

Simultaneous Abstraction and Semantic Theories

Peter Ruhrberg

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy
to the
University of Edinburgh
1996

Declaration

This thesis has been composed by myself and it has not been submitted in any previous application for a degree. The work reported within was executed by myself, unless otherwise stated.

March 1996

Acknowledgements

My first, and foremost thank goes to Robin Cooper. His influence on my work is evident for almost every part of this thesis. His use of Aczel and Lunnon's system of simultaneous abstraction as a framework for semantic theories, in particular for his Situation–Theoretic Discourse Representation Theory, has been largely my inspiration for the work presented here. Without his interest and support I would not have carried out this work, and neither would I have had the opportunity to do so, for financing a PhD abroad is no trivial matter, even for citizens of a wealthy, allegedly Euro–enthusiast country, like Germany.

My second “thank you” goes to the people at the Centre for Cognitive Science in Edinburgh. I feel privileged to have been one of them for so long, and I am continually amazed by their level of enthusiasm and commitment. Measured by such standards, one can only hope not to disappoint too severely.

Furthermore, I am grateful for the interest and helpful comments I received for this work, from Peter Aczel, Lawrence Cavedon, Susanna Kuschert, Rachel Lunnon, David Milward, Massimo Poesio, Paul Schweizer, Colin Stirling, Henk Zeevat, the members of the MaC working group at the CCS, the people at the IMS in Stuttgart and the Computerlinguistik in Saarbruecken, as well as my examiners Lex Holt and Hans Kamp. I thank my various office mates over the years, in particular Tsutomu Fujinami and Matthias Paul. For financial support I thank the DAAD, ESRC, ESPRIT BR6852 (DYANA-2, special thanks to Marc Moens), and LRE 62-051 (FraCaS).

Abstract

I present a simple *Simultaneous Abstraction Calculus*,¹ where the familiar λ -abstraction over single variables is replaced by abstraction over whole sets of them. Terms are applied to partial assignments of objects to variables. Variants of the system are investigated and compared, with respect to their semantic and proof theoretic properties.

The system overcomes the strict ordering requirements of the standard λ -calculus, and is shown to provide the kind of “non–selective” binding needed for Dynamic Montague Grammar and Discourse Representation Theory. It is closely related to a more complex system, due to Peter Aczel and Rachel Lunnon, and can be used for Situation Theory in a similar way. I present versions of these theories within an axiomatic, property–theoretic framework, based on Aczel's Frege Structures.

The aim of this work is to provide the means for integrating various semantic theories within a formal framework, so that they can share what is common between them, and adopt from each other what is compatible with them.

¹Earlier versions of parts of this work have been previously published by the author under the title *A Simultaneous Abstraction Calculus and Theories of Semantics*, in [Cooper & Groenendijk 1994], and slightly updated in [Seligman et al. 1995]. The complete draft, before corrections, has also been published under the current title, in [Cooper et al. 1996].

Contents

Acknowledgements	ii
Abstract	iii
1 Introduction	1
1.1 Basic Assumptions	1
1.2 Why Simultaneous Abstraction?	4
1.2.1 Variables: “Bound yet Free”	8
1.2.2 Axiomatic Semantics	9
1.2.3 Bealer’s Puzzle	10
1.3 Other Frameworks	11
2 Simultaneous Abstraction	13
2.1 Simple Simultaneous Abstraction	13
2.1.1 Syntax for Λ	13
2.1.2 Semantics for Λ	15
2.1.3 Basic Properties of Λ	17
2.1.4 Partial Application	20
2.2 Aczel–Lunnon Abstraction	21
2.2.1 Relating Λ , Λ_{AL} , and λ	23
2.2.2 Proof Theory for Λ_{AL}	25
2.2.3 The Church–Rosser Theorem	27
2.3 Combinatory Aczel–Lunnon Logic	31
2.3.1 Completeness for C_{AL}	33
2.3.2 Abstraction in C_{AL} -models	34

2.3.3 Completeness for Λ_{AL}	37
2.4 Structured Objects and Systems of Equations	39
3 Semantic Theories	41
3.1 Frege Structures	41
3.1.1 Consistency	46
3.2 Dynamic Montague Grammar	48
3.2.1 Some DMG Translations	50
3.3 Discourse Representation Theory	53
3.3.1 Zeevat’s DRT	53
3.3.2 ω -Properties	55
3.3.3 Kamp Structures	56
3.3.4 Consistency	58
3.4 Situation Theory	60
3.4.1 States of Affairs	61
3.4.2 Barwise Structures	63
3.4.3 Consistency	64
4 Further Directions	66
4.1 True Dynamics	66
4.2 A Typed System	68
4.3 Λ -DRT	72
4.3.1 Consistency	77
4.3.2 Some Λ -DRT Translations	78
4.4 A Note on Appropriateness and Restrictions	80
4.5 Conclusion	81
References	83
Index	86

Chapter 1

Introduction

Linguistic semantics seems to have become a true Babel of newly invented and ever changing logical formalisms, where much research is dedicated to the business of translating between them. I will add another two or three formalisms to the list. Yet my aim is not to increase the confusion, but rather to reduce it, by giving not so much a particular semantic formalism but a formal framework in which various semantic theories can be cast, in a way that allows them to share explicitly what they have in common. My faint hope is to regain some of the unity in semantics that has been lost over the last decade or so, during which the authority of Montague's Higher Order Intensional Logic (IL) was so badly eroded. The proposed systems contain no really new ideas, but rather steal elements of existing formalisms and put them together so that their common roots are clearly exposed. The most prominent common feature of the theories discussed in this work is their use of some form of *simultaneous abstraction*. I will give (variations of) a formal system of simultaneous abstraction and application, as the backbone for a set of theories which are obtained by the addition of axioms of truth, following the lead of Property Theory.

1.1 Basic Assumptions

Let me briefly outline the underlying philosophical assumptions of this work. I won't attempt to defend them here, as that would be another thesis.

I will be concerned with theories of meaning that fit into the wider realm of *denotational semantics*. I take it that utterances in some natural language are related, in virtue of their phonological, syntactic and contextual features, to their *contents* (meanings), which are relatively abstract objects such as *propositions*, in the paradigm case of assertoric sentences. What propositions are is a matter open to dispute, but I take it to be fundamental for them to be capable of being true or false, thus taking us to a *truth conditional* form of semantics. A semantic theory should make predictions of the kind that given certain things being true certain other things having to be true as well, which are to be judged against intuitions about *logical consequence*.

How to precisely make the links between meaning, truth, and logical consequence within this broader paradigm of semantics is again a matter of debate. Most widely known theories take the model theoretic line, following Tarski, and use some formal language to facilitate the interpretation. Logical consequences are then regarded as preserving truth under the reinterpretation of the non-logical symbols of that language. I do not wish to subscribe to this Tarski-Bolzano view of logic, for reasons given in [Etchemendy 1990], and thus refrain from defining logical consequence along those lines. I follow Etchemendy's view that an adequate theory of meaning must be able to separate out changes of meaning from changes of the way things are in the world, and the latter seem to lie at the heart of logic as I understand it. In following Property Theory, in the form of Frege Structures [Aczel 1980], I believe that this distinction is being made quite clearly.¹ Still, the proposed framework is pretty much open to a Tarskian view of logic, if one wishes to uphold it, and I will not try to develop and defend an alternative conception here, which would be based on the variation of circumstances which make propositions true, rather than the variation of the meaning of the "non-logical" pieces of syntax.

In emphasising matters of truth I do not mean to assert that semantics is nothing but the assignment of truth conditions to sentences, or utterances, or that propositions are to be identified with truth conditions, or distinguishable

¹One might say that meanings are fixed by the interpretation $\| \cdot \|$ in a particular \mathbb{A} -model, while matters of fact are reflected in the extension \mathcal{T} of the truth predicate in a Frege Structure, based on that \mathbb{A} -model.

only on the basis of these. The most prominent theory which follows these coarse grained lines, by identifying propositions with sets of logically possible worlds, has long been known to suffer from fundamental inadequacies in the analysis of propositional (and other) attitudes. I take it as a guiding principle of semantics that propositions and other contents of utterances must be plausible candidates for being the objects of our attitudes. The content of an assertion is something a sincere and competent speaker believes, and wishes to convey to his audience, thus hopefully contributing to the spread of knowledge in his community.

The links between the notions of content, propositions, truth, and attitudes, as I have outlined them, are largely subscribed to by the particular semantic theories I will discuss in this work, or they come at least close to it, if we push them a little in this direction.

Montague Grammar (MG) for example, [Montague 1974], in most of its forms subscribes to the identification of propositions with sets of possible worlds. These are the objects of propositional attitudes, and they are truth bearers in the sense that they may or may not contain the actual world. Versions of MG which do not make this identification, and operate along property theoretic lines instead, have been proposed, for example in [Thomason 1980] and the more recent [Chierchia 1994]. The latter work will concern me here in particular, as it presents a *dynamic* version of MG. It follows Dynamic Montague Grammar (DMG) of [Groenendijk & Stokhof 1990], but takes intensionality at face value and appears less baroque in its treatment of dynamic binding.

Discourse Representation Theory (DRT), [Kamp 1981], appears to regard propositions as rather syntactic objects, the Discourse Representation Structures (DRSs), whose truth is a matter of anchoring its Discourse Referents with respect to an appropriate model. I propose to regard DRSs as particular kinds of relations, which are easily turned into propositions by existential quantification, or by application (anchoring) to individuals. They are not to be confused with their notations, although one may argue that they are almost as highly structured as the latter. I believe this view can make sense of the kind of theory of attitudes

envisaged in [Kamp 1990], at least to some extent, without identifying the objects of attitudes with syntactic structures.² I have no quarrel with a syntactic theory of mind as such, but I do not think it is embodied in the meaning of our intensional idiom.

Situation Theory (ST) [Barwise & Perry 1983] can be taken to promote the idea that propositions have two ingredients: a situation and a state of affairs (infor, fact). Truth depends upon the “support” of the state of affairs by the situation, and the actuality of the situation, if abstract “possible” situations are admitted into the ontology. Not all versions of ST take such a narrow view of propositions, see [Barwise & Cooper 1991], and neither do I. With regards to attitudes ST tends to stress the role of situations, yet it is generally agreed, following the critique of ST in [Soames 1985], that they do not allow us to dispense with a highly structured view of propositions and states of affairs.

I stand in some distance to theories of “dynamic” semantics that stress the changes in information states brought about by the processing of sentences. There can be no doubt that such changes are of utmost importance, and that a theory of meaning worth its money must say something insightful about such matters, but this does not, on my view, relieve us from coming up with a serious theory of propositional contents as relata of attitudes. It seems to me to be in fact a precondition for any realistic dynamic theory. DMG, DRT, and ST all seem to conform fairly well to this traditional view, despite occasional rhetoric to the contrary.

1.2 Why Simultaneous Abstraction?

Since Montague's pioneering use of IL for the semantics of a fragment of English, λ -abstraction has continued to play a key role in linguistic semantics. Yet, while (repeated) unary abstraction may be all we need for mathematical logic, it looks

²Robin Cooper takes a similar view in [Cooper 1995]. In contrast to him, I do not think that simultaneous abstraction by itself provides an analysis of what Kamp calls the “sharing” of discourse referents in [Kamp 1990].

For a more radical, syntactic approach to attitudes, see [Asher 1993], who does not only regard DRSs as the “language of thought”, but also takes attitude reports to refer to these formal structures.

less satisfactory when we encounter linguistic applications, in particular to languages with more freedom in their word order than English. It is awkward, if not genuinely inadequate, to impose a strict order on the denotations of predicative expressions, which then has to be undone by sophisticated means, to cope with words coming in the “wrong” order. Moreover it seems slightly puzzling how easily logicians with Realist views on properties and relations seem to extrapolate the order of words of their preferred language into the structure of those independently existing entities. Yet, my main reason for embarking into simultaneous abstraction is that it is already being put to good use, if often in disguised forms, in linguistic semantics. Forms of simultaneous abstraction seem to be at work in the three prominent semantic approaches of the post Montagovian era to be discussed in this thesis, namely in Discourse Representation Theory, Dynamic Montague Grammar, especially when it is based on the Dynamic Property Theory (DPT) of [Chierchia 1994], and modern Situation Theory, as described in [Barwise & Cooper 1991], with its underlying theory of abstraction (AL) developed in [Aczel & Lunnon 1991].

The basic principle of simultaneous abstraction I take to be that the λ -operator takes a whole set of variables, or something to that effect, rather than just one at a time (or a list that is easily “curried” into a sequence of singular abstractions). To give a concrete example, we may write something like

$$\lambda\{x, y\}.likes(x, y)$$

in a system like the language Λ of the next chapter.³ The absence of any order in $\{x, y\}$ creates the problem of defining a sensible operation of application. We may want to express the fact that Robert Parker likes Dominus. So we need a way to unambiguously determine which of the objects *rob* and *dom* fill which of the positions abstracted over. The solution to this difficulty comes in the form

³The reader may suspend concerns regarding the order in *likes(x, y)* until the coming chapter. Notice though how application of *likes* to *x* and *y* turns an ordered relation into an unordered one, where the variables act as identifiers for the roles of the relation.

of application to assignments. I write the application in question as

$$(\lambda\{x, y\}.likes(x, y))(x.rob, y.dom)$$

which then reduces to *likes(rob, dom)*. Our abstracts denote, or correspond to, functions from assignments to objects of some kind; in this case a function from variable assignments to propositions.⁴ It is instructive to regard the free variables in an open formula as a kind of paradigm case: the denotation of an open formula is precisely a function from variable assignments to the contents of closed expressions. Other examples are not hard to find either.

DRSs for example are usually taken to be pairs $\langle U, C \rangle$ consisting of a set U of variables (the “universe of discourse markers” of the DRS) and a conjunctive set C of “conditions”. In giving truth conditions to DRSs the variables in U are read as being “non-selectively” existentially quantified, but only those of the outermost level DRS. Embedded DRSs receive all sorts of quantification over their universes, most prominently a non-selective universal reading of the discourse markers in U in a conditional of the form $\langle U, C \rangle \Rightarrow \langle U', C' \rangle$. Abstraction over its universe seems thus a plausible mechanism for giving a denotation to DRSs, which leaves the quantificational force to be fixed explicitly or implicitly by the context in which they occur. The standard semantics for DRT does not give an explicit denotation to DRSs, and thus seems to avoid needless complications involving abstraction, but once we widen our horizons and allow DRSs to occur as arguments of relations, in particular those involving attitudes, the complications have to be faced in one way or another. My inclination is to regard a DRS as a term $\lambda U.C$, where we abstract over an unordered set of variables U , rather than in the one by one fashion of IL, and C expresses a fine grained conjunction of propositions, rather than a truth value or a set of possible worlds. The content of a DRS is thus a property in as many arguments as there are variables in its universe.

DMG was developed to introduce ideas from DRT into the compositional

⁴Variables are thus used rather like key words in records. Case marking might be regarded in this way for example, when the language in question employs case rather than order for the identification of the roles of a predicate.

framework of IL, by means of a new basic type of “states” which are essentially assignments of entities to discourse markers. Chierchia’s DPT gets rid of the states as primitives, in favor of explicit assignments which are manipulated by means of two operators: \sqsupset for abstraction, and \sqcup for application. In fact the \sqsupset -operator can be understood as abstracting over the infinite set of discourse markers, while \sqcup applies a term to the current discourse marker assignment. There is no need to be quite so restrictive in what may be abstracted over and applied to, since the underlying idea is a perfectly general one. In our system any set of variables can be abstracted over, and any assignment can be applied to. This may dissolve any remaining doubts about the difference between variables and discourse markers and simplifies matters by eliminating unary abstraction and application as primitive operations from the system.

AL finally is a system of simultaneous abstraction in which λ takes one-one mappings from labels, called “role indices”, to a kind of variables, called “parameters”, as input to abstract over an unordered set of variables in one go. Application is performed on assignments of objects to those labels. The above example would look more like this in AL:

$$(\lambda(r_1 \mapsto X, r_2 \mapsto Y).likes(X, Y)) (r_1.rob, r_2.dom).$$

The essentially simple core of simultaneous abstraction and application in AL is obscured by some less obvious features of the system, which derive from its historical role in modeling Situation Theory via generalizations of the non well-founded set theory of [Aczel 1988]. In particular the theory is developed not as a formal language, but within a highly general setting of a theory of structured objects. Some of these objects are of an especially suspect looking “indeterminate” kind, namely the “parameters”. λ is an operation “in the model”, which takes role-indexed sets of parameters and parametric objects into things called “abstracts”. Much concern is devoted to the task of finding structured universes containing such abstracts, in which systems of equations involving those parameters have unique solutions in terms of anchoring the parameters to non-parametric objects. Given the rather different concerns of Linguistics, it is little wonder then

that AL has not found widespread popularity among semanticists,⁵ who remain to be convinced that such sophisticated machinery is needed to achieve their modest ambitions. This thesis offers a hopefully more accessible system of simultaneous abstraction, which shares many – though perhaps not all – of AL’s virtues, despite its comparatively simple minded, traditional design features.

1.2.1 Variables: “Bound yet Free”

A peculiar hurdle on the way to acceptance of the very idea of simultaneous abstraction is the resulting (partial) loss of α -equivalence. For example we have

$$\lambda\{x, y\}.likes(x, y) \neq_{\alpha} \lambda\{y, x\}.likes(y, x).$$

This loss is an inevitable consequence of the way in which application has to work for such terms. A renaming of variables in an abstraction term will normally change the object, as the two supposedly equivalent terms will yield different results when applied to the same assignment. For example, applying the two terms above to $(x.rob, y.dom)$ yields two quite different results. This should come as no surprise. It is of course well known that free variables cannot be renamed. Neither can the abstracted discourse markers of a DRS be changed without consequences: any subsequent anaphoric link depends on the very identity of the variables in the universe of a DRS. In the same vein no renaming is allowed for DPT discourse markers, may they be free or in the scope of a \sqsupset -abstraction. This is easily demonstrated by an application of \sqcup , which frees those discourse markers again by $\sqcup\sqsupset$ -cancellation. Parameters in AL can be renamed, if abstracted over, but role indices cannot without changing the object.

I will not embark here into an argument about whether it is wrong to speak of “abstraction” when there is no (full) α -equivalence, though it seems legitimate to me to use the term even in the case of free variables. It is more important to see that it is not a virtue in itself to be able to rename bound variables, as this means that computational work has to be done to establish a trivial identity of objects. Its virtue, as we shall see, lies in obtaining a total substitution operation, which

⁵Robin Cooper is the notable exception.

is essential for the full exploitation of β -equivalences.

The idea that variables may be essential to the identity of a semantic object is certainly now fairly accepted, due to DRT and its “dynamic” children. From a more philosophical perspective I would argue that using variables in this way is no more a contamination of semantic objects by non semantic concepts than the common view that the order of arguments to a many place function should matter. From my perspective the charge of a lack in α -equivalence exposes the corresponding lack of permutability for the curried or otherwise ordered systems.

1.2.2 Axiomatic Semantics

Given the apparent similarities between the semantic theories mentioned, it seems worthwhile to look for a common framework which brings out the similarities as well as the differences between theories like DRT, DMG, and ST more clearly and with formal precision. The core of this framework is my *Simultaneous Abstraction Calculus* (SAC). Semantic theories can be obtained from it in very much the same way as from the Lambda Calculus. I will pursue here an axiomatic approach that follows the model of Property Theory and Situation Theory in taking the fundamental semantic notions of properties, propositions, and truth pretty much at face value, and describes them by explicit axiomatic theories. One advantage of this approach lies in the possibility to incrementally accumulate such axioms as more aspects of the structure of propositions get treated, and more fine grained distinctions between propositions are introduced. One can thus be formally explicit about what is common to different semantic theories and what is not. DMG and DRT for example share a common theory of identity and propositional logic, but diverge in the kinds of quantification they employ, as well as in the syntax–semantics interface. Some form of ST can be obtained by developing a logic of situation types within such unsituated theories of propositions. Furthermore one can obtain results about the consistency of the various assumptions behind different theories, allowing for safe transfer of solutions from one to the other, such as for example the use of Situation Theoretic ideas within DRT or DMG.

My approach is minimalist in its spirit, introducing assumptions only when

necessary into ones theories, as explicit claims rather than implicitly into the definition of the formalism itself. A good example of this is the lack of any typing of our expressions. I do not want to exclude self-application and the like right from the start from my framework, and prefer to allow for different explicit theories that restrict the formation of meaningful propositions.

The topic of relating semantic objects to natural language utterances will be left largely in the background. I believe that given the spirit of axiomatic specification of this work a congenial approach would be to formalize the syntax–semantics interface in a way that leaves as much as possible open about the precise nature of the semantic objects involved.

1.2.3 Bealer’s Puzzle

In [Bealer 1989] George Bealer has argued that any theory which identifies properties with abstractions from propositions, as I am pursuing it, is not fine grained enough for dealing with attitudes. To quote his example, from the proposition ‘Jane Fonda follows Rajneesh’ ($follow(f, r)$) one can form the properties to ‘fondalee’ ($\lambda x. follow(f, x)$) and to ‘rajneesh’ ($\lambda x. follow(x, r)$), so that by functional application we get that (i) ‘Jane Fonda rajneeshes’ and (ii) ‘Rajneesh fondalees’ express the very same proposition $((\lambda x. follow(x, r))(f) = follow(f, r) = (\lambda x. follow(f, x))(r))$. If one agrees that it is possible to believe (i) without (ii), or at least accepts that a plausible example might be constructed along these lines, then one has to admit that functional application, governed by the principle of β -reduction, is guilty of erasing too many distinctions.

Bealer’s solution is to use combinators for filling and reordering argument places, which do not generate equalities like the one between (i) and (ii) above, but only logical equivalences between distinct propositions. For example instead of using abstraction to permute a relation R into $\lambda x \lambda y. R y x$, with the resulting equality $R a b = (\lambda x \lambda y. R y x) b a$, he introduces basic operations like $conv$ and $pred_0$ such that $pred_0(pred_0(conv(R), b), a) \neq pred_0(pred_0(R, a), b)$, but one is true whenever the other is.

One might envisage a transfer of this approach into an unordered setting, yet I find it plausible that the meanings of incomplete phrases, including words like

'fondalee', are learned by grasping what claim they produce in which context of other meaningful words. Sticking to some device of abstraction, it thus seems necessary to invoke a structure preserving notion of predication of propositional functions to their arguments, besides the structure erasing application operation, in order to cope with this puzzle. Bealer's introduction of two kinds of propositions in [Bealer 1982], namely unstructured "conditions" and highly structured "Cambridge propositions" seems to be very much in line with the above distinction. I thus prefer to follow Peter Aczel, [Aczel 1989], and distinguish application from predication in chapter 3.

1.3 Other Frameworks

My approach is very much inspired by the work of Robin Cooper who uses the Situation Theoretic language EKN as a framework in which various semantic theories can be expressed, as in [Cooper 1993]. One problem with his approach is that one seems to be committed to take a formidable number of Situation Theoretic assumptions on board in order to achieve ones often rather limited aims. In [Cooper & Poesio 1994] an attempt is made to deal with this charge by arranging the axioms into convenient packages, under the banner of "algebraic specification," in order to allow for a more minimalist use of the tools. Still, the theory looks quite complex, mainly due to the use of parametric objects with AL style abstraction, and the rather fundamental role that is played by situations and "infons". My hope is that my humbler system of abstraction and more prominent use of propositions can make things easier to understand and to use in applications. In fact as far I have carried out this work the number axioms has remained so small that the use of specification methods would seem out of proportion.

Another notable effort to unify the three semantic theories mentioned above is found in Reinhard Muskens' work. He does not temper with the basic aspects of the rigidly typed lambda calculus, but prefers to modify it to cope with the new challenges. Thus he introduces a new basic type of "states" which by means of some axioms are in one-one correspondence with discourse marker assignments. This allows him to formulate systems of DMG, [Muskens 1992], and DRT,

[Muskens 1994], if in a somewhat indirect manner. The systems are "dynamic" in using DPL-style relational composition, following [Groenendijk & Stokhof 1991], which allows to capture discourse markers to the right of the syntactic scope of a quantifier. Otherwise he sticks to the identification of propositions with functions from worlds to truth values, but in adopting a four values lattice of truth values in [Muskens 1989] he effectively turned possible worlds into (im)possible situations. This created a version of ST, which is very close to the "classical" theory of [Barwise & Perry 1983].

I find my system as lying in between the in some sense more conservative approach of Muskens', and Cooper's embrace of the unfamiliar.

“liker” and “likee” roles of our predicate ‘likes’.

$$(\lambda\{x, y\}. \text{likes}(z_1.x, z_2.y)) (x.\text{rob}, y.\text{dom})$$

By convention differently indexed variables are always assumed to be non-identical.

Definition 1 *The language Λ consists of TERMS $t, t' \dots$ built up from basic CONSTANTS $c, c' \dots$, and $\#$ for “undefined,” and VARIABLES $x, y \dots \in \text{Var}$ by means of ABSTRACTION $\lambda M.t$, for $M \subseteq \text{Var}$, and APPLICATION $t(x_i.t_i)_{i \in I}$.*

Notice that terms of the language Λ are fairly abstract objects, containing sets of variables and unordered records as constituents. We in fact allow infinite sets of variables $\{x_i\}_{i \in I}$ to be abstracted over and also infinite records $(x_i.t_i)_{i \in I}$ to be applied to.¹ Infinite terms are useful for theoretical purposes, even if in applications one will have to restrict oneself to FINITARY Λ , possibly extended by short hand versions of a few selected infinitary expressions.

In writing an expression like ‘ $\lambda\{x_1, \dots, x_n\}.t$ ’ or simply ‘ $\{x_1, \dots, x_n\}.t$ ’ I therefore do not mean to suggest that the terms denoted by these expressions have any order in their set of abstracted variables, as one might suspect from the indexing, and neither do I mean to imply that n is a finite number. A similar remark holds for application terms: all indeces are part of the meta language, and may be attached freely to terms, subject only to the convention that different variables should be indexed differently. To give a concrete example, I take $\{x_1, x_2\}.t$ to be the very same term as $\{x_2, x_1\}.t$, and in the same vein $t(x.t_1, y.t_2)$ and $t(y.t_2, x.t_1)$ to be identical Λ -expressions. Thus Λ is not a language that can be written, but a convenient abstraction over any systems of the appropriate structure which allow for free permutation within any particular abstraction or application. Order is only significant for multiple uses of abstraction or application, for example syntactically (and semantically) $\{x_1\}\{x_2\}.t \neq \{x_2\}\{x_1\}.t$ and $t(x.t_1)(y.t_2) \neq t(y.t_2)(x.t_1)$.

¹The notation ‘ $\{x_i\}_{i \in I}$ ’ stands here for the set $\{x_i | i \in I\}$, not for a function from indices to variables, as it is sometimes used by other authors, such as [Aczel & Lunnon 1991].

Chapter 2

Simultaneous Abstraction

In this part two slightly different formalisms for simultaneous abstraction are introduced. First I look at the conceptually simple, if somewhat unorthodox language Λ , and then at the proof theoretically more convenient Λ_{AL} , which is equivalent to the original Λ , modulo some additional variables. I give a proof theory for Λ_{AL} and show the Church–Rosser property for reduction. I investigate Λ_{AL} from a Combinatory Logic perspective, establishing a completeness result for it. Finally I discuss structured objects, and the solving of equations, which may give us non well-founded objects.

For the full understanding of chapter three the proof theoretic sections, particularly from 2.2.3 to the end of 2.3, are not essential.

2.1 Simple Simultaneous Abstraction

2.1.1 Syntax for Λ

Instead of abstracting over single variables, as in standard λ -calculus, we allow λ -abstraction over any set of them. Terms are applied to records that assign terms to variables. Ways of introducing unitary abstraction and application into such a system will be discussed later on. A purified version of the earlier oenological example, where all applications are now performed uniformly to variable assignments, would now look like this. Let us take z_1 and z_2 to function as the

2.1.2 Semantics for Λ

The semantics for Λ will be a fairly straightforward variation on the semantics of the untyped Lambda Calculus.² The crucial difference is that in our system the objects denoted by our terms apply not to objects, but rather to partial assignments of objects to variables. We thus need a set D of objects and a way to apply those objects to assignments: with every object $d \in D$ we need to associate a function $\Psi(d)$, which I call d 's **APPLICATIVE BEHAVIOR**, telling us which object in D is the value of applying d to which assignment. Thus Ψ is a mapping from D to $(Var \rightarrow D) \rightarrow D$. To deal with partial assignments we assume D comes with a distinguished element \perp_D that plays the role of the “undefined object”. We write $\langle x_i \mapsto \xi_i \rangle_{i \in I}$ for the function f such that $f(x_i) = \xi_i$ for $i \in I$ and $f(x) = \perp$ otherwise, and let $dm(f) =_{df} \{x \mid f(x) \neq \perp\}$. This is enough to make sense of application for our language: ignoring free variables, we get $\|t(x_i, t_i)_{i \in I}\| = \Psi(\|t\|)(\langle x_i \mapsto \|t_i\| \rangle_{i \in I})$.

To interpret an abstraction $\lambda M.t$ we look at the interpretation of t under variation of the objects that may be assigned to the variables in M . In an obvious generalisation from standard usage we define variants g_f^M of an assignment g for a set of variables M and an assignment f by: $g_f^M(x) = f(x)$ if $x \in M$, otherwise $g_f^M(x) = g(x)$, for all $x \in Var$. Thus, given t, M and g there is a function³ $\lambda f_{Var \rightarrow D} \|t\|^{g_f^M} \in (Var \rightarrow D) \rightarrow D$, from assignments f to the interpretation of t under g_f^M . This function defines the applicative behavior of $\lambda M.t$. For the denotation of this term all that is needed is a member of D which has precisely this applicative behavior. We thus assume a mapping Φ from $(Var \rightarrow D) \rightarrow D$ to D which maps any applicative behavior μ to an object $\Phi(\mu)$ which has this behavior, which is to say that $\Psi(\Phi(\mu)) = \mu$. We are thus dealing with a **RETRACTION** between two domains, written $D \rightrightarrows_{\Phi} C$, which is a pair of mappings $\Psi : D \rightarrow C$ and $\Phi : C \rightarrow D$ such that $\Psi \circ \Phi = id_C$.

One final point has to be observed though: $\Psi \circ \Phi = id_C$ implies that Φ is injective, hence $|D| \geq |C|$, which means there cannot be a retraction between

²See [Hindley & Seldin 1986] for an introduction into models for the untyped Lambda Calculus.

³The ‘ λ ’ in the following expression is used as a symbol of the meta language, with the standard meaning.

a set D and the full function space $(Var \rightarrow D) \rightarrow D$. One way to cut down the size of the function space without throwing away some vital functions is to work in the category CPO of complete partial orders with continuous functions between them. So a **DOMAIN** will be understood to be a cpo with a least element \perp , a mapping between cpo's to be continuous, and the operation \rightarrow to form the space of continuous functions, ordered pointwise. The two mappings of a retraction are in addition required to be **STRICT**, that is, to preserve the bottom element.⁴ Later, in section 2.3, I will use a combinatory logic to restrict the space of applicative behaviors without the use of domain theory. I will regard that version of the semantics as the official one.

Definition 2 A Λ -MODEL consists of a retraction $D \rightrightarrows_{\Phi} (Var \rightarrow D) \rightarrow D$, and an interpretation \mathfrak{S} which maps constants into D , with $\mathfrak{S}(\#) = \perp_D$. The denotation function $\|\cdot\|$ for terms, under a variable assignment g , is given by:

- $\|c\|^g = \mathfrak{S}(c)$;
- $\|x\|^g = g(x)$;
- $\|\lambda M.t\|^g = \Phi(\lambda f_{Var \rightarrow D} \|t\|^{g_f^M})$;
- $\|t(x_i, t_i)_{i \in I}\|^g = \Psi(\|t\|^g)(\langle x_i \mapsto \|t_i\|^g \rangle_{i \in I})$.

A model is **NON TRIVIAL** if $|D| > 1$. Working in CPO the denotations are only well-defined because the following holds.

Theorem 1 $\lambda f \|t\|^{g_f^M}$ is continuous for all terms t .

PROOF: Notice that $\|t\| = \lambda f \|t\|^{g_f^{Var}}$ is an instance of this. We show by induction that lub's $h^* = \bigsqcup_{n < \omega} h_n$ of ascending chains of assignments are preserved. I leave the proof of monotonicity to the reader.

- $\lambda f \|c\|^{g_f^M}(h^*) = \mathfrak{S}(c) = \bigsqcup_{n < \omega} \lambda f \|c\|^{g_f^M}(h_n)$;

⁴For more background information on these notions see also [Barendregt 1984]. Notice that every partial assignment function is continuous (if strict), given that Var is a flat domain with an added \perp , as I will assume.

- $\lambda f \|x\|^{g_f^M}(h^*) = \|x\|^{g_{h^*}^M} = \sqcup_{n < \omega} \|x\|^{g_{h_n}^M} = \sqcup_{n < \omega} \lambda f \|x\|^{g_f^M}(h_n)$;
- $\lambda f \|\lambda M'.t\|^{g_f^M}(h^*) = \|\lambda M'.t\|^{g_{h^*}^M} = \Phi \lambda f' \|t\|^{g_{h^* f'}^{MM'}} = \Phi \lambda f' \|t\| \sqcup_{n < \omega} g_{h_n f'}^{MM'}$
 $= \Phi \lambda f' \sqcup_{n < \omega} \|t\|^{g_{h_n f'}^{MM'}} = \Phi \sqcup_{n < \omega} \lambda f' \|t\|^{g_{h_n f'}^{MM'}} = \sqcup_{n < \omega} \Phi \lambda f' \|t\|^{g_{h_n f'}^{MM'}}$
 $= \sqcup_{n < \omega} \|\lambda M'.t\|^{g_f^M} = \sqcup_{n < \omega} \lambda f \|\lambda M'.t\|^{g_f^M}(h_n)$;
- $\lambda f \|t(x_i, t_i)_{i \in I}\|^{g_f^M}(h^*) = \|t(x_i, t_i)_{i \in I}\|^{g_{h^*}^M} = \Psi(\|t\|^{g_{h^*}^M})(\langle x_i \mapsto \|t_i\|^{g_{h^*}^M} \rangle_{i \in I})$
 $= \Psi(\sqcup_{n < \omega} \|t\|^{g_{h_n}^M})(\langle x_i \mapsto \sqcup_{m < \omega} \|t_i\|^{g_{h_m}^M} \rangle_{i \in I})$
 $= (\sqcup_{n < \omega} \Psi \|t\|^{g_{h_n}^M})(\sqcup_{m < \omega} \langle x_i \mapsto \|t_i\|^{g_{h_m}^M} \rangle_{i \in I})$
 $= \sqcup_{m < \omega} [(\sqcup_{n < \omega} \Psi \|t\|^{g_{h_n}^M})(\langle x_i \mapsto \|t_i\|^{g_{h_m}^M} \rangle_{i \in I})]$
 $= \sqcup_{m < \omega} \sqcup_{n < \omega} [\Psi(\|t\|^{g_{h_n}^M})(\langle x_i \mapsto \|t_i\|^{g_{h_m}^M} \rangle_{i \in I})]$
 $= \sqcup_{n < \omega} [\Psi(\|t\|^{g_{h_n}^M})(\langle x_i \mapsto \|t_i\|^{g_{h_n}^M} \rangle_{i \in I})] = \sqcup_{n < \omega} \|t(x_i, t_i)_{i \in I}\|^{g_{h_n}^M}$
 $= \sqcup_{n < \omega} (\lambda f \|t(x_i, t_i)_{i \in I}\|^{g_f^M}(h_n))$.

□

2.1.3 Basic Properties of Λ

Definition 3 The FREE VARIABLES of a term are defined by

- $FV(x) = \{x\}$;
- $FV(c) = \emptyset$;
- $FV(\lambda M.t) = FV(t) \setminus M$;
- $FV(t(x_i, t_i)_{i \in I}) = FV(t) \cup \bigcup_{i \in I} FV(t_i)$.

Lemma 2 If $x \notin FV(t)$ then $\|t\|^g = \|t\|^{g_x}$ for all g and d .

So if x is not free in a term t then the denotation of t is independent of what we assign to x , but this does not mean one could rename bound occurrences of x by some y which is fresh for t . For example, the denotation of $\{x\}.x$ is independent of x , but $\|\{x\}.x\|^g \neq \|\{y\}.y\|^g$. The latter becomes clear when we apply these two functions to suitable assignments e.g., $\|\{x\}.x(x.a, y.b)\|^g = \|a\|^g \neq \|b\|^g = \|\{y\}.y(x.a, y.b)\|^g$, presuming our model is non trivial.

Definition 4 SIMULTANEOUS SUBSTITUTION $[s_j/y_j]_{j \in J}$ (dropping the index set when no confusion can arise) is a partially defined operation given by:

- $[s_j/y_j]_{j \in J} x = s_j$, if $x = y_j$ for some $j \in J$, otherwise x ;
- $[s_j/y_j]_{j \in J} c = c$;
- $[s_j/y_j]_{j \in J} \lambda M.t = \lambda M[s_i/y_i]_{i \in I} t$, where $I = J \setminus \{j \mid y_j \in M\}$,
if $M \cap \bigcup_{i \in I} FV(s_i) = \emptyset$, otherwise undefined;⁵
- $[s_j/y_j]_{j \in J} t(x_i, t_i)_{i \in I} = [s_j/y_j]_{j \in J} t(x_i, [s_j/y_j]_{j \in J} t_i)_{i \in I}$.

An example of an undefined substitution would be $[y/x]\{y\}t_{xy}$, where replacing x in t_{xy} by y would give undesired results and renaming y is not a possibility. To avoid such trouble we require a FRESH VARIABLE for a term to occur neither free nor bound in it.

Let us say that $\models t = t'$ iff for every Λ -model and assignment $\|t\|^g = \|t'\|^g$.

Theorem 3 There are terms t such that:

1. $\not\models \{x_1 \dots x_n\}t = \{y_1 \dots y_n\}[y_i/x_i]t$,
2. $\not\models \{x_1 \dots x_n\}\{x_{n+1} \dots x_{n+m}\}t = \{x_1 \dots x_{n+m}\}t$.

This is obvious if non trivial models exist. For the C_{AL} -models of section 2.3.2 their existence follows from the non triviality of the term model via the Church–Rosser Theorem.

I will now present a few equations, where the list is not meant to give the axioms of a complete calculus for Λ . The fact that substitution is not always defined is responsible for the difficulties in giving such a list. It means that β -reduction cannot be exploited in many cases where one would usually expect it to be useful. Fact 5.3, which allows for some amount of renaming of variables, is just one example of an equation that cannot be derived from the limited β -reduction allowed by fact 5.2, because substitution is partial. It seems a good idea to look for a complete calculus for Λ_{AL} , to be defined later, rather than for Λ , since substitution can be totally defined in that system.

⁵Being undefined is meant here in the sense of not producing any result, rather than resulting in #.

Lemma 4 $\|t\|^{gM} = \|[t_i/x_i]_{i \in I} t\|^g$ if $[t_i/x_i]_{i \in I}$ is defined, where $M = \{x_i\}_{i \in I}$ and $\gamma = \langle x_i \mapsto \|t_i\|^g \rangle_{i \in I}$.

PROOF:

- obvious for atomic terms;
- $\|\lambda N. t\|^{gM} = \Phi \lambda f \|t\|^{gM'} = \Phi \lambda f \|t\|^{g_{I'}^{M'}} = \Phi \lambda f \|[t_i/x_i]_{i \in I'} t\|^{g_{I'}^{M'}}$
 $= \|\lambda N. [t_i/x_i]_{i \in I'} t\|^g = \|[t_i/x_i]_{i \in I} \lambda N. t\|^g$,
 where $I' = I \setminus \{i \mid x_i \in N\}$, $M' = \{x_i\}_{i \in I'}$, $\gamma' = \langle x_i \mapsto \|t_i\|^g \rangle_{i \in I'}$,
 and $\|t_i\|^g = \|t_i\|^{g_{I'}^{M'}}$ for all f and all $i \in I'$;
- $\|t(y_j. s_j)_{j \in J}\|^{gM} = \Psi(\|t\|^{gM})(\langle y_j \mapsto \|s_j\|^{gM} \rangle_{j \in J})$
 $= \Psi(\|[t_i/x_i]_{i \in I} t\|^g)(\langle y_j \mapsto \|[t_i/x_i]_{i \in I} s_j\|^g \rangle_{j \in J}) = \|[t_i/x_i]_{i \in I} t(y_j. s_j)_{j \in J}\|^g$.

□

Theorem 5 1. $\models \lambda M. t = \lambda N. t$ for $N = M \cap FV(t)$;

2. $\models \{x_i\}_{i \in I} t(x_j. t_j)_{j \in J} = [t_j/x_j, \#/x_i]_{j \in J, i \in I'} t$,
 where $J' = J \cap I$ and $I' = I \setminus J$, if the substitution is defined.
3. $\models (\{x_i\}_{i \in I} t)(x_i. t_i)_{i \in I} = \{z_i\}_{i \in I} [z_i/x_i]_{i \in I} t(z_i. t_i)_{i \in I}$,
 where all z_i are fresh.

PROOF:

1. From lemma 2.
2. By lemma 2 and 4 for every g in all Λ -models:
 $\|\lambda M. t(x_j. t_j)_{j \in J}\|^g = \Psi(\Phi(\lambda f \|t\|^{gM'}))(\langle x_j \mapsto \|t_j\|^g \rangle_{j \in J}) = \|t\|^{gM}$
 $= \|[t_j/x_j, \#/x_i]_{j \in J, i \in I'} t\|^g$,
 where $M = \{x_i\}_{i \in I}$, $\gamma = \langle x_j \mapsto \|t_j\|^g \rangle_{j \in J}$.
3. Again by lemma 2 and 4:
 $\|\{x_i\}_{i \in I} t(x_i. t_i)_{i \in I}\|^g = \|t\|^{gM} = \|[z_i/x_i]_{i \in I} t\|^{gM'}$
 $= \|\{z_i\}_{i \in I} [z_i/x_i]_{i \in I} t(z_i. t_i)_{i \in I}\|^g$, with $M = \{x_i\}_{i \in I}$, $M' = \{z_i\}_{i \in I}$,
 $\gamma = \langle x_i \mapsto \|t_i\|^g \rangle_{i \in I}$, and $\gamma' = \langle z_i \mapsto \|t_i\|^g \rangle_{i \in I}$.

□

2.1.4 Partial Application

It is very natural to ask for an operation for partially filling the argument roles of a relation, leaving some of them simply open for more to come. We use the notation $t[x_1. t_1, \dots, x_n. t_n]$ for partial application. The semantics is as follows:

$$\|t[x_i. t_i]_{i \in I}\|^g =_{df} \Phi \lambda f (\Psi \|t\|^g)(f_{(x_i \mapsto \|t_i\|^g)_{i \in I}}^{\{x_i\}_{i \in I}}).$$

The following properties of partial application are of particular interest, letting application associate to the left.

Theorem 6 1. $\models \{x_i\}_{i \in I} t[x_j. t_j]_{j \in J} = [t_j/x_j]_{j \in J} \{x_i\}_{i \in I'} t$, if defined,
 where $J' = J \cap I$ and $I' = I \setminus J$;

$$2. \models t[x_i. t_i]_{i \in I} [y_j. s_j]_{j \in J} = t[x_i. t_i, y_j. s_j]_{i \in I, j \in J}$$

where $J' = J \setminus \{j \mid \exists i y_j = x_i\}$;

$$3. \models t[x_i. t_i]_{i \in I} (y_j. s_j)_{j \in J} = t(x_i. t_i, y_j. s_j)_{i \in I, j \in J}$$

where $J' = J \setminus \{j \mid \exists i y_j = x_i\}$;

$$4. \models t[x_j. t_j]_{j \in J} = \{x_i\}_{i \in I}. t(x_j. t_j, x_i. x_i)_{j \in J, i \in I}$$

if $Var = \{x_i \mid i \in I\} \cup \{x_j \mid j \in J\}$, and no $x_i \in FV(t[x_j. t_j]_{j \in J})$ for $i \in I$.

Notice that the β -conversion of partial application, as expressed in equation No. 1, does reduce the set of abstracted variables, if the substitution is defined, but never removes the abstraction altogether, even if the set $\{x_i\}_{i \in I}$ is empty.

Notice also that we do not have $\models t[] = t$ unless t is an abstraction term. To have this equation hold in full generality would require our models to be **EXTENSIONAL**. For that, Ψ would have to be an injection, which turns our retraction into an isomorphism between D and $(Var \rightarrow D) \rightarrow D$. But our semantics is only **WEAKLY EXTENSIONAL** in that any two abstracts with the same applicative behavior denote the same object, as they will both denote the value of Φ for this particular behavior.

There seems to be a way to define partial application by means of the fourth equation, or some variant of it which makes equally sure that any further role of t can be filled by some further application. But this cannot be insured because

such roles/variables might occur free in some t_j ,⁶ and thus the side condition of clause 4 will be violated. In Λ_{AL} there will be no such problem for defining partial application any more.

2.2 Aczel–Lunnon Abstraction

The language Λ is closely related to Barwise and Cooper's Extended Kamp Notation (EKN), see [Barwise & Cooper 1991], which is based on Aczel and Lunnon's theory of abstraction. In EKN λ -abstraction operates on injective functions from a set of *role indices* to *parameters*. In the case of Λ those two notions are merged into one, the variables, which allows us to replace the injections by simple sets.

Variables thus play a double role in our system: as role identifiers and as means for directing the values for such roles into the intended slots. We have seen that this creates problems, as by virtue of their former function, variables can often not be renamed and thus certain substitutions, such as $[x/y]\{x\}t_{xy}$, cannot be performed. A similar problem is encountered when we try to find an abstraction term t' which uses the same role x on two levels of abstraction, so that $t'(x.a)(x.b) = t_{ab}$. The simple idea of two abstractions $\{x\}\{x\}t_{xx}$ fails as the outer role x cannot be linked to the correct position in t for reasons of interference with x 's use as a role identifier in the inner abstraction. Using different variables instead creates the problem that application can no longer be to assignments to x in both arguments.

The following system revises Λ by separating the two uses of variables, using AL-style syntax. The old variables $x, y, \dots \in Var$ will now be used as roles only, while upper case variables, called “parameters” go into the positions of terms, to assign values into their intended positions. Our standard example becomes now

$$(\lambda\langle X \mapsto x, Y \mapsto y \rangle \text{likes}(z_1.X, z_2.Y)) (x.\text{rob}, y.\text{dom}),$$

where I have turned the mapping $\langle x \mapsto X, y \mapsto Y \rangle$, as it would appear in AL,

⁶They might even be free in t itself, if it is not an abstraction.

around for convenience.⁷ The term t' we were looking for above can now be constructed as $\lambda\langle Y \mapsto x \rangle \lambda\langle Z \mapsto x \rangle t_{YZ}$. It thus seems possible that such a move may increase the expressivity for our system. I will show that the new language Λ_{AL} is in fact equivalent to the old Λ , modulo some additional variables, but the proof theory is sufficiently simpler for the new language to warrant its introduction.

Definition 5 *The TERMS t, t', \dots of Λ_{AL} are built up from CONSTANTS $c, \#$, ... and PARAMETERS X, Y, \dots by means of ABSTRACTION $\lambda\chi.t$ over injective functions χ from parameters to variables (=roles), and APPLICATION $t(x_i, t_i)_{i \in I}$.*

Semantically we stay in the same space of domains $D \rightleftharpoons_{\Phi} (Var \rightarrow D) \rightarrow D$ as for Λ , with the only difference being that we now interpret under assignments of objects to parameters, called ANCHORS, and that these have to be obtained from assignments to roles when we define the meaning of abstraction.

Definition 6 *A Λ_{AL} -MODEL is a Λ -model, where the denotation function $\|\cdot\|$ for terms, under an anchor g , is given by:*

- $\|c\|^g = \mathfrak{Q}(c)$;
- $\|X\|^g = g(X)$;
- $\|\lambda\chi.t\|^g = \Phi \lambda f_{Var \rightarrow D} \|t\|^g \stackrel{dm(\chi)}{f \circ \chi}$;
- $\|t(x_i, t_i)_{i \in I}\|^g = \Psi(\|t\|^g)(\langle x_i \mapsto \|t_i\|^g \rangle_{i \in I})$.

Given that we always have enough fresh parameters at our disposal, which can be insured by requiring $|Par| > |Var|$, we can define PARTIAL APPLICATION by

$$t[x_i, t_i]_{i \in I} =_{df} \lambda\langle X_j \mapsto x_j \rangle_{j \in J} t(x_i, t_i, x_j.X_j)_{i \in I, j \in J},$$

⁷In AL, the mappings in abstractions take roles to parameters injectively. In turning them around we could be more general, by allowing several parameters to be mapped to the same role, which is equivalent to replacing those parameters with one of them, which then gets mapped to the role in question. Notice that associating one parameter with many roles in one abstraction clearly makes no sense. My semantics for Λ_{AL} will not presuppose injectivity, but I will assume it here nevertheless for convenience.

where $\{x_j | j \in J\} = \text{Var} \setminus \{x_i | i \in I\}$ and the X_j are fresh. The resulting semantics for partial application is the same as for the basic operation we added to Λ . Notice that infinitary abstraction is used to achieve such expressivity.

The move to Λ_{AL} solves the problems with substitution in Λ , as substitution can now be totally defined by means of renaming parameters, so that for example $[X/Y]\lambda\langle X \mapsto z \rangle.t_{XY} = \lambda\langle Z \mapsto z \rangle.t_{ZX}$.

2.2.1 Relating Λ , Λ_{AL} , and λ

The question that needs to be addressed is whether we gain more from the switch from Λ to Λ_{AL} than just convenience. There might be terms in the new language which have no equivalent in the original language Λ . To show that no serious gain in expressive power is made we have to face the earlier difficulty for Λ again, namely that of allowing multiple uses of roles in different levels of abstraction, without clash of variables. Such “role recycling” is made easy in Λ_{AL} by using different parameters in a term like $\lambda\langle Y \mapsto x \rangle\lambda\langle Z \mapsto x \rangle.t_{YZ}$. In creating a Λ -term with the same role x used twice we faced the problem of the outer abstraction in $\{x\}\{x\}.t_{xx}$ being vacuous, thus failing to connect x to the intended position. The solution is to inject an intermediate role y to arrive at $\{x\}(\{\{y\}\{x\}t_{yx})(y.x)$. To have such roles available at all times we expand our language Λ to Λ' with its new set of variables $x, y, \dots \in \text{Var}' = \text{Var} \cup \text{Par}$. The idea can now be applied in full generality.

Definition 7 The TRANSLATION $[x_i/X_i]_{i \in I}^\dagger : \Lambda_{AL} \rightarrow \Lambda'$ under a substitution is defined by:

- $[x_i/X_i]_{i \in I}^\dagger X_i = x_i;$
- $[x_i/X_i]_{i \in I}^\dagger c = c;$
- $[x_i/X_i]_{i \in I}^\dagger \lambda\langle Y_j \mapsto y_j \rangle_{j \in J}.t$
 $= (\lambda\{X_i\}_{i \in I'} \lambda\{y_j\}_{j \in J}. [y_j/Y_j, x_i/X_i]_{j \in J, i \in I'}^\dagger t) (X_i, x_i)_{i \in I'}$,
where $K = I \setminus \{i | \exists j \in J X_i = Y_j\}$, $I' = K \setminus \{i | \exists j \in J x_i = y_j\}$,
and $I'' = K \cap \{i | \exists j \in J x_i = y_j\};$
- $[x_i/X_i]_{i \in I}^\dagger t(y_j.t_j)_{j \in J} = [x_i/X_i]_{i \in I}^\dagger t(y_j.[x_i/X_i]_{i \in I}^\dagger t_j)_{j \in J}.$

It is of course possible to interpret Λ_{AL} -terms in the expanded Λ' -models. So for any such model we have the following fact.⁸

Lemma 7 $\|t\|^g = \|[x_i/X_i]_{i \in I}^\dagger t\|^{g \circ \pi}$, where $\pi = \langle x_i \mapsto X_i \rangle_{i \in I}$.

PROOF: The proof proceeds by induction over Λ_{AL} -terms, with the abstraction clause providing the only non-trivial step. For this let us write M for $\{y_j\}_{j \in J}$, μ for $\langle Y_j \mapsto y_j \rangle_{j \in J}$, and π' for $\langle x_i \mapsto X_i \rangle_{i \in I'}$. Then

$$\begin{aligned} & \|[x_i/X_i]_{i \in I}^\dagger \lambda\langle Y_j \mapsto y_j \rangle_{j \in J}.t\|^{g \circ \pi} \\ &= \|\lambda\{X_i\}_{i \in I'} \lambda\{y_j\}_{j \in J} [x_i/X_i, y_j/Y_j]_{i \in I', j \in J}^\dagger (X_i, x_i)_{i \in I'}\|^{g \circ \pi} \\ &= \|\lambda\{y_j\}_{j \in J} [x_i/X_i, y_j/Y_j]_{i \in I', j \in J}^\dagger\|^{g \circ \pi'} \\ &= \Phi \lambda f \|[x_i/X_i, y_j/Y_j]_{i \in I', j \in J}^\dagger\|^{(g \circ \pi')^M} \\ &= \Phi \lambda f \|[x_i/X_i, y_j/Y_j]_{i \in I', j \in J}^\dagger\|_{f \circ \mu}^{dm(\mu) \circ (\pi' \cup \mu^{-1})} \\ &= \Phi \lambda f \|t\|_{f \circ \mu}^{dm(\mu)} \\ &= \|\lambda\langle Y_j \mapsto y_j \rangle_{j \in J}.t\|^g \end{aligned}$$

□

Translation from Λ to Λ_{AL} is a fairly trivial matter.

Definition 8 Let $\rho : \text{Var} \rightarrow \text{Par}$ be some fixed injection. We define an injection $\ddagger : \Lambda \rightarrow \Lambda_{AL}$, based on ρ , by

- $c^\ddagger = c;$
- $x^\ddagger = \rho(x);$
- $(\lambda M.t)^\ddagger = \lambda\langle x^\ddagger \mapsto x \rangle_{x \in M}.t^\ddagger;$
- $t(x_i.t_i)_{i \in I}^\ddagger = t^\ddagger(x_i.t_i^\ddagger)_{i \in I}.$

Theorem 8 Λ can be embedded in Λ_{AL} and Λ_{AL} can be embedded in Λ' .

Let us turn to λ , the untyped λ -calculus, see [Hindley & Seldin 1986]. It is clear that we can define unary abstraction $\lambda X.t$ and application $t(t')$ for Λ_{AL} in

⁸This Lemma does not depend on any assumptions about the size of Par . Hence the following encodability results go through with $|\text{Var}| = |\text{Par}| = |\text{Var}'|$.

terms of a designated role z , used solely for that purpose, as $\lambda\langle X \mapsto z \rangle.t$ and $t(z.t')$ respectively. By the translations between Λ_{AL} and Λ' we see that unary abstraction/application for Λ -terms can be encoded in Λ' ,⁹ allowing us to use expressions such as $\lambda x.t$ and $t(t')$ as shorthands for Λ_{AL} - or Λ' -terms, according to our preferences.

Corollary 9 *λ -abstraction and application for Λ -terms can be encoded in Λ' .¹⁰*

I will from now on take Λ_{AL} as my official language for the Simultaneous Abstraction Calculus, and regard the use of Λ as well as standard λ -terms as syntactic sugar. Occurrences of small variables x as terms, or in abstractions $\lambda x.t$ and $\lambda\{\dots, x, \dots\}.t$, can be taken as shorthands for their corresponding \dagger -translations into Λ_{AL} , based on the “upper-case” mapping from variables to parameters, and the encoding of λ .

2.2.2 Proof Theory for Λ_{AL}

The complications in clause three of the translation from Λ_{AL} to Λ indicate why a complete and reasonably simple set of equations for Λ will be hard to find. The slightly more complex language Λ_{AL} seems more suitable for this task. The notions of free parameters and substitution are assumed to be defined in straightforward adaption of these notions from the λ -calculus, to be found in [Hindley & Seldin 1986].

⁹It is obvious that λ -terms, for example $\lambda x\lambda y.xy$, cannot be encoded in Λ in the naive way, as $\lambda\{x\}\lambda\{y\}.x(z.y)$ or something of similar simplicity. If we follow the path suggested, via Λ_{AL} , the example term becomes $\lambda\langle X \mapsto z \rangle\lambda\langle Y \mapsto z \rangle.X(z.Y)$. Translation into Λ' proceeds as follows:

$$\begin{aligned} \llbracket \dagger \lambda\langle X \mapsto z \rangle\lambda\langle Y \mapsto z \rangle.X(z.Y) \rrbracket &= \\ (\lambda\{\} \lambda\{z\} [z/X] \dagger \lambda\langle Y \mapsto z \rangle.X(z.Y)) () &=_{\beta} \\ \lambda\{z\} [z/X] \dagger \lambda\langle Y \mapsto z \rangle.X(z.Y) &= \\ \lambda\{z\} ((\lambda\{X\} \lambda\{z\} [z/Y] \dagger X(z.Y))(X.z)) &= \\ \lambda\{z\} ((\lambda\{X\} \lambda\{z\}.X(z.z))(X.z)). \end{aligned}$$

Notice that the α -equivalence of $\lambda x\lambda y.xy$ and $\lambda x'\lambda y.x'y$ becomes, under this encoding, an instance of Theorem 5.3, which allows us to rename X by X' in the resulting Λ' -term.

¹⁰The infinitary nature of Λ , and the treatment of application to over- and under-defined assignments, rule out a similarly straightforward encoding in λ . I have not investigated the matter any further. Notice that λ is encoded in finitary Λ_{AL} .

Definition 9 *We write $Ax \vdash_{\beta} t = t'$ iff $t = t'$ can be derived when the set of equations Ax is added to the calculus below. $t =_{\beta} t'$ stands for $\vdash_{\beta} t = t'$, and $t \triangleright t'$ iff $\vdash_{\beta} t = t'$ can be proved without using *R1*.*

A1 $t = t$;

A2 $t(x_i.t_i, x_j.\#)_{i \in I, j \in J} = t(x_i.t_i)_{i \in I}$;

A3 $\lambda\chi.t = \lambda\xi.t$ for $\xi = \chi|_{FP(\xi)}$;

A4 $\lambda\langle X_i \mapsto x_i \rangle_{i \in I} t = \lambda\langle Y_i \mapsto x_i \rangle_{i \in I} [Y_i/X_i]_{i \in I} t$ for fresh Y_i 's;

A5 $\lambda\langle X_i \mapsto x_i \rangle_{i \in I} t (x_j.t_j)_{j \in J} = [t_j/X_j, \#/X_i]_{j \in J, i \in I'} t$
with $J' = J \cap I$ and $I' = I \setminus J$;

A6 $\#(x_j.t_j)_{j \in J} = \#$;

A7 $\lambda\chi.\# = \#$;

R1 $t = t' / t' = t$;

R2 $t = t', t' = t'' / t = t''$;

R3 $t = t', s_i = s'_i (i \in I) / t(x_i, s_i)_{i \in I} = t'(x_i, s'_i)_{i \in I}$;

R4 $t = t' / \lambda\chi.t = \lambda\chi.t'$.

Given infinitary terms, we are stretching the notion of “proof” somewhat with this system: “Being a proof” is not recursive, as termhood is not, and we also have a potentially infinitary rule *R3*. If we restrict the calculus to equations between finite terms we get a notion of proof which is much better behaved. I will call that system the **FINITARY β -CALCULUS**.¹¹ Notice that β -reduction of a finite term never produces an infinite one, even in the infinitary calculus, as there is no Axiom or rule other than *R1*, by which to obtain an equation $t = s$ between a finite t and an infinite s from equations that are not of this form.

Theorem 10 (Soundness) $t =_{\beta} t' \Rightarrow \models t = t'$.

¹¹It is clear that “having a proof” is not going to be recursive even for the finitary calculus, by the embedding of λ in finitary Λ_{AL} .

2.2.3 The Church–Rosser Theorem

The reduction relation \triangleright is of crucial importance for computational concerns, as it provides us with a way of simplifying terms, possibly to a *normal form*, which is a term that cannot be further reduced. By the embedding of the λ -calculus, given earlier, it is obvious though that reduction does not always lead to smaller terms, or terminate at some normal form.¹² In such a situation it is highly desirable to be sure that reduction cannot lead into blind alleys, from which a possibly existing normal form cannot be found. It is also critical to have a unique one, if existent, at least up to the renaming of bound parameters. Both these facts are consequences of the **CHURCH–ROSSER PROPERTY**, or **CONFLUENCE**, of the calculus, which says that any two terms that are β -equal reduce to a common term. The following proof uses Tait's method, as described in [Hindley & Seldin 1986], to establish confluence for reduction, and may be skipped without loss for the understanding of later sections.

- Definition 10** 1. A **REDEX** is an occurrence of a term t , possibly inside another term s , which can be contracted by one of the axioms $A2, A3, A5, A6, A7$.
2. $t =_{\alpha} t'$ iff $t = t'$ can be proved from $A1, A4, R1 \dots R4$. This is called a **CHANGE OF BOUND PARAMETERS**.
3. $t \triangleright_{A_n} t'$ iff $t = t'$ can be proved from $A1, R1 \dots R4$ and a single contraction of a redex r in t by A_n , where $n \in \{2, 3, 5, 6, 7\}$. This is called a **ONE STEP REDUCTION** of t over redex r .

A reduction $t \triangleright t'$ can thus be split into a sequence of changes of bound parameters and one step reductions. From a one step reduction of t to t' we get various **RESIDUALS** r' of occurrences of terms r in t . Let s be the redex contracted to s' in a one step reduction:

- if r is not part of s or vice versa then the unchanged r is r' 's residual;
- if $r = s$ then r has no residual;

¹²For example take $t = (\lambda x.f(xx))(\lambda x.f(xx))$, which reduces to $t \triangleright ft \triangleright f(ft) \triangleright \dots$ without ever finding a normal form.

- if s is contained in r then the r' which results from replacing s by s' in r is r' 's residual;
- the case of r being part of s will not be relevant here.

The notion of residuals is transitively extended to sequences of one step reductions.

Definition 11 A **MINIMAL COMPLETE DEVELOPMENT (MCD)** for a term t and a set of redexes r_1, \dots, r_n in t is a sequence of one step reductions, one per r_i , such that any residual r'_i of a redex r_i is contracted after every residual r'_j of any redex r_j inside r_i .¹³ This may be followed by some changes of bound parameters. We write $t \triangleright_{mcd} s$ iff there is an mcd for t resulting in s .

Lemma 11 1. $t \triangleright s \Rightarrow FP(s) \subseteq FP(t)$.

$$2. t =_{\alpha} t' \wedge s_i =_{\alpha} s'_i (i \in I) \Rightarrow [s_i/X_i]_{i \in I} t =_{\alpha} [s'_i/X_i]_{i \in I} t'.$$

$$3. t \triangleright_{mcd} s \wedge t =_{\alpha} t' \Rightarrow t' \triangleright_{mcd} s.$$

Lemma 12 $t \triangleright_{mcd} t' \wedge s_i \triangleright_{mcd} s'_i (i \in I) \Rightarrow [s_i/X_i]_{i \in I} t \triangleright_{mcd} [s'_i/X_i]_{i \in I} t'$.

PROOF: By the previous lemma we can assume without loss of generality that no X_i or free parameter of any s_i (and thus of any s'_i) is bound in t . Also we may assume the given mcd's not to contain any α -steps. By induction on t :

- $t = X, t = c$ are trivial.
- $t = \lambda \xi.p$
Let $t' = \lambda \zeta.p'$, with $p \triangleright_{mcd} p'$ and either (i) $\zeta = \xi$ or (ii) $\zeta = \xi|_{FP(p')}$ $A3$ -contracting t 's residual in the last step. Then $[s_i/X_i]t = \lambda \xi.[s_i/X_i]p \triangleright_{mcd} \lambda \xi.[s'_i/X_i]p' = [s'_i/X_i]t'$ in case (i), while in case (ii) $\lambda \xi.[s'_i/X_i]p' \triangleright_{A3} \lambda \zeta.[s'_i/X_i]p' = [s'_i/X_i]t'$, which is still an mcd.
- Case (iii) is $t' = \#$ from $A7$ applied to $\lambda \xi.\#$. Then $[s_i/X_i]t = \lambda \xi.[s_i/X_i]p \triangleright_{mcd} \lambda \xi.[s'_i/X_i]\# = \lambda \xi.\# \triangleright_{A7} \#$ is an mcd.

¹³The reduction order may not be fully determined by this.

- $t = p(y_j, q_j)_{j \in J}$

For case (i) where $t' = p'(y_j, q'_j)_{j \in J'}$, $p \triangleright_{mcd} p'$, $q_j \triangleright_{mcd} q'_j$ for $(j \in J)$ and $J = J'$ or t 's residual was $A2$ -contracted in the last step, we have $[s_i/X_i]t = [s_i/X_i]p (y_j, [s_i/X_i]q_j)_{j \in J} \triangleright_{mcd} [s'_i/X_i]p' (y_j, [s'_i/X_i]q'_j)_{j \in J} = [s'_i/X_i]t'$ or $\triangleright_{A2} [s'_i/X_i]t'$, which is an mcd.

Case (ii) has $t' = [q'_j/Y_j, \#/Y_k]_{j \in J', k \in K'} m$ from using $A5$ on t 's residual, with $p \triangleright_{mcd} p' = \lambda \langle Y_k \mapsto y_k \rangle_{k \in K} m$ and $q_j \triangleright_{mcd} q'_j$ ($j \in J$). Then $[s_i/X_i]t = [s_i/X_i]p (y_j, [s_i/X_i]q_j)_{j \in J} \triangleright_{mcd} [s'_i/X_i]p' (y_j, [s'_i/X_i]q'_j)_{j \in J} = \lambda \langle Y_k \mapsto y_k \rangle_{k \in K} [s'_i/X_i]m (y_j, [s'_i/X_i]q'_j)_{j \in J} \triangleright_{A5} [[s'_i/X_i]q'_j/Y_j, \#/Y_k]_{j \in J', k \in K'} [s'_i/X_i]m = [s'_i/X_i]t'$, which is an mcd.

Case (iii) is $t' = \#$ from $A6$ applied to $\#(y_j, q'_j)_{j \in J}$. Then $[s_i/X_i]t = [s_i/X_i]p (y_j, [s_i/X_i]q_j)_{j \in J} \triangleright_{mcd} [s'_i/X_i]\# (y_j, [s'_i/X_i]q'_j)_{j \in J} \triangleright_{A6} \#$, which is an mcd.

□

Lemma 13 $t \triangleright_{mcd} u \wedge t \triangleright_{mcd} v \Rightarrow \exists w u \triangleright_{mcd} w \wedge v \triangleright_{mcd} w$.

PROOF: We may again assume the given mcd's not to contain any α -steps. By induction on t :

- $t = X$, $t = c$ are trivial.
- $t = \lambda \xi p$

Let $u = \lambda \xi' p'$ and $v = \lambda \xi'' p''$, both possibly derived by an $A3$ -contraction of t 's residual in the last step. By induction hypothesis there is a p^+ with $p' \triangleright_{mcd} p^+$ and $p'' \triangleright_{mcd} p^+$. Chose $w = \lambda \xi^+, p^+$, where $\xi^+ = \xi|_{FP(p^+)}$.

If u or v are $\#$, using $A7$ in the last step, choose $w = \#$. The mcd is obtained by $p^+ = \#$ using $A7$.

- $t = p(y_j, q_j)_{j \in J}$

(i) For the case that $u = p'(y_j, q'_j)_{j \in J'}$ and $v = p''(y_j, q''_j)_{j \in J''}$, both possibly involving an $A2$ -contraction on t 's residual, we choose $w = p^+(y_j, q_j^+)_{j \in J^+}$, where $p' \triangleright_{mcd} p^+$, $p'' \triangleright_{mcd} p^+$, $q'_j \triangleright_{mcd} q_j^+$, $q''_j \triangleright_{mcd} q_j^+$ for $j \in J$, and $J^+ = \{j \in J \mid q_j^+ \neq \#\}$.

(ii) If u or v are $\#$, from $A6$ in the last step, choose $w = \#$. The mcd is obtained by $p^+ = \#$ as in (i), using $A6$ for the last step.

(iii) Otherwise u_i and possibly v_i are derived using $A5$ for t 's residual. So let $t = \lambda \langle Y_k \mapsto y_k \rangle_{k \in K} m (y_j, q_j)_{j \in J}$, and u, v come from $u' = \lambda \langle Y_k \mapsto y_k \rangle_{k \in K} m' (y_j, q'_j)_{j \in J}$ and $v' = \lambda \langle Y_k \mapsto y_k \rangle_{k \in K} m'' (y_j, q''_j)_{j \in J}$. There are m^+, q_j^+ with $m' \triangleright_{mcd} m^+$, $m'' \triangleright_{mcd} m^+$, $q'_j \triangleright_{mcd} q_j^+$, and $q''_j \triangleright_{mcd} q_j^+$ for $j \in J$. Choose $w = [q_j^+/Y_j, \#/Y_k]_{j \in J^+, k \in K^+} m^+$, with $J^+ = J \cap K$ and $K^+ = K \setminus J$.

Let $u = [q'_j/Y_j, \#/Y_k]_{j \in J', k \in K'} m'$ come from u' by $A5$, possibly contracting p 's residual by $A3$ beforehand. Then $u \triangleright_{mcd} w$ by the previous lemma.

If v comes from v' in the same way, the same argument applies. Otherwise $v = v'$ or it is derived from v' by $A2$ or $A3$. Let $v = \lambda \langle Y_k \mapsto y_k \rangle_{k \in K^*} m'' (y_j, q''_j)_{j \in J^*}$. Then $v \triangleright_{mcd} \lambda \langle Y_k \mapsto y_k \rangle_{k \in K^*} m^* (y_j, q_j^*)_{j \in J^*}$ with $m^* =_\alpha m^+$ and $q_j^* =_\alpha q_j^+$, without any α -steps. By a final $A5$ -step we get $[q_j^*/Y_j, \#/Y_k]_{j \in J^*, k \in K^*} m^* =_\alpha w$ by lemma 13.2, whence $v \triangleright_{mcd} w$.

□

Theorem 14 (Church–Rosser Theorem) $t =_\beta t' \Rightarrow \exists s t \triangleright s \wedge t' \triangleright s$.

PROOF: If $t =_\beta t'$ then there are $t = t_1, t_2, \dots, t_n = t'$ such that $t_i \triangleright_{mcd} t_{i+1}$ or $t_{i+1} \triangleright_{mcd} t_i$. The claim follows by induction on n using the previous lemma. □

2.3 Combinatory Aczel–Lunnon Logic

In this section I will develop Combinatory Aczel–Lunnon Logic (CALL) which relates to the SAC in the way that standard Combinatory Logic relates to the λ -calculus. It provides us with a more general notion of a model for the SAC than the previous domain-theoretic one. I will obtain a completeness result for the proof theory of Λ_{AL} relative to these models.

This section is quite technical, and presupposes some familiarity with Combinatory Logic and its relation to the λ -calculus, on the level of the relevant parts of [Hindley & Seldin 1986]. It may be skipped without much loss for the later chapters.

Definition 12 *The language C_{AL} consists of TERMS T, U, T', \dots , built up from CONSTANTS $\#, c, c', \dots, \mathbf{S}$ and \mathbf{K}_x for each $x \in \text{Var}$, and PARAMETERS X, Y, \dots by means of APPLICATION $T(x_i, T_i)_{i \in I}$.*

We use z as a distinguished variable for unary application, writing $T(T')$ for $T(z.T')$. For the interpretation of C_{AL} we do not use cpo's but flat sets D_\perp with an added “undefined” object \perp . Hence functions are no longer required to be continuous either. The FREE PARAMETERS $FP(T)$ of a term T are simply the parameters occurring in T , as we have no abstraction mechanism in C_{AL} .

Definition 13 *A C_{AL} -ALGEBRA consists of a set D_\perp , an operation $\bullet : (D_\perp \times (\text{Var} \rightarrow D_\perp)) \rightarrow D_\perp$, and an interpretation $\mathfrak{S} : \text{Const} \rightarrow D_\perp$, such that $\mathfrak{S}(\#) = \perp_D$, $\mathfrak{S}(\mathbf{S}) = \mathbf{s}$, and $\mathfrak{S}(\mathbf{K}_x) = \mathbf{k}_x$ for all $x \in \text{Var}$, where for all $f, h, k \in (\text{Var} \rightarrow D_\perp)$:*

1. $\mathbf{k}_x \bullet f \bullet h = f(x)$ for $x \in \text{Var}$;
2. $\mathbf{s} \bullet f \bullet h \bullet k = f(z) \bullet k \bullet \langle x \mapsto h(x) \bullet k \rangle_{x \in \text{Var}}$;
3. $\perp_D \bullet f = \perp_D$.

The denotation of terms under an anchoring $g : \text{Par} \rightarrow D_\perp$ is inductively obtained by:

1. $\|c\|^g = \mathfrak{S}(c)$;
2. $\|X\|^g = g(X)$;
3. $\|T(x_i, T_i)_{i \in I}\|^g = \|T\|^g \bullet \langle x_i \mapsto \|T_i\|^g \rangle_{i \in I}$.

Lemma 15 $X \notin FP(T) \Rightarrow \|T\|^g = \|T\|^g \mathbf{x}$.

We write $\models_{CL} T = T'$ if $\|T\|^g = \|T'\|^g$ in every C_{AL} -algebra. The proof theory for CALL looks like this.

Definition 14 *We write $T =_{CL} T'$ iff $T = T'$ can be proved from the axiom schemata and rules below.*

A1 $T = T$;

A2 $T(x_i, T_i, x_j, \#)_{i \in I, j \in J} = T(x_i, T_i)_{i \in I}$;

A3 $\mathbf{K}_{x_j}(x_i, T_i)_{i \in I}(y_k, U_k)_{k \in K} = T_j$,
if $j \in I$, otherwise $\#$;

A4 $\mathbf{S}(x_i, T_i)_{i \in I}(y_j, T'_j)_{j \in J}(v_k, T''_k)_{k \in K} = T_i(v_k, T''_k)_{k \in K}(y_j, T'_j(v_k, T''_k)_{k \in K})_{j \in J}$,
if $z = x_i$ for some $i \in I$, otherwise $\#$;

A5 $\#(x_i, T_i)_{i \in I} = \#$;

R1 $T = T' / T' = T$;

R2 $T = T', T' = T'' / T = T''$;

R3 $T = T', U_i = U'_i (i \in I) / T(x_i, U_i)_{i \in I} = T'(x_i, U'_i)_{i \in I}$;

Theorem 16 (Soundness) $T =_{CL} T' \Rightarrow \models_{CL} T = T'$.

2.3.1 Completeness for C_{AL}

Definition 15 Let $[T] =_{df} \{T' \mid T =_{CL} T'\}$. We define the C_{AL} -TERM ALGEBRA \mathcal{T}_{CL} by $D_{\perp} =_{df} \{[T] \mid T \in C_{AL}\}$, $\perp =_{df} [\#]$, $\mathfrak{S}(c) =_{df} [c]$ for all constants, and $[T] \bullet \langle x_i \mapsto [T_i] \rangle_{i \in I} =_{df} [T(x_i, T_i)_{i \in I}]$.

Lemma 17 \mathcal{T}_{CL} is a C_{AL} -algebra.

PROOF: By A2 and R3 the operation \bullet is well-defined for \mathcal{T}_{CL} . We need to check the axioms.

1. $[K_{x_k}] \bullet \langle x_i \mapsto [T_i] \rangle_{i \in I} \bullet \langle y_j \mapsto [T'_j] \rangle_{j \in J} = [K_{x_k}(x_i, T_i)_{i \in I}(y_j, T'_j)_{j \in J}] = [T_k]$ if $k \in I$, else $[\#]$, whence $\langle x_i \mapsto [T_i] \rangle_{i \in I}(x_k)$.
2. $[S] \bullet \langle x_i \mapsto [T_i] \rangle_{i \in I} \bullet \langle y_j \mapsto [T'_j] \rangle_{j \in J} \bullet \langle v_k \mapsto [T''_k] \rangle_{k \in K}$
 $= [T_i(v_k, T''_k)_{k \in K}(y_j, T'_j(v_k, T''_k)_{k \in K})_{j \in J}]$, if $z = x_i$ for some $i \in I$,
otherwise $= [\#]$, and thus
 $= \langle x_i \mapsto [T_i] \rangle_{i \in I}(z) \bullet \langle v_k \mapsto [T''_k] \rangle_{k \in K} \bullet \langle y_j \mapsto [T'_j] \rangle_{j \in J} \bullet \langle v_k \mapsto [T''_k] \rangle_{k \in K}$.
3. $[\#] \bullet \langle x_i \mapsto [T_i] \rangle_{i \in I} = [\#(x_i, T_i)_{i \in I}] = [\#]$.

□

Lemma 18 $\|T\|^g = [T]$, where $g(X) = [X]$ for $X \in Par$.

PROOF: $\|c\|^g = [c]$ and $\|X\|^g = [X]$ by definition. By induction $\|T(x_i, T_i)_{i \in I}\|^g = \|T\|^g \bullet \langle x_i \mapsto \|T_i\|^g \rangle_{i \in I} = [T] \bullet \langle x_i \mapsto [T_i] \rangle_{i \in I} = [T(x_i, T_i)_{i \in I}]$. □

Theorem 19 (Completeness) $\models_{CL} T = T' \Rightarrow T =_{CL} T'$.

PROOF: If $T \not\equiv_{CL} T'$ then $[T] \neq [T']$ and thus $\mathcal{T}_{CL} \not\models_{CL} T = T'$ by the previous lemma. □

2.3.2 Abstraction in C_{AL} -models

The operations $\bullet : (D_{\perp} \times (Var \rightarrow D_{\perp})) \rightarrow D_{\perp}$ and $\Psi : D_{\perp} \rightarrow ((Var \rightarrow D_{\perp}) \rightarrow D_{\perp})$ are interdefinable by $d \bullet f = \Psi(d)(f)$. The range of Ψ are the REPRESENTABLE FUNCTIONS of a given C_{AL} -algebra, written $((Var \rightarrow D_{\perp}) \rightarrow D_{\perp})_{rep}$. For Λ -models these were just the continuous functions w.r.t. a given order. Now we have \mathfrak{s} and the \mathbf{k}_x to make sure we have enough representable functions in order to be able to simulate abstraction. A function $h : D_{\perp} \rightarrow D_{\perp}$ is represented by any $d \in D_{\perp}$ such that $\forall f: Var \rightarrow D_{\perp} \ d \bullet f = h(f(z))$. I will write $d \bullet b$ for $d \bullet \langle z \mapsto b \rangle$ if no confusion will arise.

Let $\mathbf{K} =_{df} \mathbf{K}_z$ and $\Pi_x =_{df} \mathbf{SK}_x \mathbf{K}$. This gives us projections, as $\Pi_{x_j}(x_i, T_i)_{i \in I} =_{CL} T_j$ if $j \in I$, otherwise $\#$. We can now define terms that mirror abstractions as follows.

- Definition 16**
1. $\lambda^* \langle X_i \mapsto x_i \rangle_{i \in I}. X_j =_{df} \Pi_{x_j}$, if $j \in I$;
 2. $\lambda^* \zeta.T =_{df} \mathbf{K}(T)$, if $dm(\zeta) \cap FP(T) = \emptyset$;
 3. $\lambda^* \zeta.(T(x_i, T_i)_{i \in I}) =_{df} S(\lambda^* \zeta.T)(x_i, \lambda^* \zeta.T_i)_{i \in I}$,
if 2 doesn't apply.

Lemma 20 $\|\lambda^* \zeta.T\|^g \bullet f = \|T\|^g \bullet f$.

Lemma 21 $X \in dm(\zeta) \Rightarrow X \notin FP(\lambda^* \zeta.T)$.

Let APPLICATIVE STRUCTURES $\langle D_{\perp}, \bullet \rangle$ be defined like C_{AL} -algebras without mentioning \mathfrak{s} or any \mathbf{k}_x , and V range over terms not containing \mathfrak{S} and the \mathbf{K}_x . I call such expressions PURE terms.

Definition 17 An applicative structure is COMBINATORIALLY COMPLETE if for any pure term V and $\xi_1 \dots \xi_n$ with $FP(V) \subseteq \bigcup_{i=1..n} dm(\xi_i)$, there is a $d \in D_{\perp}$ such that for all $f_1 \dots f_n$ and arbitrary g :

$$d \bullet f_1 \bullet \dots \bullet f_n = \|V\|^g \bullet f_1 \bullet \dots \bullet f_n$$

Theorem 22 Every C_{AL} -algebra is combinatorially complete.

PROOF: From the previous lemmata by taking $d =_{df} \|\lambda^* \xi_1 \dots \lambda^* \xi_n . V\|^g$. \square

We now have ways to translate from Λ_{AL} to C_{AL} and vice versa.

Definition 18 $(\cdot)_{CL} : \Lambda_{AL} \rightarrow C_{AL}$ is defined by:

1. $X_{CL} =_{df} X$ for all parameters;
2. $c_{CL} =_{df} c$ for all constants;
3. $(t(x_i, t_i)_{i \in I})_{CL} =_{df} t_{CL}(x_i, (t_i)_{CL})_{i \in I}$;
4. $(\lambda \zeta . t)_{CL} =_{df} \lambda^* \zeta . t_{CL}$.

Definition 19 $(\cdot)_{\Lambda} : C_{AL} \rightarrow \Lambda_{AL}$ is defined by:

1. $X_{\Lambda} =_{df} X$ for all parameters;
2. $(K_x)_{\Lambda} =_{df} \lambda \langle X \mapsto x \rangle \lambda \langle \cdot \rangle . X$;
3. $S_{\Lambda} =_{df}$
 $\lambda Z \lambda \langle X_i \mapsto x_i \rangle_{x_i \in Var} \lambda \langle Y_j \mapsto y_j \rangle_{y_j \in Var} . Z(y_j, Y_j)_{y_j \in Var} (x_i, X_i(y_j, Y_j)_{y_j \in Var})_{x_i \in Var}$
4. $c_{\Lambda} =_{df} c$ for all other constants;
5. $(T(x_i, T_i)_{i \in I})_{\Lambda} =_{df} T_{\Lambda}(x_i, (T_i)_{\Lambda})_{i \in I}$.

Theorem 23 $T =_{CL} T' \Rightarrow T_{\Lambda} =_{\beta} T'_{\Lambda}$.

As in the case of the λ -calculus and standard CL we do not have $t =_{\beta} t' \Rightarrow t_{CL} =_{CL} t'_{CL}$. Given the completeness for $=_{CL}$, we would therefore not have soundness for β -equality if we interpreted Λ_{AL} -terms naively via their C_{AL} translations. To interpret Λ_{AL} by means of C_{AL} -algebras we need a canonical representative to be singled out for each representable function.

Definition 20 A C_{AL} -MODEL is a C_{AL} -algebra with an additional strict map Φ such that $D_{\perp} \xrightarrow{\Phi} ((Var \rightarrow D_{\perp}) \rightarrow D_{\perp})_{rep}$ is a retraction and $\Phi \circ \Psi$ is represented by some $e \in D_{\perp}$.

Theorem 24 For any C_{AL} -model there is a unique, well-defined interpretation function $\|\cdot\| : \Lambda_{AL} \rightarrow D_{\perp}$ such that

- $\|c\|^g = \mathfrak{S}(c)$;
- $\|X\|^g = g(X)$;
- $\|t(x_i, t_i)_{i \in I}\|^g = \Psi(\|t\|^g)(\langle x_i \mapsto \|t_i\|^g \rangle_{i \in I})$;
- $\|\lambda \chi . t\|^g = \Phi \lambda f \|t\|^g_{f \circ \chi}$.

PROOF: We show by induction on t that the following hold simultaneously:

1. $\|t\|^g$ is well-defined for all g ;
2. $\|t\|^g = \|t\|^g_{\mathfrak{X}}$ if $X \notin FP(t)$;
3. For all $\xi_1 \dots \xi_n$ with $FP(t) \subseteq \bigcup_{i=1..n} dm(\xi_i)$, there is a $d \in D_{\perp}$ such that for all $f_1 \dots f_n$ and arbitrary g :

$$d \bullet f_1 \bullet \dots \bullet f_n = \|t\|^g_{f_1 \circ \xi_1 \dots f_n \circ \xi_n}.$$

- For pure terms, in particular for constants and parameters, 1 \perp 3 are trivial, or follow from combinatory completeness.
- For applications $t(x_i, t_i)_{i \in I}$ 1 and 2 are trivial. Let d_t and d_{t_i} satisfy 3 for t and the t_i . By the previous item the following object b exists, where we write $\bar{\chi}$ for $(\chi(X).X)_{X \in dm_{\chi}}$ and require that $rng(\chi_i) = Var$ and $dm(\chi_i) \cap dm(\chi_j) = \emptyset$ for $i \neq j = 1..n$ and the X_i are fresh as well.

$$b =_{df} \|\lambda Y \lambda \langle X_i \mapsto x_i \rangle_{i \in I} \lambda \chi_1 \dots \lambda \chi_n . Y \bar{\chi}_1 \dots \bar{\chi}_n(x_i, X_i \bar{\chi}_1 \dots \bar{\chi}_n)\|^h.$$

Then $d = b \bullet d_t \bullet \langle x_i \mapsto d_{t_i} \rangle_{i \in I}$ satisfies 3.

- For abstractions $\lambda \chi . t$ let d_t fulfill 3 for t and $\xi_1 \dots \xi_n, \chi_i$ and let $f_i(x) = g(\xi_i^{\perp 1}(x))$ for $i = 1..n$. Then

$$a =_{df} d_t \bullet f_1 \bullet \dots \bullet f_n$$

represents $\lambda f \|t\|^{g_{f \circ \chi}^{dm(\chi)}}$. Hence $\mathbf{e} \bullet a = \|\lambda \chi . t\|^g$. This proves 1 and 2. For 3 let $\chi_1 \dots \chi_n$ be as above and X, Y fresh. Let

$$b =_{df} \|\lambda X \lambda Y \lambda \chi_1 \dots \lambda \chi_n . X(Y \bar{\chi}_1 \dots \bar{\chi}_n)\| \bullet \mathbf{e} \bullet d_t$$

Hence $b \bullet f_1 \bullet \dots \bullet f_n = \mathbf{e} \bullet (d_t \bullet f_1 \bullet \dots \bullet f_n) = \|\lambda \chi . t\|^{g_{f_1 \circ \bar{\chi}_1 \dots f_n \circ \bar{\chi}_n}^{dm \bar{\chi}_1 \dots dm \bar{\chi}_n}}$.

□

2.3.3 Completeness for Λ_{AL}

We write $Ax \models_{CLM} t = t'$ if $\|t\|^g = \|t'\|^g$ in every C_{AL} -model that satisfies all the equations in Ax .

Theorem 25 (Soundness) $Ax \vdash_{\beta} t = t' \Rightarrow Ax \models_{CLM} t = t'$.

Definition 21 Let $[t] =_{df} \{t' \mid Ax \vdash_{\beta} t = t'\}$. We define the Λ_{AL} -TERM MODEL for Ax $\mathcal{T}_{\Lambda}(Ax)$ by $D_{\perp} =_{df} \{[t] \mid t \in \Lambda_{AL}\}$, $\perp =_{df} [\#]$, $\mathfrak{S}(c) =_{df} [c]$ for all constants, and

- $\Psi([t])(\langle x_i \mapsto t_i \rangle_{i \in I}) =_{df} [t(x_i, t_i)_{i \in I}]$;
- $\mathbf{s} =_{df} [S_{\Lambda}]$;
- $\mathbf{k}_x =_{df} [(K_x)_{\Lambda}]$ for all $x \in Var$;
- $\mathbf{e} =_{df} [\lambda Z \lambda \langle X_i \mapsto x_i \rangle_{x_i \in Var} . Z(x_i, X_i)_{x_i \in Var}]$;
- $\Phi(f) =_{df} \mathbf{e} \bullet [t]$, if $[t]$ represents f .

Lemma 26 $\mathcal{T}_{\Lambda}(Ax)$ is a C_{AL} -model satisfying Ax .

PROOF: By A2 and R3 the operation Ψ , and thus \bullet , is well-defined for $\mathcal{T}_{\Lambda}(Ax)$. Φ is also well-defined: let $[t]$ and $[t']$ both represent f , then $Ax \vdash t(x_i, X_i)_{x_i \in Var} = t'(x_i, X_i)_{x_i \in Var}$, hence by R4 $Ax \vdash \lambda \langle X_i \mapsto x_i \rangle_{x_i \in Var} . t(x_i, X_i)_{x_i \in Var} = \lambda \langle X_i \mapsto x_i \rangle_{x_i \in Var} . t'(x_i, X_i)_{x_i \in Var}$, and therefore $\mathbf{e} \bullet [t] = \mathbf{e} \bullet [t']$.

We need to check the axioms for C_{AL} -algebras.

1. $\mathbf{k}_{x_k} \bullet \langle x_i \mapsto [t_i] \rangle_{i \in I} \bullet \langle y_j \mapsto [t'_j] \rangle_{j \in J} = [(K_{x_k})_{\Lambda}(\langle x_i, t_i \rangle_{i \in I} \langle y_j, t'_j \rangle_{j \in J})] = [t_k]$
if $k \in I$, else $[\#]$, whence $= \langle x_i \mapsto [t_i] \rangle_{i \in I} (x_k)$.
2. $\mathbf{s} \bullet \langle x_i \mapsto [t_i] \rangle_{i \in I} \bullet \langle y_j \mapsto [t'_j] \rangle_{j \in J} \bullet \langle v_k \mapsto [t''_k] \rangle_{k \in K}$
 $= [S_{\Lambda}(\langle x_i, t_i \rangle_{i \in I} \langle y_j, t'_j \rangle_{j \in J} \langle v_k, t''_k \rangle_{k \in K})]$
 $= [t_i(v_k, t''_k)_{k \in K} \langle y_j, t'_j \rangle_{j \in J} \langle v_k, t''_k \rangle_{k \in K}]$, if $z = x_i$ for some $i \in I$,
otherwise $\#$ and thus
 $= \langle x_i \mapsto [t_i] \rangle_{i \in I} (z) \bullet \langle v_k \mapsto [t''_k] \rangle_{k \in K} \bullet \langle y_j \mapsto [t'_j] \rangle \bullet \langle v_k \mapsto [t''_k] \rangle_{k \in K} \langle y_j \in J \rangle$.
3. $[\#] \bullet \langle x_i \mapsto [t_i] \rangle_{i \in I} = [\#(x_i, t_i)_{i \in I}] = [\#]$.

The equations in Ax obviously hold. For the additional conditions on C_{AL} -models we have by definition that $\Phi \circ \Psi$ is represented by \mathbf{e} . For strictness of Φ we notice that $[\#]$ represents $\perp_{(Var \rightarrow D) \rightarrow D}$ by item 3 above. Hence $\Phi(\perp_{(Var \rightarrow D) \rightarrow D}) = \mathbf{e} \bullet \perp_{(Var \rightarrow D) \rightarrow D} = [\lambda \langle X_i \mapsto x_i \rangle_{x_i \in Var} . \#(x_i, X_i)_{x_i \in Var}] = [\#]$ by A6 and A7.

What remains to be checked is the retraction condition that $\Psi \circ \Phi(f) = f$ for all $f : ((Var \rightarrow D_{\perp}) \rightarrow D_{\perp})_{rep}$. So if $\mu = \Psi[t]$ then for all $\langle x_i \mapsto [t_i] \rangle_{i \in I}$:

$$\Psi \circ \Phi(\mu)(\langle x_i \mapsto [t_i] \rangle_{i \in I}) = \Psi(\Phi \circ \Psi[t])(\langle x_i \mapsto [t_i] \rangle_{i \in I}) = \mathbf{e} \bullet [t] \bullet \langle x_i \mapsto [t_i] \rangle_{i \in I} = [t(x_i, t_i)_{i \in I}] = \Psi[t](\langle x_i \mapsto [t_i] \rangle_{i \in I}) = \mu(\langle x_i \mapsto [t_i] \rangle_{i \in I}). \quad \square$$

Lemma 27 $\|t\|^g = [[t_j/X_j]_{j \in J} t]$ in $\mathcal{T}_{\Lambda}(Ax)$, where $g(X_j) = [t_j]$ for $j \in J$.

PROOF: $\|c\|^g = [c]$ and $\|X_i\|^g = [t_i]$ by definition. By induction

- $\|t(x_i, t_i)_{i \in I}\|^g = \|t\|^g \bullet \langle x_i \mapsto \|t_i\|^g \rangle_{i \in I} = [[t_j/X_j]_{j \in J} t] \bullet \langle x_i \mapsto [[t_j/X_j]_{j \in J} t_i] \rangle_{i \in I} = [[t_j/X_j]_{j \in J} t(x_i, t_i)_{i \in I}]$;
- For any f choose s_j^f such that $[s_j^f] = f \circ \chi(X_j)$ for those $j \in J' \subseteq J$ for which $\chi(X_j)$ is defined. The choice is arbitrary by the rules R3 and R4. Also by axiom A4 we may assume that $dm(\chi) \cap \bigcup_{j \in J \setminus J'} FP(t_j) = \emptyset$ and no X_i gets accidentally bound in the application of \mathbf{e} . Then
 $\|\lambda \chi . t\|^g = \Phi \lambda f \|t\|^{g_{f \circ \chi}^{dm(\chi)}} = \Phi \lambda f [[s_j^f/X_j]_{j \in J'} [t_j/X_j]_{j \in J \setminus J'} t]$
 $= \Phi (\Psi[\lambda \chi . [t_j/X_j]_{j \in J \setminus J'} t]) = \mathbf{e} \bullet [\lambda \chi . [t_j/X_j]_{j \in J \setminus J'} t]$
 $= [\lambda \langle X_i \mapsto x_i \rangle_{x_i \in Var} . (\lambda \chi . [t_j/X_j]_{j \in J \setminus J'} t)(x_i, X_i)_{x_i \in Var}]$
 $= [\lambda \langle X_i \mapsto x_i \rangle_{x_i \in Var} . [X_i/\chi^{\perp 1}(x_i)]_{x_i \in \text{rng}(\chi)} [t_j/X_j]_{j \in J \setminus J'} t]$
 $= [\lambda \chi . [t_j/X_j]_{j \in J \setminus J'} t] = [[t_j/X_j]_{j \in J} \lambda \chi . t].$

□

Theorem 28 (Completeness) $Ax \models_{CLM} t = t' \Rightarrow Ax \vdash_{\beta} t = t'$.

PROOF: If $Ax \not\vdash_{\beta} t = t'$ then $[t] \neq [t']$ in $\mathcal{T}_{\Lambda}(Ax)$ and thus $\|t\|^g \neq \|t'\|^g$, setting $g(X) = [X]$ for all $X \in Par$. Therefore $Ax \not\models_{CLM} t = t'$. \square

Corollary 29 *The finitary β -calculus is complete for finitary Λ_{AL} .*

PROOF: If $\models_{CLM} t = t'$ then $\vdash_{\beta} t = t'$, and hence by Church-Rosser $t \triangleright s \wedge t' \triangleright s$ for some s . But reduction of finite terms cannot produce infinite terms. Thus, there is a proof of $t = t'$, by final application of R1 and R2, which involves no infinite terms. \square

2.4 Structured Objects and Systems of Equations

Structured objects are distinguished from ordinary objects in that they can be uniquely decomposed into the component parts, from which they were put together. Ordered pairs are typical examples, forming an object $\langle a, b \rangle$ from two objects a, b , such that there are projection functions π_1, π_2 with $\pi_1(\langle a, b \rangle) = a$ and $\pi_2(\langle a, b \rangle) = b$. The definition of ordered pairs in the λ -calculus will be generalised for our purposes here. I use the following operation, which allows us to build STRUCTURED OBJECTS with any number of components.

$$\langle\langle (x_i, t_i^1)_{i \in I^1}; \dots; (x_i, t_i^n)_{i \in I^n} \rangle\rangle =_{df} \lambda y. (y(x_i, t_i^1)_{i \in I^1} \dots (x_i, t_i^n)_{i \in I^n}) \quad (y \text{ fresh}).$$

Theorem 30 $\langle\langle (x_i, t_i^1)_{i \in I^1}; \dots; (x_i, t_i^n)_{i \in I^n} \rangle\rangle = \langle\langle (x_i, s_i^1)_{i \in I^1}; \dots; (x_i, s_i^n)_{i \in I^n} \rangle\rangle$
 $\rightarrow \bigwedge_{k=1..n} \bigwedge_{i \in I^k} t_i^k = s_i^k$.

PROOF: The x_i -component of the k th member is recovered by the projection

$$\pi_{x_i}^{n,k} = \lambda y. y(\lambda\{1\} \dots \lambda\{k-1\} \lambda\{x_i\} \lambda\{k+1\} \dots \lambda\{n\}. x_i).$$

\square

For ordered n -tuples we write $\langle\langle \dots; t; \dots \rangle\rangle$ for $\langle\langle \dots; (z, t); \dots \rangle\rangle$, where z is as usual the designated role to encode unary abstraction/application. By $n = 1$ one can encode assignments, which we write as $\langle\langle x_i, t_i \rangle\rangle_{i \in I}$. The values of an encoded assignment are retrieved by projection functions $\pi_{x_i} =_{df} \lambda y. y(\{x_i\} x_i)$. We have

$$\pi_{x_i}(\langle\langle x_j, t_j \rangle\rangle_{j \in I}) = t_i \quad \text{for } i \in I.$$

Notice that for any $x \neq y$, if $\pi_x = \pi_y$ then for all $t, t' : t = \pi_x(\langle\langle x, t, y, t' \rangle\rangle) = \pi_y(\langle\langle x, t, y, t' \rangle\rangle) = t'$. Hence, in a non trivial model all projections π_x must be different. We conclude.¹⁴

Theorem 31 *Any non trivial SAC-model is infinite.*

A particular demand of ST, at the focus of [Aczel & Lunnon 1991], is to have non well-founded objects, obtained from cyclic sets of equations.¹⁵ To solve systems of equations $(x_i = t_i)_{i \in I}$ we use the above encoding of assignments. We say that an (encoded) assignment \bar{a} SOLVES A SYSTEM OF EQUATIONS $(x_i = t_i)_{i \in I}$ iff for all $i \in I$ $\pi_{x_i}(\bar{a}) = [\pi_{x_j}(\bar{a})/x_j]_{j \in I} t_i$.

Theorem 32 *Every system of equations has a solution.*

PROOF: Recall that $\Upsilon =_{df} \lambda f. (\lambda x. f(x x))(\lambda x. f(x x))$ is a fixed point combinator with $t(\Upsilon t) = (\Upsilon t)$ for any term t . Define from a system of equations $(x_i = t_i)_{i \in I}$ the functor $F =_{df} \lambda z. \langle\langle x_i, [\pi_{x_j}(z)/x_j]_{j \in I} t_i \rangle\rangle_{i \in I}$. Then (ΥF) solves the system, as $\pi_{x_i}(\Upsilon F) = \pi_{x_i}(F(\Upsilon F)) = \pi_{x_i}(\langle\langle x_i, [\pi_{x_j}(\Upsilon F)/x_j]_{j \in I} t_i \rangle\rangle_{i \in I}) = [\pi_{x_j}(\Upsilon F)/x_j]_{j \in I} t_i$. \square

As things stand we do not necessarily have unique solutions to such equations. It may be consistent to assume uniqueness at least for some kinds of systems, to get closer to the original aims of Aczel and Lunnon.

¹⁴Notice that this holds for finitary versions of the SAC as well.

¹⁵An example of a cyclic equation would be $x = \langle\langle x, c \rangle\rangle$, any solution of which must provide a non well-founded pair p whose first component is p itself.

In the following, we need to assume that at least three variables do not occur in such a system, to allow us to form the terms below without accidentally binding some of the x_i in them.

Chapter 3

Semantic Theories

I now show how the SAC can provide a framework for specifying semantic theories, especially recent ones, which do not easily fit into the traditional λ -calculus. Perhaps the key idea for semantics, as conceived of in this thesis, is to think of properties as functions from assignments to propositions. I will use the term *ω -properties* to distinguish them from ordinary functions which take entities into propositions.

In the following I will freely mix the languages Λ_{AL} , Λ , and λ , according to whatever notation seems most convenient. The reader is expected to switch between the systems according to the translations given in section 2.2.1, though it will not be necessary to mentally translate every term into Λ_{AL} , I hope. To the contrary, the use of Λ and λ should help to avoid tedious complications, which are the price we pay for Λ_{AL} 's simple proof theory.

The following section on Frege Structures will be presupposed in all three semantic theories to be discussed, but otherwise those sections can be read relatively independently.

3.1 Frege Structures

For semantics, as I understand it, we need a theory of truth for the propositions that we hope to denote by our terms. The usual way to approach such a theory is to give an inductive definition of truth-in-a-model for terms of the right type,

where the notion of a model is suitably enriched to fix the base cases. In IL and similar systems for example we find a basic type of truth values (usually two), and the theory of truth is given as conditions that determine which truth value is denoted by a complex formula, depending on its parts. The matter is complicated by the presence of a number of parameters with respect to which a formula is evaluated, most prominently an index for possible worlds. The whole setup leans towards building theoretic assumptions into the system via modeling choices and implicit consequences of certain definitions, rather than taking things at face value and saying explicitly what we assume to hold about them.

My aim here is to provide a flexible framework in which different semantic assumptions can be expressed and tried out. It is thus a good idea to look elsewhere, namely to Property Theories, for ways to avoid prejudging too many issues before we can even begin to put any of the ideas we have produced into practise. The kinds of things I want to assume for the envisaged framework are these: some terms denote propositions, and propositions are the things that are true or false. Ω -properties are things that form propositions when applied to assignments of objects to certain variables, which we think of as argument roles for those ω -properties. The notion of truth does not primarily apply to formal expressions, in relation to various parameters of evaluation, but to propositions, which are things that need no further fixing of any such parameters. This does not prevent us from introducing such parameters into our framework, but rather allows us to do so without changing the theory of truth: if an expression has to be evaluated with respect to certain entities, worlds, situations, time intervals etc., it simply does not express a proposition, but an incomplete object, which needs some open argument places filled.

I do not claim that this approach is the only possible way of making use of the theory of simultaneous abstraction developed in the preceding chapter, but merely that this way of proceeding delivers reasonable versions of (the underlying logics of) DMG, DRT and ST in a simple, economical, and philosophically sound way. The addition of new structure in this framework, such as for example a theory of situations and states of affairs into a Montagovian or Kamp style semantics, does not require us to revise the basic axioms of truth for DMG or DRT, which in turn are perfectly compatible extensions of the theory of Proposition

Structures. What has to be revised in switching from one system to another are axioms about the lexical meaning of words, and possibly other aspects of the syntax–semantics interface, but not the fundamental apparatus of abstraction, application, propositions, and truth.

Let me then introduce a formal first order language in which to state such a theory of truth. One might do the job in an informal model theoretic way along the lines of [Aczel 1980] but I will make things look more like the formal treatment of [Turner 1990], without attaching any philosophical importance to this.¹ In line with our convention to use Λ_{AL} as our official term language I regard its parameters as variables v, v_1, v_2, \dots of FOL_{SAC} , defined below.²

Definition 22 We define *WFFs of FOL_{SAC}* over a set of n -ary predicates R_i^n :

- if $t_1, \dots, t_n, t, t' \in \Lambda_{AL}$ then $R_i^n(t_1, \dots, t_n), t = t' \in WFF$,
- if $\psi, \phi \in WFF$ then $\neg\phi, \phi \wedge \psi, \exists v\phi \in WFF$.

Other connectives and quantifiers are defined as usual. A *MODEL M* for FOL_{SAC} is a C_{AL} -model with an interpretation \mathfrak{S} for the predicates, such that the usual conditions obtain:

- $M, g \models R_i^n(t_1, \dots, t_n)$ iff $\langle \|t_1\|^g, \dots, \|t_n\|^g \rangle \in \mathfrak{S}(R_i^n)$;
- $M, g \models t = t'$ iff $\|t\|^g = \|t'\|^g$;
- $M, g \models \phi \wedge \psi$ iff $M, g \models \phi$ and $M, g \models \psi$;
- $M, g \models \neg\phi$ iff $M, g \not\models \phi$;
- $M, g \models \exists v\phi$ iff $\exists d \in D_{\perp} : M, g_d^v \models \phi$.

¹It has technical advantages though, as this allows us to use FOL's well understood proof theory to show that a (finite) term t implies another (finite) term t' , relative to a particular theory of truth. For this, one would have to derive $T(t')$ in FOL from $T(t)$ plus the axioms of truth and (β -) identity. It seems possible to provide a completeness theorem for this.

²If we were to stick to Λ -terms here it would be useful to add to the Λ -variables a set of variables that do not enter into the formation of complex terms, in order to always have fresh ones when needed.

We single out a unary relation T as a truth predicate. We define $F(t) =_{df} T(\perp t)$ (falsity) and $P(t) =_{df} T(t) \vee F(t)$ (being a proposition). For dealing with quantification we define $PTY^n(t) =_{df} \forall v_1 \dots v_n P(t(v_1) \dots (v_n))$ (being an n -place property).

Logical operations in our calculus are treated as Λ_{AL} -constants whose semantic behavior is captured by a theory of truth.³ The following logical constants are singled out: $\cap, \cup, \perp, \supset, \Sigma, \Pi, \approx, \Delta$, intended to be the operations of conjunction, disjunction, negation, implication, existential and universal quantification, equality, and structured predication⁴. I will take $\cap, \perp, \Sigma, \approx, \Delta$ as primitive, and the others to be defined in terms of these.⁵ I assume the logical combinators, other than Δ , to have one or two distinguished argument roles z_1, z_2 . One can of course curry them for standard Montagovian techniques to run smoothly.⁶ In any case I write them in the conventional form, $t \cap t'$ for $\cap(z_1 t, z_2 t')$, $\Sigma x t$ for $\Sigma(z_1, \lambda x.t)$ etc. and $(x_i, t_i)_{i \in I} \Delta t$ for $\Delta(z_1.t)(x_i, t_i)_{i \in I}$, to enhance readability.

Definition 23 1. A *PROPOSITION STRUCTURE* is a model of FOL_{SAC} satisfying

$$\begin{aligned} P(t \approx t') &\wedge (T(t \approx t') \leftrightarrow t = t') \\ P(t) &\rightarrow (P(\perp t) \wedge (T(\perp t) \leftrightarrow \neg T(t))) \\ P(t) \wedge P(t') &\rightarrow (P(t \cap t') \wedge (T(t \cap t') \leftrightarrow T(t) \wedge T(t'))). \end{aligned}$$

2. A *Proposition Structure WITH PREDICATION* has the additional truth–axiom

$$P(t(x_i, t_i)_{i \in I}) \rightarrow (P((x_i, t_i)_{i \in I} \Delta t) \wedge (T((x_i, t_i)_{i \in I} \Delta t) \leftrightarrow T(t(x_i, t_i)_{i \in I})))$$

³These constants are not to be confused with the logical symbols of FOL_{SAC} , which we use to express our theories of truth.

⁴The SAC, in contrast to Aczel and Lunnion's theory, does not take the idea of structured objects as a starting point. I take the notions of abstraction and application as fundamental here, rather than structure preserving predication. It is crucial that these two notions of application/predication are kept distinct. Occasionally, forms of Situation Theory have been guilty of conflating them, when they assumed “infons” to be composed out of a unique relation and assignment, plus relations to be infon-abstracts which *apply* to assignments to form structured infons. This implies that $r(x_i, t_i)_{i \in I} = (x_i, t_i)_{i \in I} \Delta r$ for all relations r , which quickly leads to the triviality of the model, by the argument given in [Aczel 1989].

⁵There may be reasons not to do this, such as to avoid unintended identities in attitude contexts, or to be able to use the less symmetric notion of “strong” implication in [Aczel 1980].

⁶Avoiding all forms of currying is useful if the predication axiom is to be fully exploited.

plus the axiom of structure preservation

$$((x_i, t_i)_{i \in I} \Delta t) = ((x_i, t'_i)_{i \in I} \Delta t') \rightarrow t = t' \wedge \bigwedge_{i \in I} t_i = t'_i.$$

3. A FREGE STRUCTURE⁷ is a Proposition Structure satisfying

$$PTY^1(t) \rightarrow (P(\Sigma t) \wedge (T(\Sigma t) \leftrightarrow \exists v T(t(v)))).$$

4. I speak of a STRONG Proposition / Frege Structure (with predication) if we have biconditionals for propositionhood, that is $P(t) \leftrightarrow P(\perp t)$, $P(t) \wedge P(t') \leftrightarrow P(t \cap t')$, $P(t(x_i, t_i)_{i \in I}) \leftrightarrow P((x_i, t_i)_{i \in I} \Delta t)$, and $PTY^1(t) \leftrightarrow P(\Sigma t)$.

The class of propositions guaranteed to exist by these axioms is rather restricted, in that the relations which have an internal representation by some property only involve equality and logical constants. In semantic applications one needs a much larger stock of basic properties. So it is useful to consider the addition of further predicates to FOL_{SAC} , whose extensions may be fixed in arbitrary ways. The question arises whether these extensions also correspond to properties that are internally represented in the Frege Structure, and thus are denotable by a (possibly newly introduced) term.

Definition 24 A term t INTERNALLY DEFINES a predicate ϕ_{v_1, \dots, v_n} with free variables $v_1, \dots, v_n \notin FV(t)$, in some FOL_{SAC} -model, if:

$$PTY^n(t) \wedge \forall v_1..v_n T(tv_1..v_n) \leftrightarrow \phi_{v_1..v_n}.$$

We may apply this terminology to denotations as well, and say in such cases that a particular relation is internally defined by an n-place property. As Aczel pointed out in [Aczel 1980] there are limits to what can be internally defined in

⁷For simplicity, I leave out the axiom for strong implication: $(T(t) \rightarrow P(t')) \rightarrow P(t \supset t') \wedge T(t \supset t') \leftrightarrow (T(t) \rightarrow T(t'))$ which may have a useful role to play: it allows us to form universally quantified propositions $\Pi v(F(v) \supset G(v))$ even when F and G are not properties, as long as for all d for which $F(d)$ is true, $G(d)$ is a proposition.

a Proposition Structure. For example the truth predicate T cannot be internally defined, for it would lead to the construction of a liar proposition.⁸

3.1.1 Consistency

I more or less repeat the proof of the existence of Frege Structures from [Aczel 1980] here, in a somewhat sketchy manner.

Take any non-trivial model for the SAC. To get a model for our axioms we need a set \mathcal{T} that interprets T , such that the axioms are true. The other predicates, like P, F, \dots are defined in terms of T , but it is more convenient to work with pairs $(\mathcal{P}, \mathcal{T})$ of subsets $\mathcal{T} \subseteq \mathcal{P}$, called PROPOSITIONS and TRUTHS, to interpret P and T . Such pairs of subsets form a cpo as follows, where $\mathcal{F} =_{df} \mathcal{P} \setminus \mathcal{T}$ are the FALSITIES:

$$(\mathcal{P}, \mathcal{T}) \sqsubseteq (\mathcal{P}', \mathcal{T}') \Leftrightarrow \mathcal{T} \subseteq \mathcal{T}' \text{ and } \mathcal{F} \subseteq \mathcal{F}'.$$

Lub's of ascending chains are given by set union. Any monotone operator over a cpo has a least fixed point. We define a monotone operator θ_F whose fixed points give us FOL_{SAC} -models for the Frege Structure axioms when $\mathfrak{S}(T) = \mathcal{T}$.

As any non-trivial model is infinite we can select for each logical constant l semantically distinct terms r_l such that we can use structured objects to avoid possible overlaps between the results of applying different logical constants.⁹ For binary l we assign the following interpretation:

$$d_l =_{df} \|\lambda\{z_1, z_2\} \cdot \langle\langle r_l; z_1; z_2 \rangle\rangle\|^g.$$

Similarly for unary ones. Predication is encoded as

$$d_\Delta =_{df} \lambda Z \lambda \langle x_i \mapsto X_i \rangle_{x_i \in Var} \cdot \langle\langle r_\Delta; Z; (x_i, X_i)_{x_i \in Var} \rangle\rangle$$

⁸Let l be a term that solves the equation $x = \perp true(x)$, and assume that $true$ internally defines T . Then $P(l)$, and hence $T(l) \leftrightarrow T(\perp true(l)) \leftrightarrow \neg T(true(l)) \leftrightarrow \neg T(l)$, which is a contradiction.

⁹The proof thus would not work for theories that explicitly deny their propositions to be this highly structured.

Let $d_{l,f,h}$ stand for $d_l \bullet f \bullet h$ if $l = \Delta$, otherwise $d_l \bullet f$. The truth-axiom Ax_l for a logical operator d_l can be expressed semantically with reference to a PROPOSITION CONDITION $PC_l(\mathcal{P}, f, h)$ and a TRUTH CONDITION $TC_l(\mathcal{T}, f, h)$ as

$$(Ax_l) \quad PC_l(\mathcal{P}, f, h) \Rightarrow (d_{l,f,h} \in \mathcal{P} \text{ and } (d_{l,f,h} \in \mathcal{T} \Leftrightarrow TC_l(\mathcal{T}, f, h)))$$

For example, the proposition condition for \cap is ' $f(z_1) \in \mathcal{P}$ and $f(z_2) \in \mathcal{P}$ ', and the truth condition for \perp is ' $f(z_1) \notin \mathcal{T}$ '.

Definition 25 $\theta_F(\mathcal{P}, \mathcal{T}) =_{df} (\mathcal{P}', \mathcal{T}')$, where for the logical constants l of Frege Structures:

- \mathcal{P}' is the smallest set such that for each l, f, h :
 $PC_l(\mathcal{P}, f, h) \Rightarrow d_{l,f,h} \in \mathcal{P}'$;
- \mathcal{T}' is the smallest set such that for each l, f, h :
 $PC_l(\mathcal{P}, f, h) \text{ and } TC_l(\mathcal{T}, f, h) \Rightarrow d_{l,f,h} \in \mathcal{T}'$.

Lemma 33 Any fixed point for θ_F satisfies Ax_F .

PROOF: Let $\theta_F(\mathcal{P}, \mathcal{T}) = (\mathcal{P}, \mathcal{T})$.

- (i) If $PC_l(\mathcal{P}, f, h)$ then $d_{l,f,h} \in \mathcal{P}$.
- (ii) If $PC_l(\mathcal{P}, f, h)$ and $TC_l(\mathcal{T}, f, h)$ then $d_{l,f,h} \in \mathcal{T}$.
- (iii) Given $PC_l(\mathcal{P}, f, h)$ and $d_{l,f,h} \in \mathcal{T}$ there are l', f', h' with $d_{l,f,h} = d_{l',f',h'}$ and $TC_{l'}(\mathcal{T}, f', h')$. But then $r_l = r_{l'}$ and $f = f'$, and in case $l = \Delta$ also $h = h'$, and hence $TC_l(\mathcal{T}, f, h)$. \square

By the fact that Ax_{\perp} is satisfied we see that \mathcal{P} is the extension of the predicate P as it was defined, and thus the axioms Ax_F hold if and only if the formal axioms for Frege Structures are true. Notice that the reverse direction of (i) is true as well, giving us in fact a strong Frege Structure. The fact that predication is structure preserving is obvious. So we conclude:

Theorem 34 Every non trivial SAC-model can be extended to a strong Frege Structure with Predication.

As Aczel has shown in [Aczel 1980], it is consistent to add infinitely many axioms for the internal definition of relations to the theory of Frege Structures, as long as the extensions of the possibly newly introduced predicates $\phi_{v_1 \dots v_n}$ in the SAC-model are given in advance, without reference to T . In the construction of the Frege Structures they are treated just as we treated identity, which is already internally defined.

Theorem 35 Every non trivial SAC-model can be extended to a strong Frege Structure with Predication in which a countable number of previously given relations are internally defined by n -place properties.

3.2 Dynamic Montague Grammar

There are several versions of DMG on the market, which use different "dynamic" relatives of Montague's original logic IL. One such logic, namely the Dynamic Property Theory (DPT) of [Chierchia 1994], is particularly close to the system Λ of the preceding chapter. The additions we have to make to obtain DPT are very small indeed, as Λ was first developed as a generalisation of DPT's limited means of simultaneous abstraction and application.

In DPT, the variables are divided into two kinds: the DISCOURSE MARKERS DM , and the remaining ones, which I call META VARIABLES MV . I use Greek letters $\alpha, \beta \dots$ for meta variables, dotted letters $\dot{x}, \dot{y} \dots$ for discourse markers, and $x, y \dots$ for variables that can be of either kind. The sorting of the variables is not a matter of the kinds of denotations they take. It is rather a matter of getting the two crucial operators of the system, \sqsupset -abstraction and \sqcup -application, to operate on a certain set of variables, namely the discourse markers, only. We need to protect the other variables, i.e. the meta variables, from the influence of \sqsupset -abstraction so that we can bind variables inside the scope of \sqsupset , as will become evident from the examples later on.

The "dynamic" part of DMG can be captured easily by abstracting and applying to the infinite set of discourse markers via the following definitions:

- $\ulcorner t =_{df} \lambda DM.t$, and
- $\urcorner t =_{df} t(\dot{x}.x)_{\dot{x} \in DM}$.

All other abstractions and applications will be the traditional unary ones.¹⁰ We use them to get a fairly standard system that does not require us to rethink the setup of DMG. Let me call DPT-TERMS any expressions built from unary abstraction and application and the \ulcorner and \urcorner operators.

Convenient as DPT-terms may be for the purposes of DMG, one would expect them to make trouble in the proof theory, given the inconveniences of Λ in that respect. On the positive side, notice that every discourse marker is bound in a term of the form $\ulcorner t$, and occurs free in $\urcorner t$. We thus can β -convert a term $\lambda \alpha.t(\ulcorner t')$ even if some discourse markers of t' end up in the scope of an abstraction in t . Discourse markers are not free in $\ulcorner t'$ and hence do not become bound by a new operator inside t . Another reduction we use in DMG beside the more standard β -conversions is $\ulcorner \ulcorner t = t$.

On the negative side, there are DPT-terms which cannot be reduced to simpler DPT-expressions, although their translations into Λ_{AL} may be contractable into new Λ_{AL} -terms which are not translations of any DPT expressions. An example would be a term $(\lambda \alpha.\ulcorner \alpha)(\dot{x})$ which translates to $(\lambda(\alpha^\dagger \mapsto z)\lambda(\dot{x}^\dagger \mapsto \dot{x})_{\dot{x} \in DM}.\alpha^\dagger)(z.\dot{x}^\dagger)$. As a DPT-term $(\lambda \alpha.\ulcorner \alpha)(\dot{x})$ is not reducible. In particular it is not semantically equivalent to $\ulcorner \dot{x}$, which means the same thing as the Λ -term $\{\dot{x}\}.\dot{x}$. The translated term on the other hand boils down to $[\dot{x}^\dagger/\alpha^\dagger]\lambda(\dot{x}^\dagger \mapsto \dot{x})_{\dot{x} \in DM}.\alpha^\dagger \triangleright \lambda(\dot{x}^\dagger).\dot{x}^\dagger$, which is the translation of the Λ -term $\{\dot{x}\}.\dot{x}$, but not of any DPT expression. Luckily the terms actually used in DMG do not belong to the problematic sort. I will think of DPT-expressions as convenient shorthands for the underlying Λ_{AL} -terms, for which I have already given a complete proof theory in 2.2.2, and a calculus for DPT as being obtained via the standard translation given in section 2.2.1. For computational applications this stance is of course impractical, as the terms involving \ulcorner and \urcorner will be infinite, and thus in real life something closer to actual DPT-expressions will have to be used. But for an

¹⁰As an alternative to taking these unary operations as defined in terms of Λ_{AL} - or Λ' -expressions, one could introduce them as basic operations, and give them a semantics in an expanded domain $D \rightleftharpoons_{\mathbb{V}}^{\mathbb{V}} ((Var \rightarrow D) \rightarrow D) + (D \rightarrow D)$, in the way [Chierchia 1994] does.

understanding of how these expressions have to be handled the translation into Λ_{AL} still is instructive.

The sentence 'A man walks in' can now be rendered in our notation as

$$\lambda \alpha.\Sigma \dot{x}(man(\dot{x}) \cap walkin(\dot{x}) \cap \ulcorner \alpha)$$

which can applied to $\ulcorner whistle(\dot{x})$ ('he whistles') to yield

$$\Sigma \dot{x}(man(\dot{x}) \cap walkin(\dot{x}) \cap whistle(\dot{x}))$$

The important action happens at the level of conjunction of ω -properties, which is achieved by abstracting over "possible continuations" of the discourse.¹¹

3.2.1 Some DMG Translations

I only give a short sketch here and refer the reader to [Chierchia 1994] and [Chierchia 1992] for a more complete treatment. Sentence denotations in DMG are not propositions, but *context change potentials (ccp's)*, where a ccp is a function from ω -properties to propositions. As a notational convention I write $\ulcorner t$ as $[t]$. Two operations are handy to convert propositions into ccp's and back: $\uparrow t =_{df} \lambda \alpha(t \cap \ulcorner \alpha)$ and $\downarrow t =_{df} t[true]$. I assume the scope of operators to be narrowest for \ulcorner , and increasingly wider for application (bracketing to the left), \cap , \perp , \cup , \uparrow , \downarrow and unary λ -abstraction.

Ccp's allow for clever ways of leaving holes within quantified structures to be filled by later material in the discourse so that one can use functional composition $t; t' =_{df} \lambda \alpha.t[t'(\alpha)]$ for discourse sequencing:

$$\begin{aligned} & \ulcorner [a \text{ man}] \text{ walks in. } \ulcorner [he] \text{ whistles.} \urcorner \implies \\ & (\lambda \gamma \Sigma \dot{x}(man(\dot{x}) \cap walkin(\dot{x}) \cap \ulcorner \gamma)) ; (\lambda \gamma. whistle(\dot{x}) \cap \ulcorner \gamma) \\ & = \lambda \gamma(\Sigma \dot{x} man(\dot{x}) \cap walkin(\dot{x}) \cap whistle(\dot{x}) \cap \ulcorner \gamma) \end{aligned}$$

¹¹Lambda conversion on $\ulcorner whistle(\dot{x})$ is only one way to make the conjunction happen. Unification of $whistle(\dot{x})$ with the unabstracted variable α , as in UCG, see [Zeevat 1991], is just as intuitive for this. Hence the choice of the term 'meta variables', reminiscent of its use in UCG.

The flexibility of ccp's is shown by various possibilities to define dynamic kinds of universal quantification for donkey sentences. Here is an example derivation of the “strong” reading for them. To make things concrete I use a basic, purely applicative Categorical Grammar, operating on sequents $? \Longrightarrow A : t_i$ of sequences $?$ of (indexed) strings, and category–meaning pairs $A : t$. We have lexical axioms, plus the slash elimination rules ($/E$) and ($\backslash E$),¹² interpreted by “intensional” function application.

$$\begin{aligned} (/E) \text{ if } ?_1 \Longrightarrow A/B : t_1 \text{ and } ?_2 \Longrightarrow B : t_2 \text{ then } ?_1?_2 \Longrightarrow A : t_1[t_2] \\ (\backslash E) \text{ if } ?_1 \Longrightarrow B : t_1 \text{ and } ?_2 \Longrightarrow B \backslash A : t_2 \text{ then } ?_1?_2 \Longrightarrow A : t_2[t_1] \end{aligned}$$

$$\begin{aligned} {}^{\text{st}}[\text{every}]^{\text{t}} &\Longrightarrow (s/(n \backslash s))/cn : \lambda \alpha \lambda \beta. \uparrow \Pi \dot{x} \perp (({}^{\text{u}}\alpha \dot{x}) [\perp \downarrow ({}^{\text{u}}\beta \dot{x})]) \\ \text{'man'} &\Longrightarrow cn : \lambda x. \uparrow man(x) \\ \text{'with'} &\Longrightarrow (cn \backslash cn)/(s/(n \backslash s)) : \lambda \alpha \lambda \beta \lambda \delta \lambda \gamma. ({}^{\text{u}}\beta \delta) [\uparrow \alpha [\lambda y \uparrow with(\delta, y)] (\gamma)] \\ \text{'a'} &\Longrightarrow (s/(n \backslash s))/cn : \lambda \alpha \lambda \beta \lambda \gamma. \Sigma \dot{y} (({}^{\text{u}}\alpha \dot{y}) [\uparrow \beta \dot{y} \gamma]) \\ \text{'donkey'} &\Longrightarrow cn : \lambda x. \uparrow donkey(x) \\ \text{'beats'} &\Longrightarrow (n \backslash s)/(s/(n \backslash s)) : \lambda \alpha \lambda \delta. {}^{\text{u}}\alpha [\lambda y \uparrow beat(\delta, y)] \\ \text{'it'} &\Longrightarrow (s/(n \backslash s)) : \lambda \alpha. {}^{\text{u}}\alpha \dot{y} \end{aligned}$$

$$\begin{aligned} \text{'a donkey'} &\Longrightarrow s/(n \backslash s) : \\ \lambda \beta \lambda \gamma. \Sigma \dot{y} (({}^{\text{u}}[\lambda x \uparrow donkey(x)] \dot{y}) [\uparrow \beta \dot{y} \gamma]) \\ \lambda \beta \lambda \gamma. \Sigma \dot{y} ((\uparrow donkey(\dot{y})) [\uparrow \beta \dot{y} \gamma]) \\ \lambda \beta \lambda \gamma. \Sigma \dot{y} (donkey(\dot{y}) \cap {}^{\text{u}}\beta \dot{y} \gamma) \end{aligned}$$

$$\begin{aligned} \text{'with a donkey'} &\Longrightarrow cn \backslash cn : \\ \lambda \beta \lambda \delta \lambda \gamma. ({}^{\text{u}}\beta \delta) [\uparrow [\lambda \beta \lambda \gamma \Sigma \dot{y} (donkey(\dot{y}) \cap {}^{\text{u}}\beta \dot{y} \gamma)] [\lambda y \uparrow with(\delta, y)] (\gamma)] \end{aligned}$$

¹²For a full Lambek calculus we need to be able to “hypothesize” constituents of any description. This can be achieved by adding axioms, slash introduction rules, and Cut:

$$\begin{aligned} (Ax) \quad A : {}^{\text{u}}\alpha \Longrightarrow A : {}^{\text{u}}\alpha \\ (/I) \quad \text{if } \Gamma, B : {}^{\text{u}}\alpha \Longrightarrow A : t \text{ then } \Gamma \Longrightarrow A/B : \lambda \alpha t \\ (\backslash I) \quad \text{if } B : {}^{\text{u}}\alpha, \Gamma \Longrightarrow A : t \text{ then } \Gamma \Longrightarrow B \backslash A : \lambda \alpha t \\ (Cut) \quad \text{if } \Gamma_1, A_2 : t_2, \Gamma_3 \Longrightarrow A_1 : t_1 \text{ and } \Gamma_2 \Longrightarrow A_2 : t_2 \text{ then } \Gamma_1 \Gamma_2 \Gamma_3 \Longrightarrow A_1 : t_1 \end{aligned}$$

where α is not free in Γ . The lexical translations would have to be spiced with a stronger dose of ${}^{\text{u}}$ -s, to make this work correctly.

$$\begin{aligned} \lambda \beta \lambda \delta \lambda \gamma. ({}^{\text{u}}\beta \delta) [\Sigma \dot{y} (donkey(\dot{y}) \cap ({}^{\text{u}}[\lambda y \uparrow with(\delta, y)] \dot{y} \gamma))] \\ \lambda \beta \lambda \delta \lambda \gamma. ({}^{\text{u}}\beta \delta) [\Sigma \dot{y} (donkey(\dot{y}) \cap (\uparrow with(\delta, \dot{y})) (\gamma))] \\ \lambda \beta \lambda \delta \lambda \gamma. ({}^{\text{u}}\beta \delta) [\Sigma \dot{y} (donkey(\dot{y}) \cap with(\delta, \dot{y}) \cap {}^{\text{u}}\gamma)] \end{aligned}$$

$$\begin{aligned} \text{'man with a donkey'} &\Longrightarrow cn : \\ \lambda \delta \lambda \gamma. ({}^{\text{u}}[\lambda x \uparrow man(x)] (\delta)) [\Sigma \dot{y} (donkey(\dot{y}) \cap with(\delta, \dot{y}) \cap {}^{\text{u}}\gamma)] \\ \lambda \delta \lambda \gamma. (\uparrow man(\delta)) [\Sigma \dot{y} (donkey(\dot{y}) \cap with(\delta, \dot{y}) \cap {}^{\text{u}}\gamma)] \\ \lambda \delta \lambda \gamma. (man(\delta) \cap \Sigma \dot{y} (donkey(\dot{y}) \cap with(\delta, \dot{y}) \cap {}^{\text{u}}\gamma)) \end{aligned}$$

$$\begin{aligned} {}^{\text{st}}[\text{every}]^{\text{t}} \text{ man with a donkey'} &\Longrightarrow s/(n \backslash s) : \\ \lambda \beta. \uparrow \Pi \dot{x} \perp ({}^{\text{u}}[\lambda \delta \lambda \gamma (man(\delta) \cap \Sigma \dot{y} (donkey(\dot{y}) \cap with(\delta, \dot{y}) \cap {}^{\text{u}}\gamma))] (\dot{x}) [\perp \downarrow ({}^{\text{u}}\beta \dot{x})]) \\ \lambda \beta. \uparrow \Pi \dot{x} \perp (man(\dot{x}) \cap \Sigma \dot{y} (donkey(\dot{y}) \cap with(\dot{x}, \dot{y}) \cap {}^{\text{u}}[\perp \downarrow ({}^{\text{u}}\beta \dot{x})])) \\ \lambda \beta. \uparrow \Pi \dot{x} \perp (man(\dot{x}) \cap \Sigma \dot{y} (donkey(\dot{y}) \cap with(\dot{x}, \dot{y}) \cap \perp \downarrow ({}^{\text{u}}\beta \dot{x}))) \end{aligned}$$

$$\begin{aligned} \text{'beats it'} &\Longrightarrow n \backslash s : \\ \lambda \delta. {}^{\text{u}}[\lambda \alpha ({}^{\text{u}}\alpha \dot{y})] [\lambda y \uparrow beat(\delta, y)] \\ \lambda \delta. ({}^{\text{u}}[\lambda y \uparrow beat(\delta, y)] (\dot{y})) \\ \lambda \delta. \uparrow beat(\delta, \dot{y}) \end{aligned}$$

$$\begin{aligned} {}^{\text{st}}[\text{every}]^{\text{t}} \text{ man with a donkey beats it'} &\Longrightarrow s : \\ \uparrow \Pi \dot{x} \perp (man(\dot{x}) \cap \Sigma \dot{y} (donkey(\dot{y}) \cap with(\dot{x}, \dot{y}) \cap \perp \downarrow ({}^{\text{u}}[\lambda \delta \uparrow beat(\delta, \dot{y})] (\dot{x})))) \\ \uparrow \Pi \dot{x} \perp (man(\dot{x}) \cap \Sigma \dot{y} (donkey(\dot{y}) \cap with(\dot{x}, \dot{y}) \cap \perp \downarrow \uparrow beat(\dot{x}, \dot{y}))) \\ \uparrow \Pi \dot{x} \perp (man(\dot{x}) \cap \Sigma \dot{y} (donkey(\dot{y}) \cap with(\dot{x}, \dot{y}) \cap \perp (beat(\dot{x}, \dot{y}) \cap true))) \end{aligned}$$

An alternative to the above “strong” reading of the universal, would assign the translation:

$${}^{\text{st}}[\text{every}]^{\text{t}} \Longrightarrow \lambda \alpha \lambda \beta. \uparrow \Pi \dot{x} \perp \downarrow ({}^{\text{u}}\alpha \dot{x}) \cup ({}^{\text{u}}\alpha \dot{x}) [\downarrow ({}^{\text{u}}\beta \dot{x})]$$

resulting in the “weak” reading:

$$\begin{aligned} \uparrow \Pi \dot{x} \perp (man(\dot{x}) \cap \Sigma \dot{y} (donkey(\dot{y}) \cap with(\dot{x}, \dot{y}) \cap true)) \\ \cup (man(\dot{x}) \cap \Sigma \dot{y} (donkey(\dot{y}) \cap with(\dot{x}, \dot{y}) \cap beat(\dot{x}, \dot{y}) \cap true)) \end{aligned}$$

3.3 Discourse Representation Theory

The SAC mirrors very closely the use of discourse referents in DRSs. The system I present in this section is inspired Robin Cooper's Situation Theoretic DRT, [Cooper 1993], where he uses EKN with Aczel–Lunnon abstraction to similar effect. As in Cooper's treatment, this version of DRT is not quite faithful to standard versions, as it imposes a need for some additional abstractions at the right level. This is because only abstracted variables are linked to their antecedents, while free variables cannot in this system be “captured” in the way one has come to expect. I won't go into the resulting complications for the syntax–semantics interface here, as I will explore alternatives which use free variables more prominently in the next chapter.

3.3.1 Zeevat's DRT

First, let me briefly sketch Henk Zeevat's compositional semantics for DRT, see [Zeevat 1989], which is an important reference point for the versions to be presented in this and the next chapter.

For the usual DRT syntax, we start with atomic conditions κ , which are just atomic first order formulae. DRSs and complex conditions are defined recursively: DRSs are pairs $\langle U, C \rangle$, where U is a set of variables, and C a set of conditions; the connective \gg forms implicational conditions $\langle U, C \rangle \gg \langle U', C' \rangle$ from two DRSs.¹³

Atomic conditions have their expected semantics. DRSs, and complex conditions are treated as standardly in DRT, where $g \sim_M f$ means g and f disagree at most on the variables in M .

- $\|C\| =_{df} \{g \mid \forall \kappa \in C : g \in \|\kappa\|\}$;
- $\|\langle U, C \rangle \gg \langle U', C' \rangle\| =_{df} \{g \mid \forall f \sim_U g \wedge f \in \|C\| \rightarrow \exists h \sim_{U'} f \wedge h \in \|C'\|\}$;

¹³I will ignore negation here.

- $\langle U, C \rangle$ is true under g iff_{df} $\exists f \sim_U g \wedge f \in \|C\|$.

To this system Zeevat adds a syntactic operation \otimes , for “merging” two DRSs into one:

$$(*) \quad \langle U, C \rangle \otimes \langle U', C' \rangle =_{df} \langle U \cup U', C \cup C' \rangle.$$

DRSs are given a denotation in terms of pairs,

$$\|\langle U, C \rangle\| =_{df} \langle U, \|C\| \rangle$$

whose first and second component are notated as $\|t\|_1$ and $\|t\|_2$ respectively. He has shown that this makes $\|\cdot\|$ a homomorphism between the syntactic and the semantic algebra of DRSs. In particular, with respect to \otimes we have:

$$(**) \quad \|\langle U, C \rangle \otimes \langle U', C' \rangle\| = \langle \|\langle U, C \rangle\|_1 \cup \|\langle U', C' \rangle\|_1, \|\langle U, C \rangle\|_2 \cap \|\langle U', C' \rangle\|_2 \rangle.$$

Notice that there is no unique decomposition of DRSs with respect to \otimes . In a more “linear” format, similar to Zeevat's *BL*,¹⁴ one can have expressions of the form $DRS_1 \otimes DRS_2$, which mention \otimes explicitly and thus uniquely decompose along this operation. The fact (**) then inductively defines the denotation of such DRSs. My systems will operate along the latter lines. Given such a syntax, interpreted by (**), we get the following “merging principle” to hold:

$$(MP) \quad \|\langle U, C \rangle \otimes \langle U', C' \rangle\| = \|\langle U \cup U', C \cup C' \rangle\|,$$

which semantically reflects Zeevat's earlier syntactic definition in (*).

Let me try to bridge the gap to our framework a bit further. Syntactically, sets of conditions may be replaced by conjunctions. Semantically, the sets of assignments $\|C\|$ can be turned into functions from assignments to truth, and finer grained propositions should be put in the place of truth values in order to get closer to our highly intensional approach.

There is of course an obvious difference to be observed. The SAC has an abstraction operation, which gives us terms $\lambda U.C'$ rather than pairs $\langle U, C \rangle$, denoting

¹⁴See section 1.3, in [Zeevat 1989].

functions (under assignments), rather than pairs. In contrast to DRS formation, we can iterate abstractions indefinitely, and we have a simple notion of application at our disposal. The downside is a certain loss of structure, to be faced in the SAC, as the denotation $\|\lambda U.C\|$ may be identical to $\|\lambda U'.C\|$, although $U \neq U'$. There is no way of recovering the set of variables U that was abstracted over from the denotation of a term, while it can be read off from Zeevat's $\|\langle U, C \rangle\|$ as the first component. This fact allows him to generate the correct interpretations for \gg and \otimes in a compositional way. Thus, we have to find other ways to semantically identify certain sets of variables as the universes in the denotations of DRSs, or we need to introduce more structure of the suitable kind into the semantics for our calculus. I explore the first option here, using a suitable notion of ω -properties. In the next chapter I pursue the route of changing the semantics of the SAC itself, towards a more fine grained meaning for abstraction, which draws on Zeevat's idea even more. The resulting system of Λ -DRT does seem to be sufficiently attractive to justify such a move.

3.3.2 ω -Properties

Let us regard the universe of a DRS as the set of argument roles of an ω -property. We start by defining a propositional function in the argument roles $x_1 \dots x_n$ as follows:

$$PF^{\{x_1 \dots x_n\}}(t) =_{df} \forall v_1 \dots v_n \bigwedge_{i=1..n} v_i \neq \# \rightarrow P(t(x_1.v_1, \dots, x_n.v_n)) \quad (v_i \text{ fresh}).$$

This is not a sufficiently strict notion of ω -properties to render Kamp Structures consistent. We need a notion of PTY^M such that the set of argument roles M is uniquely determined for any given ω -property. The predicate PF^M does not possess this property as abstracts are insensitive to overdefined assignments:

$$\models \{x_1 \dots x_n\}t(x_1.t_1, \dots, x_n.t_n) = \{x_1 \dots x_n\}t(x_1.t_1, \dots, x_{n+m}.t_{n+m}).$$

How strict a notion of ω -property one ultimately wants remains to be seen. For now I take it to imply that the set of argument roles M of an ω -property is the smallest set of roles in which it is a propositional function, and that it is a

propositional function in every set of roles containing M .

Definition 26 We define the notion of ω -PROPERTY IN THE ARGUMENT ROLES M by:¹⁵

$$PTY^M(t) =_{df} \bigwedge_{M \subseteq N} PF^N(t) \wedge \bigwedge_{M \not\subseteq N} \neg PF^N(t).$$

We introduce a new notion of internal representation of relations, which reflects the definedness requirements for the argument roles of ω -properties. A relation ϕ is INTERNALLY DEFINED BY THE ω -PROPERTY t if for some x_1, \dots, x_n :

$$PTY^{\{x_1 \dots x_n\}}(t) \wedge \forall v_1 \dots v_n \bigwedge_{i=1..n} v_i \neq \# \rightarrow T(t(x_1.v_1, \dots, x_n.v_n)) \leftrightarrow \phi_{v_1 \dots v_n}.$$

3.3.3 Kamp Structures

Let us now try to give a theory of ω -properties and propositions and truth that is capable of dealing with basic ideas of DRT. We add to the basic logical operations \cap, \perp, \approx a new quantificational operator, the binary conditional \gg .

Definition 27 A KAMP STRUCTURE is a Proposition Structure which satisfies the following axiom, where the v_i are all fresh:

$$PTY^{\{x_i | i \in I\}}(t) \wedge PTY^{\{x_j | j \in J\}}(s) \rightarrow P(t \gg s) \wedge T(t \gg s) \leftrightarrow \forall_{i \in I} v_i \neq \# T(t(x_i.v_i)_{i \in I}) \rightarrow \exists_{j \in J} v_j \neq \# T(s(x_i.v_i, x_j.v_j)_{i \in I, j \in J \setminus I})$$

The notion of a STRONG Kamp Structure is as expected. Letting $\Xi =_{df} \lambda X. \perp (X \gg \{ \}. p \cap \perp p)$ for some proposition p , we obtain the unselective existential quantifier that is needed to reflect the truth conditions of non-embedded DRSs:

$$PTY^{\{x_1 \dots x_n\}}(t) \rightarrow P(\Xi t) \wedge T(\Xi t) \leftrightarrow \exists v_1 \dots v_n \neq \# T(t(x_1.v_1, \dots, x_n.v_n)).$$

¹⁵I use an infinitary version of FOL_{SAC} to express this definition. This may seem unlegant, in particular as we only need finitary Λ_{AL} -terms for DRT. I do not regard this as problematic, as I attach no philosophical significance to the use of a formal language in expressing the theory. The system Λ -DRT in section 4.3 will be axiomatised informally.

The resulting theory is distinct from standard treatments of DRT in several aspects, of which one appears to be somewhat worrying: in the axiom of the conditional only bound variables in s are quantified, in contrast to standard DRT. This means for example that in $\{x, y\}.t_{xy} \gg \{x\}.s_{xy}$ only the x in s_{xy} will be bound by universal quantification, while the occurrence of y in s_{xy} remains free, and thus unaffected by \gg . The same need to abstract over anaphoric variables reappears when we introduce an operation of “merging” of DRSs \oplus , taken from [Cooper 1993], into the system, usually written in infix notation, where Y, Z are not among the X_i .

$$\oplus =_{df} \lambda Y \lambda Z \lambda \langle X_i \mapsto x_i \rangle_{x_i \in Var}. (Y(x_i, X_i)_{x_i \in Var} \cap Z(x_i, X_i)_{x_i \in Var}).$$

Item 3 of the following theorem approximates the “merging principle” of the previous section. It exposes the limitations of \oplus , which in contrast to Zeevat’s \otimes allows no capturing of free variables/parameters. Ways of overcoming this restriction will be discussed in the next chapter.

Theorem 36 1. $PTY^M(t) \wedge PTY^N(t') \rightarrow PTY^{M \cup N}(t \oplus t')$, if the Proposition Structure is strong.

2. $t \oplus t'(x_1, t_1, \dots, x_n, t_n) =_{\beta} t(x_1, t_1, \dots, x_n, t_n) \cap t'(x_1, t_1, \dots, x_n, t_n)$.
3. $\lambda \langle X_i \mapsto x_i \rangle_{i \in I}. t \oplus \lambda \langle X_i \mapsto x_i \rangle_{i \in I'}. s =_{\beta} \lambda \langle X_i \mapsto x_i \rangle_{i \in I \cup I'}. t \cap s$,
if $\{X_i \mid i \in I \setminus I'\} \cap FP(s) = \{X_i \mid i \in I' \setminus I\} \cap FP(t) = \emptyset$.

PROOF:

1. Let $M \cup N \subseteq L$, then $PF^L(t)$ and $PF^L(t')$, hence $PF^L(t \oplus t')$.

If $M \cup N \not\subseteq L$, e.g. by $M \not\subseteq L$, then $\neg PF^L(t)$. Hence by \cap being strong $\neg PF^L(t \oplus t')$.

2. By A5.
3. We may assume no X_i to be vacuously bound. Then
 $\lambda \langle X_i \mapsto x_i \rangle_{i \in I}. t \oplus \lambda \langle X_i \mapsto x_i \rangle_{i \in I'}. s$
 $\triangleright_{A5} \lambda \langle Z_i \mapsto x_i \rangle_{x_i \in Var}. [Z_i / X_i]_{i \in I} t \cap [Z_i / X_i]_{i \in I'} s$

$$\begin{aligned} &\triangleright_{A3} \lambda \langle Z_i \mapsto x_i \rangle_{i \in I \cup I'}. [Z_i / X_i]_{x_i \in I} t \cap [Z_i / X_i]_{x_i \in I'} s \\ &\triangleright_{A4} \lambda \langle X_i \mapsto x_i \rangle_{i \in I \cup I'}. t \cap s. \end{aligned}$$

□

Using partial application we can define DRT-type quantifiers Q_x for the “duplex conditions” of [Kamp & Reyle 1993], by means of standard generalized quantifiers Q as relations between ω -properties. The weak and strong readings are obtained as follows:

$$t(Q_x)^w t' =_{df} Q(\lambda z. \exists t[x.z], \lambda z. \exists (t \oplus t')[x.z]),$$

$$t(Q_x)^s t' =_{df} Q(\lambda z. \exists t[x.z], \lambda z. t[x.z] \gg t'[x.z]).$$

3.3.4 Consistency

To prove the existence of Kamp Structures we just need to check that the proof in section 3.1.2 for Frege Structures goes through for an operator θ_K which is like θ_F but replaces Ax_{Σ} with Ax_{\gg} (or just adds it). The only problem in doing so stems from the quantification over sets of variables in the axiom:

$$(Ax_{\gg}) \quad \forall M, N \subseteq Var : PC_l(\mathcal{P}, f, h, M, N) \Rightarrow (d_l \bullet f \in \mathcal{P} \text{ and } d_l \bullet f \in \mathcal{T} \Leftrightarrow TC_l(\mathcal{T}, f, h, M, N)).$$

We thus need to adapt the definition of θ_K to this format.

Definition 28 $\theta_K(\mathcal{P}, \mathcal{T}) =_{df} (\mathcal{P}', \mathcal{T}')$, where for the logical constants l of Kamp Structures:

- \mathcal{P}' is the smallest set such that for each l, f, h, M, N :
 $PC_l(\mathcal{P}, f, h, M, N) \Rightarrow d_{l,f,h} \in \mathcal{P}'$;
- \mathcal{T}' is the smallest set such that for each l, f, h, M, N :
 $PC_l(\mathcal{P}, f, h, M, N) \text{ and } TC_l(\mathcal{T}, f, h, M, N) \Rightarrow d_{l,f,h} \in \mathcal{T}'$.

Lemma 37 Any fixed point for θ_K satisfies Ax_{\gg} .

PROOF: Only part (iii) of the proof for Frege Structures needs to be examined. The crucial fact about ω -properties that we need here is that

$$(*) \quad PTY^M(d) \text{ and } PTY^{M'}(d) \Rightarrow M = M'.$$

Let $PC_{\gg}(\mathcal{P}, f, h, M, N)$ and $d_{\gg} \bullet f \in \mathcal{T}$. Then there are l, f', h', M', N' with $d_{\gg} \bullet f = d_{l, f', h'}$ such that $PC_i(\mathcal{P}, f', h', M', N')$ and $TC_i(\mathcal{T}, f', h', M', N')$. But then by $l = \gg$ and $f = f'$ we have $PC_{\gg}(\mathcal{P}, f, h, M', N')$, which implies that $M = M'$, and $N = N'$. Therefore $TC_{\gg}(\mathcal{T}, f, h, M, N)$. \square

The other axioms are obeyed as before. Again the result of the construction is in fact a strong Kamp Structure, as can be easily proved from $(*)$.

Theorem 38 *Every non-trivial SAC-model can be extended to a Kamp Structure with Predication, where infinitely many relations are internally defined by ω -properties.*

PROOF: For each given formula $\phi_{v_1 \dots v_n}$ we take a semantically distinct term r_{ϕ} , plus a term r_{ω} distinct from all r_t , and variables x_1, \dots, x_n and add the following axiom to the theory of Kamp Structures

$$\forall v_1 \dots v_n \bigwedge_{i=1..n} v_i \neq \# \rightarrow P(\langle\langle r_{\omega}; r_{\phi}; (x_i, v_i)_{i=1..n} \rangle\rangle) \wedge (T(\langle\langle r_{\omega}; r_{\phi}; (x_i, v_i)_{i=1..n} \rangle\rangle) \leftrightarrow \phi_{v_1 \dots v_n})$$

Given these proposition conditions, the least fixed point for θ_K makes $\langle\langle r_{\omega}; r_{\phi}; (z_j, v_j)_{j \in J} \rangle\rangle$ a proposition if and only if for all $j \in J$ $v_j \neq \#$ iff $z_j = x_i$ for some $i \in \{1, \dots, n\}$. Therefore we have

$$PTY^{\{x_1, \dots, x_n\}}(\lambda\{x_1, \dots, x_n\}. \langle\langle r_{\omega}; r_{\phi}; (x_i, x_i)_{i=1..n} \rangle\rangle).$$

Together with the truth conditions, ϕ is therefore internally defined by an ω -property. \square

3.4 Situation Theory

Situation Theory is different from DMG and DRT in that it is not a semantic theory that by itself provides at least the outlines of a syntax–semantics interface. Providing such an interface for ST is the task of “Situation Semantics”.¹⁶ Situation Theory concentrates on the (onto)logical side of the enterprise, as I do in this thesis. It fits less well into this piece of work though, because simultaneous abstraction is not really at the focus of ST’s attention. AL is important for modelling forms of ST, but the central contributions of ST lie in the theory of situations and their informational content, which has very little to do with simultaneous abstraction.¹⁷ My aim here is not to give a definitive version of ST, but rather to show how some rather mild assumptions about the part-of structure of events¹⁸ buys us rather cheaply a fair amount of ST, if we make use of the theory of Proposition Structures. My aim is to get a theory that is close to that of [Barwise & Cooper 1991], but rests on rather few axiomatic assumptions over and above those already introduced in previous sections.

We add to FOL_{SAC} the predicates *Sit* (being a situation) and \leq (part of), assuming that \leq is a partial order on situations. It seems reasonable to assume them to form a (complete) lattice, but I just add the weaker condition of directedness here. Let me use e, e', \dots as variables for events, writing ‘ $\forall e \dots$ ’ instead of ‘ $\forall v \text{ Sit}(v) \rightarrow \dots$ ’.

Definition 29 *the following axioms constitute our THEORY OF SITUATIONS:*

$$\begin{array}{lll} \text{(reflexivity)} & \forall e & e \leq e \\ \text{(transitivity)} & \forall e, e', e'' & e \leq e' \wedge e' \leq e'' \rightarrow e \leq e'' \\ \text{(antisymmetry)} & \forall e, e' & e \leq e' \wedge e' \leq e \rightarrow e = e' \\ \text{(directedness)} & \forall e, e' \exists e'' & e \leq e'' \wedge e' \leq e'' \end{array}$$

¹⁶There are of course countless ways of going about this task, which may make use of ST to varying degrees.

¹⁷No unique, generally accepted, Situation Theory has emerged from these efforts. In [Barwise 1989] a number of “choice points” have been isolated, which provide a map for the various versions of ST one might encounter. I leave most of these choices open, to be fixed in the light of the semantic application or particularly strong metaphysical intuitions.

¹⁸I use ‘event’ and ‘situation’ interchangeably.

I take these axioms to be fairly uncontroversial among those who believe that events have a role to play in semantic theories. One might want to restrict the last axiom to events which are part of the same possible world, if one assumes the existence of non-actual “possible situations”, for it would lead to “impossible situations” otherwise. I will take an actualist stance here. Non-realised types of events will hopefully compensate for the absence of other worlds, though the matter is not of central concern here.

3.4.1 States of Affairs

Typically in Situation Theory one classifies situations e by certain “states of affairs” (soa’s) σ which they “support”, written $e \models \sigma$. Soa’s have traditionally been taken as fundamental entities of ST, with situations being regarded as something rather like sets of them. Even when situations are taken as primitives, and support as the ST correlate to truth, they still usually are in close correspondence with the sets $\{\sigma | e \models \sigma\}$,¹⁹ and their part-of structure is defined in terms of the subset relation on those sets.

I do not want to pursue an extreme version of ST here where all propositions are “Austinian”, that is of the form ‘ $e \models \sigma$ ’. Given a more liberal view of propositions, it seems most natural to take part-of as basic, together with truth, and define support of soa’s in terms of these: soa’s are persistent types of events, and a situation supporting a soa is just another case of an object having a certain property. This brings ST closer in line with our versions of DMG and DRT, and makes consistency a trivial matter: Our Theory of Situations poses no particular consistency problem, and so long as Sit and \leq are given in advance it is also consistent for them to be internally defined in a Proposition-, Frege-, or Kamp Structure.

A critical point for ST is the treatment of “negative soa’s”: there is a crucial partiality in that a situation may fail to support a soa as well as its negation. It is tempting to try to capture this by saying that ‘ $e \models \sigma$ ’ sometimes fails to express a proposition internal to the Proposition Structure, thus exploiting the truth value gaps allowed for in that theory. But this would not lead very far as we

¹⁹It is thus often assumed that $\{\sigma | e \models \sigma\} = \{\sigma | e' \models \sigma\}$ implies $e = e'$.

have no weak negation in that system: our negation only produces a truth if the negated object is a false proposition. The system would be inconsistent if objects without truth value could be negated, for example to form a true proposition ‘ $e \not\models \sigma$ ’ in cases where ‘ $e \models \sigma$ ’ is thought of as lacking in truth value.²⁰ Hence I have to insist that soa’s and other types of events always yield propositions when applied to an event, which means that the strong, non-bivalent negation of soa’s has to be introduced by additional axioms.

Definition 30 *We define*

- $Toe(t) =_{df} \forall e P(t(e))$ (“ t is a TYPE OF EVENTS”).
I use τ, τ', \dots for toe’s and write $e \models \tau$ for $T(\tau(e))$.
- $Soa(t) =_{df} Toe(t) \wedge \forall e, e' (e \models t \wedge e \leq e' \rightarrow e' \models t)$ (“ t is a STATE OF AFFAIRS”).
Soa’s are referred to by σ, σ', \dots . Toe’s can be conjoined, disjoined, and weakly negated. We define
- $\sqcap =_{df} \lambda X \lambda Y \lambda Z. X(Z) \cap Y(Z)$;
- $\sqcup =_{df} \lambda X \lambda Y \lambda Z. X(Z) \cup Y(Z)$;
- $\sim =_{df} \lambda X \lambda Z. \perp X(Z)$.

Theorem 39 *In every Proposition Structure obeying the Theory of Situations holds:*

1. $Soa(\sigma) \wedge Soa(\sigma') \rightarrow Soa(\sigma \sqcap \sigma')$;
2. $Soa(\sigma) \wedge Soa(\sigma') \rightarrow Soa(\sigma \sqcup \sigma')$;
3. $Toe(\tau) \rightarrow Toe(\sim \tau)$;
4. $e \models \tau \sqcap \tau' \leftrightarrow e \models \tau \wedge e \models \tau'$;
5. $e \models \tau \sqcup \tau' \leftrightarrow e \models \tau \vee e \models \tau'$;
6. $e \models \sim \tau \leftrightarrow e \not\models \tau$.

²⁰This is in contrast to Muskens’ work, cf. [Muskens 1989], who uses a typed logic in which weak negation is unproblematic.

3.4.2 Barwise Structures

We saw above that our standard notion of negation is too weak to capture the partiality of negation in Situation Theory. The intuition for strong negation seems to be that a situation supports a negative fact, $e \models \bar{\sigma}$, if some fact in e precludes σ from holding, and does so in a persistent way. Using strong implication and a property *sit* which internally defines the predicate *Sit*, one might try a definition like $\bar{\sigma} =_{df} \lambda X. \Pi Y. sit(Y) \supset \perp \sigma(Y)$. But this would make negative soa's downward persistent, and only spuriously situated. Perhaps an internal entailment relation \Rightarrow between propositions, roughly similar to the “constraints” of ST, could be used, together with an internally defined \leq_i to express the above intuition as $\bar{\sigma} =_{df} \lambda X. \Pi Y. sit(Y) \wedge X \leq Y \Rightarrow \perp \sigma(Y)$. This would presumably be genuinely partial and not downward persistent, but I have no results on how appropriate entailment relations could be internalized, and such a notion of negation may fail to derive the full set of properties of strong negation, as they are usually assumed in ST.²¹

Let me thus follow the standard ST account more closely, and assume we are given a list $u_1, \bar{u}_1, u_2, \bar{u}_2, \dots, u_n, \bar{u}_n$ of symbols for basic situated relations which apply to an assignment and a situation to form propositions.

Definition 31 A **BARWISE STRUCTURE** is a Proposition Structure satisfying the Theory of Situations such that for each $j \in \{1..n\}$ there are roles $\{x_i\}_{i \in I_j}$ such that $\forall_{i \in I_j} v_i \neq \#$:

$$\begin{aligned} & Soa(u_j(x_i, v_i)_{i \in I_j}) \wedge Soa(\bar{u}_j(x_i, v_i)_{i \in I_j}) \\ \forall e \quad & e \models \bar{u}_j(x_i, v_i)_{i \in I_j} \rightarrow e \not\models u_j(x_i, v_i)_{i \in I_j} \\ \exists e \quad & e \models u_j(x_i, v_i)_{i \in I_j} \vee e \models \bar{u}_j(x_i, v_i)_{i \in I_j} \end{aligned}$$

²¹One may of course wonder whether these assumptions are well motivated if propositions are taken to be more fundamental than soa's, as I am inclined to do.

Definition 32 We can now introduce standard notations for basic and complex negative soa's:²²

- $\ll u_j, (x_i, t_i)_{i \in I_j}; 1 \gg =_{df} u_j(x_i, t_i)_{i \in I_j}$;
- $\ll u_j, (x_i, t_i)_{i \in I_j}; 0 \gg =_{df} \bar{u}_j(x_i, t_i)_{i \in I_j}$;
- $\overline{\sigma_1 \sqcap \sigma_2} =_{df} \bar{\sigma}_1 \sqcup \bar{\sigma}_2$;
- $\overline{\sigma_1 \sqcup \sigma_2} =_{df} \bar{\sigma}_1 \sqcap \bar{\sigma}_2$;
- $\bar{\bar{\sigma}} =_{df} \sigma$.

By induction on σ and the directedness of \leq we have for all soa's σ :

Theorem 40 $\exists e \ e \models \bar{\sigma} \leftrightarrow \forall e \ e \not\models \sigma$.

3.4.3 Consistency

Theorem 41 Any SAC-model can be turned into a Barwise Structure.

PROOF: Let R_i, \bar{R}_i , for $i = 1..n$, be a set of n_i -place relations satisfying $\forall v_2..v_{n_i} \neq \#$:

- $\forall e, e' \ R_i(e, v_2, \dots, v_{n_i}) \wedge e \leq e' \rightarrow R_i(e', v_2, \dots, v_{n_i})$;
- $\forall e, e' \ \bar{R}_i(e, v_2, \dots, v_{n_i}) \wedge e \leq e' \rightarrow \bar{R}_i(e', v_2, \dots, v_{n_i})$;
- $\forall e \ (R(e, v_2, \dots, v_{n_i}) \rightarrow \forall e' \ \neg \bar{R}(e', v_2, \dots, v_{n_i}))$;
- $\exists e \ R(e, v_2, \dots, v_{n_i}) \vee \bar{R}(e, v_2, \dots, v_{n_i})$.

²²It would be nice to be able to show consistency for a theory that has strong negation as an internal operation, with appropriate axioms of support, rather than using the following metasyntactic definitions.

Such relations clearly exist, and can be internally defined by n_i -place properties t_i, \bar{t}_i . We then take

$$u_i =_{df} \lambda\{x_2, \dots, x_{n_i}\} \lambda x_1. t_i x_1 \dots x_{n_i}$$

$$\bar{u}_i =_{df} \lambda\{x_2, \dots, x_{n_i}\} \lambda x_1. \bar{t}_i x_1 \dots x_{n_i}$$

The Barwise axioms are then obeyed as required. \square

It is clear from the fixed point construction that the additional quantificational axioms of Frege- and Kamp-Structures are all consistent with this theory too. So we can throw them all together without any worries.

Theorem 42 *Any SAC-model can be turned into a Frege-Kamp-Barwise Structure with Predication.*

Chapter 4

Further Directions

In this chapter I will be mainly exploring variations of the SAC that might bring it closer to more popular versions of DRT. After a look at dynamic semantics, and a typed version of the SAC, I will pursue what seems to me a rather promising direction of giving a slightly more structured semantics to the SAC, inspired by [Zeevat 1989]. Under this semantics the set of abstracted variables is a defining ingredient to the meaning of Λ -terms. This allows us to define versions of the conditional, and the merge-operator, which are closer to the originals than those of the preceding chapter. Finally I briefly discuss the notions of restriction and appropriateness, which figure prominently in [Barwise & Cooper 1991].

An understanding of sections 3.1 to 3.3 will be presupposed for most of this chapter.

4.1 True Dynamics

The system of DRT presented in chapter 3.3 did not use any non standard kind of binding of variables beyond the scope of an appropriate operator. But such binding is prominent in so called “dynamic” forms of semantics, which thus stand in some contrast to my approach. It is time to ask what the relation is between the SAC and systems like the Dynamic Predicate Logic (DPL) of [Groenendijk & Stokhof 1991], which are based on relational interpretations. DPL replaces the non-standard DRT syntax by FOL formulas, but they have

shown how to give the relational interpretation for DRSs too. Let us view relations between assignments as functions from assignments to sets of assignments. Call sets of assignments ω -sets, and take conditions to denote just ω -sets $\|\phi\|$. We then get the following denotation for a DRS:

$$\|\{x_1 \dots x_n\}\phi\|_{DPL}^g = \{h \mid g \stackrel{\{x_1 \dots x_n\}}{\sim} h \wedge h \in \|\phi\|\}.$$

This should be compared with the extensions of such terms as they are interpreted in Kamp Structures. Let $ext(d) =_{df} \{h \mid d \bullet h \in \mathcal{T}\}$. We then also get ω -sets as denotations under an assignment, but they are different from the DPL ones:

$$ext(\|\{x_1 \dots x_n\}\phi\|_{\Lambda}^g) = \{h \mid g \stackrel{\{x_1 \dots x_n\}}{h} \in ext(\Phi(\|\phi\|_{\Lambda}))\}.$$

The point is that the h 's in our set can assign anything they like to the variables outside $\{x_1 \dots x_n\}$, while for the DPL style of denotation they have to agree with the incoming g .

A natural development of DPL, carried further in [Dekker 1993], is to view “information states” as such ω -sets, and then lift denotations of DRSs to functions from ω -sets to ω -sets, such as:

$$\|\phi\|_{DPL\uparrow}(G) = \bigcup_{g \in G} \|\phi\|_{DPL}^g$$

which is quite different from the trivial lifting from ϕ to $\lambda p.(p \oplus \phi)$ that might be used in our system to make our meanings less static.

The lack of a requirement for assignments to agree on certain variables has consequences for semantics. In contrast to DPL for example we had to abstract over variables in order to link them anaphorically to preceding discourse by our discourse sequencing operation \oplus . I will discuss other kinds of sequencing, which may allow the capturing of free variables, in due course. What is clear though is that relational composition, the hallmark of DPL-style dynamics, is not an option for SAC-style meanings. If we would use DPL style composition for discourse conjunction the referential link between identical variables could not be sustained

over more than two sentences, because in the SAC any “outgoing” assignment is free to disagree with the “incoming” one. So I opt for a version of Zeevat’s merging operation to link anaphora to their antecedents. In contrast to composition, merging is commutative¹, thus allowing us to capture cataphora in the same way. A similar divergence between the two approaches is found in the treatment of the conditional. Again we had to put anaphoric variables into the abstracted set of the consequence, but only those which are anaphoric to the antecedent, not those which refer further back.

There is something of a philosophical gap between the approach of this paper and the “truly dynamic” ones, which is brought out by these considerations. On the SAC view the functions from assignments to ω -properties are not regarded as ways of “updating” information states by incoming material. It makes no sense on this view to talk of “input” and “output” in reference to the assignments. My guiding intuition is rather that semantic objects should be plausible as things to have attitudes towards. These objects, I take it, are rather static creatures with reference to which such notions as “information change” should be defined. So if we are to introduce some way of capturing free variables into our system it should be done in a way that preserves the fundamentally static view of semantic objects that underlies the proposed framework.

4.2 A Typed System

I will sketch a typed version of the SAC here, with some operators added to formalize DRT in relatively simple, extensional fashion. The purpose is to illustrate how a simple way of typing the SAC can work, and to provide a system of DRT which looks more familiar to standard versions than the previous ones. In particular it provides a way to bind free variables outside of the scope of their abstracted antecedents, just as it was done in traditional versions of the DRT conditional, and with Zeevat’s merge-operator \otimes , described in chapter 3.3.1. By using the universe of a DRS as part of the typing information we will be able to prove

¹Strictly speaking this is only so if conjunction is commutative. Still, it’s close enough for our purposes.

as sound the merging principle of DRSs (MP) in full generality, which states that

$$(MP) \quad \models \lambda M.t \otimes \lambda M'.t' = \lambda M \cup M'.t \cap t'$$

even if some variables in M occur free in t' , or some in M' occur free in t . This means that free variables can be captured to the left and to the right, via merging with DRSs in which they are abstracted over.

It is clear that we cannot get the equation in full generality for Λ , under its original interpretation, when we look at vacuous abstractions. We would need to prove for example that $(\{x\}.true) \otimes (\{y\}.F(x))$ equals $\{x\}.(true \cap F(x))$. But under the old semantics $\models \{x\}.true = \{y\}.true$, and thus MP would imply that $\{x\}.(true \cap F(x))$ equals $\{y\}.(true \cap F(x))$, which is clearly not correct.

It should be noted that the capturing of free variables as such is not incompatible with the original semantics of Λ . One could for example capture all discourse markers in a merger of two terms by defining

$$\|t \otimes t'\|^g = \Phi \lambda f \|t\|^{g_{DM}}(f) \wedge \|t'\|^{g_{DM}}(f)$$

where \wedge is the semantic correlate of conjunction. The problem would be that we capture too much to validate the merging principle: if $M, M' \subseteq DM$ we would get $\models \lambda M.t \otimes \lambda M'.t' = \lambda DM.t \cap t'$. What we need is a semantic notion of the roles of abstracts, corresponding to the sets of their abstracted variables, here M and M' . This allows us to restrict our catch to $M \cup M'$, leaving the rest free to be caught by other abstractions. The types, with the correspondingly restricted denotations of Λ_{TDRT} , provide such a notion. Another way to provide it will be explored in due course.

In order to type SAC expressions one has to build function-types that reflect the fact that terms apply to variable assignments. So on the argument side of such a type we introduce an indication of what type the objects have to be that are assigned to which variables. A set of typed variables provides this kind of information.² Hence a function type will be a pair $\langle M, \tau \rangle$, where M is a set of

²For an AL-style system on might use typed roles, or mappings from (untyped) roles to types.

typed variables and τ is a type. We thus define the set of types and the set of variables simultaneously.

Definition 33 Let *TYPE* and *VAR* be the smallest set and family of sets such that

- $e, t \in Type$;
- $\tau \in Type \Rightarrow v_\tau^1, v_\tau^2, \dots \in Var_\tau$;
- $\tau \in Type$ and $x_i \in Var_{\tau_i}$ for $i \in I \Rightarrow \langle \{x_i | i \in I\}, \tau \rangle \in Type$.

Let $Var = \bigcup_{\tau \in Type} Var_\tau$ and $\theta(x) = \tau$ iff $x = v_\tau^n$.

Definition 34 The set of *TERMS* of Λ_{TDRT} are defined by:

- $x \in Var_\tau \Rightarrow x \in Term_\tau$;
- $\tau \in Type \Rightarrow c_\tau \in Term_\tau$;
- $t \in Term_\tau$ and $M \subseteq Var \Rightarrow \lambda M.t \in Term_{\langle M, \tau \rangle}$;
- $t \in Term_{\langle \{x_i | i \in I\}, \tau \rangle}$ and $t_i \in Term_{\theta(x_i)}$ for $i \in I \Rightarrow t(x_i, t_i)_{i \in I} \in Term_\tau$;
- $t \in Term_{\langle M, t \rangle}$ and $t' \in Term_{\langle M', t \rangle} \Rightarrow t \otimes t' \in Term_{\langle M \cup M', t \rangle}$ and $t \gg t' \in Term_t$;
- $t, t' \in Term_t \Rightarrow t \cap t', \perp t \in Term_t$;
- $t, t' \in Term_\tau \Rightarrow t = t' \in Term_t$.

Notice that two DRSs $\lambda M.t$ and $\lambda M'.t'$ are of different types if $M \neq M'$. The operations \otimes and \gg thus have to be polymorphic ones, which cannot be expressed by typed constants. Such a simple minded typing scheme is too restrictive to be of much practical use, as it prevents abstraction over DRSs with varying universes. It seems desirable to investigate conceptions of polymorphism and subtyping for the SAC to make further progress in this direction.

For illustrative purposes I will nevertheless give a simple, extensional semantics to the system. Let U be a given set of objects, $D \rightarrow D'$ be the full set of total functions from D to D' , and we define the sets D_τ and type correct variable assignments $M \rightarrow D$ as follows.

Definition 35 Given a universe U we define TYPED DOMAINS by:

- $D_e = U$;
- $D_t = \{0, 1\}$;
- $D_{(M, \tau)} = (M \rightarrow \bigcup_{x \in M} D_{\theta(x)}) \rightarrow D_\tau$,
where $f \in M \rightarrow \bigcup_{x \in M} D_{\theta(x)}$ iff $dm(f) = M$ and $f(x) \in D_{\theta(x)}$ for all $x \in M$.

Let $D =_{df} \bigcup_{\tau \in \text{Type}} D_\tau$, and \wedge, \neg be boolean conjunction and negation on D_t . The notation $\langle x_i \mapsto d_i \rangle_{i \in I}$ stands for (truly) partial assignment functions, and $f|_M$ is the restriction of f to M .

Definition 36 The DENOTATION $\|\cdot\|$ of Λ_{DRT} -terms in a model $\langle D, \mathfrak{S} \rangle$ under a type correct variable assignment g are defined by:

- $\|c_\tau\|^g = \mathfrak{S}(c_\tau)$;
- $\|v_\tau^n\|^g = g(v_\tau^n)$;
- $\|t = t'\|^g = 1$ iff $\|t\|^g = \|t'\|^g$, otherwise 0;
- $\|t \cap t'\|^g = \|t\|^g \wedge \|t'\|^g$;
- $\|\perp t\|^g = \neg \|t\|^g$;
- $\|\lambda M.t\|^g = \lambda_{f_{M \rightarrow D}} \|t\|^{g_f^M}$;
- $\|t(x_i, t_i)_{i \in I}\|^g = \|t\|^g(\langle x_i \mapsto \|t_i\|^g \rangle_{i \in I})$;
- $\|t_{(M, t)} \gg t'_{(M', t)}\|^g = 1$ iff $\forall f_{M \rightarrow D} \|t\|^g(f) = 1 \Rightarrow \exists h_{M' \rightarrow D} \|t'\|^{g_f^M}(h) = 1$, otherwise 0;
- $\|t_{(M, t)} \otimes t'_{(M', t)}\|^g = \lambda_{f_{M \cup M' \rightarrow D}} \|t\|^{g_f^M}(f|_M) \wedge \|t'\|^{g_f^{M'}}(f|_{M'})$.

Notice in particular the last clause: we conjoin the conditions s and s' when evaluating $\lambda M.s \otimes \lambda M'.s'$, while making sure that variables in $M \cup M'$ are interpreted by f wherever they are free in s' or s . This is achieved by not only applying the denotations of $\lambda M.s$ and $\lambda M'.s'$ to suitable restrictions of f , but also

overwriting g by f on the variables to be captured. Only free variables which are not in the universe of either of the two merged DRSs receive their values from the assignment g , and thus behave as ordinary placeholders whose values are fixed by the context of evaluation.

Theorem 43 (Merging Principle) $\models_{\text{TDRT}} \lambda M.t \otimes \lambda M'.t' = \lambda M \cup M'.t \cap t'$

PROOF: $\|\lambda M.t \otimes \lambda M'.t'\|^g = \lambda_{f_{M \cup M' \rightarrow D}} \|\lambda M.t\|^{g_f^M}(f|_M) \wedge \|\lambda M'.t'\|^{g_f^{M'}}(f|_{M'})$
 $= \lambda_{f_{M \cup M' \rightarrow D}} \|t\|^{g_f^M} \wedge \|t'\|^{g_f^{M'}} = \|\lambda M \cup M'.t \cap t'\|^g$. \square

In a type free, or more liberally typed system, one may need to introduce additional structure into the semantics to achieve the same result. This is taken up in the next section.

4.3 Λ -DRT

In this section I will introduce a modified semantics for Λ , enriched with the basic operation \otimes , which integrates the earlier presented ideas of Henk Zeevat into the type free SAC. It is also closely related to the system of λ -DRT [Millies & Pinkal 1992], which syntactically integrates ordinary typed λ -abstraction with DRS-formation over variables of type e . An official semantics for their system has not yet been published, and there is reason to think that giving one is not an entirely straightforward task.³

Definition 37 The language Λ_{DRT} consists of TERMS t, t', \dots built up from basic CONSTANTS $c, c', \dots, \#$, and VARIABLES $x, y, \dots \in \text{Var}$ by means of ABSTRACTION $\lambda M.t$, for $M \subseteq \text{Var}$, APPLICATION $t(x_i, t_i)_{i \in I}$, and MERGE $t \otimes t'$.

In contrast to the original SAC semantics we associate not only a particular applicative behaviour with each object of the domain D but also a unique set of

³By the time of writing, a semantics for λ -DRT, which relies on a somewhat non-standard interpretation of λ -abstraction, had been proposed in [Kuschert 1995]. This interpretation seemed somewhat problematic, and has been improved upon, in [Kohlhase et al. 1996]. A comparison between this semantics and my system should therefore be of some interest.

variables. This makes our domain even more intensional, in the sense that for two abstraction terms to denote the same object, they not only have to have the same applicative behaviour but also must abstract over exactly the same set of variables.⁴ For this to work, we need to turn the power set of Var into a domain, in a way that allows us to obey the continuity requirements of our semantics. Unfortunately the naive approach of using the subset relation to form a cpo won't work. Instead I assume that $\text{Pow}(Var)$ is a flat domain of sets of variables, with an added \perp_{PV} element, rather than a true power-domain. We need to extend the notions of set-union and assignment updating to the cases involving \perp_{PV} . The cases involving proper (possibly empty) sets of variables remain as before, and we define $M \cup \perp_{PV} = \perp_{PV} \cup M =_{df} \perp_{PV}$ and $g \uparrow_{PV} =_{df} \perp_{Var \rightarrow D}$.

The retraction by means of which abstraction and application are interpreted is thus slightly more complicated in mapping between objects d and pairs $\langle M, \mu \rangle$, where $M \in Pow(Var)$ is the set of roles of d (on the outermost level), and $\mu \in (Var \rightarrow D) \rightarrow D$ describes d 's applicative behaviour. We thus use domains D such that

$$D \rightleftharpoons_{\Phi} Pow(Var) \times ((Var \rightarrow D) \rightarrow D).$$

From this we define $\Psi_1(d)$ as the first component of $\Psi(d)$, also called the **ROLES** of d ,⁵ and $\Psi_2(d)$ as the second component, called the **APPLICATIVE BEHAVIOUR** of d . We write $d \bullet f$ for $(\Psi_2(d))(f)$.

Definition 38 A Λ_{DRT} -MODEL consists of a retraction $D \rightleftharpoons_{\Phi} Pow(Var) \times ((Var \rightarrow D) \rightarrow D)$ and an interpretation \mathfrak{S} which maps constants into D , with $\mathfrak{S}(\#) = \perp_D$. The denotation function $\|\cdot\|^g$ for terms, under a variable assignment g , is given by:

- $\|c\|^g = \mathfrak{S}(c)$;
- $\|x\|^g = g(x)$;

⁴Translated into Λ_{AL} this means that only Axioms A3 and A7 of the proof theory are no longer sound in this system. Notice that Rule 4 still holds, giving us a weakened form of extensionality. The change seems to have no further dramatic impact on the theory developed in chapter 2, as far as I can see. In particular, all encodings of λ -terms have the same roles, namely z or whatever we choose, and thus weak extensionality for λ is preserved.

⁵Such a *roles*-function is also assumed to exist in [Cooper & Poesio 1994].

- $\|\lambda M.t\|^g = \Phi \langle M, \lambda f \|\cdot\|^g \rangle$;
- $\|t(x_i, t_i)_{i \in I}\|^g = \|t\|^g \bullet \langle x_i \mapsto \|t_i\|^g \rangle_{i \in I}$;
- $\|t \otimes t'\|^g = \Phi \langle M \cup M', \lambda f. \|t\|^g \bullet f \wedge \|t'\|^g \bullet f \rangle$,
where $M = \Psi_1(\|t\|^g)$ and $M' = \Psi_1(\|t'\|^g)$.

Theorem 44 $\lambda f \|\cdot\|^g$ is continuous.

PROOF: By induction, along the lines of Theorem 1. I leave it to the reader to adapt monotonicity and the other induction steps, and only show the induction step for \otimes here. Recall that $h^* = \bigsqcup_{n < \omega} h_n$. Let $N_n = \Psi_1(\|t\|^g \bullet h_n)$, $N'_n = \Psi_1(\|t'\|^g \bullet h_n)$, $N_* = \Psi_1(\|t\|^g \bullet h^*)$ and $N'_* = \Psi_1(\|t'\|^g \bullet h^*)$. Notice that $N_* = \bigsqcup_{n < \omega} N_n$, and $N'_* = \bigsqcup_{n < \omega} N'_n$, by the induction hypothesis. Also $\bigsqcup_{n < \omega} g_{h_n f}^{MN_n} = \bigsqcup_{n < \omega} g_{h_n f}^{MN_*}$ and $\bigsqcup_{n < \omega} g_{h_n f}^{MN'_n} = \bigsqcup_{n < \omega} g_{h_n f}^{MN'_*} = g_{h_* f}^{MN'_*}$.

$$\begin{aligned} \text{Thus, } \lambda f \|t \otimes t'\|^g (h^*) &= \|t \otimes t'\|^g \bullet h^* \\ &= \Phi \langle N_* \cup N'_*, \lambda f. \|t\|^g \bullet f \wedge \|t'\|^g \bullet f \rangle \\ &= \Phi \langle N_* \cup N'_*, \lambda f. \|t\|^g \bullet f \wedge \|t'\|^g \bullet f \rangle \\ &= \Phi \langle \bigsqcup_{n < \omega} N_n \cup \bigsqcup_{m < \omega} N'_m, \lambda f. \bigsqcup_{n < \omega} \|t\|^g \bullet f \wedge \bigsqcup_{m < \omega} \|t'\|^g \bullet f \rangle \\ &= \Phi \langle \bigsqcup_{n < \omega} (N_n \cup N'_n), \bigsqcup_{n < \omega} (\lambda f. \|t\|^g \bullet f \wedge \|t'\|^g \bullet f) \rangle \\ &= \bigsqcup_{n < \omega} \Phi \langle N_n \cup N'_n, \lambda f. \|t\|^g \bullet f \wedge \|t'\|^g \bullet f \rangle \\ &= \bigsqcup_{n < \omega} \|t \otimes t'\|^g (h_n). \quad \square \end{aligned}$$

The semantics of \otimes bears close resemblance to the one in the typed system of the previous section. \wedge is the semantic operation of conjunction, i.e. $d \wedge d' =_{df} \|x \cap x'\|^g \bullet d \bullet d'$. The addition of the set of argument roles as a component of the meaning of terms allows us to capture free variables in the intended way: just as in Λ_{DRT} those free variables of a DRS which are abstracted in the other DRS of a merge-expression are caught, while others remain free to be accessed by other abstractions.

Theorem 45 (Merging Principle) $\models_{\Lambda_{DRT}} \lambda M.t \otimes \lambda M'.t' = \lambda M \cup M'.t \cap t'$.

$$\begin{aligned}
& \text{PROOF: } \|\lambda M.t \otimes \lambda M'.t'\|^g \\
&= \Phi \langle M \cup M', \lambda f. \|\lambda M.t\|^g \bullet f \wedge \|\lambda M'.t'\|^g \bullet f \rangle \\
&= \Phi \langle M \cup M', \lambda f. \|\lambda M.t\|^g \wedge \|\lambda M'.t'\|^g \rangle \\
&= \Phi \langle M \cup M', \lambda f. \|\lambda M.t\|^g \wedge \|\lambda M'.t'\|^g \rangle \\
&= \Phi \langle M \cup M', \lambda f. \|\lambda M.t\|^g \wedge \|\lambda M'.t'\|^g \rangle \\
&= \|\lambda M \cup M'.t \cap t'\|^g. \quad \square
\end{aligned}$$

Capturing free variables is clearly possible in this way, but we cannot hope to increase the catch by applications to terms containing such free variables. For example, we cannot derive an equation where the VP content $\lambda z\{x\}.Gxz$ contains an anaphoric reference 'x' to the content $\lambda P.\{x\}.Fx \otimes Px$ of the subject NP,⁶ as in

$$(\neq) (\lambda P.\{x\}.Fx \otimes Px) (\lambda z\{x\}.Gxz) = \{x\}.Fx \otimes \{x\}.Gxz.$$

It has been assumed in λ -DRT that such equations hold, without having given a semantic justification for them. But this is not a problem that can't be surmounted. We simply use the \ulcorner - and \urcorner -operators of DMG, from chapter 3.2, to get a cleaned up version of this form of variable capture. Recall their definition as $\lambda DM.t$ and $t(\dot{x}.x)_{x \in DM}$ from that section, where DM is some infinite subset of the variables. The following is valid in our Λ -DRT, given $\dot{x} \in DM$:

$$(\lambda P.\{\dot{x}\}.F\dot{x} \otimes \ulcorner Px \urcorner) \ulcorner (\lambda z\{x\}.Gxz) \urcorner = \{\dot{x}\}.F\dot{x} \otimes (\ulcorner \lambda z\{x\}.Gxz \urcorner) \dot{x} = \{\dot{x}\}.F\dot{x} \otimes \{\dot{x}\}.G\dot{x}\dot{x}$$

which by the MP reduces to $\{\dot{x}\}(F\dot{x} \cap G\dot{x}\dot{x})$.

Notice that this kind of system uses three apparently different kinds of abstraction: ordinary unary λ -abstraction, abstraction over the universes of DRSs, and \ulcorner -abstraction to "intensionalize" over discourse marker assignments. Yet these three operations are in fact, as I have shown, instances of one and the same form of abstraction, namely simultaneous abstraction, as it is formalised in Λ and Λ_{AL} . The present modification in the semantics of the SAC in this section does

⁶This example assumes that anaphora are resolved before the compositional DRS construction takes place. For a different approach see [Asher 1993].

not affect that result.

I thus propose DMG style "intensional" application $t[t']$ as a general mode of semantic composition for Λ -DRT, in order to extend the capturing of variables beyond the scope of \otimes . Having the extra level of abstraction \ulcorner in the representation $\ulcorner \lambda U.C \urcorner$ of DRSs is also useful for giving a truth-axiom for \gg . If we would insist on using free variables, as in standard DRT, we would be unable to regard \gg as an operator, or constant, which takes two terms to form a complex one, whose denotation under assignment g is a function of the denotations of its arguments under g .⁷ The additional abstraction \ulcorner over all discourse markers allows us to view \gg as a constant, as we did in the earlier system of Kamp Structures. Instead of PTY^M we therefore need a similar predicate DRS_{DM}^M , which refers to the second level of argument roles of (\ulcorner -lifted) DRSs. Given two lifted DRSs $\ulcorner \lambda U.C \urcorner$ and $\ulcorner \lambda U'.C' \urcorner$, we get a condition $\ulcorner \lambda U.C \urcorner \gg \ulcorner \lambda U'.C' \urcorner$ which is also lifted, in the sense that it needs to be applied to a discourse marker assignment to form a proposition.

I will state the new axiom of truth in semantic terms here, rather than in some variant of FOL_{SAC} . Let \gg be the operation that interprets the constant \gg .

Definition 39 1. $DRS_{DM}^M(d) \Leftrightarrow_{df} \forall f \, dm(f) \supseteq DM \Rightarrow$

$$\Psi_1(d \bullet f) = M \wedge \forall h \, dm(h) \supseteq M \Rightarrow d \bullet f \bullet h \in \mathcal{P}.$$

2. $CND_{DM}(d) \Leftrightarrow_{df} \forall f \, dm(f) \supseteq DM \Rightarrow d \bullet f \in \mathcal{P}.$

3. A KAMP STRUCTURE for Λ -DRT is a Λ_{DRT} -model with subsets \mathcal{T}, \mathcal{P} obeying the Proposition Structure axioms, such that for all $d, d' \in D$:

$$DRS_{DM}^M(d) \wedge DRS_{DM}^M(d') \Rightarrow CND_{DM}(d \gg d') \wedge$$

$$\forall f \, dm(f) \supseteq DM \Rightarrow (d \gg d') \bullet f \in \mathcal{T} \Leftrightarrow$$

$$\forall h \, dm(h) \supseteq M \wedge (d \bullet f \bullet h) \in \mathcal{T} \Rightarrow \exists k \, dm(k) \supseteq M' \wedge (d' \bullet f_h^M \bullet k) \in \mathcal{T}.$$

⁷Notice that \otimes is not treated as a term here for this reason: $\|\lambda M.t\|^g$ is not a function of $\|t\|^g$ and $\|t'\|^g$, but of $\|t\|$, $\|t'\|$ and g , similar to $\|\lambda M.t\|^g$ being a function of $\|t\|$, g and M , not of $\|t\|^g$ and M .

One might want to always abstract over all discourse markers of a DRS, thus uniformly representing DRSs $\langle U, C \rangle$ as $\sqcap \lambda U.C'$ rather than $\lambda U.C'$. The merge-operator could be redefined in a suitable way, or simply be lifted to $\sqcap(\cup t \otimes \cup t')$, with the expected result. The conjunction of conditions could be lifted similarly. Here are some simple facts about lifted DRSs and conditions.

Theorem 46 1. $DRS_{DM}^M(t) \wedge DRS_{DM}^{M'}(t) \Rightarrow M = M'$.

2. $DRS_{DM}^M(t) \wedge DRS_{DM}^{M'}(t') \Rightarrow DRS_{DM}^{M \cup M'}(\sqcap(\cup t \otimes \cup t'))$.

3. $CND_{DM}(t) \wedge CND_{DM}(t') \Rightarrow CND_{DM}(\sqcap(\cup t \cap \cup t'))$.

Yet, a sweeping use of abstractions seems unnecessary to me, and I prefer to make only minimal use of cups and caps, along the lines of the earlier DMG fragment. I hope the resulting variation in the representation of DRSs, sometimes using \sqcap , sometimes not, will not confuse too much here. Perhaps parsimony should be sacrificed for generality here.

4.3.1 Consistency

Let me show how to adapt the fixed point construction for Kamp Structures to the new axiom. We need to make sure that $(d \gg d') \bullet f$ cannot be equal to the application of any other logical constant to its arguments, or of the same logical constant to different arguments. We thus set

$$d \gg =_{df} \|\lambda\{z_1, z_2\} \lambda \langle X_i \mapsto x_i \rangle_{x_i \in Var}. \langle \langle r \gg; (z_1, z_1, z_2, z_2); (x_i, X_i)_{x_i \in Var} \rangle \rangle\|^g$$

The new axiom may be reformulated as:

$$(Ax_{\gg}) \quad PC_{\gg}(\mathcal{P}, f, h, M, N) \Rightarrow (d \gg \bullet f \bullet h \in \mathcal{P} \text{ and } (d \gg \bullet f \bullet f \in \mathcal{T} \Leftrightarrow TC_{\gg}(\mathcal{T}, f, h, M, N))).$$

with the proposition condition $DRS_{DM}^M(f(z_1)) \wedge DRS_{DM}^N(f(z_2)) \wedge dm(h) \supseteq DM$, and the truth condition $\forall k \ dm(k) \supseteq M \wedge (f(z_1) \bullet h \bullet k) \in \mathcal{T} \Rightarrow \exists k' \ dm(k') \supseteq N \wedge (f(z_2) \bullet h_k^M \bullet k') \in \mathcal{T}$.

We keep the definition of θ_K from 3.3.3, letting $d_{l,f,h} =_{df} d_l \bullet f \bullet h$ for $l = \Delta, \gg$ and $d_{l,f,h} =_{df} d_l \bullet f$ otherwise.

Lemma 47 Any fixed point for θ_K satisfies Ax_{\gg} .

PROOF: Again only case (iii) of the original proof is of interest. (iii) Let $PC_{\gg}(\mathcal{P}, f, h, M, N)$ and $d \gg \bullet f \bullet h \in \mathcal{T}$. Then there are l, f', h', M', N' with $d \gg \bullet f \bullet h = d_{l,f',h'}$, such that $PC_l(\mathcal{P}, f', h', M', N')$ and $TC_l(\mathcal{T}, f', h', M', N')$. But then $l = \gg$, $f(z_1) = f'(z_1)$, $f(z_2) = f'(z_2)$, and $h = h'$. Hence $PC_{\gg}(\mathcal{P}, f, h, M', N')$, which implies by theorem 45.1 that $M = M'$, and $N = N'$. Therefore $TC_{\gg}(\mathcal{T}, f, h, M, N)$. \square

The other logical constants behave as before. So we conclude:

Theorem 48 Every non trivial model can be extended to a Kamp Structure for Λ_{DRT} .

4.3.2 Some Λ -DRT Translations

I present the same Categorical Grammar fragment as for DMG here, where I try to give translations that are very close to DMG yet result in appropriate DRSs. We use the same categories, and slash elimination rules. I delete the tautological assertion 'true' from any conjunction, to make things more readable. To save brackets I assume the scope of operators to be narrowest for \cup , and increasingly wider for application (bracketing to the left), \cap , universe-abstraction, \otimes , and unary λ -abstraction.

$$\begin{aligned} (/E) \quad & \text{if } ?_1 \Rightarrow A/B : t_1 \text{ and } ?_2 \Rightarrow B : t_2 \text{ then } ?_1 ?_2 \Rightarrow A : t_1 [t_2] \\ (\backslash E) \quad & \text{if } ?_1 \Rightarrow B : t_1 \text{ and } ?_2 \Rightarrow B \backslash A : t_2 \text{ then } ?_1 ?_2 \Rightarrow A : t_2 [t_1] \end{aligned}$$

$$\begin{aligned}
\text{'}\ddot{x}\text{[every]}' &\Longrightarrow (s/(n\setminus s))/cn : \lambda\alpha\lambda\beta. \{\}\cup(\{[x]true \otimes \cup\alpha x\} \gg \lceil \cup\beta x \rceil) \\
\text{'man}' &\Longrightarrow cn : \lambda x. \{ \}man(x) \\
\text{'with}' &\Longrightarrow (cn \setminus cn)/(s/(n\setminus s)) : \lambda\alpha\lambda\beta\lambda\delta. (\cup\beta \delta) \otimes \cup\alpha [\lambda y. \{ \}with(\delta, y)] \\
\text{'}\ddot{y}\text{[a]}' &\Longrightarrow (s/(n\setminus s))/cn : \lambda\alpha\lambda\beta. \{ \}ytrue \otimes \cup\alpha y \otimes \cup\beta y \\
\text{'donkey}' &\Longrightarrow cn : \lambda x. \{ \}donkey(x) \\
\text{'beats}' &\Longrightarrow (n\setminus s)/(s/(n\setminus s)) : \lambda\alpha\lambda\delta. \cup\alpha [\lambda y. \{ \}beat(\delta, y)] \\
\text{'}\ddot{y}\text{[it]}' &\Longrightarrow (s/(n\setminus s)) : \lambda\alpha. \cup\alpha \ddot{y}
\end{aligned}$$

$$\text{'}\ddot{y}\text{[a] donkey}' \Longrightarrow s/(n\setminus s) :$$

$$\begin{aligned}
&\lambda\beta. \{ \}ytrue \otimes \cup[\lambda x. \{ \}donkey(x)] \ddot{y} \otimes \cup\beta \ddot{y} \\
&\lambda\beta. \{ \}ytrue \otimes \lambda x. \{ \}donkey(x) \ddot{y} \otimes \cup\beta \ddot{y} \\
&\lambda\beta. \{ \}ytrue \otimes \{ \}donkey(\ddot{y}) \otimes \cup\beta \ddot{y} \\
&\lambda\beta. \{ \}ytrue \cap donkey(\ddot{y}) \otimes \cup\beta \ddot{y} \\
&\lambda\beta. \{ \}ydonkey(\ddot{y}) \otimes \cup\beta \ddot{y}
\end{aligned}$$

$$\text{'with }\ddot{y}\text{[a] donkey}' \Longrightarrow cn \setminus cn :$$

$$\begin{aligned}
&\lambda\beta\lambda\delta. (\cup\beta \delta) \otimes \cup[\lambda\beta. \{ \}donkey(\ddot{y}) \otimes \cup\beta \ddot{y}] [\lambda y. \{ \}with(\delta, y)] \\
&\lambda\beta\lambda\delta. (\cup\beta \delta) \otimes (\lambda\beta. \{ \}donkey(\ddot{y}) \otimes \cup\beta \ddot{y}) [\lambda y. \{ \}with(\delta, y)] \\
&\lambda\beta\lambda\delta. (\cup\beta \delta) \otimes (\{ \}donkey(\ddot{y}) \otimes \cup[\lambda y. \{ \}with(\delta, y)] \ddot{y}) \\
&\lambda\beta\lambda\delta. (\cup\beta \delta) \otimes (\{ \}donkey(\ddot{y}) \otimes \{ \}with(\delta, \ddot{y})) \\
&\lambda\beta\lambda\delta. (\cup\beta \delta) \otimes (\{ \}donkey(\ddot{y}) \cap with(\delta, \ddot{y}))
\end{aligned}$$

$$\text{'man with }\ddot{y}\text{[a] donkey}' \Longrightarrow cn :$$

$$\begin{aligned}
&\lambda\delta. \cup[\lambda x. \{ \}man(x)] \delta \otimes (\{ \}donkey(\ddot{y}) \cap with(\delta, \ddot{y})) \\
&\lambda\delta. \{ \}man(\delta) \otimes (\{ \}donkey(\ddot{y}) \cap with(\delta, \ddot{y})) \\
&\lambda\delta. \{ \}man(\delta) \cap donkey(\ddot{y}) \cap with(\delta, \ddot{y})
\end{aligned}$$

$$\text{'}\ddot{x}\text{[every] man with }\ddot{y}\text{[a] donkey}' \Longrightarrow s/(n\setminus s) :$$

$$\begin{aligned}
&\lambda\beta. \{ \} \cup(\{[x]true \otimes \cup[\lambda\delta. \{ \}man(\delta) \cap donkey(\ddot{y}) \cap with(\delta, \ddot{y})] \ddot{x}\} \gg \lceil \cup\beta \ddot{x} \rceil) \\
&\lambda\beta. \{ \} \cup(\{[x]true \otimes \{ \}man(x) \cap donkey(\ddot{y}) \cap with(x, \ddot{y})\} \gg \lceil \cup\beta \ddot{x} \rceil) \\
&\lambda\beta. \{ \} \cup(\{[x, \ddot{y}]man(x) \cap donkey(\ddot{y}) \cap with(x, \ddot{y})\} \gg \lceil \cup\beta \ddot{x} \rceil)
\end{aligned}$$

$$\begin{aligned}
&\text{'beats }\ddot{y}\text{[it]}' \Longrightarrow n \setminus s : \\
&\lambda\delta. \cup[\lambda\alpha. \cup\alpha \ddot{y}] [\lambda y. \{ \}beat(\delta, y)] \\
&\lambda\delta. \cup[\lambda y. \{ \}beat(\delta, y)] \ddot{y} \\
&\lambda\delta. \{ \}beat(\delta, \ddot{y})
\end{aligned}$$

$$\begin{aligned}
&\text{'}\ddot{x}\text{[every] man with }\ddot{y}\text{[a] donkey beats }\ddot{y}\text{[it]}' \Longrightarrow s : \\
&\{ \} \cup(\{[x, \ddot{y}]man(x) \cap donkey(\ddot{y}) \cap with(x, \ddot{y})\} \gg \lceil \cup[\lambda\delta. \{ \}beat(\delta, \ddot{y})] \ddot{x} \rceil) \\
&\{ \} \cup(\{[x, \ddot{y}]man(x) \cap donkey(\ddot{y}) \cap with(x, \ddot{y})\} \gg \lceil \{ \}beat(x, \ddot{y}) \rceil)
\end{aligned}$$

4.4 A Note on Appropriateness and Restrictions

The Situation Theory of [Barwise & Cooper 1991] contains a rather sophisticated theory of *appropriateness* of assignments for abstracts. Any object carries appropriateness conditions with it, which have to be met by its arguments if application is to be defined. One particular way of introducing such appropriateness constraints on abstracts is by using a **RESTRICTION OPERATOR** \wr . It links truth to identity in a presuppositional way:

$$(*) \quad T(t') \rightarrow (t \wr t') = t \wedge \neg T(t') \rightarrow (t \wr t') = \#.$$

With such an axiom one can impose additional presuppositions $t'_{x_1 \dots x_n}$ into an abstract, to form $\lambda(X_i \mapsto x_i)_{i=1}^n. (t_{x_1 \dots x_n} \wr t'_{x_1 \dots x_n})$. This applies to $(x_i, t_i)_{i=1}^n$ to result in $[t_i/X_i]_{i=1}^n t$ only if $[t_i/X_i]_{i=1}^n t'$ is true.

Unfortunately the axiom (*) is inconsistent with Proposition Structures: take some term t which does not denote \perp , then the above version implies that $\lambda x. (t \doteq (t \wr x))$ internally defines truth, which is inconsistent, as we saw earlier. One might try to use a weaker version instead:

$$(**) \quad T(t') \rightarrow (t \wr t') = t \wedge F(t') \rightarrow (t \wr t') = \#.$$

I do not know whether (**) is consistent with respect to Proposition Structures. What is demonstrably consistent,⁸ is to have just restricted propositions,

⁸To prove this, one just has to account for the fact that the proposition condition PC_1 is

following [Plotkin 1990], such that:

$$(Ax_1) \quad P(t) \wedge T(t') \rightarrow P(t \uparrow t') \wedge T(t \uparrow t') \leftrightarrow T(t).$$

This axiom is certainly a less powerful tool for introducing presuppositions than $(*)$, which might motivate us to change our relatively naive treatment of undefinedness.⁹ As things stand, I have no notion of appropriateness for the SAC beyond the rather weak principles governing $\#$, and those obtained from the conditions for the formation of propositions.

4.5 Conclusion

I have shown that a minor change from the strictly ordered λ -calculus to a theory of simultaneous abstraction, which would seem well motivated by word order considerations alone, is enough to recreate three of the currently most influential semantic theories. I have established some fundamental results concerning variants of this form of abstraction, including equivalences between the basic SAC, an Aczel–Lunnon style system, and a Combinatory Logic version. Completeness and confluence theorems were obtained.

I have used Frege Structure style axioms to obtain Dynamic Montague Grammar, Discourse Representation Theory, and Situation Theory. DMG was obtained via Chierchia's Dynamic Property Theory. ST was shown to be easily obtained by adding a simple theory of events and strong negation to the basic Property Theory. A simple minded Property Theoretic DRT was obtained as well, by adding an axiom for conditionals. The disadvantages of this system motivated a slight change in the semantics of the SAC, which allowed for a more satisfactory formulation of DRT. The resulting system can be considered as a semantically kosher, highly intensional relative of λ -DRT, with only one underlying form of

defined with respect to \mathcal{T} , as well as \mathcal{P} , in the fixed point construction. Everything else works as before.

⁹I'm indebted to Peter Aczel for pointing out that there is no easy fix for $(*)$ by restricting the equality axiom of Proposition Structures, to produce propositions only if both sides of the equation are defined. A deeper re-think of the treatment of undefinedness is required if we want $(*)$ to hold.

abstraction, appearing in three different guises.

I believe the SAC together with Property Theoretic axioms provides a powerful framework for linguistic semantics. It provides an elegant machinery of high generality and simplicity, and appears to make good philosophical as well as methodological sense. I feel that I have only scratched the surface of the logical and linguistic issues involved, yet I hope this may suffice to suggest that such a framework is worth having, and that the effort to obtain it is relatively modest.

Bibliography

- [Aczel 1980] P. Aczel: Frege Structures and the Notions of Proposition, Truth and Set. In Barwise et al. (ed.) *The Kleene Symposium*, North Holland Studies in Logic 1980.
- [Aczel 1988] P. Aczel: Non Well-Founded Sets. CSLI Lecture Notes No. 14, Stanford 1988.
- [Aczel 1989] P. Aczel: Algebraic Semantics for Intensional Logics *I*. In G. Chierchia et al. (eds.) *Properties, Types, and Meaning, Vol. 1*. Kluwer, Dordrecht 1989.
- [Aczel & Lunnon 1991] P. Aczel and R. Lunnon: Universes and Parameters. In Barwise et al. (ed.) *Situation Theory and its Applications, Vol. 2*. CSLI, Stanford 1991.
- [Asher 1993] N. Asher: *Reference to Abstract Objects in Discourse*. Kluwer, Dordrecht 1993.
- [Barendregt 1984] H.P. Barendregt: *The Lambda Calculus. Its Syntax and Semantics*. Revised edition, North-Holland, Amsterdam 1984.
- [Barwise 1989] J. Barwise: Notes on Branch Points in Situation Theory. In J. Barwise: *it The Situation in Logic*. CSLI lecture notes No. 17, Stanford 1989.
- [Barwise & Cooper 1991] J. Barwise and R. Cooper: Simple Situation Theory and its Graphical Representation. In Jerry Seligman (ed.) *Partial and Dynamic Semantics 3*, DYANA deliverable R2.1.C, Centre for Cognitive Science, Edinburgh, 1991.

- [Barwise & Perry 1983] J. Barwise and J. Perry: *Situations and Attitudes*. MIT Press, Cambridge Mass., 1983.
- [Bealer 1982] G. Bealer: *Quality and Concept*. Clarendon Press, Oxford 1982.
- [Bealer 1989] G. Bealer: On the Identification of Properties and Propositional Functions. In *Linguistics & Philosophy 12, No. 1*, Kluwer, 1989.
- [Chierchia 1992] G. Chierchia: Anaphora and Dynamic Binding. In *Linguistics & Philosophy 15/2*, 1992.
- [Chierchia 1994] G. Chierchia: Intensionality and Context Change. Towards a Dynamic Theory of Propositions and Properties. In *Journal of Logic, Language and Information 3*, 1994.
- [Cooper 1993] R. Cooper: *Towards a General Semantic Framework*. In R. Cooper (ed.): *Integrating Semantic Theories*. DYANA-II deliverable R2.1.A, Centre for Cognitive Science, Edinburgh 1993.
- [Cooper 1995] R. Cooper: The Attitudes in Discourse Representation Theory and Situation Semantics. To appear in [Seligman et al. 1995].
- [Cooper & Groenendijk 1994] R. Cooper & J. Groenendijk (eds.) *Integrating Semantic Theories II*, DYANA-2 report R2.1.B, Amsterdam 1994.
- [Cooper & Poesio 1994] R. Cooper and M. Poesio: *Situation Semantics*. Chapter 3 in *The FraCaS Consortium: Describing the Approaches*. FraCaS deliverable D8, Centre for Cognitive Science, Edinburgh 1994.
- [Cooper et al. 1996] R. Cooper et al.: *Building the Framework*. FraCaS deliverable D15, University of Edinburgh 1996.
- [Dekker 1993] P. Dekker: *Transsentential Meditations. Ups and downs in dynamic semantics*. ILLC dissertation, Amsterdam 1993.
- [Etchemendy 1990] J. Etchemendy: *The Concept of Logical Consequence*. Harvard University Press, Cambridge Mass., 1990.

- [Groenendijk & Stokhof 1990] J. Groenendijk and M. Stokhof: *Dynamic Montague Grammar*. Technical Report, ITLI Amsterdam.
- [Groenendijk & Stokhof 1991] J. Groenendijk and M. Stokhof: Dynamic Predicate Logic. In *Linguistics & Philosophy* 14, 1991.
- [Hindley & Seldin 1986] R. Hindley and J. P. Seldin: *Introduction to Combinators and λ -Calculus*. CUP, Cambridge 1986.
- [Kamp 1981] H. Kamp: A Theory of Truth and Semantic Representation. In J. Groenendijk et al. (eds.) *Formal Methods in the Study of Language*. Mathematical Centre, Amsterdam 1981.
- [Kamp 1990] H. Kamp: Prolegomena to a Structural Theory of Belief and Other Attitudes. In C. A. Anderson and J. Owens: *Propositional Attitudes*. CSLI Lecture Notes No. 20, Stanford 1990.
- [Kamp & Reyle 1993] H. Kamp and U. Reyle: *From Discourse to Logic*. Kluwer Academic Publishers, Dordrecht 1993.
- [Kohlhase et al. 1996] M. Kohlhase, S. Kuschert, and M. Pinkal: *A Type-theoretic Semantics for λ -DRT*. Proceedings of the 10th Amsterdam Colloquium, forthcoming.
- [Kuschert 1995] S. Kuschert: *Eine Erweiterung des λ -Kalkuels um Diskursrepräsentationsstrukturen*. Diplomarbeit, Universitaet des Saarlandes, Fachbereich Informatik, Saarbruecken 1995.
- [Millies & Pinkal 1992] S. Millies und M. Pinkal: *Eine deklarative Version der DRT in typisierter Merkmalslogik*. GUK Arbeitspapier 22, Universitaet des Saarlandes, Saarbruecken 1992.
- [Montague 1974] R. Montague: The Proper Treatment of Quantification in Ordinary English. In R. H. Thomason (ed.) *Formal Philosophy: Selected Papers of Richard Montague*. Yale Univ. Press, New Haven, 1974.
- [Muskens 1989] R. Muskens: *Meaning and Partiality*. Ph.D. Thesis, Amsterdam 1989.

- [Muskens 1992] R. Muskens: *Anaphora and the Logic of Change*. ITK research report No. 34, Tilburg University 1992.
- [Muskens 1994] R. Muskens: *Categorial Grammar and Discourse Representation Theory*. In Proceedings of COLING-94.
- [Plotkin 1990] G. Plotkin: An Illative Theory of Relations. In Cooper et al. (eds): *Situation Theory and its Applications Vol. 1*, CSLI, Stanford 1990.
- [Seligman et al. 1995] J. Seligman et al. (eds.) *Language, Logic, and Computation. The Moraga 1994 Proceedings*, CSLI Publications, Stanford (to appear by end of 1995).
- [Soames 1985] S. Soames: Lost Innocence. In *Linguistics and Philosophy* 8, 1985.
- [Thomason 1980] R. Thomason: A Model Theory for Propositional Attitudes. In *Linguistics & Philosophy* 4, 1980.
- [Turner 1990] R. Turner: *Truth and Modality for Knowledge Representation*. Pitman, London 1990.
- [Zeevat 1989] H. Zeevat: A Compositional Approach to Discourse Representation Theory. In *Linguistics & Philosophy* 12, No. 1, Kluwer, 1989.
- [Zeevat 1990] H. Zeevat: *Static Semantics*. In J. van Benthem (ed.): *Partial and Dynamic Semantics I*, DYANA deliverable, Centre for Cognitive Science, Edinburgh 1991.
- [Zeevat 1991] H. Zeevat: *Aspects of Discourse Semantics and Unification Grammar*. Ph.D. Thesis, Amsterdam 1991.

Index

\wedge , 43
 \bigwedge , 71, 74
 \cap , 44
 \sqcap , 62
 \cup , 44
 \sqcup , 62
 $=$, 43
 \approx , 44
 $=_\alpha$, 27
 $=_\beta$, 26
 $=_{CL}$, 32
 \supset , 44
 \gg , 53, 56, 70, 76
 \ggg , 76
 $\|\cdot\|$, 16, 22, 71, 73
 $\langle\langle \cdot \rangle\rangle$, 39
 \leq , 60
 $\models t = t'$, 18
 $\models_{CL} T = T'$, 32
 $\models_{CLM} t = t'$, 37
 $e \models \tau$, 62
 \neg , 43
 \neg , 71
 \perp , 44
 \sim , 62
 \oplus , 57
 \otimes , 54, 70, 72
 \triangleright , 26
 \triangleright_{An} , 27
 \triangleright_{mcd} , 28
 \wr , 80
 \implies , 51
 \rightarrow , 70
 $\uparrow\downarrow$, 50
 \bullet , 31, 34, 73
 $\#$, 14, 22, 31
 \perp , 15
 α, β, \dots , 48
 \vdash_β , 26
 c, c', \dots , 14, 22, 31
 C_{AL} , 31
 $(\cdot)_{CL}$, 35
 Δ , 44
 d_l , 46
 DM , 48
 $dm(f)$, 15
 D_{τ_i} , 70
 e, e', \dots , 60
 \exists , 43
 e , 35
 e , 70

INDEX

F , 44
 FOL_{SAC} , 43
 $FP(T)$, 31
 $FV(t)$, 17
 g_f^M , 15
 \mathfrak{S} , 16, 31
 \mathbf{k}_x , 31
 \mathbf{K}_x , 31
 λ , 24
 Λ , 14
 Λ' , 23
 Λ_{AL} , 22
 Λ_{DRT} , 72
 Λ_{TDRT} , 70
 $(\cdot)_\Lambda$, 35
 λ^* , 34
 $\lambda_X.t$, 22
 $\lambda M.t$, 14, 72
 $\lambda X.t$, 24
 $\lambda x.t$, 25
 MV , 48
 P , 44
 \mathcal{P} , 46
 $PC_l(\mathcal{P}, f, h)$, 47
 $PF\{x_1..x_n\}$, 55
 Φ , 15
 $\phi_{v_1..v_n}$, 45
 Π , 44
 π_x , 40
 $Pow(Var)$, 73
 Ψ , 15, 34
 Ψ_1 , 73
 Ψ_2 , 73
 $\rightleftharpoons_\Phi^\Psi$, 15
 PTY^M , 56
 PTY^n , 44
 $\langle Q_x \rangle^s$, 58
 $\langle Q_x \rangle^w$, 58
 R_i^n , 43
 S , 31
 s , 31
 σ, σ', \dots , 62
 $\bar{\sigma}$, 64
 Σ , 44
 Sit , 60
 Soa , 62
 t, t', \dots , 14, 22
 T, T', \dots , 31
 $t(x_i, t_i)_{i \in I}$, 14
 $T(x_i, T_i)_{i \in I}$, 31
 $t[x_i, t_i]_{i \in I}$, 20, 22
 $t(t)$, 24
 $T(T')$, 31
 T , 44
 \mathcal{T} , 46
 $[T]$, 33
 $[t]$, 37
 $[t]$, 50
 \mathcal{T}_{CL} , 33
 $TC_l(\mathcal{T}, f, h)$, 47
 $\mathcal{T}_\Lambda(Ax)$, 37

τ, τ', \dots , 62

τ , 70

t , 70

θ_F , 47

θ_K , 58

$\theta(x)$, 70

Toe , 62

\sqcap_t, \sqcup_t , 49

t^\sharp , 24

$[x_i/X_i]_{i \in I}^\dagger t$, 23

$[s_j/y_j]_{j \in J} t$, 18

V , 34

v, v_1, \dots , 43

$(Var \rightarrow D) \rightarrow D$, 15

$((Var \rightarrow D_\perp) \rightarrow D_\perp)_{rep}$, 34

X, Y, \dots , 22, 31

x, y, \dots , 14

$\mathbf{x}, \mathbf{y}, \dots$, 23

\hat{x}, \hat{y}, \dots , 48

Ξ , 56

$\{x_i\}_{i \in I} t$, 14

$\langle x_i \mapsto \xi_i \rangle_{i \in I}$, 15

z , 25, 31

z_1, z_2 , 44