in which case it is uninterpretable.
- One rule will refer to type shifting, which we will explain shortly.
- We are interested in translation modulo logical equivalence. So if X translates as A and A is equivalent with B, X also translates as B.

Translation Rules

- **Basic Rule**
  $X \rightsquigarrow A$ if $X \rightsquigarrow A$ is given as a basic translation.
- **Copying**
  If $X \rightsquigarrow A$ then $[X] \rightsquigarrow A$.
- **Application**
  If $X \rightsquigarrow A$ and $Y \rightsquigarrow B$ then
  - $[XY] \rightsquigarrow AB$ if AB is well-formed;
  - $[XY] \rightsquigarrow BA$ if BA is well-formed.
- **Quantifying In**
  If $X \rightsquigarrow A$ and $Y \rightsquigarrow B$ then $[X^n Y] \rightsquigarrow A(\lambda v_n.B)$, provided $A(\lambda v_n.B)$ is well-formed.
- **Type Shifting**
  If $X \rightsquigarrow A$ and A shifts to B then $X \rightsquigarrow B$.

Abstract Notation for Types

- It is useful to have a special notation σ for the type that will be associated with sentences.
- For the first fragment we are going to consider we shall set $\sigma := t$, i.e. the meaning of a sentence will be identified with a truth value. Later, the value for σ will be redefined.
- This will allow us to retain much of our previous work when we decide that sentences should have other kinds of values.
- The abstract type associated with names will be ν (the Greek letter n). Concretely, we set $\nu := e$ (where e stands for entity). This will also be changed later.

Typographical Conventions

The following typographical conventions will be used to indicate the types of variables.

VARIABLES	TYPE	KIND OF OBJECT
x, y, z	e	entity
v	ν	
p, q	σ	proposition
P	$\nu\sigma$	predicate
Q	$(\nu\sigma)\sigma$	quantifier

For variables whose type is irrelevant, we will usually employ X and Y. Sequences of variables $X_1, \ldots, X_n$ are usually denoted $\vec{X}$. Types of the form $\alpha_1(\cdots(\alpha_n t)\cdots)$ will be written as $\vec{\alpha}t$.

Some Basic Translations

$$
\begin{aligned}
\text{doesn't} &\rightsquigarrow \lambda p.\neg p \\
\text{PRES} &\rightsquigarrow \lambda p.p \\
\text{run} &\rightsquigarrow run_{et} \\
\text{man} &\rightsquigarrow man_{et} \\
\text{find} &\rightsquigarrow \lambda y \lambda x.find_{e(et)}\, xy \\
\text{be} &\rightsquigarrow \lambda y \lambda x.x = y \\
\text{Bill} &\rightsquigarrow bill_e \\
\text{every} &\rightsquigarrow \lambda P' \lambda P.\forall x(P'x \rightarrow Px) \\
\text{some, a} &\rightsquigarrow \lambda P' \lambda P.\exists x(P'x \wedge Px) \\
\text{no} &\rightsquigarrow \lambda P' \lambda P.\neg\exists x(P'x \wedge Px) \\
\text{the} &\rightsquigarrow \lambda P' \lambda P.\exists x(\forall y(P'y \leftrightarrow x = y) \wedge Px) \\
\text{t}_n &\rightsquigarrow v_n, \text{ for each } n
\end{aligned}
$$

Comments on Basic Translations

- The basic translations just given were entered in a type that is as low as possible.
- Some of these translations are meant as paradigms; e.g., since run translates as run_{et}, we may take it that walk translates as $walk_{et}$.
- We'll add more basic translations as we go along.
- Basic translations will need to change if the choices $\sigma := t$ and $\nu := e$ change.
- The translation of the, $\lambda P' \lambda P . \exists x (\forall y (P' y \leftrightarrow x = y) \wedge P x)$, embodies Russell's theory of definite descriptions.

Bill doesn't like Mary

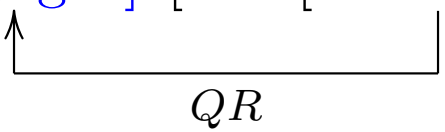
- [doesn't [Bill [like Mary]]]
  (Remember that subject raising is assumed to occur at PF)
- Mary $\rightsquigarrow mary_e$
  like $\rightsquigarrow \lambda y \lambda x.like_{e(et)}\, xy$
  [like Mary] $\rightsquigarrow \lambda x.like\, x\, mary$
  Bill $\rightsquigarrow bill_e$
  [Bill [like Mary]] $\rightsquigarrow like\, bill\, mary$
  doesn't $\rightsquigarrow \lambda p.\neg p$
  [doesn't [Bill [like Mary]]] $\rightsquigarrow \neg like\, bill\, mary$

Bill doesn't like every girl

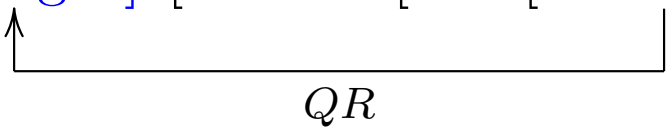
- [doesn't [Bill [like [every girl]]]]
- every $\rightsquigarrow \lambda P' \lambda P. \forall x(P'x \rightarrow Px)$
  girl $\rightsquigarrow girl_{et}$
  [every girl] $\rightsquigarrow \lambda P. \forall x(girl\, x \rightarrow Px)$
  like $\rightsquigarrow \lambda y \lambda x. like_{e(et)}\, xy$
- Problem: no translation for [like [every girl]]!!
  (The types don't match.)
- One possible solution: Quantifier Raising.

Quantifier Raising (QR, May 1977)

- Movement of DP.
- May assumed that QR adjoins the DP to S (= IP or CP), but we will allow adjunction to any higher XP node.
- [doesn't[Bill [like [every girl]]]]
- [doesn't[[every girl]^1[Bill [like t_1]]]]



- [[every girl]^1[doesn't [Bill [like t_1]]]]



- Since QR is a movement operation, the expectation is that it should be subject to the same kind of syntactic constraints as other movement operations are.

Bill doesn't like every girl

- $\mathrm{t}_1 \rightsquigarrow v_1$
  like $\rightsquigarrow \lambda y \lambda x.like_{e(et)}\, xy$
  [like t_1] $\rightsquigarrow \lambda x.like\, x\, v_1$
  [Bill [like t_1]] $\rightsquigarrow like\ bill\ v_1$
  [every girl] $\rightsquigarrow \lambda P.\forall x(girl\, x \rightarrow Px)$
  [[every girl]^1[Bill [like t_1]]] $\rightsquigarrow$
  $(\lambda P.\forall x(girl\, x \rightarrow Px))(\lambda v_1.like\ bill\ v_1)$
  [[every girl]^1[Bill [like t_1]]] $\rightsquigarrow \forall x(girl\, x \rightarrow like\ bill\, x)$
  [doesn't[[every girl]^1[Bill [like t_1]]]] $\rightsquigarrow$
  $\neg\forall x(girl\, x \rightarrow like\ bill\, x)$
- [[every girl]^1[doesn't[Bill [like t_1]]]] $\rightsquigarrow$
  $\forall x(girl\, x \rightarrow \neg like\ bill\, x)$

Every girl likes a boy 1

- [PRES[[every girl][like[a boy]]]]
  [PRES[[a boy]^2[[every girl][like t_2]]]]
- [like t_2] $\rightsquigarrow \lambda x.like\, x\, v_2$
  [every girl] $\rightsquigarrow \lambda P.\forall x(girl\, x \rightarrow Px)$
  [[every girl][like t_2]] $\rightsquigarrow \forall x(girl\, x \rightarrow like\, x\, v_2)$
  [a boy] $\rightsquigarrow \lambda P.\exists x(boy\, x \wedge Px)$
  [[a boy]^2[[every girl][like t_2]]] $\rightsquigarrow$
  $\exists x(boy\, x \wedge \forall y(girl\, y \rightarrow like\, yx))$
  [PRES[[a boy]^2[[every girl][like t_2]]]] $\rightsquigarrow$
  $\exists x(boy\, x \wedge \forall y(girl\, y \rightarrow like\, yx))$
- 'There is a specific boy that every girl likes'

Every girl likes a boy 2

- [PRES[[every girl][like[a boy]]]]
  [PRES[[a boy]^2[[every girl][like t$_2$]]]]
  [PRES[[every girl]^3[[a boy]^2[t$_3$[like t$_2$]]]]]
- [like t$_2$] $\rightsquigarrow \lambda x.\mathit{like}\, x v_2$
  [t$_3$[like t$_2$]] $\rightsquigarrow \mathit{like}\, v_3 v_2$
  [a boy] $\rightsquigarrow \lambda P.\exists x(\mathit{boy}\, x \wedge Px)$
  [[a boy]^2[t$_3$[like t$_2$]]] $\rightsquigarrow \exists x(\mathit{boy}\, x \wedge \mathit{like}\, v_3 x)$
  [every girl] $\rightsquigarrow \lambda P.\forall x(\mathit{girl}\, x \rightarrow Px)$
  [[every girl]^3[[a boy]^2[t$_3$[like t$_2$]]]] $\rightsquigarrow$
  $\forall x(\mathit{girl}\, x \rightarrow \exists y(\mathit{boy}\, y \wedge \mathit{like}\, xy))$
  [PRES[[every girl]^3[[a boy]^2[t$_3$[like t$_2$]]]]] $\rightsquigarrow$
  $\forall x(\mathit{girl}\, x \rightarrow \exists y(\mathit{boy}\, y \wedge \mathit{like}\, xy))$
- 'For every girl, there is a boy that she likes'.

Dealing with Ambiguity

- The theory predicts that the usual sentence will have many readings.
- For example, even a sentence as pedestrian as every girl doesn't like a boy is predicted to have four readings here (Exercise).
- Are all these readings really always available?
- How do language users pick out intended readings?
- The answer probably is that people make heavy use of context to do all kinds of reasoning, filtering out unintended readings.
- Our understanding of this process is very imperfect.

Computational Aspects

- In this course we concentrate on the linguistic aspects of semantic theory. However, the theory is also amenable to implementation.
- For many forms of grammar, there are efficient parsers.
- λ-reductions are easily implemented. Some programming languages are even based on lambdas.
- Assigning meanings to trees presents no computational difficulties either.
- Entailment and consistency can be checked with the help of theorem provers.
- (Blackburn and Bos 2005) is an excellent book on computational semantics. It develops techniques that are wholly consistent with the approach followed in this course.

Dealing with Ambiguity in a Computational Setting

- Ambiguity is an important bottleneck for natural language processing.
- The problem is that the number of readings of a text increases very fast as the input is read. In fact the increase is non-polynomial in the length of the input.
- This is true for semantic ambiguity but also for syntactic ambiguity.
- Computational linguists have worked on underspecified representations that allow one to avoid summing up all readings.

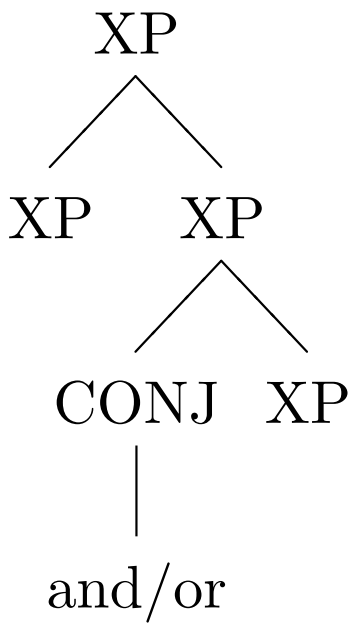
Wh Relative Clauses

- A wh relative clause is thought to result from moving a wh-phrase like who to the specifier of a CP (Complementizer Phrase).
- In (standard) English the complementizer C must be empty if the specifier of CP is not.
- $[_{\mathrm{N}'}\text{girl}[_{\mathrm{CP}}\text{who}^2[_{\mathrm{C}'}\varnothing[_{\mathrm{IP}}\mathrm{PRES}[_{\mathrm{VP}}\text{John}[_{\mathrm{V}'}\text{like } t_2]]]]]]$
- $[_{\mathrm{C}}\varnothing] \rightsquigarrow \lambda p.p$
  who $\rightsquigarrow \lambda P'\lambda P\lambda x(Px \wedge P'x)$

girl who John likes

- $[\text{girl}[\text{who}^2[\varnothing[\textsc{pres}[\text{John}[\text{like } \text{t}_2]]]]]]$
- $[\text{John}[\text{like } \text{t}_2]] \rightsquigarrow \mathit{like}\ \mathit{john}\ v_2$
  $[\textsc{pres}[\text{John}[\text{like } \text{t}_2]]] \rightsquigarrow \mathit{like}\ \mathit{john}\ v_2$
  $[\varnothing[\textsc{pres}[\text{John}[\text{like } \text{t}_2]]]] \rightsquigarrow \mathit{like}\ \mathit{john}\ v_2$
  $[\text{who}^2[\varnothing[\textsc{pres}[\text{John}[\text{like } \text{t}_2]]]]] \rightsquigarrow \lambda P \lambda x(Px \wedge \mathit{like}\ \mathit{john}\ x)$
  $[\text{girl}[\text{who}^2[\varnothing[\textsc{pres}[\text{John}[\text{like } \text{t}_2]]]]]] \rightsquigarrow \lambda x(\mathit{girl}\ x \wedge \mathit{like}\ \mathit{john}\ x)$

Coordination



- We will assume that all XPs can be coordinated according to the schema on the left.
- Barring exceptions, the semantic value of and is intersection, that of or union (Von Stechow 1974; Keenan and Faltz 1978; Gazdar 1980).
- Exceptions can be found in the DP domain: *John and Mary lifted a piano.*
- and $\rightsquigarrow \lambda R' \lambda R \lambda \vec{X}(R\vec{X} \wedge R'\vec{X})$
  or $\rightsquigarrow \lambda R' \lambda R \lambda \vec{X}(R\vec{X} \vee R'\vec{X})$
- Here R and R' can be of any type $\vec{\alpha}t$, with $\vec{X}$ a sequence of variables of types $\vec{\alpha}$.

Examples

- [$_{\text{NP}}$ linguist [and logician]]
- [$_{\text{AP}}$ green-eyed [and handsome]]
- [$_{\text{PP}}$ [in Edinburgh] [and [around the globe]]]
- [$_{\text{DP}}$ [a linguist] [or [every logician]]]
- [$_{\text{DP}}$ John [and [two girls]]]
- [$_{\text{IP}}$ every girl laughed [and two boys smiled]]
- [$_{\text{CP}}$ that every girl laughed [and that two boys smiled]]
- [girl [$_{\text{CP}}$ who every boy likes [and who some women admire]]]

green-eyed and handsome man

- [[green-eyed [and handsome]] man]
- green-eyed $\rightsquigarrow \lambda P \lambda x(\mathit{greeneyed}\, x \wedge Px)$
- handsome $\rightsquigarrow \lambda P \lambda x(\mathit{handsome}\, x \wedge Px)$
  and $\rightsquigarrow \lambda R'_{(et)(et)} \lambda R_{(et)(et)} \lambda P \lambda x(RPx \wedge R'Px)$
  [and handsome] $\rightsquigarrow \lambda R \lambda P \lambda x(RPx \wedge \mathit{handsome}\, x \wedge Px)$
  [green-eyed[and handsome]] $\rightsquigarrow$
  $\lambda P \lambda x(\mathit{greeneyed}\, x \wedge Px \wedge \mathit{handsome}\, x \wedge Px)$
  [green-eyed[and handsome]] $\rightsquigarrow$
  $\lambda P \lambda x(\mathit{greeneyed}\, x \wedge \mathit{handsome}\, x \wedge Px)$
  [[green-eyed[and handsome]]man] $\rightsquigarrow$
  $\lambda x(\mathit{greeneyed}\, x \wedge \mathit{handsome}\, x \wedge \mathit{man}\, x)$

A linguist and every logician dance

- [[[a linguist] [and [every logician]]]dance]
- [a linguist] $\rightsquigarrow \lambda P.\exists x(\mathit{linguist}\, x \wedge Px)$
  [every logician] $\rightsquigarrow \lambda P.\forall x(\mathit{logician}\, x \rightarrow Px)$
  and $\rightsquigarrow \lambda Q'\lambda Q\lambda P(QP \wedge Q'P)$
  [and[every logician]] $\rightsquigarrow \lambda Q\lambda P(QP \wedge \forall x(\mathit{logician}\, x \rightarrow Px))$
  [[a linguist] [and[every logician]]] $\rightsquigarrow$
  $\lambda P(\exists x(\mathit{linguist}\, x \wedge Px) \wedge \forall x(\mathit{logician}\, x \rightarrow Px))$
  [[[a linguist] [and[every logician]]]dance] $\rightsquigarrow$
  $\exists x(\mathit{linguist}\, x \wedge \mathit{dance}\, x) \wedge \forall x(\mathit{logician}\, x \rightarrow \mathit{dance}\, x)$

Some Problems

- [DP John [and [two girls]]]
- Here John does not translate as a relation and therefore intersection does not apply.
- [PP [in Edinburgh] [and [around the globe]]]
- If it is assumed that in translates as a relation of some type $e(\alpha t)$, it will combine with a type e term *edinburgh*; but around, presumably of the same type $e(\alpha t)$, cannot combine with the quantifier ($(et)t$-term) that translates the globe.
- Quantifier Raising is not really an option, as it is assumed (for good reason!) that that movement cannot escape the coordination.

Type Shifting

- We have decided to enter all translations at a level as low as possible. For example, we decided against translating find as a term of type $((et)t)(et)$, as Montague did (in essence), but opted for the lower $e(et)$.
- But sometimes a bit more combinatory glue is needed. For example, Montague also translated a name like John as $\lambda P.P\ john$, the set of all of John's properties. With such a translation the coordination [$_{\mathrm{DP}}$ John [and [two girls]]] no longer presents a problem.
- A solution is to introduce type shifting rules into the system. The idea is that once an item has a translation, certain other translations may be derivable.

Montague's Rule

- Montague's Rule: A_α shifts to $\lambda X_{\alpha\sigma}.XA$
- Remember the last of our translation rules: if $X \rightsquigarrow A$ and A shifts to B then $X \rightsquigarrow B$.
- Remember also that $\sigma = t$ for the moment. Later on we will make other choices for σ, but Montague's Rule will remain in force.
- The result is that John $\rightsquigarrow \lambda P.Pjohn$, which has the same type as other DPs $((et)t)$.
- Note that there is a one-to-one correspondence between a type α object a and the set of all type αt sets that have a as an element.

John and two girls dance

- [[John [and [two girls]]]dance]
- John $\rightsquigarrow$ *john*
  John $\rightsquigarrow \lambda P.Pjohn$
  two $\rightsquigarrow \lambda P'\lambda P.\exists xy(x \neq y \wedge P'x \wedge P'y \wedge Px \wedge Py)$
  (two as $at\ least\ two$)
  [two girls] $\rightsquigarrow \lambda P.\exists xy(x \neq y \wedge girl\, x \wedge girl\, y \wedge Px \wedge Py)$
  and $\rightsquigarrow \lambda Q'\lambda Q\lambda P(QP \wedge Q'P)$
  [and[two girls]] $\rightsquigarrow$
  $\lambda Q\lambda P(QP \wedge \exists xy(x \neq y \wedge girl\, x \wedge girl\, y \wedge Px \wedge Py))$
  [John[and[two girls]]] $\rightsquigarrow$
  $\lambda P(Pjohn \wedge \exists xy(x \neq y \wedge girl\, x \wedge girl\, y \wedge Px \wedge Py))$
  [[John[and[two girls]]]dance] $\rightsquigarrow$
  $dance\ john \wedge \exists xy(x \neq y \wedge girl\, x \wedge girl\, y \wedge dance\, x \wedge dance\, y)$

More Type Shifting

- Earlier on Quantifier Raising was motivated with the argument that it was needed to get the combinatorics of such phrases as [likes [every girl]] right.
- But now we have seen how type shifting may oil the combinatory process. Isn't there a type shifting rule that allows us to combine transitive verbs with their quantifier phrase objects?
- The following rule essentially is from (Partee and Rooth 1983). It (in fact: a generalization) is called Argument Raising in (Hendriks 1993).
- $A_{\nu(\vec{\alpha}\sigma)}$ shifts to $\lambda Q \lambda \vec{Y}.Q(\lambda v.Av\vec{Y})$
- E.g. $\lambda x \lambda y.love\, yx$ shifts to $\lambda Q \lambda y.Q(\lambda x.love\, yx)$

More Type Shifting, continued

- $\lambda Q \lambda y.Q(\lambda x.love\, xy)$ was Montague's original translation; with type shifting it becomes derivable.
- An alternative (we shall in fact adopt it) is to typeshift quantifiers. Let's call the following rule Quantifier Shifting.
- $A_{(\nu\sigma)\sigma}$ shifts to $\lambda R_{\nu(\vec{\alpha}\sigma)} \lambda \vec{Y}.A(\lambda v.Rv\vec{Y})$
- E.g. $\lambda P.\exists x(girl\, x \wedge Px)$ shifts to $\lambda R_{e(et)} \lambda y.\exists x(girl\, x \wedge Rxy)$.
- Applying the latter translation to $\lambda x \lambda y.love\, yx$ gives $\lambda y.\exists x(girl\, x \wedge love\, yx)$, the translation of [love[a girl]].

pen in a drawer or on a desk

- [pen [[in [a drawer]] or [on [a desk]]]]
- in $\rightsquigarrow \lambda y \lambda P \lambda x (Px \wedge in\, xy)$

  [a drawer] $\rightsquigarrow \lambda P.\exists x (drawer\, x \wedge Px)$

  [a drawer] $\rightsquigarrow \lambda R_{e((et)(et))} \lambda P \lambda x.\exists y (drawer\, y \wedge RyPx)$

  [in[a drawer]] $\rightsquigarrow \lambda P \lambda x.\exists y (drawer\, y \wedge Px \wedge in\, xy)$

  [on[a desk]] $\rightsquigarrow \lambda P \lambda x.\exists y (desk\, y \wedge Px \wedge on\, xy)$

  or $\rightsquigarrow \lambda R'_{(et)(et)} \lambda R_{(et)(et)} \lambda P \lambda x.(RPx \vee R'Px)$

  [or[on[a desk]]] $\rightsquigarrow \lambda R \lambda P \lambda x.(RPx \vee \exists y (desk\, y \wedge Px \wedge on\, xy))$

  [[in[a drawer]][or[on[a desk]]]] $\rightsquigarrow$

  $\lambda P \lambda x.(\exists y (drawer\, y \wedge Px \wedge in\, xy) \vee \exists y (desk\, y \wedge Px \wedge on\, xy))$

  [[in[a drawer]][or[on[a desk]]]] $\rightsquigarrow$

  $\lambda P \lambda x.(Px \wedge \exists y (drawer\, y \wedge in\, xy) \vee \exists y (desk\, y \wedge on\, xy))$

  [pen[[in[a drawer]][or[on[a desk]]]]] $\rightsquigarrow$

  $\lambda x.(pen\, x \wedge \exists y (drawer\, y \wedge in\, xy) \vee \exists y (desk\, y \wedge on\, xy))$

Pronouns

At least three ways to use pronouns are usually distinguished.

- The deictic or referential use.
  He doesn't drink
- The 'bound variable' use.
  Every boy likes a girl who hates him
- Other uses, e.g.
  Few congressmen admire Kennedy and they are very junior (Evans 1980)
  Every farmer who owns a donkey beats it (Geach 1962)

Deictic/Referential uses of pronouns

- Deictic pronouns are often treated as free variables.
- If he $\rightsquigarrow x$, it is clear that the rest of our rules predict that [PRES[doesn't[he drink]]] $\rightsquigarrow \neg \mathit{drink}\, x$.
- The idea is that the language user can fill in the value of the free variable on the basis of contextually salient information.
- The value of x can be set to an individual that is salient in the situation of utterance, e.g. through pointing.
- Or the value could become salient through the linguistic context, as in John went home early. He doesn't drink.
- But how to deal with A man entered. He sat down.?

Bound uses of pronouns

- In fact bound variable pronouns can also be viewed as starting their semantic life as free variables that happen to get bound in the interpretation process.
- Consider
  $[[\text{every boy}]^1[\text{t}_1[\text{love}[\text{a}[\text{girl}[\text{who}^2[\text{doesn't}[\text{t}_2[\text{hate him}]]]]]]]]]$
- Suppose the translation him $\rightsquigarrow v_1$ is chosen; then
  $[\text{t}_1[\text{love}[\text{a}[\text{girl}[\text{who}^2[\text{doesn't}[\text{t}_2[\text{hate him}]]]]]]]] \rightsquigarrow$
  $\exists x(\mathit{girl}\, x \wedge \neg \mathit{hate}\, x v_1 \wedge \mathit{love}\, v_1 x)$
- Quantifying in [every boy] then gives the desired
  $\forall y(\mathit{boy}\, y \rightarrow \exists x(\mathit{girl}\, x \wedge \neg \mathit{hate}\, xy \wedge \mathit{love}\, yx))$
- Other choices for the translation of him lead to deictic interpretations.

Indexing in Syntax

- The previous slide sketched a theory of pronouns in which they are multiply ambiguous: he $\rightsquigarrow v_n$, for all n. For some choices of n, the variable may get bound in the interpretation process.
- This competes with a theory in which pronouns are indexed by syntax. In such an approach we may find the following form at LF:
- [[every boy]^1[t$_1$[love[a[girl[who^2[doesn't[t$_2$[hate him$_1$]]]]]]]]]
- The translation rule then becomes he$_n \rightsquigarrow v_n$.
- The indexation is necessary for the syntactic Binding Theory.

E-type Pronouns

- *Few congressmen admire Kennedy and they are very junior* does not mean *for few congressmen x, x admires Kennedy and x is very junior.*
- It does mean something like
  Few congressmen admire Kennedy and the congressmen who admire Kennedy are very junior
- The theory that some ('E-type') pronouns should be treated as a definite description whose content depends on the antecedent context stems from (Evans 1980).
- What are the exact rules that regulate the formation of these descriptions?

To be or not to be

- A received wisdom of philosophy holds that the is of predication (*Wanda is a fish*) differs from the is of identity (*John is Bill*).
- However, (Montague 1973) showed that both can be covered with a single symbolization of is.
- Our translation of be, $\lambda y \lambda x.x = y$, differs from Montague's in two respects.
  - It does not take modalities into account (that will come).
  - It was entered in the lowest possible type, like other verbs.
- However, it still covers both the is of predication and the is of identity, as we shall see now.

Identity and Predication

- [PRES[John[is Bill]]] $\rightsquigarrow$ $john = bill$
- [Wanda[is [a fish]]]
- is $\rightsquigarrow$ $\lambda y \lambda x.x = y$
  [a fish] $\rightsquigarrow$ $\lambda P.\exists x(fish\ x \wedge Px)$
  [a fish] $\rightsquigarrow$ $\lambda R_{e(et)} \lambda y.\exists x(fish\ x \wedge Rxy)$
  [is[a fish]] $\rightsquigarrow$ $\lambda y.\exists x(fish\ x \wedge y = x)$
  [is[a fish]] $\rightsquigarrow$ $\lambda y.fish\ y$
  [Wanda[is[a fish]]] $\rightsquigarrow$ $fish\ wanda$
- How should one deal with Socrates is white, the oldest of all example sentences?

Adjectives 1

There are at least three kinds of adjectives:

- Intersective adjectives like green-eyed.
  - John is a green-eyed man $\models$ John is a man.
  - John is a green-eyed man, Every man is an animal $\models$ John is a green-eyed animal
- Degree adjectives like small.
  - Jumbo is a small elephant $\models$ Jumbo is an elephant
  - Jumbo is a small elephant, every elephant is an animal $\not\models$ Jumbo is a small animal
- Other adjectives like fake, alleged.
  - John is an alleged murderer $\not\models$ John is a murderer

Adjectives 2

- green-eyed $\rightsquigarrow \lambda P \lambda x(\mathit{greeneyed}\, x \wedge Px)$
  [green-eyed man] $\rightsquigarrow \lambda x(\mathit{greeneyed}\, x \wedge \mathit{man}\, x)$
- This works well for intersective adjectives, but what about the others? Montague's solution: just make them functions that change predicates into predicates (type (et)(et) in the present set-up).
- small $\rightsquigarrow \mathit{small}_{(et)(et)}$
  [small elephant] $\rightsquigarrow \mathit{small}\ \mathit{elephant}$
  [Jumbo[is[a[small elephant]]]] $\rightsquigarrow \mathit{small}\ \mathit{elephant}\ \mathit{jumbo}$
- Similar, for fake etc.
- But does this give the right logic?

Meaning Postulates

- The translation small $\rightsquigarrow$ $small_{(et)(et)}$ does not predict that small elephants are elephants.
- One possibility: find a subtler translation of small, for example one that builds in a comparison with a contextually determined value.
- Another possibility: Axiomatize the desired behaviour, by stipulating that $\forall P \forall x(\delta P x \rightarrow P x)$, where δ is *small*, *large*, *tall*, etc.
- Such lexical axioms are called meaning postulates.

More Structure

- Up till now we have primarily been concerned with the combinatorics of translating natural language into logic.
- We have looked at various constructions, have discussed the possibilities of translating these and have met some challenges.
- But the translations for sentences that were obtained were very simple, basically first-order statements about ordinary cabbages and kings individuals.
- There were no intervals of time, no possible situations or worlds, no events, etc.
- Yet, it seems that natural language abounds with such things. Moreover, it seems that language contains operators that quantify over objects like worlds and times.
- We need more structure in our models.

Natural Language Metaphysics and Real Metaphysics

- But do all the things (worlds, events, reference points, plural individuals) that semanticists dream of exist at all? Many with a philosophical outlook on things will favour a more parsimonious ontology.
- However, a distinction should be made between the claim that certain entities exist and the claim that natural language assumes that they exist.
- For example, it is consistent to hold that the description of natural language semantics needs possible worlds, but that in fact there are no such entities at all.
- We are interested here in what (Bach 1986) has called Natural Language Metaphysics, not in real metaphysics.
- What is the ontology of the linguistic system? With what kind of data structures does it work?

Introducing Worlds 1

- Many philosphers and semanticists have found it expedient to base the semantics of natural language on possible worlds.
- With the help of worlds a range of constructions can be studied: modals, counterfactual conditionals, propositional attitudes, etc.
- Usually the introduction of possible worlds is accompanied by a shift to a modal logic. Montague, for example, developed his theory with the help of a higher order modal logic IL.
- But this will not be our strategy here. We will use classical logic to talk about possible worlds.
- This follows (Gallin 1975), who embeds IL into the two-sorted type theory TY_2.

Introducing Worlds 2

- We will assume a type s of indices. For the moment indices will be identified with worlds.
- Variables i, j, and k will typically be used to range over indices.
- The abstract type σ will now be reset to the type of sets of possible worlds: $\sigma := st$. ν will continue to have the value e.
- This has consequences for our convention for the use of variables; p will now be of type st, P of type $e(st)$, etc.

Typographical Conventions

VARIABLES	TYPE	KIND OF OBJECT
i, j, k	s	world
x, y, z	e	entity
v	ν	
p, q	σ	proposition
P	$\nu\sigma$	predicate
Q	$(\nu\sigma)\sigma$	quantifier

New Basic Translations 1

- We need new basic translations.
- Each predicate should come with an extra argument for the relevant possible world now. E.g. $run_{e(st)}\ bill\ i$ now expresses that Bill is running in world i.
- The proposition *Bill is running* corresponds to a set of possible worlds: $\lambda i.run_{e(st)}\ bill\ i$, which is equivalent with $run_{e(st)}\ bill$.
- Basic translations that expect predicates or propositions as arguments must also be adapted in order to take care of these extra arguments.
- The following list gives the new basic translations.

New Basic Translations 2

doesn't	$\rightsquigarrow$	$\lambda p \lambda i. \neg pi$	PRES	$\rightsquigarrow$	$\lambda p.p$
run	$\rightsquigarrow$	$run_{e(st)}$	man	$\rightsquigarrow$	$man_{e(st)}$
find	$\rightsquigarrow$	$\lambda y \lambda x. find_{e(e(st))}\, xy$	be	$\rightsquigarrow$	$\lambda y \lambda x \lambda i. x = y$
Bill	$\rightsquigarrow$	$bill_e$	small	$\rightsquigarrow$	$small_{(e(st))(e(st))}$
who	$\rightsquigarrow$	$\lambda P' \lambda P \lambda x \lambda i. (P'xi \wedge Pxi)$	fake	$\rightsquigarrow$	$fake_{(e(st))(e(st))}$
blue	$\rightsquigarrow$	$\lambda P \lambda x \lambda i. (blue_{e(st)}\, xi \wedge Pxi)$	if	$\rightsquigarrow$	$\lambda p \lambda q \lambda i. (pi \rightarrow qi)$
and	$\rightsquigarrow$	$\lambda R' \lambda R \lambda \vec{X} (R\vec{X} \wedge R'\vec{X})$			
or	$\rightsquigarrow$	$\lambda R' \lambda R \lambda \vec{X} (R\vec{X} \vee R'\vec{X})$			
t_n	$\rightsquigarrow$	v_n, for each n			
he_n	$\rightsquigarrow$	v_n, for each n			
every	$\rightsquigarrow$	$\lambda P' \lambda P \lambda i. \forall x (P'xi \rightarrow Pxi)$			
some, a	$\rightsquigarrow$	$\lambda P' \lambda P \lambda i. \exists x (P'xi \wedge Pxi)$			
no	$\rightsquigarrow$	$\lambda P' \lambda P \lambda i. \neg \exists x (P'xi \wedge Pxi)$			
the	$\rightsquigarrow$	$\lambda P' \lambda P \lambda i. \exists x (\forall y (P'yi \leftrightarrow x = y) \wedge Pxi)$			

Complex Expressions

- Do we need translation rules and type shifting rules other than the ones we already have?
- No, these were formulated in a way that was independent from the choice of σ.
- In fact, leaving translation and type shifting rules as they are, we now have new translations for complex expressions that involve possible worlds.
- Translations have changed in a completely uniform way.
- Many of our previous considerations, based on the choice $\sigma = t$, are still valid given the new choice.
- The following slide gives an example of one of the new translations that we get. It differs only marginally from one of our previous examples.

Bill doesn't like every girl

- [doesn't [[every girl]^1[Bill [like t_1]]]]
- $t_1 \rightsquigarrow v_1$
  like $\rightsquigarrow \lambda y \lambda x.like_{e(e(st))}\, xy$
  [like t_1] $\rightsquigarrow \lambda x.like\, x\, v_1$
  [Bill [like t_1]] $\rightsquigarrow like\ bill\ v_1$
  [every girl] $\rightsquigarrow \lambda P \lambda i. \forall x(girl\, xi \rightarrow Pxi)$
  [[every girl]^1[Bill [like t_1]]] $\rightsquigarrow$
  $(\lambda P \lambda i. \forall x(girl\, xi \rightarrow Pxi))(\lambda v_1.like\ bill\ v_1)$
  [[every girl]^1[Bill [like t_1]]] $\rightsquigarrow \lambda i. \forall x(girl\, xi \rightarrow like\ bill\ xi)$
  [doesn't[[every girl]^1[Bill [like t_1]]]] $\rightsquigarrow$
  $\lambda i. \neg \forall x(girl\, xi \rightarrow like\ bill\ xi)$

The actual index

- The preceding translation procedure translates sentences into terms of type st.
- Truth values may be obtained by assuming that **i** is a constant which stands for the actual index (or the actual world if indices are identified with worlds) and applying sentence translations to **i**.
- [doesn't[[every girl]^1[Bill [like t_1]]]] $\rightsquigarrow$
$$\lambda i.\neg\forall x(\mathit{girl}\, xi \rightarrow \mathit{like}\, \mathit{bill}\, xi)$$
- Applying to **i** gives $\neg\forall x(\mathit{girl}\, x\mathbf{i} \rightarrow \mathit{like}\, \mathit{bill}\, x\mathbf{i})$

Modal Operators

- By resetting σ to its new value *st* and choosing a new set of basic translations we have 'lifted' translations uniformly to a higher type.
- Up till now this has made no difference for the entailment relation. If the translation of X entailed the translation of Y previously, the new translation of X applied to **i** will entail the new translation of Y applied to **i** now, and vice versa.
- But the new type *st* for sentences allows for the introduction of new operators. In particular, *st* invites the introduction of modal operators.

Modalities

- Modal auxiliaries like may, must, can, can't can be treated essentially as modal operators.
- $$\begin{aligned} \text{may} &\rightsquigarrow \lambda p \lambda i. \exists j (acc^1\, ij \wedge pj) \\ \text{must} &\rightsquigarrow \lambda p \lambda i. \forall j (acc^1\, ij \rightarrow pj) \\ \text{can} &\rightsquigarrow \lambda p \lambda i. \exists j (acc^2\, ij \wedge pj) \\ \text{can't} &\rightsquigarrow \lambda p \lambda i. \neg \exists j (acc^2\, ij \wedge pj) \end{aligned}$$
- Here acc^1 and acc^2 are accessibility relations of type $s(st)$.
- While the logic we are working in is still classical type theory, we have borrowed an important idea from modal logic, as the translations here essentially transcribe the usual Kripke semantics of the modal operators.
- Suitable meaning postulates for acc^1 and acc^2 (e.g. transitivity, euclideanity) will give the desired logic.

Some girl must dance

- [must[[some girl]dance]]
- [[some girl]dance] $\rightsquigarrow \lambda i.\exists x(girl\, xi \wedge dance\, xi)$
  must $\rightsquigarrow \lambda p\lambda i.\forall j(acc^1\, ij \rightarrow pj)$
  [must[[some girl]dance]] $\rightsquigarrow$
  $\lambda i.\forall j(acc^1\, ij \rightarrow \exists x(girl\, xj \wedge dance\, xj))$
- $[[\text{some girl}]^1[\text{must}[\text{t}_1\ \text{dance}]]$
- $[\text{t}_1\ \text{dance}] \rightsquigarrow dance\, v_1$
  must $\rightsquigarrow \lambda p\lambda i.\forall j(acc^1\, ij \rightarrow pj)$
  $[\text{must}[\text{t}_1\ \text{dance}]] \rightsquigarrow \lambda i.\forall j(acc^1\, ij \rightarrow dance\, v_1 j)$
  $[[\text{some girl}]^1[\text{must}[\text{t}_1\ \text{dance}]]] \rightsquigarrow$
  $\lambda i.\exists x(girl\, xi \wedge \forall j(acc^1\, ij \rightarrow dance\, xj))$

Multiple Interpretations of Modals

Modals have at least three different uses: circumstantial, epistemic, and deontic (Kratzer 1977).

- Circumstantial

  The ball must fall

  ≈ In all worlds compatible with the law of gravity, the ball falls (will fall)

- Epistemic

  You must be joking

  ≈ In all worlds compatible with what I believe to be the case, you are joking

- Deontic

  You must obey your dad

  ≈ In all worlds compatible with what I take to be morally acceptable, you obey your dad

Accessibility Contextualized

- It seems then that the accessibility relation acc^1 should at least be split into three variants: a circumstantial, an epistemic, and a deontic one. Something similar holds for acc^2.
- But that is not enough: (Kratzer 1977) shows that the set of worlds that are considered to be accessible can vary with each occasion of use.
- The solution is to make accessibility relations related to modals such as acc^1 and acc^2 context-dependent.
- One way to do this is to go the way we have gone with pronouns before: assume that acc^1 and acc^2 are variables rather than constants of type $s(st)$.
- On any occasion of use a suitable accessibility relation must be chosen.

Hintikka's Theory of Belief

- (Hintikka 1962) gives an attractive modal theory of epistemic operators. For example he equates belief with truth in all possible worlds which are doxastic alternatives for a given person in a given world.
- x believes p if p is true in all of x's doxastic alternatives.
- believe $\rightsquigarrow \lambda p \lambda x \lambda i. \forall j(Bxij \rightarrow pj)$
- Here $B_{e(s(st))}$ is the doxastic alternative relation. $Bxij$ stands for: j is a doxastic alternative for x in world i.
- The translation is not really different from that of must or can't, except that the accessibility relation is now relativized to persons.
- Again, suitable meaning postulates for B will give it the desired logic.

Ken believes Bill dates a woman

- [Ken[believes[Bill[dates[a woman]]]]]
- [Bill[dates[a woman]]] $\rightsquigarrow \lambda i.\exists x(woman\, xi \wedge date\, bill\, xi)$
  believe $\rightsquigarrow \lambda p\lambda x\lambda i.\forall j(Bxij \rightarrow pj)$
  [believes[Bill[dates[a woman]]]] $\rightsquigarrow$
  $\lambda x\lambda i.\forall j(Bxij \rightarrow \exists x(woman\, xj \wedge date\, bill\, xj))$
  [Ken[believes[Bill[dates[a woman]]]]] $\rightsquigarrow$
  $\lambda i.\forall j(B\ ken\, ij \rightarrow \exists x(woman\, xj \wedge date\, bill\, xj))$
- [[a woman]^2[Ken[believes[Bill[dates t_2]]]]] $\rightsquigarrow$
  $\lambda i.\exists x(woman\, xi \wedge \forall j(B\ ken\, ij \rightarrow date\, bill\, xj))$
- Two natural readings. But can QR cross clause boundaries?

Subject Controlled Verbs

- Subject controlled verbs like try and wish also typically combine with CP complements.
- Syntacticians assume that these CP complements contain a silent pronominal element in subject position. This element is denoted with PRO.
- [John[tries[$_{\text{CP}}$ $\varnothing$[$_{\text{IP}}$ to[$_{\text{VP}}$ PRO skate]]]]]
- We will assume that $\varnothing$, to and PRO are semantically neutral and all translate to $\lambda P.P$.
- We also stipulate try $\rightsquigarrow \lambda P \lambda x.try_{e((st)(st))}\ x(Px)$.
- Note that x appears twice in this translation. It corresponds to the subject of the matrix clause but also to the subject of the complement clause.

John tries to find a unicorn

- [John[tries[∅[to[PRO[find[a unicorn]]]]]]]
- [find[a unicorn]] $\rightsquigarrow \lambda y \lambda i.\exists x(\mathit{unicorn}\ xi \wedge \mathit{find}\ yxi)$
  [∅[to[PRO[find[a unicorn]]]]] $\rightsquigarrow$
  $\lambda y \lambda i.\exists x(\mathit{unicorn}\ xi \wedge \mathit{find}\ yxi)$
  try $\rightsquigarrow \lambda P \lambda x.\mathit{try}\ x(Px)$
  [tries[∅[to[PRO[find[a unicorn]]]]]] $\rightsquigarrow$
  $\lambda y.\mathit{try}\ y(\lambda i.\exists x(\mathit{unicorn}\ xi \wedge \mathit{find}\ yxi))$
  [John[tries[∅[to[PRO[find[a unicorn]]]]]]] $\rightsquigarrow$
  $\mathit{try}\ \mathit{john}\ (\lambda i.\exists x(\mathit{unicorn}\ xi \wedge \mathit{find}\ \mathit{john}\ xi))$
- Apply to actual index:
  $\mathit{try}\ \mathit{john}\ (\lambda i.\exists x(\mathit{unicorn}\ xi \wedge \mathit{find}\ \mathit{john}\ xi))\ \mathbf{i}$

Seek and try to find

- seek a unicorn $\approx$ try to find a unicorn
- seek $\rightsquigarrow \lambda Q \lambda x.try\, x(Q(\lambda y.find\, xy))$
- Note that this translation of seek has type $((e(st))(st))(e(st))$ (or $((e\sigma)\sigma)(e\sigma)$). This is the type that one obtains from Argument Raising the translation of a transitive verb. This type is also closely related to the type that Montague gave to the (intensions of) all translations of transitive verbs.

John seeks a unicorn 1

- [John[seeks[a unicorn]]]
- [a unicorn] $\rightsquigarrow \lambda P \lambda i.\exists x(\mathit{unicorn}\, xi \wedge Pxi)$
  seek $\rightsquigarrow \lambda Q \lambda x.\mathit{try}\, x(Q(\lambda y.\mathit{find}\, xy))$
  [seek[a unicorn]] $\rightsquigarrow \lambda y.\mathit{try}\, y(\lambda i.\exists x(\mathit{unicorn}\, xi \wedge \mathit{find}\, yxi))$
  [John[seeks[a unicorn]]] $\rightsquigarrow$
  $$\mathit{try}\, \mathit{john}\, (\lambda i.\exists x(\mathit{unicorn}\, xi \wedge \mathit{find}\, \mathit{john}\, xi))$$
- No existence of unicorns implied!

John seeks a unicorn 2

- [[a unicorn]^3[John[seeks t$_3$]]]
- t$_3 \rightsquigarrow v_3$
  t$_3 \rightsquigarrow \lambda P.Pv_3$
  seek $\rightsquigarrow \lambda Q\lambda x.try\ x(Q(\lambda y.find\ xy))$
  [seek t$_3$] $\rightsquigarrow \lambda x.try\ x(find\ xv_3)$
  [John[seeks t$_3$]] $\rightsquigarrow try\ john\ (find\ john\ v_3)$
  [a unicorn] $\rightsquigarrow \lambda P\lambda i.\exists x(unicorn\ xi \wedge Pxi)$
  [[a unicorn]^3[John[seeks t$_3$]]] $\rightsquigarrow$
  $\lambda i.\exists x(unicorn\ xi \wedge try\ john\ (find\ john\ x)\ i)$
- Here John is seeking a specific unicorn.

Introducing Times

- We have shown how modal auxiliaries can be treated.
- How about tense?
- We will show how past, present and future can in principle be dealt with.
- But the treatment will be very approximative. All aspects of aspect, for example, will be ignored.
- The basic idea is that we will now start to treat indices as world-time pairs, so that we can quantify over their time component.

World-Time Pairs

- We want indices to behave like world-time pairs $\langle w, t \rangle$.
- One way to go: shift to a type logic in which for each two types α and β there is a type $\alpha \times \beta$ with associated domain the cartesian product $D_\alpha \times D_\beta$.
- We will opt for the poor man's solution here and axiomatize pairing and projection.
- Introduce new types ω (for worlds) and τ (for times).
- And constants $wd_{s\omega}$ and $tm_{s\tau}$. These will be our projection functions.
- Think of $wd\, i$ as the world component of index i and of $tm\, i$ as its time component.

Plenitude and Ordering

- We need a principle of plenitude; all world-time pairs should be represented: $\forall w_\omega \forall t_\tau \exists i(wd\, i = w \wedge tm\, i = t)$.
- And perhaps we also need $\forall ij((wd\, i = wd\, j \wedge tm\, i = tm\, j) \rightarrow i = j)$
- This basically identifies indices with world-time pairs.
- We now come to the structure of the temporal domain D_τ.
- A lot can be said about this structure. For a thoroughgoing study, see (van Benthem 1983).
- Here we will simply assume that D_τ is linearly ordered by a relation $<$ of type $\tau(\tau t)$.

Past and Future

- A relation $<^{\mathrm{tm}}$ of type $s(st)$ can be defined as $\lambda i \lambda j(wd\, i = wd\, j \wedge tm\, i < tm\, j)$.
- (Relations such as $<$ and $<^{\mathrm{tm}}$ will be written in so-called infix notation, i.e. we write $A < B$ instead of the official $< AB$).
- $i <^{\mathrm{tm}} j$ holds if i and j have the same world component but the time component of i precedes that of j.
- PAST $\rightsquigarrow \lambda p \lambda i.\exists j(j <^{\mathrm{tm}} i \wedge pj)$
  will $\rightsquigarrow \lambda p \lambda i.\exists j(i <^{\mathrm{tm}} j \wedge pj)$
- Here will is taken to be purely temporal, which is an approximation at best.

Some girl will dance

- [will[[some girl]dance]]
- [[some girl]dance] $\rightsquigarrow \lambda i.\exists x(girl\, xi \wedge dance\, xi)$
  will $\rightsquigarrow \lambda p\lambda i.\exists j(i <^{\mathrm{tm}} j \wedge pj)$
  [will[[some girl]dance]] $\rightsquigarrow$
  $\lambda i.\exists j(i <^{\mathrm{tm}} j \wedge \exists x(girl\, xj \wedge dance\, xj))$
- $[[\text{some girl}]^1[\text{will}[\mathrm{t}_1\ \text{dance}]]$
- $[\mathrm{t}_1\ \text{dance}] \rightsquigarrow dance\, v_1$
  will $\rightsquigarrow \lambda p\lambda i.\exists j(i <^{\mathrm{tm}} j \wedge pj)$
  $[\text{will}[\mathrm{t}_1\ \text{dance}]] \rightsquigarrow \lambda i.\exists j(i <^{\mathrm{tm}} j \wedge dance\, v_1 j)$
  $[[\text{some girl}]^1[\text{will}[\mathrm{t}_1\ \text{dance}]]] \rightsquigarrow$
  $\lambda i.\exists x(girl\, xi \wedge \exists j(i <^{\mathrm{tm}} j \wedge dance\, xj))$

Taking Stock

- We started with a fragment of English in which sentences got truth values as their denotations.
- We now have seen how this translation can be lifted to a translation in higher types by choosing a higher type for sentence type σ and adapting lexical translations.
- The choice was $\sigma = st$ and this immediately led to new operators, modal ones in this case, with the help of which a better approximation to the semantics of many expressions was obtained.
- But in fact this 'lifting' is a general strategy that can be used over and over again. We will see another example shortly.

Lifting and Axiomatization

- Another strategy for obtaining more structure and, hopefully, a better fit with the data is axiomatization.
- We have, for example, assumed that accessibility relations such as acc^1, acc^2 and the doxastic accessibility relation B obey the 'right' axioms. Some discussion is possible about which axioms are 'right'.
- We have also assumed that the temporal relation $<$ obeys the axioms for strict linear order and have imposed a 'principle of plenitude' on indices.
- In fact, there is a trade-off between lifting and axiomatization: indices were effectively turned into world-time pairs by means of axiomatization, but in fact we could also have chosen the value $\omega(\tau t)$ for σ.

Hyperintensionality

- It is often held that the logical approach to semantic meaning suffers from grave foundational difficulties.
- The problem is that there are many expressions that are logically equivalent and yet differ in meaning.
- This problem of hyperintensionality has provoked an extensive philosophical and logical literature.
- In 1980, Richmond Thomason came with a solution that was ridiculously simple, but worked well (Thomason 1980).
- In this section I will explain the problem and will give a variant of Thomason's solution.
- Thomason worked with a variant of classical type logic that he called 'Intentional Logic'. We will just use the classical theory and streamline Thomason's theory a bit. The essence, though, of his theory will remain intact.

Granularity

- The possible worlds account of sentence meaning is much more finegrained than a simple truth values only account.
- If Fido is a dog and Fritz is a cat then a truth values only account cannot distinguish between Fido is a dog and Fritz is a cat because both get true as their semantic value.
- But in a possible worlds account these sentences are easily distinguished.
- The following silly argument is easily shown to be invalid in the possible worlds approach.
- Fido is a dog. Fritz is a cat. Mary believes that Fido is a dog. Therefore, Mary believes that Fritz is a cat.
- In a truth values only approach it must be denied that believe expresses a relation between the semantic value of its subject and that of its complement.

Unwanted Consequences

- While the possible worlds approach is good at blocking some unwanted inferences (for what seem the right reasons), it is not so good at blocking others.
- We will first review some unwanted inferences that we get at present, but that can in fact be blocked within a possible worlds approach, sometimes at the cost of giving up certain assumptions that may be dear to some.
- Among the problems that have a solution within the theory are the problem that belief is closed under entailment and the problem of substitution of coreferring names.
- We will discuss how such inferences can be gotten rid of. Then, we will turn to closure of belief under logical equivalence. This is the really nasty problem for which we need Thomason's move.

Closure under Entailment

- Consider the following argument:
- Mary believes that John walks if Sue talks
  Mary believes that John doesn't walk
  Therefore, Mary believes that Sue doesn't talk
- $\forall j(B\ mary\ \mathbf{i}j \rightarrow (talk\ sue\ j \rightarrow walk\ john\ j))$
  $\forall j(B\ mary\ \mathbf{i}j \rightarrow \neg walk\ john\ j)$
  $\models \forall j(B\ mary\ \mathbf{i}j \rightarrow \neg talk\ sue\ j)$
- But people are notoriously bad at doing contrapositions and Mary may fail to have drawn the Sue doesn't talk conclusion.
- So while the argument is predicted to be valid, it is in fact invalid.

Blocking Closure under Entailment

- But the validity of the argument really rested on our adoption of Hintikka's theory of belief (and propositional attitudes in general).
- Suppose we translate the word believe as follows: believe $\rightsquigarrow$ $believe_{(st)(e(st))}$, then our argument would translate as follows
- $believe\ (\lambda j.(talk\ sue\ j \rightarrow walk\ john\ j))\ mary\ \mathbf{i}$
  $believe\ (\lambda j.\neg walk\ john\ j)\ mary\ \mathbf{i}$
  $\not\models believe\ (\lambda j.\neg talk\ sue\ j)\ mary\ \mathbf{i}$
- The unwanted inference is blocked now. Propositional attitudes are no longer closed under entailment.

Explicit and Implicit Belief

- A distinction can be made between explicit and implicit belief (knowledge, etc.).
- Explicit belief is just belief (or, roughly, what you are disposed to assent to). Implicit belief is what you rationally should believe, given your explicit beliefs.
- Viewed in this light, Hintikka's theory of belief really is a theory of implicit belief:
  $\text{believe}_{imp} \rightsquigarrow \lambda p \lambda x \lambda i. \forall j(Bxij \rightarrow pj)$
  $\text{believe} \rightsquigarrow believe_{(st)(e(st))}$
- It seems reasonable to explain the doxastic alternative relation in terms of the set of all worlds compatible with someone's explicit beliefs:
  $\forall i \forall x \forall j(Bxij \leftrightarrow \forall p(believe\, pxi \rightarrow pj))$

Cicero and Tully

- Another problem of the theory sketched thus far is that it predicts that coreferential names can be substituted for one another in any context without change in truth value.
- Cicero is Tully
  Mary believes that Cicero is an orator
  Therefore, Mary believes that Tully is an orator
- $cicero = tully$
  $believe\ (orator\ cicero)\ mary\ \mathbf{i}$
  $\models believe\ (orator\ tully)\ mary\ \mathbf{i}$
- But while Cicero is Tully, Mary may not be aware of that fact and may ascribe one property to Cicero and another to Tully. Again a mismatch between predicted entailment and real entailment.

The Description Theory of Names

- Our decision to translate names such as Cicero and Tully with constants of type e is an implementation of the so-called rigid designator theory of names (Kripke 1972), which holds that the denotation of a name is independent from possible worlds.
- This theory is now widely adopted. But until the 1970s another theory was almost universally accepted. This was the description theory of names (Frege, Russell, Quine) which essentially holds that a name like Aristotle is in fact a description like the Aristotle, in which Aristotle is a predicate.
- We take Quine's view here and take it that predicates like Aristotle or is Aristotle need not be analysed (Quine 1953, Essay I).

The Description Theory of Names Implemented

- A direct implementation of the description theory of names would be the following paradigm:
  Bill $\rightsquigarrow \lambda P \lambda i. \exists x(\forall y(bill_{e(st)}\ yi \leftrightarrow x = y) \wedge Pxi)$
- However, in order to make life easier it may be better to divide the work over a translation
  Bill $\rightsquigarrow \lambda P \lambda i. \exists x(bill_{e(st)}\ xi \wedge Pxi)$
  and a meaning postulate
  $\forall i \exists x \forall y(bill\ yi \rightarrow x = y)$.
  Similar translations and postulates may be adopted for other names.
- Exercise: Show that the unwanted Cicero is Tully argument no longer comes out valid in this approach.

Closure under Co-Entailment

- Consider the following argument:
- Mary believes that if some woman walks no man talks
  $\not\models$ Mary believes that if some man talks no woman walks
- $$\begin{aligned} &\textit{believe}\ (\lambda j.(\exists x(\textit{woman}\ x j \wedge \textit{walk}\ x j) \rightarrow \\ &\qquad\qquad \neg\exists x(\textit{man}\ x j \wedge \textit{talk}\ x j)))\ \textit{mary}\ \mathbf{i} \\ &\models \textit{believe}\ (\lambda j.(\exists x(\textit{man}\ x j \wedge \textit{talk}\ x j) \rightarrow \\ &\qquad\qquad \neg\exists x(\textit{woman}\ x j \wedge \textit{walk}\ x j)))\ \textit{mary}\ \mathbf{i} \end{aligned}$$
- The two $\lambda j.(\cdots)$ terms denote the same set of worlds and they should, as the embedded sentences co-entail.
- But that means that Mary cannot stand in the believe relation to one of these sets of worlds without standing in that relation to the other and again the argument is incorrectly predicted to be valid.

Closure under Co-Entailment and Extensionality

- Remember the axiom of Extensionality:
  $\forall XY(\forall\vec{x}(X\vec{x} \leftrightarrow Y\vec{x}) \rightarrow \forall Z(ZX \rightarrow ZY))$
- This axiom is directly related to the problem of closure under co-entailment.
- Take $\lambda p.believe\ p\ mary\ \mathbf{i}$ for Z, j for $\vec{x}$, $\lambda j.\exists x(woman\ xj \wedge walk\ xj) \rightarrow \neg\exists x(man\ xj \wedge talk\ xj)$ for X, and $\lambda j.\exists x(man\ xj \wedge talk\ xj) \rightarrow \neg\exists x(woman\ xj \wedge walk\ xj)$ for Y.
- There are models for type logic that do not validate Extensionality (Fitting 2002; Benzmüller, Brown, and Kohlhase 2004; Muskens 2005a).
- But Thomason's solution of the problem can be carried out in a type theory with Extensionality.

Thomason's Move

- (Thomason 1980) introduces a basic type p that is the type of propositions.
- This means that propositions are treated as primitive objects.
- Sentences are translated as terms of type p.
- Meaning postulates correlate propositions with the values they determine. In the present case these values will be sets of possible worlds.
- But while propositions determine sets of worlds, one set of worlds may be determined by different propositions.
- Therefore one may bear some relation to one proposition, while not bearing that relation to another proposition determining the same set of worlds.

An Implementation of Thomason's Move

- We set σ to p.
- In the next few slides we will give new basic translations. The terms that are obtained from any Logical Form will look much like that Logical Form itself.
- A series of meaning postulates will connect propositions with sets of possible worlds.
- Finally, we make sure that the notion of entailment is connected with the possible worlds level as well.

Basic Translations

doesn't	$\rightsquigarrow \mathsf{not}_{pp}$	PRES	$\rightsquigarrow \lambda p.p$
run	$\rightsquigarrow \mathsf{run}_{ep}$	man	$\rightsquigarrow \mathsf{man}_{ep}$
find	$\rightsquigarrow \lambda y \lambda x.\mathsf{find}_{e(ep)}\, xy$	be	$\rightsquigarrow \lambda y \lambda x.\mathsf{is}_{e(ep)}\, xy$
Bill	$\rightsquigarrow \mathsf{bill}_{(ep)p}$	small	$\rightsquigarrow \mathsf{small}_{(ep)(ep)}$
fake	$\rightsquigarrow \mathsf{fake}_{(ep)(ep)}$	if	$\rightsquigarrow \mathsf{if}_{p(pp)}$
t_n	$\rightsquigarrow v_n$, for each n	may	$\rightsquigarrow \mathsf{may}_{pp}$
he_n	$\rightsquigarrow v_n$, for each n	must	$\rightsquigarrow \mathsf{must}_{pp}$
every	$\rightsquigarrow \mathsf{every}_{(ep)((ep)p)}$	believe	$\rightsquigarrow \mathsf{believe}_{p(ep)}$
some, a	$\rightsquigarrow \mathsf{a}_{(ep)((ep)p)}$	that	$\rightsquigarrow \lambda p.p$
no	$\rightsquigarrow \mathsf{no}_{(ep)((ep)p)}$		
the	$\rightsquigarrow \mathsf{the}_{(ep)((ep)p)}$		

who $\rightsquigarrow \lambda P' \lambda P \lambda x.\mathsf{and}_{p(pp)}(P'x)(Px)$

blue $\rightsquigarrow \lambda P \lambda x.\mathsf{and}_{p(pp)}(\mathsf{blue}_{ep}\, x)(Px)$

and $\rightsquigarrow \lambda R' \lambda R \lambda \vec{X}.\mathsf{and}_{p(pp)}(R\vec{X})(R'\vec{X})$

or $\rightsquigarrow \lambda R' \lambda R \lambda \vec{X}.\mathsf{or}_{p(pp)}(R\vec{X})(R'\vec{X})$

Translations

- [[if[PRES[[some woman]walk]]][PRES[[no man]talk]]] $\rightsquigarrow$ ((if((some woman)walk))((no man)talk))
- Mary believes that if some woman walks no man talks $\rightsquigarrow$ (mary(believe((if((some woman)walk))((no man)talk))))
- In these translations almost nothing happens. The real work must be done by the meaning postulates.
- All items in sans serif are non-logical constants. So, for the moment, there is no logic at all operative on the type p terms exemplified above.
- It can consistently be assumed that two type p terms that are not syntactically identical denote differrent propositions.

Meaning Postulates 1

- In order to connect our type p terms to a level at which the logic is operative we must give some meaning postulates.
- In the following, many of the constants in *italic* are those we have already met in the previous, possible worlds, translation.
- In fact, the net effect of our new translation plus the meaning postulates will be very similar to that translation.
- One crucial difference: the behaviour of propositional attitude verbs like believe.
- The meaning postulates axiomatize the behaviour of a function r of type $p(st)$.

Meaning Postulates 2

$\forall p(r(\mathsf{not}\ p) = \lambda i.\neg rpi)$
$\forall pp'(r(\mathsf{and}\ pp') = \lambda i.rpi \wedge rp'i)$
$\forall pp'(r(\mathsf{or}\ pp') = \lambda i.rpi \vee rp'i)$
$\forall pp'(r(\mathsf{if}\ pp') = \lambda i.rpi \rightarrow rp'i)$
$\forall P(r(\mathsf{bill}\ P) = r(Pbill))$
$\forall xy(r(\mathsf{is}\ xy) = \lambda i.(x = y))$
$\forall xy(r(\mathsf{find}\ xy) = find\ xy)$
$\forall x(r(\mathsf{man}\ x) = man\ x)$
$\forall x(r(\mathsf{run}\ x) = run\ x)$
$\forall x(r(\mathsf{blue}\ x) = blue\ x)$
$\forall p \forall x(r(\mathsf{believe}\ px) = believe_{p(e(st))}\ px)$
$\forall P \forall x(r(\mathsf{fake}\ Px) = fake(\lambda y.r(Py))x)$
$\forall P \forall x(r(\mathsf{small}\ Px) = small(\lambda y.r(Py))x)$
$\forall PP'(r(\mathsf{every}\ P'P) = \lambda i.\forall x(r(P'x)i \rightarrow r(Px)i))$
$\forall PP'(r(\mathsf{a}\ P'P) = \lambda i.\exists x(r(P'x)i \wedge r(Px)i))$
$\forall PP'(r(\mathsf{no}\ P'P) = \lambda i.\neg\exists x(r(P'x)i \wedge r(Px)i))$
$\forall PP'(r(\mathsf{the}\ P'P) = \lambda i.\exists x(\forall y(r(P'y)i \leftrightarrow x = y) \wedge r(Px)i))$
$\forall p(r(\mathsf{must}\ p) = \lambda i.\forall j(acc^1ij \rightarrow rpj))$
$\forall p(r(\mathsf{may}\ p) = \lambda i.\exists j(acc^1ij \wedge rpj))$

Two Derivations

$$
\begin{aligned}
r(((\text{if}((\text{a woman})\text{walk}))((\text{no man})\text{talk})) &= \\
\lambda i.r((\text{a woman})\text{walk})i \rightarrow r((\text{no man})\text{talk})i &= \\
\lambda i.\exists x(r(\text{woman } x)i \wedge r(\text{walk } x)i) \rightarrow r((\text{no man})\text{talk})i &= \\
\lambda i.\exists x(r(\text{woman } x)i \wedge r(\text{walk } x)i) \rightarrow \neg\exists x(r(\text{man } x)i \wedge r(\text{talk } x)i) &= \\
\lambda i.\exists x(\mathit{woman}\ xi \wedge \mathit{walk}\ xi) \rightarrow \neg\exists x(\mathit{man}\ xi \wedge \mathit{talk}\ xi)
\end{aligned}
$$

$$
\begin{aligned}
r(\text{mary}(\text{believe}((\text{if}((\text{some woman})\text{walk}))((\text{no man})\text{talk})))) &= \\
r(\text{believe}((\text{if}((\text{some woman})\text{walk}))((\text{no man})\text{talk})))\mathit{mary} &= \\
\mathit{believe}((\text{if}((\text{some woman})\text{walk}))((\text{no man})\text{talk}))\mathit{mary}
\end{aligned}
$$

Entailment

- We can define a derived notion of entailment by stipulating that type p terms $A_1, \ldots A_n$ entail a type p term B if

$$\mathrm{MP}, rA_1\mathbf{i}, \ldots, rA_n\mathbf{i} \models_g rB\mathbf{i} \ ,$$

  where MP is the set of meaning postulates and $\mathbf{i}$ is the special constant intended to denote the actual world.
- ((if((a woman)walk))((no man)talk)) entails ((if((a man)talk))((no woman)walk)) and vice versa.
- But these terms need not be coreferential, therefore
- (mary(believe((if((a woman)walk))((no man)talk)))) does not entail (mary(believe((if((a man)talk))((no woman)walk))))

Implicit Belief Again

- We can retain our analysis of implicit belief $\text{believe}_{imp} \rightsquigarrow \lambda p \lambda x \lambda i. \forall j(Bxij \rightarrow pj)$,
- while the doxastic alternative relation is still explained in terms of the set of all worlds compatible with someone's explicit beliefs. However, this should now be formalized as $\forall i \forall x \forall j(Bxij \leftrightarrow \forall p(believe\, pxi \rightarrow rpj))$.

What has been Achieved?

- Thomason's Move enabled us to get maximally finegrained distinctions between propositions: no two terms of type p need be assumed to corefer, unless they are syntactically identical (modulo λ-conversion).
- Linguistic expressions that do not contain verbs of propositional attitude and the like are associated with the same sets of worlds as they were associated with by our previous translation.
- But propositional attitudes are no longer closed under entailment, co-entailment, or the substitution of coreferential names.
- Isn't blocking all entailment in opaque contexts a bit too strong?
- And what are propositions?

What are Propositions?

- Thomason's theory treats propositions as primitive objects but one may well ask what kinds of things they really are.
- An answer to this ontological question may also give more grips on the identity criteria of propositions.
- There have been many different answers, e.g.:
  - Structured Meanings (Carnap 1947; Lewis 1972; Cresswell 1985)
  - Sets of possible and impossible worlds (Montague 1970b; Cresswell 1972; Hintikka 1975)
  - Algorithms (Moschovakis 1994)
- The beauty of Thomason's theory is that it abstracts from the ontological question but seems compatible with each of these answers. E.g. (Muskens 2005b) argues that a slightly revised version of Thomason's theory dovetails well with Moschovakis' Senses-as-Algorithms idea.

Introducing Dynamics 1

- In the following slides we will review some of the considerations that led to the development of Discourse Representation Theory (DRT, Kamp 1981; Heim 1982; Kamp and Reyle 1993).
- Time constraints will not allow us to do DRT the justice it deserves. For an excellent introduction to the theory consult (Kamp and Reyle 1993).
- Two kinds of phenomena in particular have led to the development of DRT: cross-sentential anaphora and so-called donkey sentences. We will consider these shortly.

Introducing Dynamics 2

- Today's DRT is concerned with a very wide range of phenomena and can be considered to work out a representationalist hypothesis about natural language semantics, claiming psycholinguistic adequacy of its representations.
- Many aspects of DRT are consistent with the approach followed in this course, however, and we will see how a reduced form of DRT can be included into the present theory.
- The strategy will be to first identify types in which the basic representations of DRT can reasonably be modeled and to then instantiate σ and ν as those types.
- Before we do that let's look at DRT and the way it works.

Cross-sentential Anaphora

- A man entered. He ordered a beer.
- An approach in which a man is treated as before and he is treated as a free variable will fail: the existential quantifier associated with a man will not bind the free variable.
- One approach is to treat the he of the second sentence as an E-type pronoun in the sense of (Evans 1980):
- A man entered. The man who entered ordered a beer.
- DRT follows another path. The basic idea is that a man, like other indefinites, sets up a discourse referent and that anaphoric pronouns can pick up discourse referents.

Discourse Representation

- Text is interpreted in the context of a Discourse Representation Structure (DRS).
- The interpretation of any sentence makes the DRS grow.
- A DRS consists of two parts: 1) a set of discourse referents, called the universe of the DRS, and 2) a set of conditions.
- A man entered. He ordered a beer.
- $$\boxed{\begin{array}{l} x \quad y \quad z \\ \hline man\ x \\ enter\ x \\ order\ yz \\ beer\ z \\ y = x \end{array}}$$

Incremental Interpretation

- Start with the empty DRS.
- $\boxed{}$ A man entered $\boxed{\begin{array}{l} x \\ \hline man\ x \\ enter\ x \end{array}}$ He ordered a beer $\boxed{\begin{array}{lll} x & y & z \\ \hline man\ x & & \\ enter\ x & & \\ order\ yz & & \\ beer\ z & & \\ y = x & & \end{array}}$
- Each sentence is interpreted in context but also adds to the context.
- Essential for interpretation is the DRT construction algorithm. It deconstructs syntactic trees of input sentences and constructs the growing DRS. We will say little about the construction algorithm but will replace it with a Montague-like translation procedure.

Truth Conditions

- While DRT offers an alternative to Montague Semantics, it is still very much within the logical, truth-conditional approach to semantics.
- A verifying embedding of a DRS is a function from its discourse referents to the domain of a first-order model, such that all its conditions come out true under that interpretation.
- A DRS is true if it has a verifying embedding.

Coreferentiality of a man and he

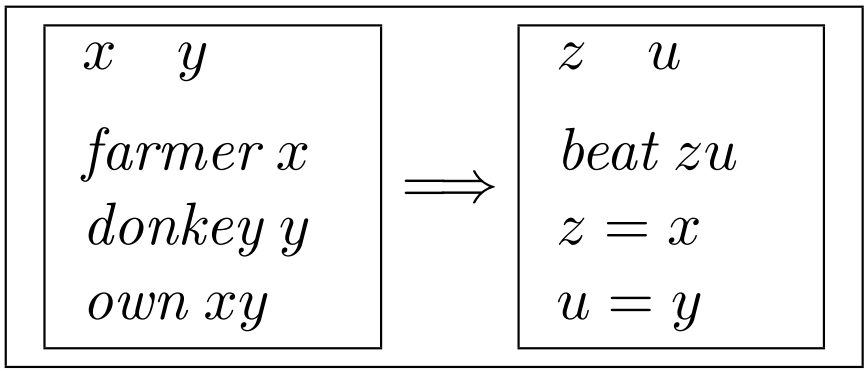
- What these truth conditions boil down to, is that $\boxed{\begin{array}{l} x \\ \hline man\, x \\ enter\, x \end{array}}$,

  for example, is true iff $\exists x(man\, x \wedge enter\, x)$ is true, while

  the DRS $\boxed{\begin{array}{lll} x & y & z \\ \hline man\, x \\ enter\, x \\ order\, yz \\ beer\, z \\ y = x \end{array}}$, is true iff

  $\exists xz(man\, x \wedge enter\, x \wedge order\, xz \wedge beer\, z)$ is.
- The coreferentiality of a man and he is captured now.

More Operators

- It is clear that the kind of simple DRSs we have seen thus far, with all conditions atomic sentences of predicate logic, are not expressive enough for treating reasonable parts of language.
- The core part of DRT adds negation, implication, and disjunction (DRT proper adds much more).
- If a farmer owns a donkey he beats it
- 

Syntax of Core DRT

- We move to a linear representation of DRSs and conditions, as this is easier to work with in some circumstances.
- The following sets up a simple DRS language. It is assumed that some first-order vocabulary (without function constants) is given.
  - Each atomic formula is a condition;
  - If K_1 and K_2 are DRSs, then **not** K_1, $K_1 \Rightarrow K_2$, and K_1 **or** K_2 are conditions;
  - If $\gamma_1, \ldots, \gamma_n$ are conditions and $x_1, \ldots, x_m$ are variables, then $[x_1 \ldots x_m \mid \gamma_1, \ldots, \gamma_n]$ is a DRS.
- $[\!| \, [xy \mid \mathit{farmer}\, x, \mathit{donkey}\, y, \mathit{own}\, xy] \Rightarrow [zu \mid \mathit{beat}\, zu, z = x, u = y]]$

Restrictions on Anaphora

- John has a cat. Mary feeds it.
  John doesn't have a cat. #Mary feeds it.
  A man I know has a cat. Mary feeds it.
  Every man I know has a cat. #Mary feeds it.
  Every man who has a cat feeds it.
  John has a cat and Mary feeds it.
  #John has a cat or Mary feeds it.
- There are systematic restrictions on the life spans of discourse referents (Karttunen 1976).
- DRT endeavors to capture these restrictions with the help of the notion of accessibility.

Accessibility

- A discourse referent x associated with a pronoun or other dependent element may be identified with a discourse referent y, previously declared in the universe of some DRS, if some conditions are met.
- One condition being that y must be accessible to x.
- Accessibility is a geometric notion. From a given position in a DRS some positions are accessible, others are not.
- Accessibility is a relation between occurrences of discourse referents and if y is accessible to x, y must occur in the universe of some DRS, while x must occur in some atomic condition.

An Accessibility Calculus

- Define $\mathbf{adr}([x_1 \ldots x_m \mid \gamma_1, \ldots, \gamma_n]) = \{x_1, \ldots, x_m\}$
- The following is a calculus that computes the set $\mathbf{acc}(x, \Phi)$ of referents accessible to x in any DRS or condition Φ. The calculus proceeds in a top-down way.

$$
\begin{aligned}
&\mathbf{acc}(x, \varphi) && = \varnothing, \text{ if } \varphi \text{ is atomic}\\
&\mathbf{acc}(x, \mathbf{not}\, K) && = \mathbf{acc}(x, K)\\
&\mathbf{acc}(x, K_1 \,\mathbf{or}\, K_2) && = \mathbf{acc}(x, K_1), \text{ if } x \text{ occurs in } K_1\\
& && = \mathbf{acc}(x, K_2), \text{ if } x \text{ occurs in } K_2\\
&\mathbf{acc}(x, K_1 \Rightarrow K_2) && = \mathbf{acc}(x, K_1), \text{ if } x \text{ occurs in } K_1\\
& && = \mathbf{acc}(x, K_2) \cup \mathbf{adr}(K_1),\\
& && \qquad \text{if } x \text{ occurs in } K_2\\
&\mathbf{acc}(x, [x_1 \ldots x_m \mid \gamma_1, \ldots, \gamma_n]) && = \mathbf{acc}(x, \gamma_i) \cup \{x_1, \ldots, x_m\},\\
& && \qquad \text{if } x \text{ occurs in } \gamma_i
\end{aligned}
$$

Accessibility Explains Restrictions on Anaphora

- The accessibility requirement now explains why some anaphoric relations cannot exist. For example:
- Every man owns a cat. #Mary feeds it.
- $[xu \mid x = mary, [y \mid man\ y] \Rightarrow [z \mid cat\ z, own\ yz],$
  $feed\ xu, u = ?]$
- The discourse referents accessible to the red occurrence of u in this DRS are x and u.
- x is out for various syntactic reasons. u should not be resolved as u. Crucially, z is not accessible.
- This explains why the discourse is out. The impossibility of other anaphora is explained in a similar fashion.

Donkey Sentences

- If a farmer owns a donkey he beats it
- $[\!| \, [xy \mid \mathit{farmer}\, x, \mathit{donkey}\, y, \mathit{own}\, xy] \Rightarrow [zu \mid \mathit{beat}\, zu, z = x, u = y]]$
- Every farmer who owns a donkey beats it
- $[\!| \, [xy \mid \mathit{farmer}\, x, \mathit{donkey}\, y, \mathit{own}\, xy] \Rightarrow [u \mid \mathit{beat}\, xu, u = y]]$
- A reasonable translation of these sentences into predicate logic is $\forall xy((\mathit{farmer}\, x \wedge \mathit{donkey}\, y \wedge \mathit{own}\, xy) \rightarrow \mathit{beat}\, xy)$
- This treats the indefinites a farmer and a man as universal quantifiers, not as existential ones!
- The DRT truth definition takes care of this.

From Discourse Representation to Logic

The function † defined below sends conditions and DRSs to predicate logic.

$$
\begin{array}{ll}
\gamma^\dagger & = \gamma, \text{ if } \gamma \text{ is atomic} \\
(\textbf{not}\ K)^\dagger & = \neg K^\dagger \\
(K_1\ \textbf{or}\ K_2)^\dagger & = K_1^\dagger \vee K_2^\dagger \\
([x_1 \ldots x_m \mid \gamma_1, \ldots, \gamma_n] \Rightarrow K)^\dagger & = \forall x_1 \ldots x_m((\gamma_1^\dagger \wedge \ldots \wedge \gamma_n^\dagger) \rightarrow K^\dagger) \\
[x_1 \ldots x_m \mid \gamma_1, \ldots, \gamma_n]^\dagger & = \exists x_1 \ldots x_m(\gamma_1^\dagger \wedge \ldots \wedge \gamma_n^\dagger)
\end{array}
$$

Example:

$$[|\ [xy \mid \mathit{farmer}\, x, \mathit{donkey}\, y, \mathit{own}\, xy] \Rightarrow [u \mid \mathit{beat}\, xu, u = y]]^\dagger =$$

$$\forall xy((\mathit{farmer}\, x \wedge \mathit{donkey}\, y \wedge \mathit{own}\, xy) \rightarrow \exists u(\mathit{beat}\, xu \wedge u = y))$$

$$\Longleftrightarrow$$

$$\forall xy((\mathit{farmer}\, x \wedge \mathit{donkey}\, y \wedge \mathit{own}\, xy) \rightarrow \mathit{beat}\, xy)$$

DRT: What Logical Animal?

- DRT can be translated into predicate logic, but seems inherently richer. The mechanism to create and pass on discourse referents across sentence boundaries has no correlate in predicate logic.
- Then what kind of logical beast is DRT? Since the 1980s logicians have given many different answers.
- An answer that is interesting in its own right is the idea that DRT is a kind of abstract programming language. This idea is implicit or explicit in (Barwise 1987; Rooth 1987; Staudacher 1987; Groenendijk and Stokhof 1990; Groenendijk and Stokhof 1991)
- If DRSs are programs, much of their value-passing behaviour falls into place.

DRT as a Programming Language

- The following correspondences seem enlightening:
- DRSs $\approx$ programs
  conditions $\approx$ Boolean conditions
  discourse referents $\approx$ variables or registers
- A program state is a function that assigns values to variables in a program. E.g. $\{x \mapsto 3, y \mapsto 7\}$.
- A program changes the program state:
  $\{x \mapsto 3, y \mapsto 7\}x := x + y\{x \mapsto 10, y \mapsto 7\}$
- One way to model the meaning of a program is as a binary relation between program states.
- This idea can be transposed to DRT.

The Semantics of DRT and that of Programming

- If DRT shows behaviour that is reminiscent of the behaviour of programming languages, then its semantics may perhaps take a form inspired by the semantics of programming languages.
- The latter have been studied extensively. Underlying some of our considerations is work done in Dynamic Logic (Pratt 1976; Harel 1984).
- In the following, which is based on (Muskens 1996), we will depend very much upon a dynamic semantics for DRT that was given by (Groenendijk and Stokhof 1991), as a side product of their Dynamic Predicate Logic.
- We will not present Groenendijk and Stokhof's semantics directly but will transcribe it, much in the way we have transcribed modal logic when dealing with modalities.

Registers and Program States in Type Logic 1

- Registers behave just like variables and program states then are just assignments (functions from variables to values).
- But variables are part of the syntax of our logic and assignments are objects that we use to talk about its interpretation.
- A direct identification of registers with variables and of states with assignments would blur the distinction between the object level and the metalevel of our logic and would land us into all kinds of formal difficulty.
- Instead of doing that we will axiomatize the behaviour of registers and states directly.
- Think of registers as memory locations; places where you can store some bits. These bits can later be consulted; can be updated; etc.

Registers and Program States in Type Logic 2

- Formally, registers will be objects of type ν and states will be objects of type s.
- There will be two kinds of constants of type ν. One kind will be denoted with u (with primes, subscripts, etc.). These will be called updatable referents. The second kind, for which we typically use *John, Mary, Bill* and the like, will be called non-updatable referents.
- It might seem strange to have constants that denote objects that are going to behave just like variables (the updatable referents), but if you think of registers as memory locations, much of this strangeness disappears.
- The updatable referents are memory locations that you can read and write to; the non-updatable ones are read-only.

Registers and Program States in Type Logic 3

- We will use a non-logical constant V of type $\nu(se)$. $V\delta i$ will denote the value of register δ in state i.
- While states are primitive objects $\lambda v.Vvi$ will assign a value to each register, for any state i.
- $i[u_1, \ldots, u_n]j$ will be short for
$$\forall v((v \neq u_1 \wedge \ldots \wedge v \neq u_n) \rightarrow Vvi = Vvj)$$
- 'i and j agree everywhere, except possibly in $u_1, \ldots, u_n$'

Axioms

1. Update Axiom Scheme:
   $\forall i \forall x \exists j(i[u]j \wedge Vuj = x)$, if u is an updatable referent.
2. $u_n \neq u_m$, if u_n and u_m are syntactically distinct.
3. $\forall i((V\ John_\nu\ i) = john_e)$,
   $\forall i((V\ Mary_\nu\ i) = mary_e)$, etc.

- 'updatable registers can be selectively updated with any value'
- 'two distinct updatable registers do not denote the same memory location (although they may have the same values in their memory locations)'
- 'the memory locations associated with non-updatable referents contain the expected value'

Selective Updating

	i	j
u_1:	bob	bob
u_2:	tim	tim
u_3:	tim	tim
u_4:	sue	ann
u_5:	pat	pat
	⋮	⋮
Tom:	tom	tom
Ann:	ann	ann
	⋮	⋮

Unselective Binding

- States effectively correspond to lists of values.
- Quantification over states can have the effect of multiple quantification over values.
- The following is easily seen to hold provided that j is not free in φ. ($\{x_1 := A_1, \ldots, x_n := A_n\}$ stands for simultaneous substitution of the A_i for x_i).
- $\forall i(\exists j(i[u_1, \ldots, u_n]j \wedge \varphi\{x_1 := (V u_1 j), \ldots, x_n := (V u_n j)\}) \leftrightarrow \exists x_1 \ldots x_n \varphi)$
- (Lewis 1975) argues that this kind of binding of many arguments in one fell swoop takes place in natural language and calls it unselective binding.

Conditions and DRSs as Abbreviations

- In the following we introduce DRT conditions and DRSs as abbreviations into our type logic. Conditions will have type st; DRSs type $s(st)$. γ ranges over conditions; K over DRSs.
- Write

$R\{u_1, \ldots, u_n\}$	for	$\lambda i.R(Vu_1i)\ldots(Vu_ni)$
δ_1 **is** δ_2	for	$\lambda i.(V\delta_1 i) = (V\delta_2 i)$
not K	for	$\lambda i.\neg\exists j\ Kij$
K **or** K'	for	$\lambda i.\exists j(Kij \vee K'ij)$
$K \Rightarrow K'$	for	$\lambda i.\forall j(Kij \rightarrow \exists k K'jk)$
$[u_1 \ldots u_m \mid \gamma_1, \ldots, \gamma_n]$	for	$\lambda i\lambda j.(i[u_1, \ldots, u_m]j \wedge \gamma_1 j \wedge \ldots \wedge \gamma_n j)$
$K; K'$	for	$\lambda i\lambda j.\exists k(Kik \wedge K'kj)$

Informal Meanings

- $[u_1 \ldots u_m \mid \gamma_1, \ldots, \gamma_n] \approx$
  indeterministically update $u_1, \ldots, u_m$ so that $\gamma_1, \ldots, \gamma_n$ will hold (if that is possible, otherwise fail).
- $K; K' \approx$ first do K, then K'.
- $R\{u_1, \ldots, u_n\} \approx$ test whether the values of $u_1, \ldots, u_n$ stand in the relation R, return if that is so, otherwise fail.
- δ_1 **is** $\delta_2 \approx$ test whether the value of δ_1 is identical to the value of δ_2.
- **not** $K \approx$ it is not possible to execute K.
- K **or** $K' \approx$ it is possible to execute K or K'.
- $K \Rightarrow K' \approx$ after any succesful execution of K it is possible to execute K'.

Sequencing and Conjunction

- Since the logical space associated with DRSs here consists of binary relations it is natural to consider relational composition.
- The semantics of $K; K'$ is the relational composition of that of K and that of K'.
- Relational composition is the semantics that is usually given to the sequencing of two programs (do P_1 then P_2).
- In a more realistic setting, where programs are deterministic and hence functional, relational composition reduces to functional composition.
- The operator ; will play the role of conjunction in the translation we will define.

Merging

Let $[u_1 \ldots u_m \mid \gamma_1, \ldots, \gamma_n]$ and $[u'_1 \ldots u'_\ell \mid \gamma'_1, \ldots, \gamma'_q]$ be DRSs, both built up using the abbreviations we have just defined and such that none of the $u'_1, \ldots, u'_\ell$ occurs in any of the $\gamma_1, \ldots, \gamma_n$. Then

$$[u_1 \ldots u_m \mid \gamma_1, \ldots, \gamma_n]; [u'_1 \ldots u'_\ell \mid \gamma'_1, \ldots, \gamma'_q]$$
$$=$$
$$[u_1 \ldots u_m u'_1 \ldots u'_\ell \mid \gamma_1, \ldots, \gamma_n, \gamma'_1, \ldots, \gamma'_q]$$

Combining DRT and Montague Semantics

- We now have imported discourse representations into our type logic. Conditions are certain terms of type st, DRSs are terms of type $s(st)$, and discourse referents are terms of type ν.
- It is also legitimate now to mix DRSs with lambda notation.
- Let us see how we can use this to import the core part of DRT into our Montague-like framework.
- According to our general strategy we must fix the values of the abstract types σ and ν. The type ν has already been identified with discourse referents. For σ we choose $s(st)$, the type of DRSs.
- As a next step we should give basic translations and then we are in business...

Basic Translations 1

doesn't	$\rightsquigarrow$	$\lambda p[\ \mid \mathbf{not}\ p]$
PRES	$\rightsquigarrow$	$\lambda p.p$
run	$\rightsquigarrow$	$\lambda v.[\ \mid run_{e(st)}\{v\}]$
man	$\rightsquigarrow$	$\lambda v.[\ \mid man_{e(st)}\{v\}]$
find	$\rightsquigarrow$	$\lambda v'\lambda v.[\ \mid \mathit{find}_{e(e(st))}\{v, v'\}]$
be	$\rightsquigarrow$	$\lambda v'\lambda v.[\ \mid v\ \mathbf{is}\ v']$
Bill	$\rightsquigarrow$	$Bill_\nu$
if	$\rightsquigarrow$	$\lambda p\lambda q.[\ \mid p \Rightarrow q]$
who	$\rightsquigarrow$	$\lambda P'\lambda P\lambda v.(Pv\ ;\ P'v)$
and	$\rightsquigarrow$	$\lambda R'\lambda R\lambda \vec{X}(R\vec{X}\ ;\ R'\vec{X})$
or	$\rightsquigarrow$	$\lambda R'\lambda R\lambda \vec{X}[\ \mid R\vec{X}\ \mathbf{or}\ R'\vec{X}]$
t_n	$\rightsquigarrow$	v_n, for each n
he	$\rightsquigarrow$	δ, for any discourse referent δ

Basic Translations 2

For each n that is fresh to the translation,

$$\begin{array}{ll} \text{some, a} & \rightsquigarrow \lambda P'\lambda P.([u_n \mid] \,;\, P'u_n \,;\, Pu_n) \\ \text{no} & \rightsquigarrow \lambda P'\lambda P.[\mid \mathbf{not}([u_n \mid] \,;\, P'u_n \,;\, Pu_n)] \\ \text{every} & \rightsquigarrow \lambda P'\lambda P.[\mid ([u_n \mid] \,;\, P'u_n) \Rightarrow Pu_n] \end{array}$$

- Each of these set up a discourse referent u_n.
- n may freely be chosen, provided it has not been used in a translation before.
- This is a bit of a stop-gap. A more correct treatment would let the translation do an increment on n (van Eijck), but we do not want to develop the necessary machinery here.

A farmer owns a donkey

- [PRES[[a farmer][own[a donkey]]]
- a $\rightsquigarrow \lambda P'\lambda P.([u_1 \mid]\ ;\ P'u_1\ ;\ Pu_1)$

  donkey $\rightsquigarrow \lambda v.[\ \mid donkey\{v\}]$

  [a donkey] $\rightsquigarrow \lambda P.([u_1 \mid]\ ;\ [\ \mid donkey\{u_1\}]\ ;\ Pu_1)$

  [a donkey] $\rightsquigarrow \lambda P.([u_1 \mid donkey\{u_1\}]\ ;\ Pu_1)$ (merge)

  [a donkey] $\rightsquigarrow \lambda R_{\nu(\nu\sigma)}\lambda v.([u_1 \mid donkey\{u_1\}]\ ;\ Ru_1v)$ (quantifier shifting)

  own $\rightsquigarrow \lambda v'\lambda v.[\ \mid own\{v, v'\}]$

  [own[a donkey]] $\rightsquigarrow \lambda v.([u_1 \mid donkey\{u_1\}]\ ;\ [\ \mid own\{v, u_1\}])$

  [own[a donkey]] $\rightsquigarrow \lambda v.([u_1 \mid donkey\{u_1\}, own\{v, u_1\}])$

  [a farmer] $\rightsquigarrow \lambda P.([u_2 \mid farmer\{u_2\}]\ ;\ Pu_2)$

  [a farmer[own[a donkey]]] $\rightsquigarrow$

  $[u_2 \mid farmer\{u_2\}]\ ;\ [u_1 \mid donkey\{u_1\}, own\{u_2, u_1\}]$

  $[u_1u_2 \mid farmer\{u_2\}, donkey\{u_1\}, own\{u_2, u_1\}]$

A farmer owns a donkey. He beats it.

- We will suppose that if S and S′ are CPs, the structure $[_{\text{CP}}\text{S}[\varnothing_{\text{CONJ}}\ \text{S}']]$ can be obtained and that $\varnothing_{\text{CONJ}} \rightsquigarrow \lambda p \lambda q.q\ ;\ p$
- $[_{\text{PRES}}[\text{a farmer}[\text{own}[\text{a donkey}]]]] \rightsquigarrow [u_1 u_2 \mid \mathit{farmer}\{u_2\}, \mathit{donkey}\{u_1\}, \mathit{own}\{u_2, u_1\}]$

  $\text{he} \rightsquigarrow u_2 \quad \text{it} \rightsquigarrow u_1$

  $\text{beat} \rightsquigarrow \lambda v' \lambda v.[\ \mid \mathit{beat}\{v, v'\}]$

  $[\text{he}[\text{beat it}]] \rightsquigarrow [\ \mid \mathit{beat}\{u_2, u_1\}]$

  $[_{\text{PRES}}[\text{he}[\text{beat it}]]] \rightsquigarrow [\ \mid \mathit{beat}\{u_2, u_1\}]$

  $[\varnothing_{\text{CONJ}}[_{\text{PRES}}[\text{he}[\text{beat it}]]]] \rightsquigarrow \lambda q.q\ ;\ [\ \mid \mathit{beat}\{u_2, u_1\}]$

  $[[_{\text{PRES}}[\text{a farmer}[\text{own}[\text{a donkey}]]]]\ [\varnothing[_{\text{PRES}}[\text{he}[\text{beat it}]]]]] \rightsquigarrow$

  $[u_1 u_2 \mid \mathit{farmer}\{u_2\}, \mathit{donkey}\{u_1\}, \mathit{own}\{u_2, u_1\}, \mathit{beat}\{u_2, u_1\}]$

Constraints on Anaphora

- When giving the translation of A farmer owns a donkey. He beats it. we translated he and it directly by picking the 'right' discourse referents, u_2 and u_1 respectively.
- Other choices (say u_5 for he) would have led to a translation in which the discourse referent remained 'free'. This is allright as we may take it that this corresponds to a deictic interpretation of he.
- Yet other choices (e.g. u_1 for he) would lead to readings that are impermissible on syntactic grounds.
- We may think about constraints on anaphora as being contributed by various levels of grammar.

A Processing Strategy

- Suppose that syntax has given us some constraints on how a certain anaphoric element can be interpreted, but that there is still a choice between possibilities that lead to deictic and bound readings.
- How do we efficiently get a bound reading?
- One possibility: translate anaphoric elements as variables of type ν initially and instantiate them only later, once a full DRS is obtained. Use accessibility to ensure binding.
- Example: Translate he as v and it as v' initially. Then A farmer owns a donkey. He beats it. $\rightsquigarrow$

  $$[u_1 u_2 \mid \mathit{farmer}\{u_2\}, \mathit{donkey}\{u_1\}, \mathit{own}\{u_2, u_1\}, \mathit{beat}\{v, v'\}]$$

- Accessibility will restrict the choice for v to u_1 and u_2. Feature constraints tell that u_1 is out and that v hence must be instantiated as u_2. Similar for v'.

Every farmer who owns a donkey beats it 1

- [every[farmer[who^2[$\varnothing$[PRES[t$_2$[own [a donkey]]]]]]][beat it]]
  [every[farmer[who^2[t$_2$[own [a donkey]]]]][beat it]]
- [own[a donkey]] $\rightsquigarrow \lambda v.([u_1 \mid donkey\{u_1\}, own\{v, u_1\}])$
  [t$_2$[own[a donkey]]] $\rightsquigarrow [u_1 \mid donkey\{u_1\}, own\{v_2, u_1\}]$
  who $\rightsquigarrow \lambda P' \lambda P \lambda v.(Pv \,;\, P'v)$
  [who^2[t$_2$[own[a donkey]]]] $\rightsquigarrow$
  $\lambda P \lambda v.(Pv \,;\, [u_1 \mid donkey\{u_1\}, own\{v, u_1\}])$
  farmer $\rightsquigarrow \lambda v.[\ \mid farmer\{v\}]$
  [farmer[who^2[t$_2$[own[a donkey]]]]] $\rightsquigarrow$
  $\lambda v.([\ \mid farmer\{v\}] \,;\, [u_1 \mid donkey\{u_1\}, own\{v, u_1\}])$

Every farmer who owns a donkey beats it 2

$[\text{farmer}[\text{who}^2[\text{t}_2[\text{own}[\text{a donkey}]]]]] \rightsquigarrow$
$\lambda v.([\ |\ farmer\{v\}]\ ;\ [u_1\ |\ donkey\{u_1\}, own\{v, u_1\}])$
$\text{every} \rightsquigarrow \lambda P'\lambda P.[\ |\ ([u_2\ |\ ]\ ;\ P'u_2) \Rightarrow Pu_2]$
$[\text{every}[\text{farmer}[\text{who}^2[\text{t}_2[\text{own}[\text{a donkey}]]]]]] \rightsquigarrow$
$\lambda P.[\ |\ ([u_2\ |\ ]\ ;\ [\ |\ farmer\{u_2\}]\ ;$
$[u_1\ |\ donkey\{u_1\}, own\{u_2, u_1\}]) \Rightarrow Pu_2]$
$[\text{every}[\text{farmer}[\text{who}^2[\text{t}_2[\text{own}[\text{a donkey}]]]]]] \rightsquigarrow$
$\lambda P.[\ |\ [u_1u_2\ |\ farmer\{u_2\}, donkey\{u_1\}, own\{u_2, u_1\}] \Rightarrow Pu_2]$
$\text{it} \rightsquigarrow u_1$
$[\text{beat it}] \rightsquigarrow \lambda v.[\ |\ beat\{v, u_1\}]$
$[[\text{every}[\text{farmer}[\text{who}^2[\text{t}_2[\text{own}[\text{a donkey}]]]]]][\text{beat it}]] \rightsquigarrow$
$[\ |\ [u_1u_2\ |\ farmer\{u_2\}, donkey\{u_1\}, own\{u_2, u_1\}]$
$\Rightarrow [\ |\ beat\{u_2, u_1\}]]$

Truth Again

- A DRS K will be said to be true in a state i (in some given model) if $\exists j K i j$ holds (in that model).
- The initial i may be thought of as an initial context that assigns initial values to memory locations. These initial values can be used for deictic reference.
- The existential quantification over j corresponds to the existence of a verifying embedding in the original DRT semantics.
- A weakest precondition calculus can be used to easily compute a first-order statement equivalent to $\exists j K i j$ for any translation K.

Truth Calculus 1

Let † be a bijection from the set of updatable referents to the variables of type e and write $x^{-\dagger} = u$ if $u^{\dagger} = x$. Extend † to the non-updatable referents by setting $Tom_{\nu}^{\dagger} = tom_e$, $Pat_{\nu}^{\dagger} = pat_e$, etc. Define the functions tr from conditions to type t terms and wp from DRSs and type t terms to type t terms as follows

$$
\begin{aligned}
&\mathsf{tr}(R\{\delta_1, \ldots, \delta_n\}) && = R\delta_1^{\dagger}, \ldots, \delta_n^{\dagger} \\
&\mathsf{tr}(\delta \textbf{ is } \delta') && = \delta^{\dagger} = \delta'^{\dagger} \\
&\mathsf{tr}(\textbf{not } K) && = \neg\, \mathsf{wp}(K, \top) \\
&\mathsf{tr}(K_1 \textbf{ or } K_2) && = \mathsf{wp}(K_1, \top) \vee \mathsf{wp}(K_2, \top) \\
&\mathsf{tr}(K \Rightarrow K') && = \neg\, \mathsf{wp}(K_1, \neg\, \mathsf{wp}(K_2, \top) \\
&\mathsf{wp}([\delta_1 \ldots \delta_m \mid \gamma_1, \ldots, \gamma_n], \varphi) && = \exists \delta_1^{\dagger} \ldots \delta_m^{\dagger}(\mathsf{tr}(\gamma_1) \wedge \ldots \wedge \mathsf{tr}(\gamma_n) \wedge \varphi) \\
&\mathsf{wp}(K \,;\, K', \varphi) && = \mathsf{wp}(K, \mathsf{wp}(K', \varphi))
\end{aligned}
$$

Truth Calculus 2

- Weakest precondition calculi come from the semantics of programming languages and were first put to the present use by (van Eijck and de Vries 1992).
- Write φ^i for the result of substituting $Vx^{-\dagger}i$ for each free variable x in φ. For every condition γ, DRS K and state i:

$$\gamma i \leftrightarrow \mathsf{tr}(\gamma)^i$$
$$\exists j K i j \leftrightarrow \mathsf{wp}(K, \top)^i$$

- He owns a cat $\rightsquigarrow [u_1 \mid cat\, u_1, own\, u_2 u_1]$
- $\mathsf{wp}([u_1 \mid cat\, u_1, own\, u_2 u_1], \top) = \exists x_1 (cat\, x_1 \wedge own\, x_2 x_1)$
- $\mathsf{wp}([u_1 \mid cat\, u_1, own\, u_2 u_1], \top)^i = \exists x_1 (cat\, x_1 \wedge own\, (V u_2 i) x_1)$

Taking Stock 1

- We have imported many aspects of Discourse Representation Theory into a type logical system of formal semantics.
- The strategy was to first establish the fundamental character of DRT. What kind of beast is it? What kind of logical space is its natural habitat?
- Groenendijk and Stokhof's answer: DRT is an abstract programming language. It feels happy in a space of relations between program states.
- There are other answers, e.g. (Zeevat 1989), and these could also be used, but Groenendijk and Stokhof's idea has served well as a first approximation.

Taking Stock 2

- Once the logical character of DRT had been established, the rest was technique.
- We gave some axioms that made the type ν and σ domains behave as desired.
- And after we had given new basic translations we were basically done.
- The procedure is very general and seems applicable in many instances where some form of logical interpretation of natural language is considered.
- A result of importing other systems into type logical semantics is that ideas can be combined. For example, ideas about generalized coordination are now combined with the dynamic approach of DRT.

Bach, E. (1986).
Natural Language Metaphysics.
In R. Marcus, G. Dorn, and P. Weingartner (Eds.), *Logic, Methodology and Philosophy of Science VII*, pp. 573–595. Amsterdam: North-Holland.

Barwise, J. (1987).
Noun Phrases, Generalized Quantifiers and Anaphora.
In P. Gärdenfors (Ed.), *Generalized Quantifiers, Logical and Linguistic Approaches*, pp. 1–29. Dordrecht: Reidel.

van Benthem, J. (1983).
The Logic of Time.
Dordrecht: Reidel.

Benzmüller, C., C. E. Brown, and M. Kohlhase (2004).
Higher Order Semantics and Extensionality.
Journal of Symbolic Logic 69.

Blackburn, P. and J. Bos (2005).
Representation and Inference for Natural Language: A First Course in Computational Semantics.
Stanford, CA: CSLI.
www.blackburnbos.org.

Carnap, R. (1947).
Meaning and Necessity.
Chicago: Chicago UP.

Cresswell, M. (1972).
Intensional Logics and Logical Truth.
Journal of Philosophical Logic 1, 2–15.

Cresswell, M. (1985).
Structured Meanings.
Cambridge, MA: MIT Press.

van Eijck, J. and F.-J. de Vries (1992).
Dynamic Interpretation and Hoare Deduction.
Journal of Logic, Language, and Information 1, 1–44.

Evans, G. (1980).
Pronouns.
Linguistic Inquiry 11(2), 337–362.

Fitting, M. (2002).
Types, Tableaus, and Gödels God.
Dordrecht: Kluwer Academic Publishers.

Frege, G. (1882).
Über die wissenschaftliche Berechtigung einer Begriffsschrift.
Zeitschrift für Philosophie und philosophische Kritik 81, 48–56.

Gallin, D. (1975).
Intensional and Higher-Order Modal Logic.
Amsterdam: North-Holland.

Gamut, L. (1991).
Logic, Language and Meaning.
Chicago: University of Chicago Press.

Gazdar, G. (1980).
A Cross-Categorial Semantics for Coordination.
Linguistics and Philosophy 3, 407–409.

Geach, P. (1962).
Reference and Generality.
Ithaca, NY: Cornell University Press.

Groenendijk, J. and M. Stokhof (1990).
Dynamic Montague Grammar.
In L. Kálmán and L. Pólos (Eds.), *Papers from the Second Symposium on Logic and Language*, pp. 3–48. Budapest: Akadémiai Kiadó.

Groenendijk, J. and M. Stokhof (1991).
Dynamic Predicate Logic.
Linguistics and Philosophy 14, 39–100.

Harel, D. (1984).
Dynamic Logic.
In D. Gabbay and F. Guenthner (Eds.), *Handbook of Philosophical Logic*, Volume II, pp. 497–604. Dordrecht: Reidel.

Heim, I. (1982).
The Semantics of Definite and Indefinite Noun Phrases.
Ph. D. thesis, Umass, Amherst.
published in 1989 by Garland, New York.

Hendriks, H. (1993).
Studied Flexibility: Categories and Types in Syntax and Semantics.
Ph. D. thesis, University of Amsterdam.

Henkin, L. (1950).
Completeness in the Theory of Types.
Journal of Symbolic Logic 15, 81–91.

Hintikka, J. (1962).
Knowledge and Belief.
Ithaca, NY: Cornell University Press.

Hintikka, J. (1975).
Impossible Possible Worlds Vindicated.
Journal of Philosophical Logic 4, 475–484.

Hodges, W. (1977).
Logic.
Harmondsworth: Penguin.

Kamp, H. (1981).
A Theory of Truth and Semantic Representation.
In J. Groenendijk, T. Janssen, and M. Stokhof (Eds.), *Formal Methods in the Study of Language*, pp. 277–322. Amsterdam: Mathematisch Centrum.

Kamp, H. and U. Reyle (1993).
From Discourse to Logic.
Dordrecht: Kluwer.

Karttunen, L. (1976).
Discourse Referents.
In J. McCawley (Ed.), *Syntax and Semantics 7: Notes from the Linguistic Underground*, pp. 363–385. New York: Academic Press.

Keenan, E. and L. Faltz (1978).
Logical Types for Natural Language.
UCLA Occasional Papers in Linguistics, 3.

Klein, E. and I. Sag (1985).
Type-Driven Translation.
Linguistics and Philosophy 8, 163–201.

Kratzer, A. (1977).
What "must" and "can" must and can mean.
Linguistics and Philosophy 1, 337–355.

Kripke, S. (1972).
Naming and Necessity.
In D. Davidson and G. Harman (Eds.), *Semantics of Natural Language*, pp. 253–355. Dordrecht: Reidel.

Lewis, D. (1972).
General Semantics.
In D. Davidson and G. Harman (Eds.), *Semantics of Natural Language*, pp. 169–218. Dordrecht: Reidel.

Lewis, D. (1975).
Adverbs of Quantification.
In E. Keenan (Ed.), *Formal Semantics of Natural Language*, pp. 3–15. Cambridge University Press.

Montague, R. (1970a).
English as a Formal Language.
In B. V. et al. (Ed.), *Linguaggi nella Società e nella Tecnica*, pp. 189–224. Milan: Edizioni di Comunità.
Reprinted in (Thomason 1974).

Montague, R. (1970b).
Universal Grammar.
Theoria 36, 373–398.
Reprinted in (Thomason 1974).

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).

Moschovakis, Y. (1994).
Sense and Denotation as Algorithm and Value.
In *Logic Colloquium '90 (Helsinki 1990)*, Volume 2 of *Lecture Notes in Logic*, pp. 210–249. Berlin: Springer.

Muskens, R. (1996).
Combining Montague Semantics and Discourse Representation.
Linguistics and Philosophy 19, 143–186.

Muskens, R. (2005a).
Intensional Models for the Theory of Types.
Submitted for Publication.

Muskens, R. (2005b).
Sense and the Computation of Reference.
Linguistics and Philosophy.
To appear.

Partee, B. and M. Rooth (1983).
Generalized Conjunction and Type Ambiguity.
In R. Baüerle, C. Schwarze, and A. von Stechow (Eds.), *Meaning, Use and Interpretation of Language*, pp. 361–383. Berlin: de Gruyter.

Pratt, V. (1976).
Semantical Considerations on Floyd-Hoare Logic.
In *Proc. 17th IEEE Symp. Found. Comp. Sci.*, pp. 46–57.

Quine, W. (1953).
From a Logical Point of View.
New York: Harper and Row.

Rooth, M. (1987).
Noun Phrase Interpretation in Montague Grammar, File Change Semantics, and Situation Semantics.
In P. Gärdenfors (Ed.), *Generalized Quantifiers, Logical and Linguistic Approaches*, pp. 237–268. Dordrecht: Reidel.

Russell, B. (1946).
A History of Western Philosophy.
London: George Allen & Unwin.

Santorini, B. and A. Kroch (2000).
The syntax of natural language: An online introduction using the Trees program.
http://www.ling.upenn.edu/ beatrice/syntax-textbook.

Sauerland, U. and P. Elbourne (2002).
Total Reconstruction, PF Movement, and Derivational Order.
Linguistic Inquiry 33(2), 283–319.

Staudacher, P. (1987).
Zur Semantik indefiniter Nominalphrasen.
In B. Asbach-Schnitker and J. Roggenhofer (Eds.), *Neuere Forschungen zur Wortbildung und Historiographie der Linguistik: Festgabe für Herbert E. Brekle zum 50. Geburtstag*, pp. 239–258. Gunter Narr Verlag.

Thomason, R. (Ed.) (1974).
Formal Philosophy, Selected Papers of Richard Montague.
Yale University Press.

Thomason, R. (1980).
A Model Theory for Propositional Attitudes.
Linguistics and Philosophy 4, 47–70.

Von Stechow, A. (1974).
ϵ-λ kontextfreie Sprachen: Ein Beitrag zu einer natürlichen formalen Semantik.
Linguistische Berichte 34, 1–33.

Zeevat, H. (1989).
A Compositional Approach to Discourse Representation Theory.
Linguistics and Philosophy 12, 95–131.