

Chapter 3. Compositional DRT

1. Introduction

The main goal of this chapter is to reformulate DPL (i.e. DPL without unselective and selective generalized quantification.) in type logic so that we will be able to define an interpretation procedure for English sentences that is both dynamic and compositional at the sub-sentential / sub-clausal level.

Section 2 provides the basics of the dynamic type-logical system, i.e. it defines the underlying dynamic logic, which I will label Dynamic Ty2¹. I follow the Compositional DRT (CDRT) and the Logic of Change in Muskens (1996) as closely as possible, the single most obvious exception being that I model discourse referents (dref's) as functions – as Muskens (1995b) does.

The choice to model dref's as functions is motivated by the fact that the resulting system can be more easily compared with situation-based D-/E-type approaches (in which pronouns are basically functions from minimal situations to individuals) and by the fact that the system can be more easily generalized to account for:

- multiple donkey sentences involving both weak and strong donkey anaphora, e.g.
Every person who buys a book on [amazon.com](https://www.amazon.com) (strong) and has a credit card (weak) uses it (the credit card) to pay for it (the book);
- plural anaphora, e.g. *Some / Three men came in. They sat down;*
- modal anaphora and modal subordination, e.g. *A wolf might enter the cabin. It would attack John first* (example based on Roberts (1989))

Section 3 shows how to translate the DPL system into Dynamic Ty2.

¹ Dynamic Ty2 is basically the Logic of Change in Muskens (1991, 1995b, 1996). I label it "Dynamic Ty2" to make more transparent the fact that it actually is a generalization of Dynamic Predicate Logic (Groenendijk & Stokhof 1991).

Building on the translation of DPL into type logic, the following two sections introduce compositionality at the sub-sentential level: section 4 describes a rough-and-ready syntax for the English fragment we will be concerned with, while section 5 defines its semantics (i.e. a type-driven translation procedure). The resulting version of CDRT is the basis for all the formal systems introduced throughout the remainder of the dissertation.

Section 6 provides the CDRT analyses of a fairly wide range of examples – for example, we show how CDRT derives the two possible quantifier scopings of the '*every man loves a woman*' kind of examples.

The differences between the material in this chapter and Muskens (1995b, 1996) are for the most part presentational. The main four differences are:

- the fact that section 2 provides a complete, detailed definition of the underlying Dynamic Ty2 logic;
- the fact that Dynamic Ty2 allows static objects of *arbitrary* types as dref values;
- the different analysis of proper names I end up adopting;
- the novel dynamic analysis of *ditransitive* verbs and the *scoping properties* of their Dative and Accusative objects in section 6.

Building on the foundations layed out in this chapter, the next chapter will add to the previous literature in a more substantial way by reformulating the DPL-style definitions of *unselective* and *selective* generalized quantification in type logic and, thus, extending CDRT to CDRT+GQ in a way that enables it to account for the weak / strong donkey ambiguity and the proportion problem.

The chapter concludes with a summary of the main results (section 7).

2. Dynamic Ty2

I have already indicated in the previous chapter that Compositional DRT (CDRT) combines Montague semantics and DRT in a formalism based on ordinary type logic. As Muskens (1991) puts it:

"[The unification is] based on two assumptions and one technical insight. The first assumption is that meaning is compositional. The meaning of words (roughly) are the smallest building blocks of meaning and meanings may combine into larger and larger structures by the rule that the meaning of a complex expression is given by the meaning of its parts.

The second assumption is that meaning is computational. Texts effect change, in particular, texts effect changes in context. The meaning of a sentence or text can be viewed as a relation between context states, much in the way that the meaning of a computer program can be viewed as a relation between program states.

[...] The technical insight [...] is that virtually all programming concepts to be found in the usual imperative computer languages are available in classical type theory. We can do any amount of programming in type theory. This suggests that type theory is an adequate tool for studying how languages can program context change. Since there is also some evidence that type theory is also a good vehicle for modelling how the meaning of a complex expression depends on the meaning of its parts, we may hope that it is adequate for a combined theory: a compositional theory of the computational aspects of natural language meaning."(Muskens (1991): 3-4²)

2.1. Preliminaries

The logic that underlies the entire present investigation is Ty2 (Gallin 1975; see also Janssen 1986 and Carpenter 1998). The set of basic types is $\{t, e, s\}$. Type t is the type of truth values; the logic is bivalent and total: the domain of type t (D_t) is $\{T, F\}$. Type e is the type of individuals; I assume (for the time being) that D_e contains only atomic entities, i.e. there are no pluralities. The domain of type s (D_s) models DPL's variable assignments; several axioms will be needed to ensure that the entities of type s do actually behave as DPL variable assignments.

Dref's are modeled as functions that take 'assignments' as arguments (i.e. entities of type s) and return a static object as value, e.g. an individual (type e). A dref for individuals will therefore be of type se . This is not as different from the DPL way of modeling dref's as it might seem: DPL models dref's as variables and a variable x is basically an instruction to look in the current info state, i.e. the current variable assignment g , and retrieve whatever individual the current info state associates with x – that individual is, of course, $g(x)$.

² Page references are to the online version.

Therefore, instead of working directly with variables, we can work with their 'type-lifted' versions, i.e. instead of x , we can take a dref to be a function of the form $\lambda g. g(x)$, which is basically the (set-theoretic) x^{th} projection function, which projects the sequence g onto the coordinate x .

This is precisely what Dynamic Ty2 does: instead of modeling discourse referents as atomic entities (variables) and info states as functions taking dref's as arguments (i.e. variable assignments), we model info states as atomic entities (of type s)³ and dref's as functions taking info states as arguments. Thus, dref's are similar to Montague's individual concepts: they do not refer directly, but only as a function of the current discourse context.

2.2. Definitions and Abbreviations

Let us turn now to the definition of Dynamic Ty2. For the most part, the definitions are the usual Ty2 ones. I will state what makes this logic a *Dynamic* Ty2 in plain words before or after each definition. The reader should feel free to just glance at these observations and move on to section 3, which shows how to translate DPL into Dynamic Ty2 and, by doing this, indirectly provides the empirical motivation for Dynamic Ty2.

The definition of types in (1) below isolates a subset of types as the types of dref's: these are functions from 'assignments' (type s) to static objects of arbitrary type. This seems to be more than sufficient for all the analyses in the present work⁴. We restrict our dref's to functions from 'variable assignments' to *static* objects of arbitrary types because, if we allow for arbitrary dref types, e.g. $s(st)$, we might run into counterparts of Russell's paradox – see Muskens (1995b): 179-180, fn. 10.

³ We can define a notion of 'info state' g that is closer to the DPL variable assignments, e.g. for any 'assignment' i of type s , let g^i be $\lambda v_{\tau}. vi$, where $\tau \in \mathbf{DrefTyp}$. If we consider only dref's for individuals, i.e. $\tau := se$, g^i is a function of type $(se)e$, that assigns values to dref's much like a DPL variable assignment assigns values to variables.

⁴ But see Stone & Hardt (1999) for an account of strict/sloppy readings that employs 'dynamic' drefs, i.e. dref's of type $s(s(\dots))$. These are just the *pointers* introduced in Janssen (1986).

In fact, throughout this work, I will be much more restrictive and use dref's that have *basic* static types (other than t) as codomains. The fact that we do not require dref's for higher-order static objects in our analyses can be to a certain extent taken to support the *empirical* hypothesis that natural languages strongly prefer to use discourse referents for entities of a basic static type⁵.

The *logic*, however, should allow for dref's that have any arbitrary static type as their codomain, given that the logic should provide a framework within which any plausible analysis can be formulated (including analyses involving dref's for higher-order static objects) and compared with alternative accounts.

1. **Dynamic Ty2** – the set of dref types **DRefTyp** and the set of types **Typ**.
 - a. The set of basic static types **BasSTyp**: $\{t, e\}$ (truth-values and individuals).
 - b. The set of static types **STyp**: the smallest set including **BasSTyp** and s.t., if $\sigma, \tau \in \text{STyp}$, then $(\sigma\tau) \in \text{STyp}$.
 - c. The set of dref types **DRefTyp**: the smallest set s.t., if $\sigma \in \text{STyp}$, then $(s\sigma) \in \text{DRefTyp}$.
 - d. The set of basic types **BasTyp**: $\text{BasSTyp} \cup \{s\}$ ('variable assignments').
 - e. The set of types **Typ**: the smallest set including **BasTyp** and s.t., if $\sigma, \tau \in \text{Typ}$, then $(\sigma\tau) \in \text{Typ}$.

The definition in (2) below provides some typical examples of expressions of various types and introduces several notational conventions that will improve the readability of the subsequent analyses.

2. **Dynamic Ty2 – basic expressions.**

For any type $\tau \in \text{Typ}$, there is a denumerable set of τ -constants **Con _{τ}** and a denumerably infinite set of τ -variables **Var _{τ}** = $\{v_{\tau,0}, v_{\tau,1}, \dots\}$, e.g.

Con _{e} = $\{\text{john}, \text{mary}, \text{dobby}, \dots, a, a', \dots, b, b', \dots, a_0, a_1, a_2, \dots\}$

Con _{$_{et}$} = $\{\text{donkey}, \text{farmer}, \text{house_elf}, \text{witch}, \dots, \text{leave}, \text{drunk}, \text{walk}, \dots\}$

Con _{$_{e(et)}$} = $\{\text{fall_in_love}, \text{own}, \text{beat}, \text{have}, \dots\}$

⁵ See also the related "No higher-order variables" hypothesis in Landman (2006).

$$\mathbf{Con}_s = \{h, h', \dots, i, i', \dots, j, j', \dots, k, k', \dots, h_0, h_1, \dots, i_0, i_1, \dots\}$$

$$\mathbf{Con}_{se} = \{u, u', u'', \dots, u_0, u_1, u_2, \dots\}$$

Notational conventions:

$x, x', \dots, y, y', \dots, z, z', \dots, x_0, x_1, \dots$ are variables of type e ;

$h, h', h'', \dots, i, i', i'', \dots, j, j', j'', \dots$ are variables of type s ;

$f, f', f'', \dots, f_0, f_1, f_2, \dots$ are variables over terms of type τ ,

for any $\tau \in \mathbf{STyp}$;

$v, v', v'', \dots, v_0, v_1, v_2, \dots$ are variables over terms of type τ ,

for any $\tau \in \mathbf{Typ}$.

The definition in (3) introduces the term $i[\delta]j$ of type t that is meant to model the DPL random variable assignment. Intuitively, the formula $i[\delta]j$ requires the info states i and j to differ at most with respect to the value they assign to def δ . Unlike Muskens (1995b, 1996), I introduce this as a basic formula of the language and not as an abbreviation, because the set **DRefTyp** of dref types is infinite and the abbreviation would require an infinite conjunction of formulas (as indicated in (4d) below).

I also introduce identity as a basic operator of the language; although it can be defined when the logic is interpreted relative to standard frames (as in (4a) below), I want to allow for the possibility of interpreting it relative to generalized (Henkin) frames, in which case identity is not definable anymore, just as it is not in many-sorted first-order logic.

Finally, note that proper names with capitals, e.g. *John*, are dref's for individuals (type se) and they are constant functions, a.k.a. *specific* dref's (see Muskens (1996)). They are defined in terms of the corresponding constant of type e , e.g. *john*.

3. Dynamic Ty2 – terms.

For any type $\tau \in \mathbf{Typ}$, the set of τ -terms **Term** $_{\tau}$ is the smallest set s.t.:

$$\mathbf{Con}_{\tau} \cup \mathbf{Var}_{\tau} \subseteq \mathbf{Term}_{\tau};$$

$$\alpha(\beta) \in \mathbf{Term}_{\tau} \text{ if } \alpha \in \mathbf{Term}_{\sigma\tau} \text{ and } \beta \in \mathbf{Term}_{\sigma} \text{ for any } \sigma \in \mathbf{Typ};$$

$$(\lambda v. \alpha) \in \mathbf{Term}_{\tau} \text{ if } \tau = (\sigma\rho), v \in \mathbf{Var}_{\sigma} \text{ and } \alpha \in \mathbf{Term}_{\rho} \text{ for any } \sigma, \rho \in \mathbf{Typ};$$

$$(\alpha = \beta) \in \mathbf{Term}_{\tau} \text{ if } \tau = t \text{ and } \alpha, \beta \in \mathbf{Term}_{\sigma} \text{ for any } \sigma \in \mathbf{Typ};$$

$(i[\delta]) \in \mathbf{Term}_\tau$ if $\tau = t$ and $i, i' \in \mathbf{Var}_s$ and $\delta \in \mathbf{Term}_\sigma$, for any $\sigma \in \mathbf{DRefTyp}$.

Abbreviations (the subscripts on terms indicate their type):

$$\begin{aligned}
 John_{se} &:= \lambda i_s. john_e, Mary_{se} := \lambda i_s. mary_e \dots; \\
 \mathbf{T} &:= \lambda f_t. f = \lambda f_t. f; & \mathbf{F} &:= \lambda f_t. f_t = \lambda f_t. \mathbf{T}; \\
 \neg &:= \lambda f_t. f_t = \mathbf{F}; & \wedge &:= \lambda f_t f'_t. (\lambda f''_{tt}. f''(f) = f') = \lambda f''_{tt}. f''(\mathbf{T}) \text{ }^6; \\
 \rightarrow &:= \lambda f_t f'_t. (f \wedge f') = f; & \vee &:= \lambda f_t f'_t. \neg f \rightarrow f'; \\
 \forall v(\phi) &:= \lambda v. \phi = \lambda v. \mathbf{T}; & \exists v(\phi) &:= \lambda v. \neg \phi \neq \lambda v. \mathbf{T}.
 \end{aligned}$$

Definition (4) introduces four axioms that Dynamic Ty2 models have to satisfy. These axioms make sure that the entities of type s actually behave as variable assignments intuitively do⁷.

First, **Axiom1** employs a non-logical constant **udref** to identify *unspecific* dref's, i.e. the dref's that are supposed to behave as the DPL variables, e.g. u_0, u_1 etc. The constant function *John* ($John := \lambda i. john_e$ – see (3) above) for example is a *specific* dref: although it is of type se , i.e. the type of dref's for individuals, it does not behave as a DPL variable – its value does not vary from 'assignment' to 'assignment'; if anything, specific dref's are the counterpart of DPL constants, not variables.

Axiom2 makes sure that all the unspecific dref names actually name different functions: if two distinct names denoted the same function, we would accidentally update *both* whenever we would update one of them.

Axiom3 ensures that, just like DPL variable assignments, two 'assignments' (i.e. two entities of type s) are different only if they assign different values to some dref δ . If they assign the same values to all dref's, the 'assignments' are identical.

Finally, **Axiom4** ensures that we have enough 'assignments': for any given 'assignment' i , any unspecific dref v and any possible dref value (i.e. static object) f of the

⁶ Equivalently, $\wedge := \lambda f_t f'_t. \forall f''_{tt} (f''(f, f') = f''(\mathbf{T}, \mathbf{T}))$ or $\wedge := \lambda f_t f'_t. \forall f''_{tt} (f = (f''(f) = f''(f')))$.

⁷ To get a better grasp of the axioms, the reader might find it instructive to construct a model for them, i.e. to construct the domain D_s given the domain D_e and the set of **udref** names.

appropriate type, there is an 'assignment' j that differs from i at most with respect to the value it assigns to v and which in fact assigns f as the value of v .

4. Dynamic Ty2 – frames, models, assignments, interpretation and truth.

a. A *standard frame* F for Dynamic Ty2 is a set $D = \{D_\tau : \tau \in \mathbf{Typ}\}$ s.t. D_e, D_t and D_s are pairwise disjoint sets and $D_{\sigma\tau} = \{f : f \text{ is a total function from } D_\sigma \text{ to } D_\tau\}$, for any $\sigma, \tau \in \mathbf{Typ}$.

b. A *model* M for Dynamic Ty2 is a pair $\langle F^M, \|\cdot\|^M \rangle$ s.t.:

- F^M is a standard frame for Dynamic Ty2;

- $\|\cdot\|^M$ assigns an object $\|\alpha\|^M \in D_\tau^M$ to each $\alpha \in \mathbf{Con}_\tau$ for any $\tau \in \mathbf{Typ}$,

i.e. $\|\cdot\|^M$ respects typing;

- M satisfies the following axioms⁸:

Axiom1 ("Unspecific dref's"): $\mathbf{udref}(\delta)$,

for any unspecific dref name δ of any type $(s\tau) \in \mathbf{DRefTyp}$,

e.g. u_0, u_1, \dots but not $John, Mary, \dots$;

\mathbf{udref} is a non-logical constant⁹ intuitively identifying the 'variable' dref's,

i.e. the non-constant functions of type $s\tau$ (for any $\tau \in \mathbf{STyp}$)

intended to model DPL-like variables.

Axiom2 ("Dref's have unique dref names"): $\mathbf{udref}(\delta) \wedge \mathbf{udref}(\delta') \rightarrow \delta = \delta'$,

for any two distinct dref names δ and δ' of type τ ,

for any type $\tau \in \mathbf{DRefTyp}$,

i.e. we make sure that we do not accidentally update a dref δ' when we update δ .

Axiom3 ("Identity of 'assignments'"): $\forall i, j, s (i[s]j \rightarrow i=j)$.

Axiom4 ("Enough 'assignments'"): $\forall i, s \forall v_{s\tau} \forall f_\tau (\mathbf{udref}(v) \rightarrow \exists j_s (i[v]j \wedge vj=f))$,

for any type $\tau \in \mathbf{STyp}$.

⁸ The axioms / axiom schemes are based on Muskens (1995b, 1996).

⁹ In fact, \mathbf{udref} stands for an infinite family of non-logical constants of type (τ) for any $\tau \in \mathbf{DRefTyp}$. Alternatively, we can assume a polymorphic type logic with infinite sum types, in which \mathbf{udref} is a polymorphic function. For a discussion of sum types, see for example Carpenter (1998): 69 et seqq.

c. An M -assignment θ is a function that assigns to each variable $v \in \mathbf{Var}_\tau$ an element $\theta(v) \in D^M_\tau$ for any $\tau \in \mathbf{Typ}$. Given an M -assignment θ , if $v \in \mathbf{Var}_\tau$ and $d \in D^M_\tau$, then $\theta^{v/d}$ is the M -assignment identical with θ except that it assigns d to v .

d. The interpretation function $\|\cdot\|^{M,\theta}$ is defined as follows:

$$\begin{aligned}
\|\alpha\|^{M,\theta} &= \|\alpha\|^M && \text{if } \alpha \in \mathbf{Con}_\tau \text{ for any } \tau \in \mathbf{Typ}; \\
\|\alpha\|^{M,\theta} &= \theta(\alpha) && \text{if } \alpha \in \mathbf{Var}_\tau \text{ for any } \tau \in \mathbf{Typ}; \\
\|\alpha(\beta)\|^{M,\theta} &= \|\alpha\|^{M,\theta} (\|\beta\|^{M,\theta}); \\
\|\lambda v. \alpha\|^{M,\theta} &= \langle \|\alpha\|^{M,\theta^{v/d}} : d \in D^M_\sigma \rangle && \text{if } v \in \mathbf{Var}_\sigma; \\
\|\alpha = \beta\|^{M,\theta} &= \mathbf{T} && \text{if } \|\alpha\|^{M,\theta} = \|\beta\|^{M,\theta} \\
&= \mathbf{F} && \text{otherwise.} \\
\|i[\delta]j\|^{M,\theta} &= \mathbf{T} && \text{if } \delta \in \mathbf{Term}_\sigma, \sigma \in \mathbf{DRefTyp} \text{ and} \\
&\|\forall v_\sigma(\mathbf{udref}(v) \wedge v \neq \delta \rightarrow vi = vj)\|^{M,\theta} = \mathbf{T} \text{ and} \\
&\|\forall v_\tau(\mathbf{udref}(v) \rightarrow vi = vj)\|^{M,\theta} = \mathbf{T} \text{ for all } \tau \neq \sigma, \tau \in \mathbf{DRefTyp} \\
&= \mathbf{F} && \text{otherwise.}
\end{aligned}$$

e. **Truth:** A formula $\phi \in \mathbf{Term}_t$ is *true* in M relative to θ iff $\|\phi\|^{M,\theta} = \mathbf{T}$.

A formula $\phi \in \mathbf{Term}_t$ is *true* in M iff it is true in M relative to any assignment θ .

3. Translating DPL into Dynamic Ty2

In this section, we will see how to encode DPL (and therefore classical DRT / FCS) in Dynamic Ty2. We do this by providing a list of abbreviations that follows closely the definition of DPL in the previous chapter: the *definiendum* has the form of a DPL expression, while the *definiens* is a term of Dynamic Ty2. As soon as the abbreviations are in place, we will see how they are employed by analyzing the examples we have previously used as empirical motivation for DPL.

3.1. Definitions and Abbreviations

Definition (5) below corresponds to the DPL definition in the preceding chapter. Note that ' \wedge ' is the Dynamic Ty2 conjunction, i.e. the official, type-logical conjunction, and ' \neg ' is the Dynamic Ty2 negation, i.e. the official, type-logical negation. In contrast, dynamic conjunction ';' and dynamic negation '~' are simply abbreviations.

Note also that the DPL notion of random assignment $[x]$ has as its direct correspondent the random assignment $[u]$ of Dynamic Ty2.

The DPL distinction between conditions and DRS's is formulated in terms of types. Conditions are terms of type st , i.e. they denote sets of 'assignments'; intuitively, conditions denote the set of 'assignments' that satisfy them. DRS's are terms of type $s(st)$, i.e. binary relations between 'assignments'; intuitively, a DRS D is satisfied by a pair of two 'assignments' i and j , i.e. $Dij = \mathbf{T}$ ¹⁰, iff the output 'assignment' j is the result of non-deterministically updating the input 'assignment' i with D .

5. DPL in Dynamic Ty2 (subscripts on terms represent their types).

a. Atomic conditions – type st :

$$R\{u_1, \dots, u_n\} := \lambda i_s. R(u_1 i, \dots, u_n i),$$

for any non-logical constant R of type $e^n t$,

where $e^n t$ is defined as follows: $e^0 t := t$ and $e^{m+1} t := e(e^m t)$ ¹¹

$$u_1 = u_2 := \lambda i_s. u_1 i = u_2 i$$

b. Atomic DRS's (DRS's containing exactly one atomic condition) – type $s(st)$

(corresponding to DPL atomic formulas):

$$[R\{u_1, \dots, u_n\}] := \lambda i_s j_s. i = j \wedge R\{u_1, \dots, u_n\} j$$

$$[u_1 = u_2] := \lambda i_s j_s. i = j \wedge u_1 j = u_2 j$$

c. Condition-level connectives (negation), i.e. non-atomic conditions:

$$\sim D := \lambda i_s. \neg \exists k_s (D i k) \text{ }^{12}, \quad \text{where } D \text{ is a DRS (term of type } s(st))$$

$$\text{i.e. } \sim D := \lambda i_s. i \notin \mathbf{Dom}(D), \quad \text{where } \mathbf{Dom}(D) := \{i_s : \exists j_s (D i j)\}$$

d. Tests (generalizing 'atomic' DRS's):

¹⁰ Recall that \mathbf{T} and \mathbf{F} are the model-theoretic objects intuitively modeling 'True' and 'False', while their bolded counterparts \mathbf{T} and \mathbf{F} are the Dynamic Ty2 constants whose semantic values are \mathbf{T} and \mathbf{F} respectively.

¹¹ The definition of $e^n t$ is due to Muskens (1996): 157-158.

¹² Strictly speaking, the Dynamic Ty2 translation of DPL negation is defined as $\mathbf{TR}(\sim \phi) := [\sim \mathbf{TR}(\phi)]$, i.e. $\mathbf{TR}(\sim \phi) := [\lambda i_s. \neg \exists k_s (\mathbf{TR}(\phi) i k)]$. \mathbf{TR} is the translation function from DPL to Dynamic Ty2 which is recursively defined in the expected way, e.g. for DPL atomic formulas, we have that $\mathbf{TR}(R(x_1, \dots, x_n)) := [R\{u_1, \dots, u_n\}]$ and $\mathbf{TR}(x_1 = x_2) := u_1 = u_2$.

$$[C_1, \dots, C_m] := \lambda i_s j_s. i=j \wedge C_{1j} \wedge \dots \wedge C_{mj}^{13},$$

where C_1, \dots, C_m are conditions (atomic or not) of type st .

e. DRS-level connectives (dynamic conjunction):

$$D_1; D_2 := \lambda i_s j_s. \exists h_s(D_1 i h \wedge D_2 h j), \quad \text{where } D_1 \text{ and } D_2 \text{ are DRS's (type } s(st))$$

f. Quantifiers (random assignment of value to a dref):

$$[u] := \lambda i_s j_s. i[u]j$$

g. Truth: A DRS D (type $s(st)$) is true with respect to an input info state i_s iff $\exists j_s(D i j)$, i.e. $i \in \mathbf{Dom}(D)$ ¹⁴.

The abbreviations introduced in definition (6) below correspond to the DPL abbreviations defined in the previous chapter. ' \exists ' and ' \forall ' are the official type-logical existential and universal quantifiers, while ' \forall ' and ' \exists ' are the abbreviations corresponding to the dynamic (DPL-style) existential and universal quantifiers. I use ' \rightarrow ' and ' \vee ' both for the official Dynamic Ty2 and for the dynamic DPL-style implication and, respectively, disjunction; they can be easily disambiguated in context.

6. a. Additional abbreviations – condition-level connectives (closure, disjunction, implication):

$$!D := \sim[\sim D]^{15},$$

$$\text{i.e. } !D := \lambda i_s. \exists k_s(D i k) \text{ or simply: } !D := \mathbf{Dom}(D)$$

$$D_1 \vee D_2 := \sim([\sim D_1]; [\sim D_2]),$$

$$\text{i.e. } D_1 \vee D_2 := \sim[\sim D_1, \sim D_2]$$

$$\text{i.e. } D_1 \vee D_2 := \lambda i_s. \exists k_s(D_1 i k \vee D_2 i k);$$

¹³ Alternatively, $[C_1, \dots, C_m]$ can be defined using dynamic conjunction as follows:

$$[C_1, \dots, C_m] := \lambda i_s j_s. ([C_1]; \dots; [C_m]) i j, \text{ where } [C] := \lambda i_s j_s. i=j \wedge C j.$$

¹⁴ Or, equivalently, $i \in !D$ – see the abbreviation of ' $!$ ' in (6) below.

¹⁵ Strictly speaking, DPL anaphoric closure is translated in Dynamic Ty2 as $\mathbf{TR}(!\phi) := [\sim \mathbf{TR}(\sim \phi)]$, i.e. $\mathbf{TR}(!\phi) := [\sim[\sim \mathbf{TR}(\phi)]] = [\sim[\lambda j_s. \neg \exists l_s(\mathbf{TR}(\phi) j l)]] = [\lambda i_s. \neg \exists k_s([\lambda j_s. \neg \exists l_s(\mathbf{TR}(\phi) j l)] i k)]$, i.e. $\mathbf{TR}(!\phi) := [\lambda i_s. \neg \exists k_s(i=k \wedge \neg \exists l_s(\mathbf{TR}(\phi) k l))] = [\lambda i_s. \neg(\neg \exists l_s(\mathbf{TR}(\phi) i l))]$, i.e. $\mathbf{TR}(!\phi) := [\lambda i_s. \exists l_s(\mathbf{TR}(\phi) i l)] = \mathbf{Dom}(\mathbf{TR}(\phi))$.

\mathbf{TR} is the translation function from DPL to Dynamic Ty2 – see fn. 12 above.

equivalently: $D_1 \vee D_2 := \mathbf{Dom}(D_1) \cup \mathbf{Dom}(D_2)$ ¹⁶

$$D_1 \rightarrow D_2 := \sim(D_1; [\sim D_2]),$$

$$\text{i.e. } D_1 \rightarrow D_2 := \lambda i_s. \forall h_s(D_1 i h \rightarrow \exists k_s(D_2 h k))$$
 ¹⁷,

$$\text{i.e. } D_1 \rightarrow D_2 := \lambda i_s. (D_1) i \subseteq \mathbf{Dom}(D_2), \quad \text{where } (D) i := \{j_s: D i j\}$$

b. Additional abbreviations – DRS-level quantifiers (multiple random assignment, existential quantification):

$$[u_1, \dots, u_n] := [u_1]; \dots; [u_n]$$

$$\exists u(D) := [u]; D$$

c. Additional abbreviations – condition-level quantifiers (universal quantification):

$$\forall u(D) := \sim([u]; [\sim D]),$$

$$\text{i.e. } \sim[u \mid \sim D] \text{ or } [u] \rightarrow D \text{ or equivalently } \sim \exists u([\sim D]),$$

$$\text{i.e. } \forall u(D) := \lambda i_s. \forall h_s(i[u] h \rightarrow \exists k_s(D h k)),$$

$$\text{i.e. } \forall u(D) := \lambda i_s. ([u]) i \subseteq \mathbf{Dom}(D)$$

d. Additional abbreviations – DRS's (a.k.a. linearized 'boxes'):

$$[u_1, \dots, u_n \mid C_1, \dots, C_m] := \lambda i_s j_s. ([u_1, \dots, u_n]; [C_1, \dots, C_m]) i j,$$

where C_1, \dots, C_m are conditions (atomic or not),

$$\text{i.e. } [u_1, \dots, u_n \mid C_1, \dots, C_m] := \lambda i_s j_s. i[u_1, \dots, u_n] j \wedge C_1 i j \wedge \dots \wedge C_m i j.$$

3.2. Cross-sentential Anaphora

Going through the examples that motivated DPL and classical DRT / FCS in the first place will help us get familiar with Dynamic Ty2 translations and, at the same time, provide the empirical motivation for various features of the formal system. Consider again the mini-discourse in (7-8) below, containing two instances of cross-sentential anaphora.

¹⁶ $D_1 \vee D_2 := \sim([\sim D_1]; [\sim D_2]) = \lambda i. \neg \exists k(([\sim D_1]; [\sim D_2]) i k) = \lambda i. \neg \exists k l([\sim D_1] i l \wedge [\sim D_2] l k) = \lambda i. \neg \exists k l(i=l \wedge \neg \exists h(D_1 i h) \wedge l=k \wedge \neg \exists h'(D_2 l h')) = \lambda i. \neg(\neg \exists h(D_1 i h) \wedge \neg \exists h'(D_2 i h')) = \lambda i. \exists h(D_1 i h) \vee \exists h'(D_2 i h') = \lambda i. \exists k(D_1 i k \vee D_2 i k).$

¹⁷ $D_1 \rightarrow D_2 := \sim(D_1; [\sim D_2]) = \lambda i. \neg \exists k((D_1; [\sim D_2]) i k) = \lambda i. \neg \exists k l(D_1 i l \wedge [\sim D_2] l k) = \lambda i. \neg \exists k l(D_1 i l \wedge l=k \wedge \neg \exists h(D_2 l h)) = \lambda i. \neg \exists k(D_1 i k \wedge \neg \exists h(D_2 k h)) = \lambda i. \forall k(D_1 i k \rightarrow \exists h(D_2 k h)).$

7. A u_1 house-elf fell in love with a u_2 witch.
8. He u_1 bought her u_2 an u_3 alligator purse.

I provide its DRT-style representation in DPL and its DRT-style representation in Dynamic Ty2 in (9) and (10) below. The formula in (11) is the 'unpacked' type-logical term of type $s(st)$ translating the discourse in (7-8). Finally, the formula in (12) provides the truth-conditions associated with the Dynamic Ty2 term in (11), derived on the basis of the definition of truth for DRS's in (5g) and the "Enough States" axiom (**Axiom4** in (4b) above).

Note that the formula in (12) capturing the truth-conditions of discourse (7-8) contains a vacuous λ -abstraction over input 'assignments', which is intuitively correct given that the discourse does not contain any item whose reference is dependent on the input context, as for example a deictically used pronoun would be.

9. $[x, y \mid \text{house_elf}(x), \text{witch}(y), \text{fall_in_love}(x, y)]; [z \mid \text{alligator_purse}(z), \text{buy}(x, y, z)]$
10. $[u_1, u_2 \mid \text{house_elf}\{u_1\}, \text{witch}\{u_2\}, \text{fall_in_love}\{u_1, u_2\}]; [u_3 \mid \text{alligator_purse}\{u_3\}, \text{buy}\{u_1, u_2, u_3\}]$
11. $\lambda i_{js}. i[u_1, u_2, u_3]j \wedge \text{house_elf}(u_{1j}) \wedge \text{witch}(u_{2j}) \wedge \text{fall_in_love}(u_{1j}, u_{2j}) \wedge \text{alligator_purse}(u_{3j}) \wedge \text{buy}(u_{1j}, u_{2j}, u_{3j})$
12. $\lambda i_s. \exists x_e \exists y_e \exists z_e (\text{house_elf}(x) \wedge \text{witch}(y) \wedge \text{fall_in_love}(x, y) \wedge \text{alligator_purse}(z) \wedge \text{buy}(x, y, z))$

3.3. Relative-Clause Donkey Sentences

Let us turn now to the relative-clause donkey example in (13) below. The formula in (14) is its DPL translation (abbreviated DRT-style), while the corresponding Dynamic Ty2 formula is provided in (15). Note the double square brackets on the left- and right-hand side of the Ty2 representation: the external square brackets are due to the fact that dynamic implication ' \rightarrow ' is a condition-level connective (see definition (6a) above), so we need the extra square brackets to obtain a test, i.e. a DRS (which is a term of type $s(st)$), out of a condition of type st .

13. Every^{*u*₁} house-elf who falls in love with a^{*u*₂} witch buys her^{*u*₂} an^{*u*₃} alligator purse.

14. $[x, y \mid \text{house_elf}(x), \text{witch}(y), \text{fall_in_love}(x, y)]$
 $\rightarrow [z \mid \text{alligator_purse}(z), \text{buy}(x, y, z)]$

15. $[[u_1, u_2 \mid \text{house_elf}\{u_1\}, \text{witch}\{u_2\}, \text{fall_in_love}\{u_1, u_2\}]]$
 $\rightarrow [u_3 \mid \text{alligator_purse}\{u_3\}, \text{buy}\{u_1, u_2, u_3\}]]$

The DRT-style representation in (15) above is 'unpacked' as the type-logical term in (16) below, whose truth-conditions are given in (17); note again the vacuous λ -abstraction over 'assignments', followed by a static first-order formula that captures the intuitively correct truth-conditions for sentence (13) above.

16. $\lambda i_s j_s. i=j \wedge \forall h_s(i[u_1, u_2]h \wedge \text{house_elf}(u_1h) \wedge \text{witch}(u_2h) \wedge \text{fall_in_love}(u_1h, u_2h))$
 $\rightarrow \exists k_s(h[u_3]k \wedge \text{alligator_purse}(u_3k) \wedge \text{buy}(u_1k, u_2k, u_3k))$

17. $\lambda i_s. \forall x_e \forall y_e(\text{house_elf}(x) \wedge \text{witch}(y) \wedge \text{fall_in_love}(x, y))$
 $\rightarrow \exists z_e(\text{alligator_purse}(z) \wedge \text{buy}(x, y, z))$

3.4. Conditional Donkey Sentences

The conditional donkey sentence repeated in (18) below receives the same Dynamic Ty2 translation and the same truth-conditions as the relative-clause donkey sentence in (13) above (see (15), (16) and (17)). Thus, just as DPL, the Dynamic Ty2 translations capture the intuitive correspondence between the generalized determiner *every* and bare conditional structures.

18. If a^{*u*₁} house-elf falls in love with a^{*u*₂} witch, he^{*u*₁} buys her^{*u*₂} an^{*u*₃} alligator purse.

Finally, we turn to the intuitively equivalent negative donkey sentences we have analyzed in DPL in the previous chapter – repeated in (19), (20) and (21) below.

19. No^{*x*} house-elf who falls in love with a^{*y*} witch buys her^{*y*} an^{*z*} alligator purse.

20. If a^{*x*} house-elf falls in love with a^{*y*} witch, he^{*x*} never buys her^{*y*} an^{*z*} alligator purse.

21. If a^{*x*} house-elf falls in love with a^{*y*} witch, he^{*x*} doesn't buy her^{*y*} an^{*z*} alligator purse.

Just as in the case of the DPL, we can translate sentence (19) in Dynamic Ty2 in two different ways, i.e. we can translate the determiner *no* either by means of a combination of negation and existential quantification or by means of a combination of negation and universal quantification. But this is not a problem for Dynamic Ty2 just as it wasn't for DPL: as expected, the DPL partial duality between existential and universal quantification is inherited in Dynamic Ty2, so the two translations, i.e. the two *st* terms in (22) below, are identical.

The terms in (22) are of type *st* because both dynamic negation ' \sim ' and universal quantification \forall are condition-level connectives. The corresponding tests – which are DRS's, i.e. terms of type $s(st)$ – are obviously identical if the conditions they are based on are identical.

$$22. \sim\exists u(D; D') = \forall u([D \rightarrow [\sim D']])^{18}$$

And, given the identity in (22), we can translate sentence (19) either way¹⁹, as shown in (23) and (25) below. Furthermore, these equivalent translations are also equivalent to the DRT-style formulas in (24) and (26).

Note that the universal quantification over pairs of house-elves and witches is exhibited in the clearest way by (26), since any dref introduced in the antecedent of a conditional ends up being quantified over universally²⁰.

$$\begin{aligned}
 &^{18} \forall u(D \rightarrow [\sim D']) = \lambda i_s. \forall h_s([u]ih \rightarrow \exists k_s((D \rightarrow [\sim D'])hk)) \\
 &= \lambda i_s. \forall h_s([u]ih \rightarrow \exists k_s(h=k \wedge \forall h'_s(Dhh' \rightarrow \exists k'_s([\sim D']h'k))) = \lambda i_s. \forall h_s([u]ih \rightarrow \forall h'_s(Dhh' \rightarrow \exists k'_s(h'=k' \wedge (\sim D')k'))) \\
 &= \lambda i_s. \forall h_s([u]ih \rightarrow \forall h'_s(Dhh' \rightarrow (\sim D')h')) = \lambda i_s. \forall h_s([u]ih \rightarrow \forall h'_s(Dhh' \rightarrow \neg \exists k_s(Dh'k)) \\
 &= \lambda i_s. \forall h_s([u]ih \rightarrow \forall h'_s(Dhh' \rightarrow \neg \exists k_s(Dh'k)) = \lambda i_s. \neg \exists h_s([u]ih \wedge \neg \forall h'_s(Dhh' \rightarrow \neg \exists k_s(Dh'k)) \\
 &= \lambda i_s. \neg \exists h_s([u]ih \wedge \exists h'_s \neg (Dhh' \rightarrow \neg \exists k_s(Dh'k)) = \lambda i_s. \neg \exists h_s([u]ih \wedge \exists h'_s(Dhh' \wedge \exists k_s(Dh'k)) \\
 &= \lambda i_s. \neg \exists h_s \exists h'_s \exists k_s([u]ih \wedge Dhh' \wedge Dh'k) = \lambda i_s. \neg \exists k_s(([u]; D; D')ik) = \sim \exists u(D; D')
 \end{aligned}$$

¹⁹ I assume that terms that are equivalent to (Dynamic Ty2 translations of DPL) translations of English sentences are also acceptable as translations.

²⁰ It is easily checked that the following identities hold:

$$\forall u([D \rightarrow D']) = [u] \rightarrow [D \rightarrow D'] = ([u]; D) \rightarrow D' = \exists u(D) \rightarrow D'.$$

23. $[\sim\exists u_1([house_elf\{u_1\}]; \exists u_2([witch\{u_2\}, fall_in_love\{u_1, u_2\}]);$
 $\exists u_3([alligator_purse\{u_3\}, buy\{u_1, u_2, u_3\}]))]$
24. $[\sim[u_1, u_2, u_3 \mid house_elf\{u_1\}, witch\{u_2\}, fall_in_love\{u_1, u_2\},$
 $alligator_purse\{u_3\}, buy\{u_1, u_2, u_3\}]]]$
25. $[\forall u_1([house_elf\{u_1\}]; \exists u_2([witch\{u_2\}, fall_in_love\{u_1, u_2\}])$
 $\rightarrow [\sim\exists u_3([alligator_purse\{u_3\}, buy\{u_1, u_2, u_3\}])]])]$
26. $[[u_1, u_2 \mid house_elf\{u_1\}, witch\{u_2\}, fall_in_love\{u_1, u_2\}]$
 $\rightarrow [\sim[u_3 \mid alligator_purse\{u_3\}, buy\{u_1, u_2, u_3\}]]]$

Note also that the formula in (26) is in fact the compositional translation of the negative conditional sentences in (20) and (21) above.

The Dynamic Ty2 truth-conditions for all three sentences are most easily derived from formula (24) – and they are provided in (27) below. Just as before, we have vacuous λ -abstraction over 'assignments', followed by a static first-order formula that captures the intuitively correct truth-conditions for the three English sentences under consideration.

27. $\lambda i_s. \neg\exists x_e\exists y_e\exists z_e(house_elf(x) \wedge witch(y) \wedge fall_in_love(x, y) \wedge$
 $alligator_purse(z) \wedge buy(x, y, z))$

Thus, we see that Dynamic Ty2 can capture everything that DPL (hence classical DRT / FCS) does – including compositionality down to sentence- / clause-level. However, with Dynamic Ty2, we have all the ingredients to go compositional at the sub-sentential / sub-clausal level, which is what sections 4 and 5 below endeavor to do. I conclude this section with a brief discussion of the Dynamic Ty2 analysis of proper names.

Intermezzo: Proper Names in Dynamic Ty2

The main choice for the analysis of proper names in Dynamic Ty2 is between: (i) a *pronoun-like* analysis, whereby a proper name is basically interpreted as a deictically used pronoun, whose referent is specified by the input discourse context, and (ii) an *indefinite-like* analysis, whereby a proper name introduces a new individual-level dref

whose referent is constrained to be the individual (rigidly) designated by the proper name.

Following Muskens (1996), I have introduced specific dref's corresponding to proper names, i.e. constant functions from 'assignments' to individuals, e.g. $John := \lambda i_s. john_e$. However, unlike Muskens – who chooses the pronoun-like analysis of proper names –, I will not interpret proper names as denoting such specific dref's, but I will instead let proper names introduce an unspecific dref and an identity condition between the unspecific dref and the specific dref that is the Dynamic Ty2 correspondent of the proper name. For example, the proper name *John* is represented as shown in (28) below.

$$28. John^u \rightsquigarrow [u \mid u=John], \quad \text{i.e. } \lambda i_s j_s. i[u]j \wedge uj=Johnj, \quad \text{i.e. } \lambda i_s j_s. i[u]j \wedge uj=john$$

As (28) shows, the newly introduced unspecific dref is constrained to have the value $john_e$ in the output info state j . This interpretation of proper names is in fact equivalent to the *external anchoring* of proper names in Kamp & Reyle (1993): 248 – and it is similar to the interpretation of proper names in Kamp (1981). Moreover, pronouns anaphoric to proper names are taken to be anaphoric to the unspecific dref introduced by the proper name, as exemplified by (29) below.

$$29. \dots John^u \dots he_u \dots$$

As Muskens (1996): 151-153 observes, this kind of representation seems needlessly complex: why not simply take the proper name to be anaphoric to its corresponding specific dref? This would basically be equivalent to using the proper name as a deictic anaphor, interpreted directly relative to the input context (a.k.a. info state or 'assignment')²¹. Moreover, a pronoun anaphoric to a proper name would be anaphoric to the corresponding specific dref, as shown in (30) below. From this perspective, the use of a pronoun anaphoric to a proper name and the use of the proper name itself are not really different.

²¹ Which is basically what the causal chain theory of proper names proposes – see Kripke (1972) and Kaplan (1977/1989a, 1989b) among others.

30. ... *John*_{John} ... *he*_{John} ...

This conflation of proper names and pronouns requires additional justification, as the two are different in at least two respects. First, a proper name is completely felicitous in a discourse initial position, while a felicitous pronoun requires a suitable context (linguistic or non-linguistic) to have previously been set up – as shown in (31) and (32) below.

31. Dobby entered The Three Broomsticks.

32. ??He_{Dobby} entered The Three Broomsticks.

Second, when the proper name has been (recently) mentioned, using an anaphoric pronoun is felicitous, while using the proper name again is usually not, as shown in (33) and (34) below.

33. Dobby entered the Three Broomsticks. He_{Dobby} ordered a butterbeer.

34. Dobby entered the Three Broomsticks. ??Dobby ordered a butterbeer.

These two observations seem to argue for the indefinite-like and against the pronoun-like analysis of proper names.

However, the pronoun-like analysis of proper names, i.e. representing proper names as deictic pronouns together with the assumption that proper names are by default salient in the input context, is supported by the interaction between anaphora to proper names and negation. Generally, an indefinite introduced in the scope of negation is not anaphorically accessible to a subsequent pronoun, as shown in (35) below. In contrast, a proper name is anaphorically accessible when it occurs in the scope of negation, as (36) shows.

35. Hermione didn't see a^u / any^u house-elf in the Three Broomsticks.

#He_u was on vacation in the Bahamas.

36. Hermione didn't see Dobby in the Three Broomsticks.

He_{Dobby} was on vacation in the Bahamas.

The fact that dynamic negation is defined as a condition in (5c) above, i.e. as *externally static*, correctly predicts the infelicity of anaphora in (35): the pronoun, despite

being co-indexed with the indefinite, ends up being interpreted as a deictic pronoun, picking up whatever the input context associates with the dref u .

And this is the reason for the infelicity of the discourse in (35): the co-indexation of the indefinite and the pronoun formally encodes that the pronoun should be 'bound' by the indefinite, i.e., as far as the speaker is concerned, the indefinite and the pronoun should be co-referent / anaphorically connected. However, the pronoun is actually 'unbound', i.e. independent of the individual non-deterministically made salient by the indefinite, since the pronoun ends up referring to some (arbitrary) individual that is already salient in the input discourse context – and this happens despite the fact that the speaker intended the pronoun to refer to the individual made salient by the indefinite.

Therefore, the hypothesis that proper names are by default salient in the input context (which underlies the representation of proper names as deictically used pronouns of some sort) correctly predicts that the anaphoric pronoun in (36) is felicitously used – while the indefinite-like analysis of proper names, according to which they introduce an unspecific dref to which subsequent pronouns are anaphoric to, makes incorrect predictions: we would expect anaphora to proper names introduced under negation to be infelicitous just as the corresponding anaphora to indefinites.

The very simple (and simplified²²) data presented above does not completely support either the indefinite-like or the pronoun-like analysis of proper names – and it is not the goal of the present investigation to settle these difficult matters²³. I will henceforth use the indefinite-like analysis only because it is more easily made compatible with the independently motivated formal developments in the following chapters – and the above discussion was only meant to lend some plausibility to this choice²⁴.

²² There are many other factors that can influence the accessibility of referents in discourse and that should be taken into account, e.g. information structure, epistemic specificity in the sense of Farkas (2002) etc.

²³ For a recent in-depth discussion of the linguistic and philosophical issues raised by the interpretation of proper names, see Cumming (2006).

²⁴ The development I have in mind is the van den Berg-style analysis of dynamic generalized quantifiers (see chapter 6 below), which requires the introduction of a dummy/'exception'/'undefined' individual # (# is

I will account for the felicitous anaphora in (36) above, which is problematic for the indefinite-like analysis of proper names, by assuming that pronouns can be indexed not only with unspecific drefs, but also with specific drefs like *Dobby* or *John*. That is, in addition to the anaphoric pattern in (29) above, I will also allow for the kind of connection between a pronoun and a proper name schematically represented in (37) below.

37. ... $John^u$... he_{John} ...

Strictly speaking, the pronoun is not co-referring with the proper name, i.e. the pronoun he_{John} is different from the co-indexed pronoun he_u as far as their context-change potentials go. However, the truth-conditional import of the two pronouns is the same in most cases; an exception is, of course, discourse (36) above, where only the pronoun he_{John} can account both for the felicity of the pseudo anaphoric connection and for the truth-conditions of the discourse. Sentence (36) (repeated in (38) below with the intended indexation) is analyzed as shown in (39). The reader can easily check that the representation in (39) delivers the intuitively correct truth-conditions.

38. Hermione ^{u_1} didn't see Dobby ^{u_2} in the Three Broomsticks.

He_{Dobby} was on vacation in the Bahamas.

39. $[u_1 \mid u_1 = Hermione, \sim[u_2 \mid u_2 = Dobby, see_in_TB\{u_1, u_2\}]];$
 $[on_vacation_in_Bahamas\{Dobby\}]$

a designated element of type e ; van den Berg's symbol is in fact \star). In certain contexts, we will need some 'assignments' i to assign this dummy individual to individual drefs, e.g. for some dref u , we will have $ui = \#$.

We will ultimately have to enforce a similar requirement with respect to proper names: the mention of a proper name, e.g. *John*, will be taken to simultaneously update some 'assignments' with the actual value of the proper name, e.g. $john_e$, and other assignments with the dummy value $\#$. Interpreting proper names directly in terms of specific drefs, i.e. in terms of constant functions, e.g. $John := \lambda i_s. john_e$, does not allow for the option of introducing a dref whose values are either the individual (rigidly) designated by the proper name or the dummy individual.

4. Syntax of a Fragment of English

Now that we have translated DPL in type logic, we can go compositional at the sub-sentential/sub-clausal level. For this purpose, I will define a basic transformational syntax for a small fragment of English in the tradition of Chomsky (1981). The definition is mostly based on Muskens (1996): 159-163 and Muskens (2005).

4.1. Indexation

"The most important requirement that we impose is that the syntactic component of the grammar assigns indices to all names, pronouns and determiners" (Muskens 1996: 159). Unlike Muskens (1996), I will let indices be specific and unspecific dref's (recall that they are all constants of type *se*), e.g. *u*, *u'*, *u_l*, *Dobby* etc. Just as before, the antecedents are indexed with superscripts and dependent elements with subscripts, following the convention in Barwise (1987).

I will also allow variables that have the appropriate dref type as indices on traces of movement, e.g. *v_{se}*, *v'_{se}*, *v_{0,se}*, *v_{l,se}* etc. – but such indices appear only on traces, *because they are needed only on traces*. As Muskens (1996): 169 puts it: "In Montague's PTQ (Montague 197[4]) the Quantifying-in rules served two purposes: (a) to obtain scope ambiguities between noun phrases and other scope bearing elements, such as noun phrases, negations and intensional contexts, and (b) to bind pronouns appearing in the expression that the noun phrase took scope over. In the present set-up the mechanism of discourse referents takes over the second task".

The fact that we use distinct indices for the two purposes enables us to keep track of when our indexation makes an essentially dynamic contribution to the semantics and when it is an artifact of the particular scoping mechanism and the particular syntax/semantics interface we employ. In this way, it will be fairly straightforward for the reader to reformulate the analyses we develop in her/his favorite syntactic formalism.

Thus, the choice of a particular (version of a particular) syntactic formalism is largely orthogonal to the matters addressed in the present work and is motivated only by presentational considerations: whichever syntactic formalism the reader favors, it is a

reasonable expectation that s/he will have at least a nodding acquaintance with the Y-model (a.k.a. the T-model) of GB syntax. The syntax-semantics interface defined in this section (which is no more than a proof of concept) is merely meant to give the reader a basic idea of how to design a proper 'interface' between the semantics proposed here and her/his favorite syntactic formalism.

4.2. Phrase Structure and Lexical Insertion Rules

The Y-model of syntax has four components: D-structure (DS), S-Structure (SS), Logical Form (LF) and Phonological Form (PF). We will be interested in the first three, in particular in the level of LF, which provides the input to the semantic interpretation procedure.

The DS component consists of all the trees that can be generated by the phrase structure rules PS1-PS12 and the lexical insertion rules LI1-LI11 in (40) below. We could in fact do away with rule PS1 (the necessary recursion is already built into PS2), but I will keep it as a reminder that sequencing two sentences in discourse occurs at a supra-sentential, textual level.

40. Phrase structure rules and lexical insertion rules²⁵.

(PS 1) $\text{T}_{\text{xt}} \rightarrow (\text{T}_{\text{xt}}) \text{CP}$	(PS 5) $\text{VP} \rightarrow \text{DP V}'$	(PS 9) $\text{V}_{\text{di}}' \rightarrow \text{V}_{\text{di}} \text{DP}$
(PS 2) $\text{CP} \rightarrow (\text{CP}) \text{IP}$	(PS 6) $\text{V}' \rightarrow \text{V}_{\text{in}}$	(PS 10) $\text{DP} \rightarrow \text{D NP}$
(PS 3) $\text{CP} \rightarrow \text{C IP}$	(PS 7) $\text{V}' \rightarrow \text{V}_{\text{tr}} \text{DP}$	(PS 11) $\text{NP} \rightarrow \text{N (CP)}$
(PS 4) $\text{IP} \rightarrow \text{I VP}$	(PS 8) $\text{V}' \rightarrow \text{V}_{\text{di}}' \text{DP}$	(PS 12) $\text{X} \rightarrow \text{X}^+ \text{Conj X}$
(LI 1) $\text{D} \rightarrow a^u, \text{every}^u, \text{most}^u, \text{few}^u, \text{no}^u, \text{some}^u, \text{any}^u, a^{u'}, \text{every}^{u'}, \dots$	(LI 5) $\text{N} \rightarrow \text{farmer, house-elf, donkey, } \dots$	(LI 9) $\text{I} \rightarrow \emptyset, \text{doesn't, don't, -ed, -s, didn't, } \dots$
(LI 2) $\text{DP} \rightarrow \text{he}_u, \text{she}_u, \text{it}_u, \text{he}_{u'}, \dots, \text{he}_{\text{John}}, \text{she}_{\text{Mary}}, \dots, \text{t}_v, \text{t}_{v'}, \dots$	(LI 6) $\text{V}_{\text{tr}} \rightarrow \text{own, beat, } \dots$	(LI 10) $\text{C} \rightarrow \text{if}$
(LI 3) $\text{DP} \rightarrow \text{John}^u, \text{Mary}^u, \text{John}^{u'}, \dots$	(LI 7) $\text{V}_{\text{in}} \rightarrow \text{sleep, walk, } \dots$	(LI 11) $\text{Conj} \rightarrow \text{and, or}$
(LI 4) $\text{DP} \rightarrow \text{who, whom, which}$	(LI 8) $\text{V}_{\text{di}} \rightarrow \text{buy, give, } \dots$	

²⁵ I am temporarily overloading the symbol ' \rightarrow ', which (as it is customary in the literature) is used to define the production rules of our grammar.

Subjects are assumed to be VP-internal and this is where they remain by default even at LF (they are raised out of VP only at PF). In this way, we can interpret sentential negation as having scope over quantifiers in subject position. Similarly, V-heads move to the inflectional I-head only at PF.

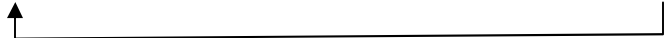
4.3. Relativization and Quantifier Raising

DS and SS are connected via the obligatory movement rule of Relativization (REL). A tree Θ' follows by REL from a tree Θ iff Θ' is the result of replacing some sub-tree of Θ of the form $[_{CP} [_{IP} X [_{DP} wh] Y]]$, where X and Y are (possibly empty) strings and wh is either *who*, *whom* or *which*, by a tree $[_{CP} [_{DP} wh]^v [_{CP} [_{IP} X t_v Y]]]$, where v is a fresh variable index (not occurring in Θ as a superscript). REL is basically CP adjunction.

41. *Relativization (REL)*: $[_{CP} [_{IP} X [_{DP} wh] Y]] \rightarrow [_{CP} [_{DP} wh]^v [_{CP} [_{IP} X t_v Y]]]$

For example, the DP *a^u girl who every^{u'} boy adores* has the syntactic representation in (42) below, obtained by an application of REL:

42. $[_{DP} a^u [_{NP} [_{N} girl] [_{CP} [_{DP} who]^v [_{CP} [_{IP} [I -s] [_{VP} [_{DP} every^{u'} boy] [_{V'} [_{V_{tr}} adore] t_v]]]]]]]]]]$



Formally, SS is the smallest set of trees that includes DS and is closed under REL; thus, $DS \subseteq SS$.

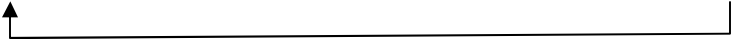
Finally, we turn to the definition of LF, the syntactic component that is the input to our semantics. This is the level where quantifier scope ambiguities are resolved. We define an optional rule of Quantifier Raising (QR) (May 1977) which adjoins DP's to IP's or DP's to VP's (we need VP-adjunction for ditransitive verbs among other things) and which is basically the Quantifying-In rule of Montague (1974).

A tree Θ' follows by QR from a tree Θ iff: **(a)** Θ' is the result of replacing some sub-tree Σ of Θ of the form $[_{IP} X [_{DP} Z] Y]$ by a tree $[_{IP} [_{DP} Z]^v [_{IP} X t_v Y]]$, where v is a fresh variable index (not occurring in Θ as a superscript); or **(b)** Θ' is the result of replacing some sub-tree Σ of Θ of the form $[_{VP} X [_{DP} Z] Y]$ by a tree $[_{VP} [_{DP} Z]^v [_{VP} X t_v Y]]$, where v is a fresh variable index (not occurring in Θ as a superscript). The conditions on the QR

rule are that Z is not a pronoun or a *wh*-word and that $[_{DP} Z]$ is not a proper sub-tree of a DP sub-tree $[_{DP} W]$ of Σ ²⁶.

43. *Quantifier Raising (QR)*: **a.** $[_{IP} X [_{DP} Z] Y] \rightarrow [_{IP} [_{DP} Z]^v [_{IP} X t_v Y]]$
b. $[_{VP} X [_{DP} Z] Y] \rightarrow [_{VP} [_{DP} Z]^v [_{VP} X t_v Y]]$

For example, the reverse scope of *every^u house-elf adores a^{u'} witch* can be obtained by QR to IP, as shown in (44) below (of course, it could also be obtained by QR to VP).

44. $[_{IP} [_{DP} a^{u'} witch]^v [_{IP} [_{I} -s] [_{VP} [_{DP} every^u house-elf] [_{V'} [_{V_{tr}} adore] t_v]]]]$
- 

LF is defined as the smallest set of trees that includes SS and is closed under QR; thus, $SS \subseteq LF$.

5. Type-driven Translation

In a Fregean / Montagovian framework, the compositional aspect of the interpretation is largely determined by the types for the 'saturated' expressions, i.e. names and sentences. Let's abbreviate them as **e** and **t**. An extensional static logic without pluralities (i.e. classical higher-order logic) is the simplest: **e** is *e* (atomic entities) and **t** is *t* (truth-values). The English verb *sleep*, for example, is represented by a term *sleep* of type (**et**), i.e. (*et*), and the generalized quantifier (GQ) *every man* by a term of type ((**et**)**t**), i.e. ((*et*)*t*).

This setup can be complicated in various ways²⁷. In particular, Dynamic Ty2 complicates it by adding another basic type *s* whose elements model DPL variable assignments, i.e. (simplified versions of) dynamic info states. A sentence is interpreted as

²⁶ For example, if the DP sub-tree $[_{DP} W]$ of Σ contains a relative clause which in its turn contains $[_{DP} Z]$, we do not want to QR $[_{DP} Z]$ all the way out of the relative clause.

²⁷ See for example Lewis (1972) and Creswell (1973), which use the same technique to introduce intensionality, i.e., in their case, $\mathbf{t} := st$ and *s* is the sort of indices of evaluation (however one wants to think of them, e.g. as worlds, <world, time> pairs etc.; see Muskens 1995a for a set of axioms that make the atomic objects of type *s* behave as <world, time> pairs).

a relation between an input and an output 'assignment', i.e. $\mathbf{t} := (s(st))$, and a name denotes an individual dref, i.e. $\mathbf{e} := (se)$ ²⁸.

The English verb *sleep* is still translated by a term of type (\mathbf{et}) , but now this means that it takes a dref u of type \mathbf{e} and it relates two info states i and i' of type s if and only if $i=i'$ and the entity denoted by u in info state i , i.e. ui , has the *sleep* property of type (\mathbf{et}) , i.e. the static 'sleep'-property.

5.1. Translating Basic Expressions

Table (45) below provides examples of basic meanings for the lexical items in (40) above: the first column contains the lexical item, the second column its Dynamic Ty2 translation and the third column its type, assuming the above two abbreviations, i.e. $\mathbf{t} := (s(st))$ and $\mathbf{e} := (se)$. Note that the abbreviated types have exactly the form we would expect them to have in Montague semantics (e.g. the translation of the intransitive verb *sleep* is of type \mathbf{et} , the translation of the pronoun *he* is of type $(\mathbf{et})\mathbf{t}$, the translations of the indefinite article *a* and of the determiner *every* are of type $(\mathbf{et})((\mathbf{et})\mathbf{t})$ etc.). The list of basic meanings constitutes rule **TR0** of our type-driven translation procedure for the English fragment.

Transitive verbs like *own* are assumed to take a generalized quantifier (GQ) as their direct object (type $(\mathbf{et})\mathbf{t}$), which captures the fact that the default quantifier scoping is subject over object. The reverse scope is obtained by QR.

Ditransitive verbs like *buy* are assumed to take two GQ's as objects; the default relative scope of the two (encoded in the lexical entry) is their left-to-right surface order, i.e. the first of them (e.g. the Dative GQ) takes scope over the second (e.g. the Accusative GQ). Arguably, this is the correct prediction, since the most salient quantifier scoping in

²⁸ Despite appearances, relativizing the interpretation of names to 'assignments' is not different from the Montagovian interpretation of names (or the Tarskian interpretation of individual constants in first-order logic): just as a name like 'John' is assigned the same individual, namely $john_e$, relative to any variable assignment g in a static Montagovian system, CDRT interprets proper names in terms of constant functions of type se , e.g. the semantic value of the name 'John' is given in terms of the constant function $John_{se}$ that maps each 'assignment' i_s to the individual $john_e$, i.e. $John_{se} := \lambda i_s. john_e$ (see also the discussion in section 0 above).

the sentence *Dobby bought every witch an alligator purse* follows the left-to-right linear order: the Dative GQ takes scope over the Accusative GQ, so that the purses co-vary with the witches. The reverse scope has to be obtained by QR (to VP or IP).

Note that the Dative GQ takes scope over the Accusative GQ despite their relative syntactic position: given the phrase structure rules **PS8** and **PS9** in (40) above, the Dative GQ is actually c-commanded by the Accusative GQ. The fact that a quantifier can take scope over another without c-commanding it syntactically is one of the advantages of working with a dynamic system, in which *quantifier scope is encoded in the order in which the updates are sequenced*.

Thus, in a dynamic framework, *syntactic structure affects quantifier scope only to the extent to which syntactic relations (e.g. c-command) are ultimately reflected in update sequencing*. The lexical entry for ditransitive verbs in (45) below 'neutralizes' syntactic c-command: it sequences the updates contributed by the two GQ objects according to their linear order and not according to their syntactic structure.

Defaulting to linear order (as opposed to syntactic c-command) has welcome empirical consequences in the case at hand: besides the fact that we capture the correlation between linear order and quantifier scope, we can also account for the fact that the Dative GQ is able to bind pronouns within the Accusative GQ without c-commanding them, as for example in *Dobby gave every^u witch her_u broom*.

It is in fact not unexpected that a dynamic system can account for pronoun binding without c-command given that donkey anaphora is a paradigmatic example of such binding without c-command. The point made by the present analysis of ditransitive verbs is that the dynamic account of donkey sentences can successfully generalize beyond the phenomena that provided the initial empirical motivation.

Pronouns of the form he_u and traces of the form t_v are interpreted as in Montague (1974), i.e. as the GQ-lift of their index, which, for pronouns, is a dref (i.e. a constant of type $e := se$) and, for traces, is an actual variable (again of type $e := se$). We will see in chapter 5 that this kind of 'lifted' interpretation for pronouns (coupled with the type-raised interpretation of transitive and ditransitive verbs) is not necessarily a 'worst case'

generalization, but it is actually motivated by the *distributive* interpretation of singular number morphology occurring on donkey pronouns.

Proper names are basically analyzed as indefinites – see the discussion at the very end of section 3 above, in particular (28). The only difference is that they are now translated as the corresponding GQ-lift.

Indefinites have the type of (dynamic) generalized determiners, as needed for the definition of the compositional interpretation procedure, but their crucial dynamic contribution is the introduction of a new dref, which has to satisfy the restrictor property and the nuclear scope property *in this order*. The DPL-style abbreviation explicitly exhibits the existential quantification built into the indefinite.

The universal quantifier *every* also has the type of generalized determiners and it is interpreted as expected: the DPL-style abbreviation speaks for itself. Note the square brackets surrounding the formula – they are due to the fact that, unlike the indefinite determiner *a*, the universal determiner *every* contributes a test – it is internally dynamic but *externally static*, just as classical DRT / FCS and DPL would have it.

The negative quantifier *no* also contributes a test; I provide its two equivalent translations, one of them based on negation and existential quantification, the other based on negation and universal quantification.

The *wh*-words that enter the construction of relative clauses are analyzed as identity functions over the property contributed by the relative clause. This property will then be 'sequenced' with the property contributed by the preceding common noun to yield a 'conjoined' property that is a suitable argument for a generalized determiner. The order in which the common noun and the relative clause are sequenced follows the linear surface order. The rule that achieves this dynamic 'conjunction' / 'sequencing' of properties generalizes both the static Predicate Modification rule in Heim & Kratzer (1998) and the dynamic Sequencing rule in Muskens (1996) – see (48) below.

45. TR 0: Basic Meanings (TN – Terminal Nodes).

Lexical Item	Translation	Type $e := se \quad t := s(st)$
$[sleep]_{V_{in}}$	$\rightsquigarrow \lambda v_e. [sleep_{et}\{v\}]$	et
$[own]_{V_{tr}}$	$\rightsquigarrow \lambda Q_{(et)t}. \lambda v_e. Q(\lambda v'_e. [own_{e(et)}\{v, v'\}])$	((et)t)(et)
$[buy]_{V_{di}}$	$\rightsquigarrow \lambda Q'_{(et)t}. \lambda Q_{(et)t}. \lambda v_e. Q'(\lambda v'_e. Q(\lambda v''_e. [buy_{e(et)}\{v, v', v''\}]))$	(ett)((ett)(et))
$[house-elf]_N$	$\rightsquigarrow \lambda v_e. [house_elf_{et}\{v\}]$	et
$[he_u]_{DP}$	$\rightsquigarrow \lambda P_{et}. P(u_e)$	(et)t
$[tv]_{DP}$	$\rightsquigarrow \lambda P_{et}. P(v_e)$	(et)t
$[he_{Dobby}]_{DP}$	$\rightsquigarrow \lambda P_{et}. P(Dobby_e)$	(et)t
$[Dobby^u]_{DP}$	$\rightsquigarrow \lambda P_{et}. [u \mid u=Dobby]; P(u)$	(et)t
$[a^u]_D$	$\rightsquigarrow \lambda P'_{et}. \lambda P_{et}. [u]; P'(u); P(u),$ i.e. $\lambda P'_{et}. \lambda P_{et}. \exists u(P'(u); P(u))$	(et)((et)t)
$[every^u]_D$	$\rightsquigarrow \lambda P'_{et}. \lambda P_{et}. [([u]; P'(u)) \rightarrow P(u)],$ i.e. $\lambda P'_{et}. \lambda P_{et}. [\forall u(P'(u) \rightarrow P(u))]$	(et)((et)t)
$[no^u]_D$	$\rightsquigarrow \lambda P'_{et}. \lambda P_{et}. [\sim([u]; P'(u); P(u))],$ i.e. $\lambda P'_{et}. \lambda P_{et}. [\sim \exists u(P'(u); P(u))]$ $\rightsquigarrow \lambda P'_{et}. \lambda P_{et}. [([u]; P'(u)) \rightarrow [\sim P(u)]],$ i.e. $\lambda P'_{et}. \lambda P_{et}. [\forall u(P'(u) \rightarrow [\sim P(u)])]$	(et)((et)t)
$[who]_{DP}$	$\rightsquigarrow \lambda P_{et}. P$	(et)(et)
$[\emptyset]_I / [-ed]_I / [-s]_I$	$\rightsquigarrow \lambda D_t. D$	tt
$[doesn't]_I / [didn't]_I$	$\rightsquigarrow \lambda D_t. [\sim D]$	tt
$[if]_C$	$\rightsquigarrow \lambda D'_t. \lambda D_t. [D' \rightarrow D]$	t(tt)
$[and]_{Conj}$	$\rightsquigarrow \lambda D'_t. \lambda D_t. D'; D$	t(tt)
$[or]_{Conj}$	$\rightsquigarrow \lambda D'_t. \lambda D_t. [D' \vee D]$	t(tt)

Non-negative inflectional heads are interpreted as identity functions over DRS meanings (type $\mathbf{t} := s(st)$). Negative inflectional heads are interpreted as expected: their value is a test, containing a condition that negates the argument DRS.

The conditional *if* is a binary DRS connective: it takes two DRS's as arguments and it returns a test containing a dynamic implication condition that relates the two argument DRS's.

The coordinating elements *and* and *or* will be discussed in more detail in section 5.1 of the following chapter (chapter 4); I provide here the entries for the simplest case, namely coordination of two sentences (i.e. DRS's).

5.2. Translating Complex Expressions

Based on **TR0**, we can obtain the translation of more complex LF structures by specifying how the translation of a mother node depends on the translations of its daughters. I provide five such rules, the last of which (**TR5: Coordination** – see (50) below) will be generalized in the following chapter.

The first rule covers non-branching nodes: the mother inherits the translation of the daughter.

46. **TR 1 – Non-branching Nodes (NN).**

If $A \rightsquigarrow \alpha$ and A is the only daughter of B,
then $B \rightsquigarrow \alpha$.

The second rule is functional application: the translation of the mother is the result of applying the translation of one daughter to the translation of the other.

47. **TR 2 – Functional Application (FA).**

If $A \rightsquigarrow \alpha$ and $B \rightsquigarrow \beta$ and A and B are the only daughters of C,
then $C \rightsquigarrow \alpha(\beta)$, provided that this is a well-formed term.

The third rule is a generalized sequencing (i.e. a generalized dynamic conjunction) rule. For one thing, it translates the meaning of complex texts (Txt) that are formed out of

a text (Txt) and a sentence (CP) – see **PS1** in (40) above. In this sense, it is a generalization of the Sequencing rule in Muskens (1996). But it also handles predicate modification in general, e.g. it translates the meaning of an NP that is formed out of a common noun N and a relative clause CP – see **PS11** in (40) above. In this sense, it is a generalization of the Predicate Modification rule in Heim & Kratzer (1998).

48. TR 3 – Generalized Sequencing (GSeq) (i.e. Sequencing + Predicate Modification).

If $A \rightsquigarrow \alpha$, $B \rightsquigarrow \beta$, A and B are the only daughters of C in that order (i.e. $C \rightarrow A B$) and α and β are of the same type τ of the form **t** or $(\sigma\mathbf{t})$ for some $\sigma \in \mathbf{Typ}$,
 then $C \rightsquigarrow \alpha; \beta$ if $\tau = \mathbf{t}$ or $C \rightsquigarrow \lambda_{\nu\sigma}. \alpha(\nu); \beta(\nu)$, if $\tau = (\sigma\mathbf{t})$,
 provided that this is a well-formed term.

The fourth rule handles Quantifying-In, both for quantifiers and for relativizers (i.e. *wh*-words).

49. TR 4 – Quantifying-In (QIn).

If $DP^v \rightsquigarrow \alpha$, $B \rightsquigarrow \beta$ and DP^v and B are daughters of C,
 then $C \rightsquigarrow \alpha(\lambda_{\nu}. \beta)$, provided that this is a well-formed term.

The final rule handles binary coordinations (it will be generalized to an arbitrary finite number of coordinated elements in the next chapter).

50. TR 5 – Coordination (Co).

If $A_1 \rightsquigarrow \alpha_1$, $\text{Conj} \rightsquigarrow \beta$, $A_2 \rightsquigarrow \alpha_2$ and A_1 , Conj and A_2 are the only daughters of A in that order (i.e. $A \rightarrow A_1 \text{Conj} A_2$),
 then $A \rightsquigarrow \beta(\alpha_1)(\alpha_2)$,
 provided this a well-formed term and has the same type as α_1 and α_2 .

The translation procedure, i.e. the relation 'tree Θ translates as term α ', is formally defined as the smallest relation \rightsquigarrow between trees and Dynamic Ty2 terms that is conform

to **TR0-TR5** and is closed under the reduction of the type-logical terms, e.g. if tree Θ *translates as* term α and term β follows from α by λ -conversion, then Θ *translates as* β .

6. Anaphora and Quantification in Compositional DRT (CDRT)

We are now ready to go through some examples. This section will show how CDRT can account for bound variable anaphora (6.1), quantifier scope ambiguities (6.2) and quantifier scope with ditransitive verbs (6.3). Finally, we will see how to analyze in CDRT the three paradigm examples that motivate dynamic semantics: cross-sentential anaphora (6.4), relative-clause donkey sentences (6.5) and, finally, conditional donkey sentences (6.6).

6.1. Bound Variable Anaphora

First, we can capture bound anaphora in CDRT without using the syntactic rule QR (Quantifier Raising, see (43) above) and the corresponding semantic rule QIn (Quantifying-In, see (49) above): we simply need the pronoun to be co-indexed with the antecedent, as shown in (51) below.

51. Every_{*u*} house-elf hates himself_{*u*}.

Co-indexation is enough for binding because binding in CDRT (just like in DPL) is actually taken care of by the *explicit (and, in this case, unselective) quantification over 'assignments'* built into the meaning of quantifiers. In classical static logic, the quantification over assignments is only implicit (and selective, but this is not directly relevant for the matter at hand): the paradigm example is λ -abstraction, which selectively quantifies over assignments that differ at most with respect to the variable that is abstracted over. Therefore, if we want to obtain bound variable anaphora in a static system, co-indexation, i.e. using the same variable, is not enough: we also need to create a suitable λ -abstraction configuration that will ensure the semantic co-variation via selective quantification over assignments.

Sentence (51) receives the Dynamic Ty2 representation in (52) below – or, equivalently, the one in (53). The formulas deliver the intuitively correct truth-conditions, as shown in (54).

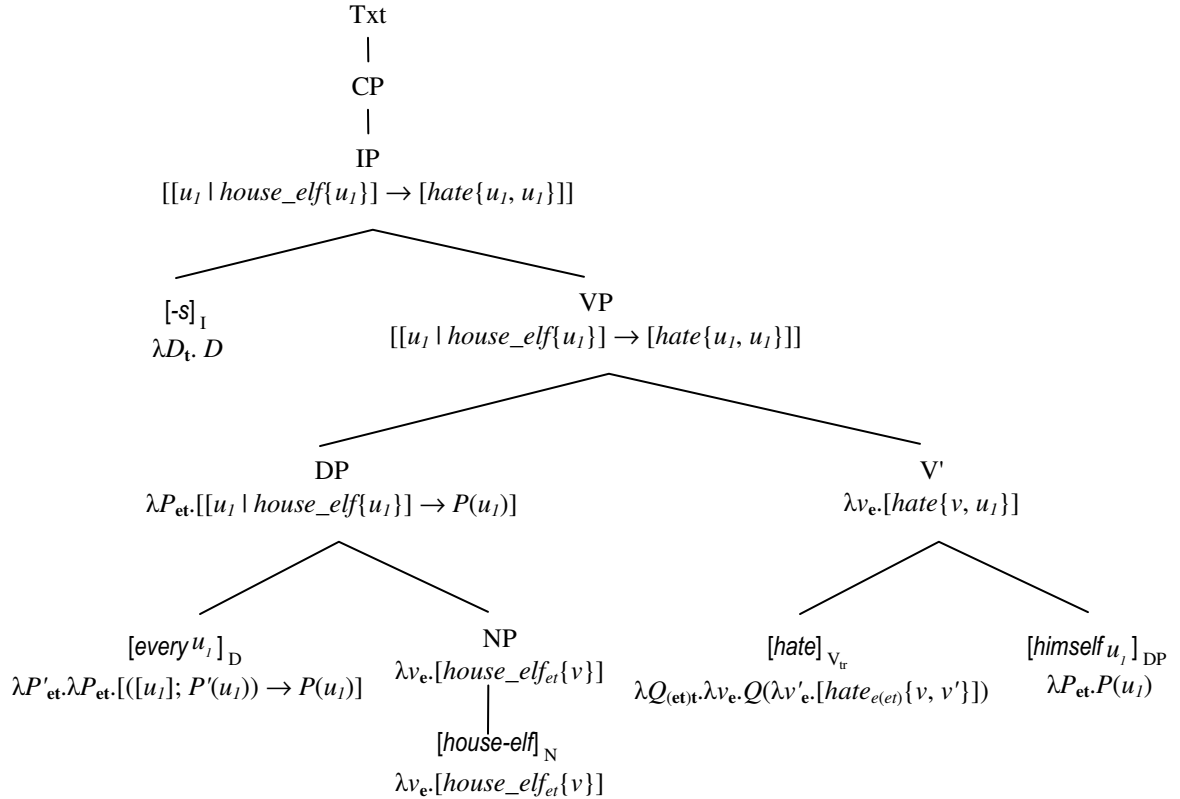
$$52. [[u_I \mid house_elf\{u_I\}] \rightarrow [hate\{u_I, u_I\}]]$$

$$53. [\forall u_I([house_elf\{u_I\}] \rightarrow [hate\{u_I, u_I\}])]$$

$$54. \lambda i_s. \forall x_e(house_elf(x) \rightarrow hate(x, x))$$

Most importantly, CDRT associates the correct Dynamic Ty2 translation with sentence (51) in a compositional way, as shown by the LF in (55) below, with the nodes in the tree decorated with their corresponding translations. I do not explicitly show what rules of type-driven translation are applied at various points in the calculation – the reader will have no difficulty identifying them. Note only that, by the NN rule for non-branching nodes (see (46) above), the translation of the topmost node Txt is the same as the translation of the IP node dominated by it.

55. Every^{*u_I*} house-elf hates himself^{*u_I*}.



Thus, we see that CDRT can compositionally account for bound anaphora in English without QR or QIn: co-indexation is enough for binding, since the basic meaning of the determiner *every* universally quantifies over assignments. This universal quantification can be ultimately traced back to dynamic negation – see the discussion of DPL universal quantification and dynamic implication in the previous chapter.

6.2. Quantifier Scope Ambiguities

We turn now to an application of QR and QIn. Consider the sentence in (56) below, which is ambiguous between two quantifier scopings: the surface-based scope $every^{u_1} >> a^{u_2}$ and the reverse scope $a^{u_2} >> every^{u_1}$. The reverse scope is obtained by an application of QR. The two LF's yield the translations in (57) and (59) below, which capture the intuitively correct truth-conditions for the two readings, as shown in (58) and (60).

56. Every ^{u_1} house-elf adores a ^{u_2} witch.

57. **every** ^{u_1} >> **a** ^{u_2} : $[[u_1 \mid house_elf\{u_1\}] \rightarrow [u_2 \mid witch\{u_2\}, adore\{u_1, u_2\}]]$

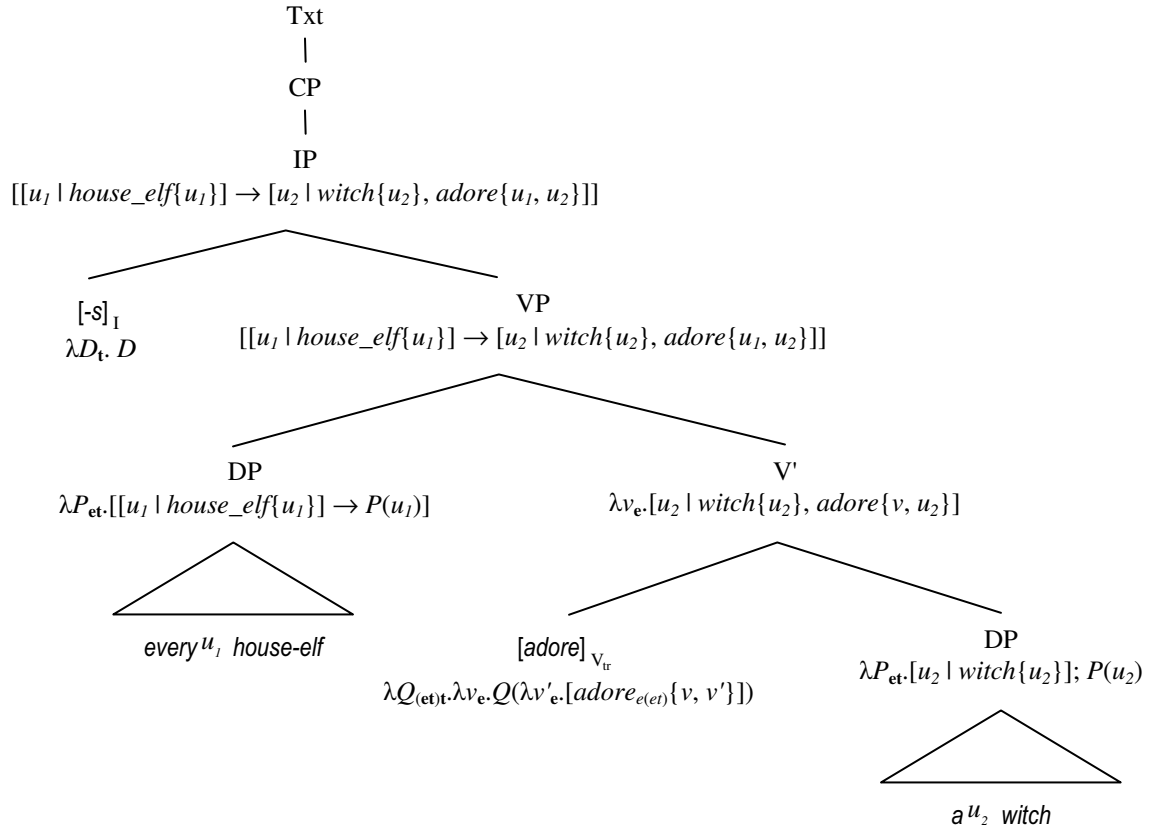
58. **every** ^{u_1} >> **a** ^{u_2} : $\lambda i_s. \forall x_e (house_elf(x) \rightarrow \exists y_e (witch(y) \wedge adore(x, y)))$

59. **a** ^{u_2} >> **every** ^{u_1} : $[u_2 \mid witch\{u_2\}, [u_1 \mid house_elf\{u_1\}] \rightarrow [adore\{u_1, u_2\}]]$

60. **a** ^{u_2} >> **every** ^{u_1} : $\lambda i_s. \exists y_e (witch(y) \wedge \forall x_e (house_elf(x) \rightarrow adore(x, y)))$

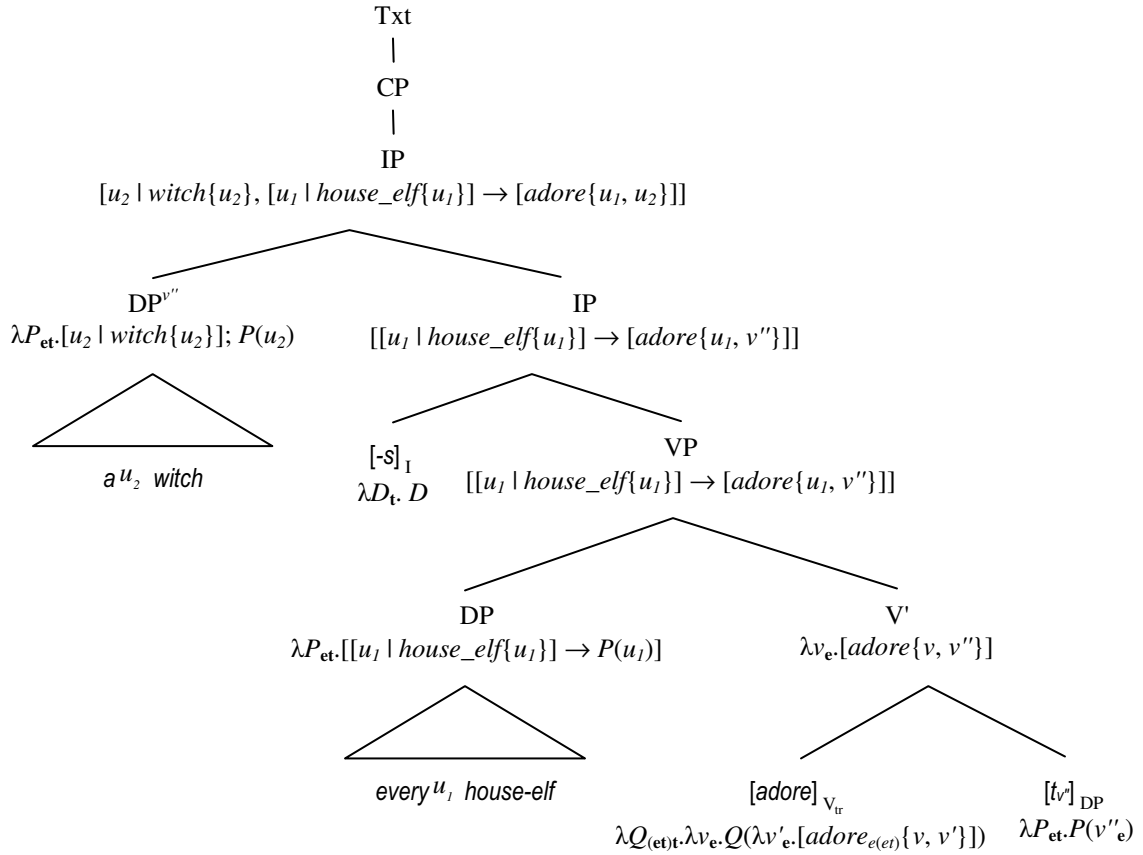
The two LF's are provided in (61) and (62) below.

61. **every**^{u₁} >> **a**^{u₂} : Every^{u₁} house-elf adores a^{u₂} witch.



The reverse scope is obtained by applying the QR rule to the indefinite DP *a^{u₂} witch*, as shown in (62) below. Note that the application of the QR rule yields the reverse scope not because it places the indefinite DP in a c-commanding position, *but because it reverses the order of updates*. From this perspective, having a syntactic level for quantifier scoping that encodes dominance in addition to linear precedence relations seems like overkill (see also the discussion of the ditransitive sentence in (63) below).

62. ($a^{u_2} \gg \text{every}^{u_1}$) Every u_1 house-elf adores a u_2 witch.



6.3. Quantifier Scope with Ditransitive Verbs

Given that donkey sentences with ditransitive verbs will feature quite prominently throughout the remainder of the dissertation, I will show in detail how sentences with ditransitive verbs are analyzed in CDRT. Consider the sentence in (63) below, in which the Dative GQ both takes scope over and binds into the Accusative GQ – without c-commanding it.

63. Dobby u_3 gave every u_1 witch her $_{u_1}$ alligator purse.

This example simultaneously exhibits two of the most interesting aspects of CDRT:

- we can have binding of pronouns without c-command and without QR, i.e. without the covert syntactic manipulations associated with the level of LF – see also (51) and (73) above;
- a quantifier can have wide scope over another without c-commanding it as long as the update it contributes is sequenced before the update of the other quantifier: the *lexical entry* for ditransitive verbs specifies that the Dative GQ update is sequenced / takes scope over the Accusative GQ update – and this is enough to nullify the fact that, *syntactically*, the former does not take scope over the latter.

Both features of CDRT point to the fact that the syntactic level of LF provides a needlessly rich, i.e. complex, input to the semantic interpretation procedure. In particular, the dominance relations that the LF level encodes are not (always) relevant for interpretation; the only two semantically relevant features of the LF level are: (i) the co-indexation of the referring expressions and (ii) the linear precedence (i.e. sequencing) of the updates²⁹.

Following the simplified LF for possessive DP's proposed in Heim & Kratzer (1998)³⁰, I analyze *her alligator purse* as the DP in (64) below³¹.

64. [DP a^{u_2} [NP [N *alligator purse*] [PP *of her_{u_1}*]]]

²⁹ I will not further pursue this perspective on meaning composition in the present work. Note however that coupling this perspective on meaning composition with the *plural* info states we will introduce in chapter 6 below (*plural* in the sense that the dynamic info states are sets of 'assignments' and not single 'assignments') promises to provide a novel and intuitively appealing analysis of cataphora on the one hand and the non-standard ('choice-function') scopal behavior of indefinites on the other hand (see for example Chierchia 1995: Chapter 3 for cataphora and Chierchia 2001 and references therein for 'choice-function' indefinites).

See also the online update of Bittner (2006), where said properties of CDRT (i.e. the fact that the only two semantically relevant features of the LF level are indexation and sequencing of updates) take center stage.

³⁰ Although the LF in (64) is similar to the one in Heim & Kratzer (1998), the analysis is different: while Heim & Kratzer (1998) take possessives to be covertly definite DP's (and adopt a Fregean analysis of definite descriptions), I analyze them here as covertly *indefinite* DP's. The indefinite analysis of possessive DP's is empirically supported by the interpretation of possessives in predicative positions, e.g. *John is her / Mary's brother*, which are not associated with uniqueness implications – I am grateful to Magdalena Schwager (p.c.) for bringing this to my attention.

³¹ I assume that the following phrase structure and lexical insertion rules are added to the syntax of our English fragment: (PS 13) NP → N (PP); (PS 14) PP → P DP; (LI 12) P → *of*.

The meaning of the preposition *of* is given in (65) below – it has the same structure as the lexical entry of a transitive verb like *own*.

$$65. [of]_P \rightsquigarrow \lambda Q_{ett}. \lambda v_e. Q(\lambda v'_e. [of_{e(et)}\{v, v'\}])$$

We compositionally derive the following translation for the DP in (64) (the subscript on the symbol \rightsquigarrow indicates the rule applied in translating the mother node):

$$66. a. [PP\ of\ her_{u_1}] \rightsquigarrow_{FA} \lambda v_e. [of\{v, u_1\}]$$

$$b. [NP\ [N\ alligator\ purse]\ [PP\ of\ her_{u_1}]] \rightsquigarrow_{GSeq} \lambda v_e. [alligator_purse\{v\}, of\{v, u_1\}]$$

$$c. (64) \rightsquigarrow_{FA} \lambda P_{et}. [u_2 \mid alligator_purse\{u_2\}, of\{u_2, u_1\}]; P(u_2)$$

The syntactic structure of the V' is provided in linearized form in (67) below and compositionally translated in (68). The Dative GQ $every^{u_1}$ *witch* takes scope over the Accusative GQ and also binds the pronoun her_{u_1} contained in it.

$$67. [v\ [_{Vdir}\ give\ every^{u_1}\ witch]\ [_{DP}\ a^{u_2}\ alligator\ purse\ of\ her_{u_1}]\]$$

$$68. [_{Vdir}\ give\ every^{u_1}\ witch]\ \rightsquigarrow_{FA}$$

$$\lambda Q_{(et)t}. \lambda v_e. [[u_1 \mid witch\{u_1\}] \rightarrow Q(\lambda v''_e. [give\{v, u_1, v''\}])]$$

$$(67) \rightsquigarrow_{FA} \lambda v_e. [[u_1 \mid witch\{u_1\}]$$

$$\rightarrow [u_2 \mid alligator_purse\{u_2\}, of\{u_2, u_1\}, give\{v, u_1, u_2\}]]$$

Thus, sentence (63) is translated as shown in (69) below and it receives the intuitively correct truth-conditions (for its most salient reading), as (70) below shows.

$$69. [u_3 \mid u_3=Dobby,$$

$$[u_1 \mid witch\{u_1\}] \rightarrow [u_2 \mid alligator_purse\{u_2\}, of\{u_2, u_1\}, give\{u_3, u_1, u_2\}]]$$

$$70. \lambda i_s. \exists z_e(z=dobby \wedge$$

$$\forall x_e(witch(x) \rightarrow \exists y_e(alligator_purse(y) \wedge of(y, x) \wedge give(z, x, y))), \text{ i.e.}$$

$$\lambda i_s. \forall x_e(witch(x) \rightarrow \exists y_e(alligator_purse(y) \wedge of(y, x) \wedge give(dobby, x, y)))$$

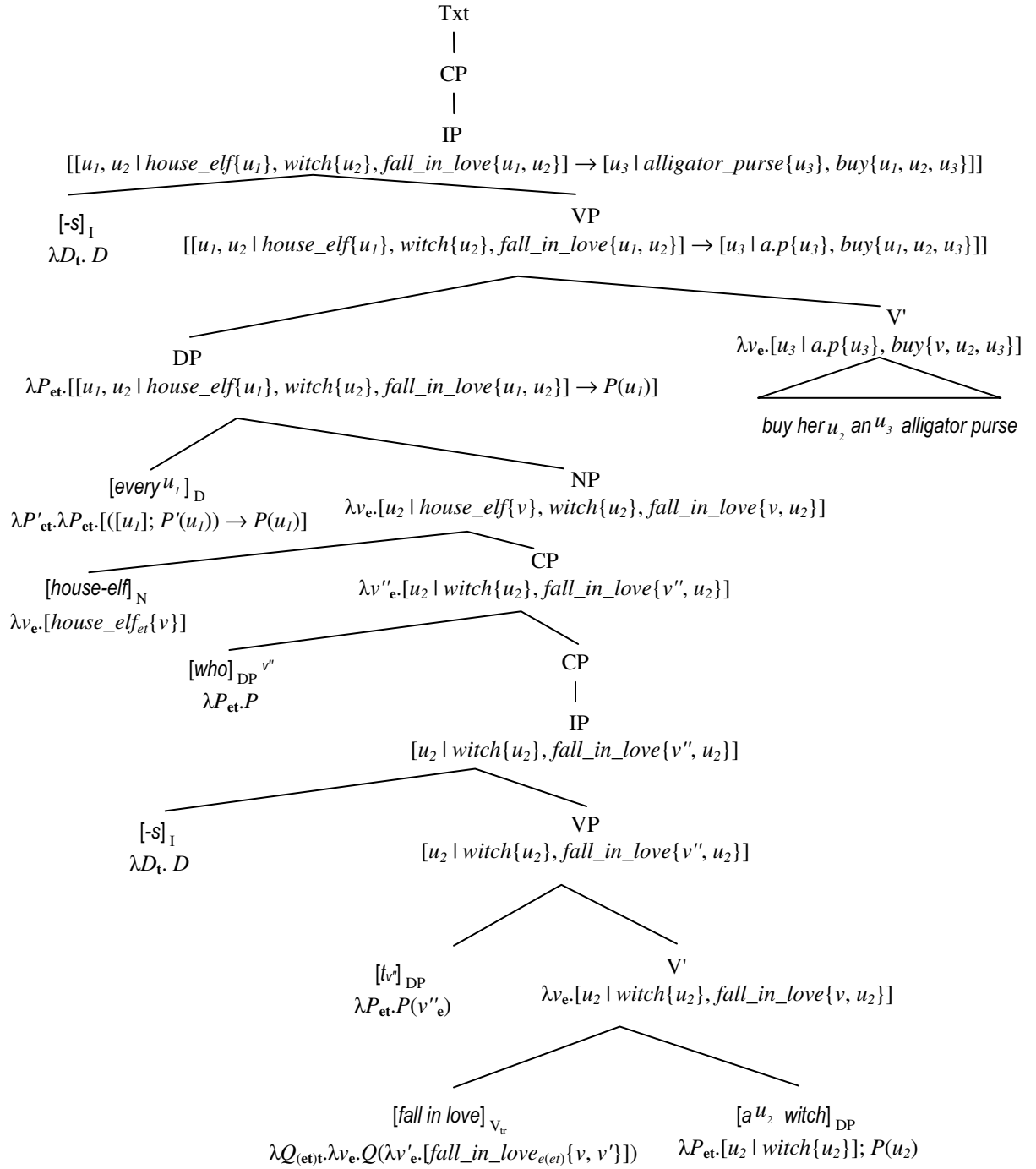
6.4. Cross-sentential Anaphora

We can also compositionally assign the intuitively correct interpretation to the three paradigm examples we have used in the previous chapter to motivate dynamic semantics. The examples are repeated in (71-72), (73) and (74) below; their LF's have two distinctive features: on the one hand, they put to use the previously otiose CP and Txt categories; on the other hand, they contain an application of the REL & QIn rules.

The analysis of donkey sentences exhibits a crucial property of CDRT we have already hinted at, namely the fact that, as long as these sentences receive the intuitively correct co-indexation, we can get the semantics of pronoun binding without c-command at the level of the LF.

6.5. Relative-clause Donkey Sentences

73. Every^{*u*₁} house-elf who falls in love with a^{*u*₂} witch buys her_{*u*₂} an^{*u*₃} alligator purse.



- to briefly recapitulate the basic empirical generalizations that motivate the dynamic approach to semantics and the basic kinds of semantic analyses that this approach makes possible.

The main achievement is the introduction of the basic compositional dynamic system couched in type logic in sections **2** and **5** above (i.e. the introduction of Dynamic Ty2 and CDRT).

The differences between the material introduced in this and the previous chapter and the cited sources are for the most part presentational. The six novel things are:

- the DPL-style definition of unselective generalized quantification that incorporates generalized quantifier *conservativity* (chapter **2**);
- the introduction of the mixed weak & strong donkey sentences, i.e. relative-clause donkey sentences with two donkey indefinites that receive different readings – one strong, the other weak –, e.g. *Every person who buys a book on amazon.com (strong) and has a credit card (weak) uses it (the credit card) to pay for it (the book)*; this kind of sentence cannot be accounted for in DRT / FCS / DPL or CDRT for that matter, even if they are extended with *selective* generalized quantification. Mixed weak & strong donkey sentences will provide one of the primary motivations for the subsequent revisions and generalizations of CDRT (see chapter **5**);
- the complete definition of the underlying Dynamic Ty2 logic (chapter **3**);
- the fact that Dynamic Ty2 allows static objects of *arbitrary* types as dref values (chapter **3**);
- the indefinite-like analysis of proper names adopted in the present version of CDRT (chapter **3**);
- the novel dynamic analysis of ditransitive verbs and of the scoping properties of their Dative and Accusative objects (chapter **3**).

Building on the foundations layed out in this chapter, the next chapter will add to the previous literature in a more substantial way by reformulating the DPL-style definitions of *unselective* and *selective* generalized quantification in type logic and, thus,

extending CDRT to CDRT+GQ in a way that enables it to account for the weak / strong donkey ambiguity and the proportion problem.