Presuppositions and Dynamic Montague Grammar

Emiel Krahmer*
ITK/KUB, PO Box 90153, 5000 LE, Tilburg

February 3, 1992

1 Introduction

The aim of this paper is a very modest one. I want to propose an extension of *Dynamic Montague Grammar* (DMG, see Groenendijk & Stokhof [2]), in which the space is created for incorporating presuppositions. Presuppositions cannot be handled in the version of MG that Groenendijk & Stokhof present, although in their article they hint at a possible solution. In this paper I will elaborate on that hint.

It is evident that creating a version of DMG in which we can incorporate an analysis of presuppositions is not the same as giving such an analysis. Unfortunately, it is not an entirely trivial task to give such an analysis in the extended version of DMG. As we shall see the definition of negation constitutes a problem; a presupposition cannot escape from it. At the end of this paper I will present a definition of negation which does not have this problem and roughly describe how it can be used to develop a full-fledged theory of presuppositions in a DMG-framework, including accommodation.

I will certainly not claim that such a theory will empirically do better than the best theory of presuppositions of the moment, which I take to be the *presuppositions-as-anaphora*-approach advocated by Van der Sandt [10] and Van der Sandt & Geurts [11]. Why undertake this exercise then? Why DMG as our framework? I do not want to be very dogmatic about this, but I think that although DMG has its weak points, at the very least its strict compositionality forces us to be very explicit about what we are doing and in this way we can hope to gain some insights in what is going on.

The structure of this paper is as follows: in section 2 we will take a closer look at Groenendijk & Stokhof's proposal. In paragraph 2.1 the representation language of Dynamic Montague Grammar is introduced: Dynamic Intensional Logic. This language is also the basis of the extended version of Dynamic Montague Grammar. In paragraph 2.2 DMG itself is presented and illustrated with some examples. In section 3 the extended version of DMG will be presented. We will illustrate with a simple example how this version can be used to treat presuppositions and point out some problems. Finally,

*Thanks are due to Paul Dekker, Bart Geurts, Jeroen Groenendijk, Reinhard Muskens, Martin Stokhof & Elias Thijsse. Every single one of the remaining errors I have made myself.

in section 4 we will try to overcome some of these problems.

2 Dynamic Montague Grammar

DMG was developed to give a fully compositional account of the semantics of discourse. Essentially, it differs from 'standard' Montague Grammar[1] in that the view on meaning is dynamic instead of static. What this means is that in Montague Grammar the meaning of a sentence resides in its truth-conditions, while in DMG the meaning of a sentence is its potential to change the context (i.e. the representation of the previous discourse).

The basic idea of DMG is very simple, and before we shall see in greater detail how it works, I will first illustrate it by means of a very simple example.

(1) A man is eating doughnuts.

Just like in traditional Montague Grammar we want to translate this sentence into a formula of a representation language, before we give it an interpretation. In the case of DMG the representation language is *Dynamic Intensional Logic* (DIL). We want to translate sentence (1) in a DIL-formula, and the resulting DIL-formula has to have a number of characteristics. First of all, we want to say that there is a discourse marker, say d, which denotes an object belonging to the set of men and to the set of doughnut-eating objects; that is to say: $\mathsf{man}(d) \wedge \mathsf{eat\_doughnut}(d)$. The existential quantifier which is part of the meaning of the indefinite determiner is just $\exists x$. We also need a link between the variable x and the discourse-marker d: this is fleshed out in the state-switcher $\{x/d\}$. This results in: $\exists x\{x/d\}(\mathsf{man}(d) \wedge \mathsf{eat\_doughnut}(d))$. Notice that this translation is very much like a static first-order representation, there are no signs that we are translating a discourse-fragment; there is still something missing. To make this formula a truly dynamic one we have to add a place –within the scope of the quantifier– where further sentences can land. Formally this is achieved in DMG by assigning each sentence α a translation of the form $\lambda p(\alpha' \wedge {}^{\vee}p)$, where p is variable of type $\langle s,t\rangle$. The result then is:

(2) $\lambda p \exists x\{x/d\}(\mathsf{man}(d) \wedge \mathsf{eat\_doughnut}(d) \wedge {}^{\vee}p)$

And this translation is indeed the DIL-translation of sentence (1). Now the state-switcher can do its work: it 'walks' through the formula and every time it hits upon a d, it replaces that particular occurrance of d with x. The state-switcher cannot be reduced any further when it hits the tail of the representation, the '${}^{\vee}p$'. So, the following formula is equivalent to (2):

(3) $\lambda p \exists x(\mathsf{man}(x) \wedge \mathsf{eat\_doughnut}(x) \wedge \{x/d\}{}^{\vee}p)$

The state-switcher remains there because it is possible that in a next sentence an anaphor occurs which refers back to the man walking on the beach; a case in point would be if we conjoin sentence (1) with sentence (4.a) (with (4.b) as its DIL-representation):

[1] Montague Grammar usually refers to the PTQ-model (Montague [7], see Gamut [1] for a thorough introduction).

(4) a. He smiles.

b. $\lambda q(\mathsf{smile}(d) \wedge {}^{\vee}q)$

The conjunction in DMG is defined is such a way that, when we utter (4.a) immediately after (2) the representation of the second sentence ends up on the hook for further sentences of the first one. The result is:

(5) $\lambda q \exists x(\mathsf{man}(x) \wedge \mathsf{eat\_doughnut}(x) \wedge \{x/d\}\mathsf{smile}(d) \wedge {}^{\vee}q)) =$
$\lambda q \exists x(\mathsf{man}(x) \wedge \mathsf{eat\_doughnut}(x) \wedge \mathsf{smile}(x) \wedge \{x/d\}{}^{\vee}q))$

Notice that this process can be repeated for a long time, just like we can add a large number of sentences following (2)-(4.a). One ingredient is still missing: how we can derive the static, 'ordinary' truth-conditions of a dynamic formula like (5)? We simply apply the DIL-formula to a tautological proposition (symbolized as **true**). The result of this is a formula of the form $\alpha' \wedge \mathbf{true}$, which is obviously equivalent to α', the standard PL-translation of α.

Now we will look in somewhat more detail how this works: first we shall look at DIL (paragraph 2.1), then we will see how DIL is used as the representation language of DMG in paragraph 2.2.

2.1 Dynamic Intensional Logic

The version of DIL that Groenendijk & Stokhof employ as the basis of DMG originates from Janssen [5], where in chapter 9 a Montague Grammar for the programming language ALGOL 68 is developed.

The types of DIL are the same as those of *Intensional Logic*, the representation language of Montague Grammar. The syntax of DIL is also the same as that of IL, although DIL-syntax differs from IL-syntax in two respects. First, DIL contains two new concepts: the *discourse markers* and the ***state-switchers***. Their syntactic behavior is, as usual, defined in terms of the set of meaningful expressions of type a, ME_a. Given sets of constants and variables for every type a (respectively CON_a and VAR_a) and a set of discourse markers DM, the new concepts are defined as follows:

Definition 1 ((new parts of the) DIL-Syntax)

1. If $\alpha \in DM$, then $\alpha \in ME_e$
2. If $d \in DM, \alpha \in ME_e, \beta \in ME_a$, then $\{\alpha/d\}\beta \in ME_a$

As can be seen from the definition only discourse-markers of type e are used. The state-switchers make it possible to identify certain states in which discourse markers have a particular value, by switching from the present state to a new one. A state can be viewed as a discourse marker assignment and so state-switchers are in fact a sort of descriptions of states.

The second difference between IL and DIL is that the operators '$^{\vee}$' and '$^{\wedge}$' are present in DIL, but will in this context be seen to express abstraction over, and application to states respectively. So the intensional vocabulary is used in DIL, but with another meaning than in IL.

Let us now direct our attention to the semantics of DIL. A model M is a triple $\langle D, S, F\rangle$, with D and S as two disjunct, non-empty sets of individuals and states as their respective elements, and F as a function which interprets the constants of the language. So if $\alpha \in CON_a$, then $F(\alpha) \in D_a$. The definition of these DIL domains is just as one would expect them to be, except of course that we use S here as the set of states. Groenendijk & Stokhof force the states to behave like discourse marker assignments, by means of two postulates: *distinctness* and *update*. Together these two postulates guarantee that for each state s, discourse marker d and object $\mathbf{d}$, there exists a unique state s' which differs from s at most in this respect that the denotation of d in s' is $\mathbf{d}$. The notation used for this unique s' is $\langle d \leftarrow \mathbf{d}\rangle s$. Notice the similarity between this notation and the familiar $g[v/\mathbf{d}]$: states behave as assignments of values to discourse markers.

The semantics is defined by means of the notion $[\![\alpha]\!]_{M,s,g}$, which stands for the interpretation of α with respect to model M, state s and assignment g.

Definition 2 (DIL-semantics)

1. $[\![c]\!]_{M,s,g} = F(c)$, for every constant c
   $[\![v]\!]_{M,s,g} = g(v)$, for every variable v
2. $[\![d]\!]_{M,s,g} = F(d)(s)$, for every discourse marker d
3. $[\![\alpha(\beta)]\!]_{M,s,g} = [\![\alpha]\!]_{M,s,g}([\![\beta]\!]_{M,s,g})$
4. $[\![\neg\varphi]\!]_{M,s,g} = 1$ iff $[\![\varphi]\!]_{M,s,g} = 0$
   $[\![\varphi \wedge \psi]\!]_{M,s,g} = 1$ iff $[\![\varphi]\!]_{M,s,g} = [\![\psi]\!]_{M,s,g} = 1$
5. $[\![\exists v\varphi]\!]_{M,s,g} = 1$ iff there is a $\mathbf{d} \in D_a$ such that $[\![\varphi]\!]_{M,s,g[v/\mathbf{d}]} = 1$, where the type of v is a
6. $[\![\lambda v\alpha]\!]_{M,s,g}$ = that function $h \in {D_a}^{D_b}$ such that $h(\mathbf{d}) = [\![\alpha]\!]_{M,s,g[v/\mathbf{d}]}$ for every $\mathbf{d} \in D_b$, where the type of α is a and the type of v is b
7. $[\![\{\alpha/d\}\beta]\!]_{M,s,g} = [\![\beta]\!]_{M,\langle d \leftarrow [\![\alpha]\!]_{M,s,g}\rangle s,g}$
8. $[\![^\wedge\alpha]\!]_{M,s,g}$ = that function $h \in {D_a}^S$ such that $h(s') = [\![\alpha]\!]_{M,s',g}$ for every $s' \in S$, where the type of α is a
9. $[\![^\vee\alpha]\!]_{M,s,g} = [\![\alpha]\!]_{M,s,g}(s)$

Several things are worth noticing about this definition. First of all we see in clause 2 that the interpretation of discourse markers is state-dependent. They are interpreted as 'individual concepts', that is functions from S to D_e. Clause 8 presents the interpretation of the state-switcher: the interpretation of $\{\alpha/d\}\beta$ in a given state s is the interpretation of β is a new state s' which differs from s at most in this respect that the value of d in s' is the object which is the denotation of α in s. From clause 9 and 10 we see, as

was announced above, that $^{\vee}\alpha$ and $^{\wedge}\alpha$ are interpreted as application and abstraction over states respectively.

The notions of truth, validity and entailment are defined in the usual way: (truth) φ is *true* w.r.t. M, s, and g iff $[\![\varphi]\!]_{M,s,g}$, (validity) $\models \varphi$ iff φ is true for all M, s and g, and (entailment) $\varphi \models \psi$ iff for all M, s and g it holds that if $[\![\varphi]\!]_{M,s,g} = 1$, then $[\![\psi]\!]_{M,s,g} = 1$.

The behavior of state-switchers is characterized in the following fact:

Fact 1 (Behavior of state-switchers)

1. If β is a constant, a variable or an expression of the form $^{\wedge}\alpha$, then $\{\alpha/d\}\beta$ is equivalent to β
2. $\{\alpha/d\}d$ is equivalent to α
   $\{\alpha/d\}d'$ is equivalent to d', if $d' \neq d$
3. $\{\alpha/d\}(\beta(\gamma))$ is equivalent to $\{\alpha/d\}\beta(\{\alpha/d\}(\gamma))$
4. $\{\alpha/d\}(\varphi \wedge \psi)$ is equivalent to $\{\alpha/d\}\varphi \wedge \{\alpha/d\}\psi$ (similarly for '$\vee$', '$\rightarrow$', '$=$' and '$\neg$')
5. $\{\alpha/d\}\exists v\varphi$ is equivalent to $\exists v\{\alpha/d\}\varphi$, provided that v does not occur freely in α (similarly for $\forall v\varphi$ and $\lambda v\varphi$)
6. $\{\alpha/d\}^{\vee}\beta$ is *not* equivalent to $^{\vee}\beta$

This last item of the definitions is of prime-importance for the following as we shall see. Finally, we note that the following familiar facts also hold in DIL:

Fact 2 ($^{\vee}{}^{\wedge}$-elimination) $^{\vee}{}^{\wedge}\alpha$ is equivalent to α

Fact 3 (λ-conversion) $\lambda v\alpha(\beta)$ is equivalent to $[\beta/v]\alpha$ if:

1. all free variables in β are free for v in α, and
2. β is a constant, a variable or an expression of the form $^{\wedge}\alpha$ (that is: β in an *intensionaly closed expression*), or there are no occurrences of v in α in the scope of a '$^{\wedge}$' or a state-switcher $\{\gamma/d\}$

Some notation: when two DIL expressions α and β are equivalent module $^{\vee}{}^{\wedge}$-elimination we write this is as $\alpha =_{\vee\wedge} \beta$. When α reduces to β by means of a (finite) number of λ-conversions we write this as $\alpha =_{\lambda} \beta$.

Before we show how DMG makes use of DIL it is worth pointing out that the use of discourse markers apart form 'ordinary' variables is not crucial for Groenendijk & Stokhof's analysis. The only reason why Groenendijk & Stokhof make use of them is because they make it more clear what is going on. In fact, in Groenendijk & Stokhof [2] (page 14) it is shown that the addition of discourse markers and state-switchers is in fact nothing else than the addition of a second sort 'variable-and-binding device'.

2.2 Dynamic Montague Grammar

Now we will construe a dynamic version of MG, making use of DIL: *Dynamic Montague Grammar* (DMG). In DMG, as in classical MG, natural language expressions are translated in a logical language (DIL) which is associated with a model-theoretic interpretation. The meaning of complex constituents is determined in a compositional fashion by the meaning of the respective parts and the operations which have formed the complex constituent. The use of the DIL-formulas in DMG is facilitated by the introduction of a number of notation-conventions, which we shall introduce in a minute.

First, however a general remark concerning the notion of meaning in DMG. In classical MG an (indicative) sentence is translated into an IL-expression of type t: the extension of a sentence is a truth-value, the intension a proposition. In other words: in MG –as one would expect– we find the *static* view on meaning. Needless to say, this is not the view on meaning we would expect in DMG. And indeed, in DMG we find the *dynamic* view on meaning. This has some consequences for the type of the DIL-translation for a sentence. A natural language sentence is translated into a DIL-formula of type $\langle\langle s,t\rangle,t\rangle$. In the following we use the abbreviation τ for this type. The meaning of a sentence is an object of type $\langle s,\tau\rangle$, i.e. a function from states to sets of propositions. The standard, static meaning still resides in DMG, but only as a secondary notion.

Let us now look in more detail how this is formalized in DMG. In what follows we will use uppercase Greek letters as meta-variables for formulas of type τ and lowercase Greek letters as meta-variables for formulas of type t.

The first notation-convention of the DMG-language is the uparrow: $\uparrow$.

Definition 3 (uparrow) $\uparrow\varphi = \lambda p(\varphi \wedge {}^{\vee}p)$, where φ is a formula of type t, and p is a variable of type $\langle s,t\rangle$ with no free occurrences in φ

Basically, what the $\uparrow\varphi$ does is making place for possible continuations of φ, the ${}^{\vee}p$ figures as the 'hook' on which further pieces of discourse can be hanged. If φ is true in some state s, then $\uparrow\varphi$ denotes the set of all true propositions in s. And when φ is false in s, $\uparrow\varphi$ denotes the empty set.

There is also a converse operation downarrow: $\downarrow$:

Definition 4 (downarrow) $\downarrow\Phi = \Phi(\hat{}\mathbf{true})$

The expressions Φ of type τ are typically the kind of formulas into which natural language sentences will be translated. This is the level on which the dynamic bindings are created. **True** is a constant for which the following holds: $F(\mathbf{true}) = 1$. So, as we have seen above, what the $\downarrow$-operator does is free a formula of its dynamic burden and deliver its standard static truth conditions. The formula $\downarrow\Phi$ is true in some state s iff Φ can be processed successfully in s.

Notice that the following fact holds:

Fact 4 ($\downarrow\uparrow$-elimination)$\downarrow\uparrow\varphi = \varphi$

Just as ${}^{\wedge\vee}\varphi = \varphi$ doesn't hold in general, $\uparrow\downarrow\Phi = \Phi$ does not either. Especially

when Φ still has dynamic potential (i.e. active quantifiers), this does not hold. Basicly $\uparrow\downarrow\Phi$ can be seen as the static closure of Φ. Whenever you want to strip a formula of its dynamic potential but still want it to be followed by new sentences, it can be closed of with $\uparrow\downarrow$. A context in which this is desirable is for instance *negation*. Consider this mini-discourse:

(6) It is not the case that a_1 man eats a_2 doughnut. *He$_1$ likes it$_2$.

We do not want the quantifiers associated with a_1 *man* and a_2 *doughnut* to bind the a variable corresponding with a pronoun in the second sentence. This is achieved by defining negation in terms of static closure:

Definition 5 (Static negation) $\sim\Phi = \uparrow\neg\downarrow\Phi$

Besides negation there are two more basic DMG-operators, viz. the existential quantifier and the conjunction, both dynamic:

Definition 6 (Existential quantifier and conjunction)

1. $\mathcal{E}d\Phi = \lambda p\exists x\{x/d\}(\Phi(p))$, where p has no free occurrences in Φ.
2. $\Phi\,;\Psi = \lambda p(\Phi(\wedge(\Psi(p))))$, where p has no free occurrences in either Φ or Ψ.

Notice that the conjunction is strictly linear, that is to say it is both *associative* (i.e $(\Phi\,;\Psi)\,;\Upsilon = \Phi\,;(\Psi\,;\Upsilon)$) and *non-commutative* (i.e. $\Phi\,;\Psi \neq \Psi\,;\Phi$). An important fact for the translation of discourse-fragments is the following:

Fact 5 $\mathcal{E}d\Phi\,;\Psi = \mathcal{E}d(\Phi\,;\Psi)$

This fact guarantees that a quantifier (*a man*) can bind variables (*he*) which are not in its syntactic scope.

The other DMG-notions can, in the usual fashion, be defined in terms of conjunction, negation and existential quantification:

Definition 7 (Other DMG notions)

1. $\Phi \Rightarrow \Psi = \sim(\Phi\,;\sim\Psi)$
2. Φ **or** $\Psi = \sim(\sim\Phi\,;\sim\Psi)$
3. $\mathcal{A}d\Phi = \sim\mathcal{E}d\sim\Phi$

The implication is internally dynamic, because the antecedent is not closed of under negation. The other two operators do not have any dynamic potential.

Another important fact for the analysis of 'donkey'-sentences is the following, which takes care of the universal reading in implications.

Fact 6 $\mathcal{E}d\Phi \Rightarrow \Psi = \mathcal{A}d(\Phi \Rightarrow \Psi)$

The next fact makes it easier to replace dynamic operators for their static counter-parts:

Fact 7 (From dynamic operators to static ones)

1. $\sim\uparrow\varphi = \neg\uparrow\varphi$
2. $(\uparrow\varphi\,;\uparrow\psi) = \uparrow(\varphi \wedge \psi)$
3. $(\uparrow\varphi \textbf{ or } \uparrow\psi) = \uparrow(\varphi \vee \psi)$
4. $(\uparrow\varphi \Rightarrow \uparrow\psi) = \uparrow(\varphi \rightarrow \psi)$

Now we have all the tools needed for a small fragment of English. The fragment has as basic categories IV (intransitive verbs), CN (common nouns) and S ((sequences of) sentences). The derived categories are of the form A/B, A and B any category. Used here are: NP (= S/IV, noun phrases), DET (= NP/CN, determiners) and TV, (= IV/NP, transitive verbs).

In what follows, x is a variable of type e, P and Q are variables of type $\langle s,\langle e,\tau\rangle\rangle$ and T of type $\langle s,\langle\langle s,\langle e,\tau\rangle\rangle,\tau\rangle\rangle$; *leland* is a constant of type e, doughnut and smile are constants of type $\langle e,t\rangle$ and eat (in its transitive form) of type $\langle e,\langle e,t\rangle\rangle$; the d_n's are discourse markers. In the table below some examples of lexical items are listed:

Definition 8 (Lexical items)

(DET)	a_n	$\rightsquigarrow$	$\lambda P\lambda Q\mathcal{E}d_n({}^\vee P(d_n)\,;{}^\vee Q(d_n))$
(DET)	$every_n$	$\rightsquigarrow$	$\lambda P\lambda Q\mathcal{A}d_n({}^\vee P(d_n)\Rightarrow{}^\vee Q(d_n))$
(CN)	*doughnut*	$\rightsquigarrow$	$\lambda x\uparrow\mathsf{doughnut}(x)$
(IV)	*smile*	$\rightsquigarrow$	$\lambda x\uparrow\mathsf{smile}(x)$
(TV)	*eat*	$\rightsquigarrow$	$\lambda T\lambda x({}^\vee T({}^\wedge\lambda y\uparrow\mathsf{eat}(x,y)))$
(NP)	he_n	$\rightsquigarrow$	$\lambda Q{}^\vee Q(d_n)$
(NP)	$Leland_n$	$\rightsquigarrow$	$\lambda Q\{leland/d_n\}{}^\vee Q(d_n)$

The thing to notice about these definitions is that determiners, pronouns and proper names are indexed. We need to chose a particular discourse marker in the translation, otherwise we are not able to make the anaphoric link. In other words, the fragment does not translate English sentences as such, but rather indexed structures.

The fragment has the following construction rules:

Definition 9 (Construction rules)

1. Functional application: $\alpha_{A/B}\beta_B \rightsquigarrow \alpha'({}^\wedge\beta')$
2. Sentence sequencing: $\alpha.\beta \rightsquigarrow \alpha'\,;\beta'$
3. Conditional sentences: *If* α, *(then)* $\beta \rightsquigarrow \alpha' \Rightarrow \beta'$
4. Sentence negation: *It is not the case that* $\alpha \rightsquigarrow \sim\alpha'$

To end this section I would like to illustrate DMG with two examples.

A_1 man sees a_2 doughnut. He_1 eats it_2.
Functional applications and some reductions give the following translation of this example.

(7) $\mathcal{E}d_1(\uparrow\mathsf{man}(d_1)\,;\,\mathcal{E}d_2(\uparrow\mathsf{doughnut}(d_2)\,;\,\uparrow\mathsf{see}(d_1,d_2)))\,;\,\uparrow\mathsf{eat}(d_1,d_2)$

Notice that we can lift the existential quantifier corresponding to a_2 *doughnut* since that doesn't create new bindings.

(8) $\mathcal{E}d_1\mathcal{E}d_2(\uparrow\mathsf{man}(d_1)\,;\,\uparrow\mathsf{doughnut}(d_2)\,;\,\uparrow\mathsf{see}(d_1,d_2))\,;\,\uparrow\mathsf{eat}(d_1,d_2) =_{\text{fact 7.3}}$
$\mathcal{E}d_1\mathcal{E}d_2\uparrow(\mathsf{man}(d_1)\wedge\mathsf{doughnut}(d_2)\wedge\mathsf{see}(d_1,d_2))\,;\,\uparrow\mathsf{eat}(d_1,d_2)$

Writing out the definitions of the existential quantifier (def. 6.2) and the uparrow (def. 3) and some reductions, this gives us:

(9) $\lambda p\exists x\exists y(\mathsf{man}(x)\wedge\mathsf{doughnut}(y)\wedge\mathsf{see}(x,y)\wedge\{x/d_1\}\{y/d_2\}{}^{\vee}p)\,;$
$\lambda p(\mathsf{eat}(d_1,d_2)\wedge{}^{\vee}p)$

Now we write out the dynamic conjunction of the two sentences (def. 6.1):

(10) $\lambda r(\lambda p\exists x\exists y(\mathsf{man}(x)\wedge\mathsf{doughnut}(y)\wedge\mathsf{see}(x,y)\wedge\{x/d_1\}\{y/d_2\}{}^{\vee}p)$
$({}^{\wedge}(\mathsf{eat}(d_1,d_2)\wedge{}^{\vee}r))) =_{\lambda,\vee\wedge,\text{fact 1}}$
$\lambda r\exists x\exists y(\mathsf{man}(x)\wedge\mathsf{doughnut}(y)\wedge\mathsf{see}(x,y)\wedge\mathsf{eat}(x,y)\wedge\{x/d_1\}\{y/d_2\}{}^{\vee}r)$

When we apply the $\downarrow$-operator to this last formula we end up with the standard truth conditions:

(11) $\exists x\exists y(\mathsf{man}(x)\wedge\mathsf{doughnut}(y)\wedge\mathsf{see}(x,y)\wedge\mathsf{eat}(x,y))$

If a man_1 sees a_2 doughnut, he_1 eats it_2.
Next we translate the 'donkey'-sentence. We translate the antecedent and the consequent and combine them by means of the construction rule 9.3:

(12) $\lambda p\exists x\exists y(\mathsf{man}(x)\wedge\mathsf{doughnut}(y)\wedge\mathsf{see}(x,y)\wedge\{x/d_1\}\{y/d_2\}{}^{\vee}p)\Rightarrow$
$\lambda p(\mathsf{eat}(d_1,d_2)\wedge{}^{\vee}p) =_{\text{def. 7.1}}$
$\sim(\lambda p\exists x\exists y(\mathsf{man}(x)\wedge\mathsf{doughnut}(y)\wedge\mathsf{see}(x,y)\wedge\{x/d_1\}\{y/d_2\}{}^{\vee}p)\,;$
$\sim(\lambda p(\mathsf{eat}(d_1,d_2)\wedge{}^{\vee}p)))$

As far as the second conjunct is concerned: $\sim\uparrow\varphi$ by fact 7.2 equals $\uparrow\neg\varphi$. Next we eliminate the dynamic conjunction, which after two λ-conversion and further pushing of state-switchers results in:

(13) $\sim(\lambda r\exists x\exists y(\mathsf{man}(x)\wedge\mathsf{doughnut}(y)\wedge\mathsf{see}(x,y)\wedge\neg\mathsf{eat}(x,y)\wedge$
$\{x/d_1\}\{y/d_2\}{}^{\vee}r))$

Writing out the outermost negation, this eventually results in:

(14) $\lambda q\neg(\exists x\exists y(\mathsf{man}(x)\wedge\mathsf{doughnut}(y)\wedge\mathsf{see}(x,y)\wedge\neg\mathsf{eat}(x,y))\wedge{}^{\vee}q)$

Notice that this translation is still of type τ, but further sentences will no longer land *inside* the scope of the quantifiers. This explains why this is not a grammatical discourse:

(15) If a_1 man sees a_2 doughnut, he_1 eats it_2. *He_1 really loves doughnuts, particularly with black coffee.

The pronoun *he_1* cannot be bound by the relevant quantifier.

Applying the $\downarrow$-operator to the formula in (14), we get the standard truth-conditions:

(16) $\neg(\exists x \exists y(\mathsf{man}(x) \wedge \mathsf{doughnut}(y) \wedge \mathsf{see}(x,y) \wedge \neg\mathsf{eat}(x,y))) =_{\text{equivalent}}$
$\forall x \forall y((\mathsf{man}(x) \wedge \mathsf{doughnut}(y) \wedge \mathsf{see}(x,y)) \rightarrow \mathsf{eat}(x,y))$

3 Presuppositions in DMG

In the previous section we have seen how DMG succeeds in giving a strictly compositional account of discourse semantics. In this section we will see how we can extend DMG in such a way that presuppositions can be incorporated in it. I will restrict myself here to presuppositions triggered by definite NPs. All definite NPs have the property of presupposing their descriptive content, so a proper name like *Laura Palmer* presupposes that there exists someone with that name, a definite description like *the doughnut* presupposes that there exists a doughnut etc. Take the following example:

(17) The doughnut is delicious .

This sentence presuppose that 'there exists a doughnut'. What does it mean for a sentence to presuppose something? One possible answer to this question is that a presupposition is a kind of test; when we encounter a presupposition like 'there exists a doughnut', we test if the context (which we will simplify to the representation of the previous discourse) satisfies this presupposition. If this is the case we simply continue, if not, we say that the sentence containing the trigger of the relevant presupposition (in the case of example (17), the definite determiner) is *infelicitous*. Note that this view on presuppositions is not the only possible one. In fact, in section 4 we will discuss a somewhat different view on presuppositions. But for now, suppose that we want to give an account of presuppositions which tests the context for presupposition-satisfaction and only continues if this test succeeds. We cannot express that in the version of DMG which we saw in the previous section. The reason for this is simple; when we translate a sentence into a DIL-formula of type τ, only further sentences are anticipated, the previous discourse is not taken into consideration. However, if we want to formulate a test to check whether or not a context satisfies a certain presupposition we dò need the previous discourse.

Basically this problem follows from the simplification Groenendijk & Stokhof make concerning their notion of meaning. As we have seen they refer to the meaning of a sentence as a function of type $\langle s, \tau \rangle$. This is strictly speaking not the type one would expect for a system concerned with the *context change potential* (CCP) of sentences. According to this view, the CCP of a sentence is a function from information states to information states. It should be noted that it is not the states which are the information states

but rather the sets of sets of states. That this is indeed the case can be seen when we consider the following: in DMG we are interested in (partial) information about the reference of discourse markers, a state completely determines the references of discourse markers, hence it is obvious that the sets of sets of states, and not the states themselves are the real information states. This means that the notion of meaning as the context change potential of a sentence should be represented as a function of type $\langle \tau, \tau \rangle$. Groenendijk & Stokhof observe two things. First of all: this notion can be derived from functions of type $\langle s, \tau \rangle$ Second: Groenendijk & Stokhof note that *(...) incorporating other aspects of meaning, such as presuppositions, may very well necessitate the use of the more complex notion as our basic notion* (Groenendijk & Stokhof [2], page 22). This is exactly the line that is going to be explored in this section. With one small extension of the type: we will take type $\langle \langle s, \tau \rangle, \tau \rangle$ as our basic type. Notice that this type has an initial s added to the type corresponding with the CCPs. For the purposes of this section the type without the initial s (i.e. $\langle \tau, \tau \rangle$) will do. But later on we will need the slightly higher type. In order not to create too much confusion, I will use the higher type from the beginning.

But before we start being formal, I will first informally describe how we can translate sentences like (17) into DIL-formulas. Basically such a formula will have to consist of two parts: a part which tests if the previous discourse satisfies the presupposition and a part which is just the semantic representation of the sentence. So we need two new things: first of all we have to define a version of DMG which has the higher notion of meaning as its basis. And we need an operation which tests if presupposition-satisfaction. The first part entails that a sentence α will not be translated as $\lambda q(\alpha' \wedge {}^{\vee}q)$, but instead as $\lambda P \lambda q^{\vee} P(\hat{}(\alpha' \wedge {}^{\vee}q)$, where P is a variable of type $\langle s, \tau \rangle$.

The second new thing is the presupposition-test. For this I will introduce the following notation: $[\chi \gg \xi]$ (a context χ satisfies a presupposition ξ). This test will be a static one (i.e. of type t).[2] Notice that for such a definition of presupposition satisfaction we need a partial semantics for DIL. Because when the context does not satisfy the presupposition, we do not want the sentence to be false, but rather truth-value-less. The simplest way to achieve this is by letting the $\gg$-operation return a third truth-value (i.e. not true or false), say: *error*, when a presupposition is not satisfied. We will say that a presupposition is satisfied if, when the previous discourse is true, the presupposition is also true. Then the rest of the semantics of DIL has to be defined in such a way that when somewhere in a formula an error occurs, this can never be undone: the whole formula is in error (hence infelicitous). In particular: the negation of a formula which is in error, remains in error. We shall not spell out these definitions here, because they would run in a similar fashion as the rules of *error-state semantics* proposed by Van Eijck [9].

[2] Strictly speaking it has to be (internally) dynamic before it is brought back to the truthconditional level, for instance because a presupposition might contain free discourse markers (*Cooper saw a nice doughnut next to his cup of strong black coffee.* The doughnut he was going to eat *looked really delicious.*), but we will ignore this for reasons of simplicity.

Now we can state the intended DIL translation for a sentence containing a presupposition-trigger, like example (17). This translation will not only anticipate possible further sentences, but also previous sentences; it has a hook for both previous and further sentences. The hook for the previous sentences will be needed twice: once for the presupposition-test and once for the semantic representation. In the presupposition-part it will be tested if the static meaning of the previous discourse satisfies the presupposition. The semantic part will simply be 'delicious(d)'. When we add this all up we get the following DIL-representation:

(18) $\lambda P\lambda q([\Downarrow {}^{\vee}P \gg \exists x \mathsf{doughnut}(x)] \wedge {}^{\vee}P({}^{\wedge}(\mathsf{delicious}(d) \wedge {}^{\vee}q)))$

(Where $\Downarrow$ is the operation which brings a dynamic formula back to a static one).

3.1 Dynamic Montague Grammar revisited

When we want to extend DMG in such a way that we can work with DIL-formulas as the one in (18), this will lead to a different set of notation-conventions. We will refer to this version of DMG as DMG_p.

First of all, we define the operation which lifts DIL-expressions of type t to ones of type $\langle\langle s,\tau\rangle,\tau\rangle$.

Definition 10 (Uparrow) $\Uparrow\varphi = \lambda P\lambda q {}^{\vee}P({}^{\wedge}(\varphi \wedge {}^{\vee}q))$, where φ is a DIL-formula of type t, P a variable of type $\langle s,\tau\rangle$ and q of type $\langle s,t\rangle$

Notice that the $\uparrow$ from DMG is encapsuled in this definition. We also define the converse operation, bringing back formulas from type $\langle\langle s,\tau\rangle,\tau\rangle$ to type t. (In what follows we again use uppercase Greek letters for dynamic formulas (now of type $\langle\langle s,\tau\rangle,\tau\rangle$), and lowercase Greek letters for static formulas.)

Definition 11 (Downarrow)$\Downarrow\Phi = \Phi({}^{\wedge}\lambda p {}^{\vee}p)({}^{\wedge}\mathbf{true})$, in which Φ is a formula of type $\langle\langle s,\tau\rangle,\tau\rangle$ and p is a variable of type $\langle s,t\rangle$

Again we have the fact that $\Downarrow\Uparrow\varphi = \varphi$ The converse still does not hold in general: $\Uparrow\Downarrow\Phi \neq \Phi$.

The three primary DMG_p operators are defined in 12.

Definition 12 (DMG_p-operators)

1. $\Phi\,;\Psi = \lambda Q\Psi({}^{\wedge}(\Phi(Q)))$, Q not free in either Φ or Ψ.
2. $\mathcal{E}d_n = \lambda P\lambda q {}^{\vee}P(\exists x\{x/d_n\}{}^{\vee}q)$
3. $\sim\Phi = \Uparrow\neg\Downarrow\Phi$

Several things are worth noticing here. First of all notice that the definition of negation is the same as the one in DMG, except for the types. The existential quantifier is defined categorematically. This has been done to prevent free variables in previous sentences from being bound by the quantifier. A quantification $\mathcal{E}d\Phi$ will be read as a conjunction $\mathcal{E}d\,;\Phi$. That is, first we random assign a value to d and the dynamics is created by the composition

with Φ. In a sense this brings us closer to dynamic logic. The definition of conjunction might seem somewhat strange at first: the second sentence has the first one in its scope. But suppose the second sentence contains a definite determiner which triggers a presupposition. Then we want the representation of the previous discourse to land inside the test by means of an instance of λ-conversion and that is only possible when the second sentence has scope over the previous sentence(s). Apart from that the conjunction acts just the same as the one in DMG. So it will come as no surprise that the conjunction in DMG_p is also both associative and non-commutative. Notice that the DMG_p version of the important fact that an existential quantifier can bind variables which are outside its syntactic scope (see fact 5) is just a special case of the associativity fact in DMG_p:

Fact 8 $(\mathcal{E}d_n\,;\Phi)\,;\Psi = \mathcal{E}d_n\,;(\Phi\,;\Psi)$

The secondary DMG_p operators are again defined in terms of conjunction, negation and existential quantification.

Definition 13 (Secondary DMG_p-operators)

1. $\Phi \Rightarrow \Psi = \sim(\Phi\,;\sim\Psi)$
2. $\mathcal{A}d_n\,;\Phi = \sim(\mathcal{E}d_n\,;\sim\Phi)$
3. Φ **or** $\Psi = \sim(\sim\Phi\,;\sim\Psi)$

The definitions for the disjunction and implication are the same as their DMG counterparts, except of course for the types. The definition of the universal quantifier is somewhat different because we have introduced the existential quantifier categorematically. Again, it can easily be seen that the universal quantifier is internally dynamic. Another interesting thing to note is the structural similarity between the universal quantifier and the implication, as familiar from DRT. The fact that guarantees a correct reading for the donkey-sentences is in DMG_p formulated as follows:

Fact 9 $\mathcal{E}d_n\,;\Phi \Rightarrow \Psi = \mathcal{A}d_n\,;(\Phi \Rightarrow \Psi)$

In DMG_p we can replace dynamic operators with static ones exactly the same as in DMG, i.e. just like in fact 7, but with $\Uparrow$'s stand in for $\uparrow$'s. It is worth noticing that the static nature of negation in DMG_p is crucial for the DMG_p version of this fact. In this way, for instance the negation corresponding with the consequent of the implication does not take scope over the antecedent.[3]

We can construct a fragment for English in a parallel fashion as we did in section 2.2. The interesting definitions are of course the presupposition triggers, with as their most important representative the definite determiner. The translation of the definite article can be defined as follows (leaving aside uniqueness-issues):

[3] **Proof:** $\Uparrow\varphi \Rightarrow \Uparrow\psi = \sim(\Uparrow\varphi\,;\sim\Uparrow\psi) = \Uparrow\neg\Downarrow(\Uparrow\varphi\,;\Uparrow\neg\Downarrow\Uparrow\psi) = \Uparrow\neg\Downarrow(\Uparrow\varphi\,;\Uparrow\neg\psi) = \Uparrow\neg\Downarrow\Uparrow(\varphi \wedge \neg\psi) = \Uparrow\neg(\varphi \wedge \neg\psi) = \Uparrow(\varphi \rightarrow \psi)$

(19) *(DET)* $the_n \rightsquigarrow \lambda P\lambda Q\lambda R\lambda q([\Downarrow^{\vee} R \gg \Downarrow \mathcal{E} d_n {}^{\vee} P(d_n)] \wedge {}^{\vee} Q(d_n)(R)(q))$

I shall illustrate DMG_p with a simple example:

A_1 man eats a_2 doughnut. The_2 doughnut is delicious.
The first sentence translates into:

(20) $\lambda P\lambda q^{\vee} P(^{\wedge}(\exists x \exists y(\mathsf{man}(x) \wedge \mathsf{doughnut}(y) \wedge \mathsf{eat}(x,y) \wedge \{x/d_1\}\{y/d_2\}^{\vee} q)))$

The second sentence translates as follows:

(21) $\lambda R\lambda q([\Downarrow^{\vee} R \gg \Downarrow(\mathcal{E} d_1 \,;\, \Uparrow \mathsf{doughnut}(d_1))] \wedge \Uparrow \mathsf{delicious}(d_1)(R)(q))$

This can be reduced by writing out the second $\Uparrow$-operator and two instances of λ-conversion:

(22) $\lambda R\lambda q([\Downarrow^{\vee} R \gg \Downarrow(\mathcal{E} d_1 \,;\, \Uparrow \mathsf{doughnut}(d_1))] \wedge {}^{\vee} R(^{\wedge}(\mathsf{delicious}(d_1) \wedge {}^{\vee} q)))$

Next we dynamically conjoin the two representations using notation convention 12.1 and one instance of λ-conversion:

(23) $\lambda Q(\lambda R\lambda q[\Downarrow^{\vee} R \gg \Downarrow(\mathcal{E} d_1 \,;\, \Uparrow \mathsf{doughnut}(d_1))] \wedge {}^{\vee} R(^{\wedge}(\mathsf{delicious}(d_1) \wedge {}^{\vee} q))$
$(^{\wedge}(\lambda q^{\vee} Q(^{\wedge}(\exists x \exists y(\mathsf{man}(x) \wedge \mathsf{doughnut}(y) \wedge \mathsf{eat}(x,y) \wedge$
$\{x/d_1\}\{x/d_2\}^{\vee} q))))))$

Now we do two more instances of λ-conversion and see that the representation of the first sentence lands in both the presupposition-test and the representation of the second sentence:

(24) $\lambda Q\lambda q([\exists x \exists y(\mathsf{man}(x) \wedge \mathsf{doughnut}(y) \wedge \mathsf{eat}(x,y)) \gg \exists z \mathsf{doughnut}(z)] \wedge$
${}^{\vee} Q(^{\wedge}(\exists x \exists y(\mathsf{man}(x) \wedge \mathsf{doughnut}(y) \wedge \mathsf{eat}(x,y) \wedge \mathsf{delicious}(y) \wedge$
$\{x/d_1\}\{y/d_2\}^{\vee} q))))$

It shall be clear that the previous discourse satisfies the relevant presupposition. In fact, the presupposition-test is a tautology, so what we end up with is the second conjunct. When we apply the $\Downarrow$-operator to it we end up with the following static truth conditions:

(25) $\exists x \exists y(\mathsf{man}(x) \wedge \mathsf{doughnut}(y) \wedge \mathsf{eat}(x,y) \wedge \mathsf{delicious}(y))$

Notice that this exactly the representation we would find for the same discourse, with *it_2* instead *the_2 doughnut*.

The DMG_p approach to presuppositions sketched above is very simple and by no means complete. One thing that has to be altered is the definition of negation: it is still the static closure. But there is also another, more basic problem with the approach. A presupposition is analyzed as a test on the context (previous discourse). If the test is satisfied we continue; if not, we render the sentence infelicitous. The problem is that this is not the way it works in real life. Humans tend to be much more coöperative. Suppose A says to B:

(26) Sorry I'm late, but I wanted to eat my doughnut first.

it is not very likely that B will protest that this doughnut was never properly introduced. Rather he will *accommodate* (as Lewis [6] called it) the information that A 'owned' a doughnut, triggered by the definite NP *my doughnut.* An accommodated presupposition has its descriptive content simply *added* to the representation of the discourse, in a suitable place. This process has been studied in some detail by Heim [4] and particularly Van der Sandt (Van der Sandt [10], Van der Sandt & Geurts [11]).

Accommodation is for dynamic semantics an important and interesting mechanism. Take the use of PNs. PNs have a different anaphoric behavior than indefinite NPs. When an indefinite NP is used in an if-then clause it cannot be the antecedent for further reference outside that clause, as we have seen in (15). When we replace the indefinite NP with a PN, the situation changes radically:

(27) If Cooper$_1$ sees a$_2$ doughnut, he$_1$ eats it$_2$. He$_1$ really loves doughnuts, particularly with black coffee.

This phenomenon led Kamp to define a rule which stated that the discourse marker for a PN is always introduced in the principal DRS, no matter how deeply embedded it occurs. This is rather ad hoc. But, as Zeevat [12] noticed, with accommodation we have a sort of explanation for this: what happens is that the presupposition triggered by the PN (that is its descriptive content; it is after all a definite NP) is accommodated from the embedded clause to the top-level. In the last se tion of this paper, I will roughly describe how such an accommodation mechanism can be formalized in DMG$_p$. I will illustrate it with PNs, because they do not carry very much descriptive content, but the procedure is general enough to be applied to other definite NPs, like -most notably- definite descriptions. In order to do this we also have to change the definition of negation, as announced a few paragraphs above.

4 Accommodation in DMG$_p$

The prime difference between DMG and DMG$_p$ is that the former only anticipates further discourse, while the latter also takes the previous discourse in consideration. This means that it should be possible to modify the representation of the previous discourse in DMG$_p$, and that is basically what accommodation does. The idea is the following: when a sentence contains a presupposition-trigger, we test if the previous discourse satisfies this presupposition. If that is the case we continue, but if that is not the case we now try to *accommodate* the presupposition. The lexical entry for a presupposition-trigger contains a test which amounts to the following: if the presupposition is satisfied nothing happens, if the presupposition is not satisfied we let the descriptive content of the presupposition end up in front of the hook for the representation of the previous discourse.

An example will hopefully clarify this. Suppose A tells B the following:

(28) If Leland$_1$ is happy, he$_1$ dances.

and they haven't discussed this Leland before (i.e. the presupposition triggered by the definite NP *Leland* is not satisfied by the representation of the

previous discourse). Then B will assume (i.e. accommodate) that there exists someone whose name is Leland and that when that person is happy he whistles. In other words, the presupposition 'escapes' from the conditional and gets accommodated at a 'higher' level of representation. This also explains the fact that we can refer back to the individual named Leland in sentences which follow the conditional (see the difference between example (15) and example (27)).

How can we translate sentence (28) into DMG_p? I will assume that we have passed the testing-stage and that the result was that Leland was not present in the context. As said, we then let the descriptive content (which in the case of a PN in DMG is simply a state-switcher) end up in front of the representation of the clause in which the trigger occurred:

(29) $\{leland/d_1\}\Uparrow\mathsf{happy}(d_1) \Rightarrow \Uparrow\mathsf{dance}(d_1)$

When we write out the first occurrance of the $\Uparrow$-operator and the definition of the dynamic implication we get the following formula:

(30) $\sim(\lambda P\lambda q\{leland/d_1\}{}^{\vee}P({}^{\wedge}(\mathsf{happy}(d_1) \wedge {}^{\vee}q)) ; \sim\Uparrow\mathsf{dance}(d_1))$

Notice that the state-switcher 'hops' over the lambdas and keeps hanging in front of the ${}^{\vee}P$, which is the hook for the previous discourse. In order to make things work we need two things, both having to do with the closure. The only closure we have seen above is the full static closure, i.e. $\Uparrow\Downarrow\Phi$. That is also the form of closure in terms of which we defined the DMG negation. Notice that with the higher $\langle\langle s,\tau\rangle,\tau\rangle$, we have in fact two possible definitions of closure besides the full closure: one which allows for dynamic binding but stops further accommodation, and one which does the exact opposite.

To begin with the first one. In DMG_p we have two hooks to relate the representation of a sentence with the representation of the rest of the discourse; one for the previous discourse and one for further discourse. But there is a significant difference between them: we can keep anticipating further sentences as long as we like, but we cannot do the same with the previous discourse, which is of a fixed length. We do not want a presupposition which is in the process of being accommodated to stay in front of the hook for previous sentences too long; at a certain point we want the state-switcher to be able to walk through the formula, replacing discourse markers where possible. That is where the first closure operator comes into play, which I will call *backward closure*: there is no longer any space for previous sentences, while further sentences are still anticipated. In other words: accommodation is blocked, while dynamic binding is still possible. Or: backwards closure brings a DMG_p formula back to a normal DMG formula. A formula which no longer has a place for previous sentences (that is, a formula of type τ) will be symbolized as $\vec{\varphi}$.

Definition 14 (Backward closure: $\nearrow\!\!\swarrow\Phi$)

1. $\swarrow\Phi = \Phi(^\wedge\lambda p^\vee p)$, where p is a variable of type $\langle s,t\rangle$
2. $\nearrow\!\vec{\varphi} = \lambda P\lambda q P(^\wedge(\vec{\varphi}(q)))$, P and q are variables of types τ and $\langle s,t\rangle$ respectively.

Of course we have the following fact: $\swarrow\!\!\nearrow\vec{\varphi} = \vec{\varphi}$ However, in this case we are more interested in the fact that the following *doesn't* hold in general: $\nearrow\!\!\swarrow\Phi = \Phi$. As familiar from the literature, there is a strong preference for global accommodation (over local accommodation). Usually local accommodation is only an option, when global accommodation is out of the question (for instance because it leads to a contradiction). For now I simply want to propose that the translation of a sentence (but not of a clause!) is closed off under backwards closure.

The most problematic parts of presupposition-testing and accommodating are the negation and the constructions defined in terms of it. The problem is that presuppositions escape from the scope of negation. In section 3.1 we have defined negation in terms of static closure. Such a definition of negation entails that a PN cannot be the antecedent of further reference, because accommodation fails (both the hook for previous and the one for further discourse are closed off). That this is incorrect follows from example (31.a) However, it is true that an indefinite NP inside a negated sentence cannot bind variables outside that sentence (except of course in the modal subordination cases), as can be seen from sentence (31.b).

(31) a. It is not the case that Leland is unhappy. He whistles.

b. It is not the case that a man is unhappy. *He whistles.

What this calls for is a form of closure which stops the dynamic potential, but allows for accommodation (negation 'as a hole' in Karttunenian terminology). Then a state-switcher is allowed to escape the clause and go to the beginning of the discourse representation and from there it can bind possible anaphors. A formula which allows for accommodation, but not dynamic binding (i.e. a formula of type $\langle\langle s,\tau\rangle,t\rangle$) will be symbolized as $\overleftarrow{\varphi}$.

Definition 15 (Forward closure: $\triangle\!\!\triangledown\Phi$)

1. $\triangledown\Phi = \Uparrow\Phi(^\wedge\mathbf{true})$
2. $\triangle\!\overleftarrow{\varphi} = \lambda Q_{\langle s,\tau\rangle}\lambda q_{\langle s,t\rangle}\overleftarrow{\varphi}(^\wedge\lambda r_{\langle s,t\rangle}{}^\vee Q(^\wedge(^\vee r \wedge {}^\vee q)))$

Again we have the fact that $\triangledown\!\triangle\overleftarrow{\varphi} = \overleftarrow{\varphi}$. But not in general: $\triangle\!\triangledown\Phi = \Phi$. It is unfortunately not so simple that we can define negation as follows: $\sim\Phi = \triangle\neg\triangledown\Phi$

There is a way out, although I admit it is bit trickery. What we want from negation 'as a hole' is that it closes of the possibility of further binding, allows for accommodation, negates the information in its scope and brings

the whole back to the original type. When we look at the definition of $\triangle$ it is clear where the predicates that need to be negated will end up, so we can easily define another lifting which is similar to $\triangle$, but with a negation added:

Definition 16 $\triangle_{\neg}\overleftarrow{\varphi} = \lambda Q \lambda q \overleftarrow{\varphi}(\hat{}\lambda r^{\vee} Q(\hat{}(\neg^{\vee} r \wedge {}^{\vee} q)))$

Then we can define $\sim\Phi$ simply as follows:

Definition 17 $\sim\Phi = \triangle_{\neg}\triangledown\Phi$

Once we have this definition of negation we can again define implication, disjunction, and universal quantification in terms of it. These definitions are exactly the same as those in 13, but now their behavior is rather different; they allow presuppositions to be accommodated out of their scope.

Let us now look once again at example (28):

If Leland$_1$ is happy, he$_1$ dances.
The translation remains the same as the one we have seen in (29), except that it we apply the backwards closure to it:

(32) $\nearrow\!\!\swarrow(\{leland/d_1\}\Uparrow\mathsf{happy}(d_1) \Rightarrow \Uparrow\mathsf{dance}(d_1))$

Writing out the two $\Uparrow$-operators and the dynamic implication, we get the following:

(33) $\nearrow\!\!\swarrow(\sim(\lambda P \lambda q\{leland/d_1\}^{\vee} P(\hat{}(\mathsf{happy}(d_1) \wedge {}^{\vee} q)); \sim \lambda P \lambda q^{\vee} P(\hat{}(\mathsf{dance}(d_1) \wedge {}^{\vee} q))))$

Next we write the definition of the inner negation out, which reduces to:

(34) $\nearrow\!\!\swarrow(\sim(\lambda P \lambda q\{leland/d_1\}^{\vee} P(\hat{}(\mathsf{happy}(d_1) \wedge {}^{\vee} q)); \lambda P \lambda q^{\vee} P(\hat{}(\neg\mathsf{dance}(d_1) \wedge {}^{\vee} q))))$

Dynamic conjunction of antecedent and consequent results in the following formula:

(35) $\nearrow\!\!\swarrow(\sim(\lambda Q \lambda q\{leland/d_1\}^{\vee} Q(\hat{}(\mathsf{happy}(d_1) \wedge \neg\mathsf{dance}(d_1) \wedge {}^{\vee} q))))$

Then we write the definition of the outer negation out:

(36) $\nearrow\!\!\swarrow(\triangle_{\neg}\triangledown(\lambda Q \lambda q\{leland/d_1\}^{\vee} Q(\hat{}(\mathsf{happy}(d_1) \wedge \neg\mathsf{dance}(d_1) \wedge {}^{\vee} q))))$

Following that we eliminate the $\triangledown$-operator, lowering the type and avoiding dynamic binding:

(37) $\nearrow\!\!\swarrow(\triangle_{\neg}(\lambda Q\{leland/d_1\}^{\vee} Q(\hat{}(\mathsf{happy}(d_1) \wedge \neg\mathsf{dance}(d_1)))))$

And then we do the same with the $\triangle_{\neg}$-operator, negating the sentence and bringing the formula back to its original type:

(38) $\nearrow\!\!\swarrow(\lambda R \lambda q \lambda Q\{leland/d_1\}^{\vee} Q(\hat{}(\mathsf{happy}(d_1) \wedge \neg\mathsf{dance}(d_1)))(\hat{}\lambda r^{\vee} R(\hat{}(\neg^{\vee} r \wedge {}^{\vee} q)))) =_{\lambda}$
$\nearrow\!\!\swarrow(\lambda R \lambda q\{leland/d_1\}^{\vee} R(\hat{}(\neg(\mathsf{happy}(d_1) \wedge \neg\mathsf{dance}(d_1)) \wedge {}^{\vee} q))) =_{\text{equivalent}}$
$\nearrow\!\!\swarrow(\lambda R \lambda q\{leland/d_1\}^{\vee} R(\hat{}(\mathsf{happy}(d_1) \rightarrow \mathsf{dance}(d_1)) \wedge {}^{\vee} q))$

We see that the '$^\vee q$' is outside the implication (that is, the implication is externally static), and the state-switcher corresponding to $Leland_1$ is standing in front of the variable anticipating the previous discourse (that is, the presupposition triggered by the PN is accommodated to top-level). Now we can write out the definition of the forward closure, allowing the state-switcher to walk through the representation. This will result in the following formula:

(39) $\lambda P \lambda q^\vee P(^\wedge((\mathsf{happy}(leland) \rightarrow \mathsf{dance}(leland)) \wedge \{leland/d_1\}^\vee q))$

5 Conclusions and future directions

In this paper I have described how we can extend Groenendijk & Stokhof's DMG to a framework in which presuppositions can be analyzed. In section 2.2 we saw a first approximation of a treatment of presuppositions in DMG. This treatment is not the desired one, because (i) the definition of DMG-negation is still defined in terms of full-static closure, hence a presupposition cannot escape from its scope and (ii) in case a presupposition is not satisfied by the context the sentence containing the trigger of the presupposition is always labelled infelicitous; there is no possibility for accommodating presuppositions. To overcome these two points we can define two more flexible notions of closure, one which allows accommodation but not dynamic binding and one that does the exact opposite. We can use the latter to give a somewhat trickery definition of DMG-negation which does the job. I would like to stress that the above is by no means a complete theory of presuppositions, it just hands the tools which are likely to be needed when we want to give such a theory in a DMG-framework..

In the future several things need to be investigated, a number of which I will mention shortly in this section. First of all, we have to find a way to constrain the accommodation-process. Van der Sandt [10] presents some constraints. These are a little too strong, but empirically they do rather well. An interesting line of research is to look at exactly what conditions should be used and how they can be incorporated in the approach I sketched above.

Besides that we have to generalize the approach to other presuppositions. I think that the outline of the procedure presented above is general enough to include other presuppositions. For instance definite descriptions. There is one important difference between PNs and definite descriptions: the descriptive content of the first is just a state-switcher, while the second is a state-switcher with one or more constraints on it. Suppose we want to accommodate a presupposition triggered by the_1 *doughnut*. That means that we want to place a state-switcher $\{x/d_1\}$ in front of the hook for the previous discourse, but we also want to say that x is a doughnut. What we seem to need is a more specific form of state-switching.

I guess another thing section 4 shows, is that the formulas tend to get rather complex and highly typed. This entails that we still do not gain much insights in what is going on when we accommodate presuppositions (except perhaps the observation that it is a very strong process). A last point for further research therefor is to try if it is possible to give an analysis along

the lines presented in this paper in an 'easier' framework like Groenendijk & Stokhof's DPL [3] or a related framework like Muskens' version of DMG [8].

6 References

[1] Gamut, L. T. F., *Logic, Language and Meaning, Vol. 2, Intensional Logic and Logical Grammar*, The University of Chicago Press, Chicago, 1991.

[2] Groenendijk, J. & M. Stokhof, "Dynamic Montague Grammar," in *Papers from the Second Symposium on Logic and Language*, L. Kálmán & L. Pólós, eds., Akadémiai Kiadó, Budapest, 1990.

[3] ———, "Dynamic Predicate Logic," *Linguistics and Philosophy* 1 (1991), 39–100.

[4] Heim, I., "On the Projection Problem for Presuppositions," in *Proceedings of the Second West Coast Conference on Formal Linguistics*, M. Barlow e.a., ed., Stanford University, Stanford, 1983, 114 – 125.

[5] Janssen, T., *Foundations and Applications of Montague Grammar*, CWI, Amsterdam, 1986.

[6] Lewis, D., "Scorekeeping in a Language Game," *Journal of Philosophical Logic* (1979).

[7] Montague, R., "The Proper Treatment of Quantification in Ordinary English," in *Formal Philosophy, selected papers of Richard Montague*, R. Thomason, ed., Yale University Press, New Haven, 1974, 247 –270.

[8] Muskens, R., "Anaphora and the Logic of Change," in *JELIA '90, European Workshop on Logics in AI*, J. Van Eijck, ed., Springer Lecture Notes, Springer Verlag, Berlin, 1990, 414–430.

[9] Van Eijck, J., "The Dynamics of Description," in *Papers from the first CLIN-meeting*, T. Van der Wouden & W. Sijtsma, eds., OTS working papers, 1991.

[10] Van der Sandt, R., "Presupposition and Discourse Structure," in *Semantics and Contextual Expression*, R. Bartsch, J. Van Benthem & P. Van Emde Boas, eds., Foris Publications, Dordrecht, 1989.

[11] Van der Sandt, R. & B. Geurts, "Presupposition, Anaphora and Lexical Content," in *Text understanding in LILOG*, O. Herzog & C.-R. Rollinger, eds., Springer Lecture Notes, Springer Verlag, Berlijn, 1991, 259–296.

[12] Zeevat, H., "Discourse Semantics and Unification Grammar," University of Amsterdam, dissertation, 1991.