Interpretation of Discourse and Proof Theory: Questions and Directions

David Israel

May 25, 1999

Abstract

A major theme of Johan's **Exploring Logical Dynamics** is the move from using the tools of logic to analyze the products of cognitive processes to using those tools, including perhaps some new ones, to analyze the processes themselves. One kind of cognitive product is a grammatically well-formed sentence of a natural language apt for statement-making. Another, much more complex product, is a multi-person discourse or conversation, thought of as a complex structure of some sort, some of whose components may be statement-making sentences. Of course, a discourse isn't simply a set or even a sequence of such things; it has a structure that results both from the content of the components and from a variety of other structural features. This particular pair of products suggests another: a statement of mathematics and a proof (or derivation) of that statement. The latter, too, is a complex structure; it isn't simply a set or even a sequence of statements. Is there more to this suggestion or is it merely the result of more than free association? This is a question worth thinking—even thinking out loud—about. And, of course, since it is a question worth thinking about, Johan van Benthem has thought about it. So herein is my contribution to what I hope will be a continuing dialogue.

Contents

1 Introduction **2**

2 Proofs and Programs **3**

3 Categorial Type Logics **4**

4 Proof Search and Computation **5**

5 Discourse and Discourse Understanding **6**

6 Proofs and Beyond **7**

7 Discourse and Games **9**

8 Interaction and Games **10**

9 Summing Up **11**

10 CONGRATULATIONS **12**

1 Introduction

A distinctive hallmark of the thinking and writing of Johan van Benthem is the many ways it evinces a mind constantly on the lookout for interesting and illuminating connections amongst seemingly disparate intellectual disciplines and approaches. Simply put, there is no one quite like Johan when it comes to breadth and openness of vision.[1] In this little gift (?) to Johan on his 50th, I am, humbly and fearfully, going to stumble along one or another bridge, perhaps some of them are rope bridges, that Johan has already erected across at least one seeming intellectual chasm.

A major theme of Johan's **Exploring Logical Dynamics** (**ELD**) is the move from using the tools of logic to analyze the products of cognitive processes to using those tools, and perhaps some new ones, to analyze the processes themselves. One kind of cognitive product is a grammatically well-formed sentence of a natural language apt for statement-making. Another, much more complex product, is a multi-person discourse or conversation, thought of as a complex structure of some sort, some of whose components may be statement-making sentences. Of course, a discourse isn't simply a set or even as sequence of such things; it has a structure that results both from the content of the components and from a variety of structural features. This particular pair of products suggests another: a statement of mathematics and a proof (or derivation) of that statement. The latter, too, is a complex structure; it isn't simply a set or even a sequence. Is there more to this suggestion or is it the result of more than free association? This is a question worth thinking—thinking out loud—about. One minor subtheme of **ELD** involves precisely thinking out loud about this question.

Before doing a little of this exploring ourselves, let us look at the relation between the question of the relation between proofs and discourses and another theme of Johan's recent work: that "natural languages are programming languages for cognition (minds)". The idea here is, I suppose, (i) that the actions of human beings are directly responsive to their beliefs, desires, plans, etc. and that the only systematic and fairly reliable means for one human to have some indirect control over another's actions is by effecting the beliefs and desires of that other and (ii) the principal systematic and fairly reliable means for getting such indirect control is by way of linguistic communication with that other person. So, in some loose but plausibly illuminating sense, we use language as a way of "programming" other people's minds, thereby indirectly controlling their actions—or as a computer scientist might say, their (observable) behavior.

Of course, one could have quite another picture in mind, a purely "individualistic" one: that cognition is computation. In just a little more detail, this is the idea that the creation and manipulation within a human mind of its beliefs, desires, etc. and the relations between these cognitive products and actions is itself a computational process. This has certainly been the paradigm of much cognitive science over the last quarter century or so. Given that, it's best to

[1]And then, too, there is the somewhat irritating productivity and fertility of that mind. *Giving us so much to read and think about; so little time.*

keep both possibilities in mind. And, also, to keep in mind that while the first picture, of our programming each other's minds, and indirectly their behaviors, by the use of language, does not depend on accepting the computationalist cognitive science paradigm, it certainly resonates well with it.

2 Proofs and Programs

So, to return: what do proofs have to do with programs and programming languages? Here we have at least two (families of) answers: (i) proofs are (things from which) programs (can be extracted)[2] and (ii) program execution can be thought of as the search for or the construction of proofs. The slogan associated with the first answer is *Propositions are Types; Proofs are Programs*; with the latter, *Logic is a Programming Language*. The magic names associated with the first answer are Curry, Howard, DeBruijn; those associated with the second, Robinson, Kowalski and Colmerauer.

When introducing proof theoretic ideas into discussions of natural language analysis, in **ELD** and elsewhere, Johan seems mostly to have in mind the first, Curry-Howard-DeBruijn answer. Very crudely, the key technical idea behind this answer are the correspondences between (i) the types of a typed λ-calculus and the formulae of a logic, (ii) typed terms of the calculus and derivations in a natural deduction (or sequent calculus) for the logic, and (iii) reduction or evaluation of typed terms and normalization (or cut-elimination) of derivations. One reason this connection seems natural is that this strand of proof theory, sometimes called *structural proof theory*, focuses on aspects of the structure of proofs. Indeed, a part of what might be called the Gentzen-Prawitz thesis is that natural deduction systems, in particular, are adequate representations of the structure of naturally occurring proofs.

Without going into much detail, a word more should be said about how the terms that enter into the Curry-Howard-DeBruijn correspondence to be understood. Here we might as well introduce yet another hyphenated triad: Brouwer-Heyting-Kolmogorov, or more precisely, and shortly, the BHK interpretation of Intuitionistic Logic. According to BHK the meaning of a complex proposition (or of the logical operators that form such propositions) is to be given in terms of constructions that prove or realize the proposition. For the critical case of an implication, $A \rightarrow B$, a construction is a constructive method (function) that transforms an arbitrary construction for A into a construction for B. And, of course, the typed λ-calculi are systems for representing such functions. This interpretation supports the construal of the terms quite directly as linear (1-dimensional) representations for proof structures, representing the constructions that those structures represent less explicitly and in tree form.

The foregoing is a crude précis of the intuitions behind some of the most important work on the semantics of functional programming languages of the last 20 years—and thereby, behind the most important connection between proof theory and computer science. For given that one can think of typed λ-calculi as (the kernels of) functional programming languages, and of the terms

[2]Remember, I said that there were at least two (families of) answers.

themselves as programs, one can think of the reduction of such terms to normal forms as the evaluation or execution of those programs. So any suggestion that the semantics of natural language discourse be looked at through anything like this proof-theoretic prism carries with it (though not ineluctably) a suggestion that it be looked at through a particular computational prism.

3 Categorial Type Logics

What does this picture of the relation between programs and proofs suggest with regard to the interpretation of natural language? It has, in fact, suggested a good deal of great interest to researchers looking at the analysis of individual sentences. I have in mind the work of Moortgat, Morrill and others, including van Benthem himself, working in the categorial grammar tradition of Lambek (whose name could very well, if a little misleadingly, be added to those of Curry, Howard and DeBruijn). This work has focused on the first two aspects of the correspondence, and not on the third, distinctively computational or process-oriented aspect: the relation between normalization/cut-elimination procedures and the execution of programs.

The central aim of the work in question is linguistic analysis by way of proof—"parsing as deduction" or grammar as logic—for example, in the basic case, proof of the well-formedness, more generally of the category-membership, of strings. The aim is not especially to analyze these proofs or the relations among them. The fact that the deductive calculi enjoy the cut-elimination property is important largely because it underwrites sensible proof-search; though detailed studies of the process of search is also not a focus of this research. This all makes good sense so long as one keeps in mind that the work is not primarily aimed at modeling a (cognitive) process at all, but rather at providing a clean mathematical regimen for the analysis of various properties of cognitive products.

Indeed, even with respect to the correspondence between terms and proofs, there has not been much attention paid to what I take to be a distinctive feature of a properly computationalist approach to proof-theory: the fact that there can be importantly different proofs of the same theorem; alternatively, that there can be importantly different programs/algorithms computing the same function.[3] There is, of course, an apparent crucial exception here; namely the the way in which different interpretations (*readings*) of a (non-lexically) ambiguous string are displayed via different (cut-free) proofs that the string is, e.g., a sentence.

In a sense, though, this apparent exception makes the case especially strong: In the case of a truly ambiguous string, there will be at least two different derivations, involving different λ-terms being assigned at one or another intermediate point and at the end sequent.

[3]Continuing at the level of sophistication and detail I have earlier set for myself: one of the central complaints about classical logic in this regard is that, in its standard presentations, all proofs of a given theorem are created equal. Here, if I could, I would execute my best approximation to a Gallic sneer.

How are we to understand these terms? Roughly speaking, just as we are to understood typed λ-terms in Montague grammar, that is, as representing the denotations—the semantic values, *classically construed*—of the linguistic expressions. In this regard, it should be remembered that the terms of Montague's **IL** are expressions in an ω-order logic, not in a typed computational calculus.[4]

Moortgat, in particular, is very clear on the differences between the syntax-semantic interface within Montague-style grammars, on the one hand, and the categorial type systems, on the other.[5] But on the purely semantic side of the interface, the picture is very much like Montague's. The denotations of the terms are thought of as elements in a type-theoretic universe of discourse; that is, as things that components of the sentence refer to. Even when they are functions, they certainly need not be thought of as computable; and they certainly are not thought of as representing the proof, e.g., of well-formedness of the sentence in the end sequent. They are not representations of the structure of the proof of well-formedness; they are, rather, representations of part of what is to be proved; to wit, that a given string, say, has a certain meaning, according to the grammar of the language. Thus, in the case of a structurally ambiguous string, each of the various proofs is to the effect that the string has such and such a semantic representation or denotation.

While (much) more needs to be said about the proofs-as-programs paradigm and its relation to work in "proof-theoretically oriented" theoretical linguistics, it is time to sum up: Despite the fact that the logics involved in categorial type systems have a relational semantics and despite some of the elements of what might be called computational proof theory, the main focus of the work within this framework is aimed, and well-aimed, at illuminating analysis of various properties of linguistic products, e.g., sentences, where these—both properties and products—are construed in ways familiar from classical theoretical, as opposed to computational, linguistics.

4 Proof Search and Computation

We mentioned that there were two main families of approaches to relating programs and programming languages to proofs. We have scouted, though briefly, one such approach. What of the other? This is the logic programming view that that holds, again roughly, that programs are specifications in some logic or other and that the execution of a program is the search for a proof. In the proofs as programs paradigm, a proof is a program, or something from which a program can be extracted, and program execution is essentially evaluation of the program. In the logic programming paradigm, the program is given explicitly via a set of axioms, the program clauses, and computation consists of a controlled process of constructing—searching for—a proof from those axioms.

[4]The only real connection between the Church of the 20's and 30's who created and with his great graduate students, Kleene and Rosser, analyzed various λ-calculi and the Church of the 40's and on who studied high-order (intensional) logics, is the relation of *being stages in the same life as*.

[5]See [Moortgat97]; see also [Morrill94].

But a proof of what? Logic programming languages are refutation-oriented; what is involved is proving that the program clauses and the negation of the *goal* is inconsistent. This, of course, is a way of proving that program clauses entail the goal. In Horn clause logic programming, for example, the negation of a goal has the force of an existentially quantified conjunction; so proving inconsistency, or dually proving entailment, is proving that there is at least one thing that meets a certain condition. Rather than establishing the relation of logical consequence, from a programming perspective, however, what is important is establishing a witness, an *answer substitution*—or all such—verifying the existential in the intended model.[6] And, of course, if the negation of the goal isn't entailed, and the refutation fails, the programmer again wants to be able to understand the process as the construction of a counterexample.

What has this picture of the relation between proofs and programs to offer in the way of illuminating process of discourse understanding? To answer this, we had best say at least a little about discourse.

5 Discourse and Discourse Understanding

So far we have been operating at a very abstract level, merely sketching various conceptions of the relations between proofs in a logic and programs and program execution or computation. Alas we're going to stay at that level, but now we'll be sketching at least one conception of what is involved in discourse, and so in understanding discourse.

The conception we have in mind, or its main outline, is due to Grice and Stalnaker. As noted above, the typical goal of discourse is to affect the minds, in particular the beliefs, desires and intentions of the participants and (typically) to do this more or less transparently or openly.[7] This attempt involves a tacit mutual understanding of the starting point of the discourse: that is, of a set of propositions that all parties mutually assume are assumed or taken-for-granted, at least for the purposes (and life) of the discourse. These starting points sound a bit as if they function as axioms. And, of course one can take the various speakers' contributions, at least those of the statement-making variety as adding further axioms. But where in all this is model construction?

Among these propositions are those that are about the state of the discourse itself. Using jargon from the Web—itself sometimes thought of as a space in which multiple conversations can occur–these propositions express *metadata* about the discourse. Thus, one subclass of such propositions might be to the effect that certain objects (and events) of certain sorts are already, even before the discourse proper begins, mutually established as accessible objects of reference for particular classes of referential devices. For instance, if all parties mutually take it for granted that each of the parties is acquainted with one particular person named "Johan" or one such person especially relevant to the issues (mutually assumed) to be discussed, then any one of the parties might

[6]Or at any rate in the least Herbrand model.

[7]For much much more on what this might mean, see [Grice58, Grice67]. Stalnaker's contributions begin with [Stalnaker72]. See also [Stalnaker74].

use the proper name "Johan", more or less without further ado, to refer to that particular person.

As the conversation goes on, other objects become more or less accessible to reference—differentially accessible via different referential devices or modes, the degree and mode of accessibility depending, *inter alia*, on the history within the discourse of references to those and other objects, including the history of the use of various referential devices. Imagine one speaker starts off a discourse by saying "I met *a Dutch logician* yesterday. *He* seemed to know a helluva lot about just about everything." and a second speaker answers, "Oh *that* must have Johan vB," the second speaker can be understood or modeled as taking it as mutually assumed that, after the first statement, all parties to the discussion have opened a mental file for an otherwise unspecified and unidentified Dutch logician and that the use of "he" in the second sentence is linked to that file somehow so that the predication of relative omniscience gets added to that of being (now) a male Dutch logician. With all that assumed as given, the second speaker can assume that it will be mutually assumed that the use of the discourse demonstrative "that" will be taken to be a reference to the person so far identified as a very smart male Dutch logician whose name is now being provided to the conversants for future use in the conversation.

This can begin to sound a bit like model construction: especially if one thinks of the mental files that get triggered by the understanding of sentences as represented by individual constants (discourse markers) in some language and then identifies the elements of the model with those constants .... Rather than pursue this theme, though, I intend to take a step back to more general and abstract considerations.[8]

6 Proofs and Beyond

Let's remind ourselves of the original quest: bridges, however shaky (at first), between conceptions of proof and conceptions of discourse. First let's distinguish different kinds of discourse. Obviously if we include monologues, including written monologues, as discourses, then some discourses can indeed be modeled more or less straightforwardly as arguments, if not as proofs; for some discourses just are arguments, in the sense of attempts to systematically marshal evidence for a particular conclusion. And, of course, some arguments really are proofs. So some discourses are proofs. Let's agree to ignore them and let's focus on those discourses that are multi-person conversations.[9]

Notice that the Grice-Stalnaker model is in any case aimed principally at such conversations. And, though I didn't highlight it above, that model often assumes that one of the parties, at least, to the conversation has a goal or goals in mind that motivates his/her initiating the conversation. Indeed, early and continuing research in Computational Linguistics and Artificial Intelligence has often focused on dialogues in which one of the agents is in charge: it is that

[8]Of course others, especially working within the framework of DRT, have pursued this theme of discourse understanding as model construction. See for example [KohKon:mgfdrt99].

[9]Needless to say, Johan has thoughts about the logic of arguments, too. See [JvB 94].

agent's goal and the subgoals it generates that supplies the most important structuring relationships to the conversation. These reflections put us in a position to add another bit to the analogy between discurse understanding and proof search.

Some conversations, then, can be seen as multi-agent activities aimed at a single goal, say of a single of the agents. As such they can be seen as attempts to realize a plan of action by that agent to realize that goal. That plan will, perforce, include the planning of linguistic acts, the agent's contributions to the conversation. The task of the other agents, who are—in this scheme—the witting and willing assistants of the prime mover, is to catch on to what that plan is and what their role in it is and, if all goes well, to perform that role. That task may of course require planning of their own, including the planning of linguistic acts. Happily there are deep connections between proof search and planning. (And vice-versa, of course. After all proof search is just the kind of complex, single-minded goal-oriented activity that motivates work in planning.) Indeed one way to implement planning is precisely via proof search: the search for a plan that gets one from a starting state to a goal state is turned into the search for a proof—and an "effective", executable witness—that there is a sequence or more generally a *program* of actions that if performed in a state that meets the initial conditions will eventuate in a state that meets the goal conditions. So here is further confirmation that if wants to look at dialogue understanding via proof-theoretic models, and hence via computational ones, a proper prism is that of computation as proof search.

Still, let us remind ourselves that in general, all the parties to a conversation might also be pursuing goals in and through that conversation, perhaps conflicting goals, perhaps independent goals, perhaps even complementary ones, and during segments of the conversation those goals might come to dominate or at least take precedence over those of the initial and initiating player.

With my use of that last word, I have let enough of the cat out of the bag. For while it is true that a multi-agent planning model might apply to the more complex kinds of dialogues mentioned above, it is also true that it seems to leave something out. Maybe the understanding of monologues and single-agent initiated and controlled dialogues can be adequately modeled as proof (or argument) search; but most real discourses are not univocally goal-oriented in the relevant sense. Rather many potentially conflicting concerns are being addressed, some only implicitly, and there are many complexly related criteria of success held by the participating agents. Each of the agents has to respond appropriately, given his/her own beliefs, desires and plans to the moves, linguistic and otherwise, of the other agents. In this respect conversations are much more like games—not purely competitive games, of course, but collaborative games, signaling games, etc.

7 Discourse and Games

With talk of games we come to yet another of Johan's most compelling (and recurring) interests.[10] So what do games have to do with proofs? And is there a connection between games and computation of the kind sketched above between proofs and programs? Oddly enough, quite a lot. And, yes there is. But here the story is quite tangled and still aborning and I don't have the space (or the competence) to tell it here. Just a few points and pointers.[11]

There is Lorenzen's game or *dialogue* semantics for Intuitionistic logic [Felscher86] and, even more to the point, what may be thought of as Lorenzen-style game semantics for (various fragments of) Linear Logic. These are due originally to Blass [Blass92]. Lorenzen's account is of proof as successfully meeting challenges. Blass's account(s), especially of Linear Logic(s) are conceptually both more nuanced and more complicated.[12] Extensions and variants of this style of semantic account have been applied with great success to fragments of Linear Logic [AJ94] and, following up on the Curry-Howard-DeBruijn Correspondences, to certain outstanding problems in the semantics of programming languages. The model of computation in these languages is that of sequential and deterministic computation of higher-order functions. See, especially, the work of Hyland and Ong [H095], and Abramsky *et. al.* [AJM94] on **PCF**. (See also [Abramsky97, Hyland97].) But this conception, as noted above, I take to be too singly (univocally) goal-oriented. One sign of this is the particular asymmetry between the two players in the game semantics of Intuitionistic Linear Logic and indeed, of Lorenzen's dialogue semantics for Intuitionism. This asymmetry is not surprising: after all, theorems are meant to be proved—the Player, who is the agent for the theorem, so to speak, is supposed to win. Or think of the BHK semantics for Intuitionism: at a purely intuitive conceptual level, the semantics of the complex types (propositions) is given in terms of constructions taking *inputs* and yielding *an output*. This same asymmetry is also critical in the game semantics of sequential programming languages.

I think a better model than function computation, but again to be found within computer science, is that of *interactive systems*. These are systems whose behavior arises out of interaction with other processes (including users) in their environments and whose purpose is to provide various computational resources and execute various procedures over an unlimited stretch of time, or at least one that has neither a predetermined bound nor is characterized by a point-wise (instantaneous) success condition. Is there any connection with games here? Yes, but with this we are at the frontier of current work in the semantics of concurrency and interaction and also at that frontier where the theory of economic games *might* be accommodated within an account of the semantics of programming constructs.

[10] See [JvB88, JvB98]. There is almost no conceptual space outside his interests; it's rather like traveling around inside closed curve models of the Universe.

[11] In what follows I will have nothing to say about model construction via games. I intend to stick to talk of proofs and computations.

[12] For an extended and wonderfully idiosyncratic discussion of many of the issues involved in game semantics for Linear Logic, see [Girard98].

8 Interaction and Games

The immediately foregoing has been a bit dense, I realize. So in just a little more detail: Linear Logic is often said to be a resource-conscious logic. It's clear enough what this means within the context of proofs within the logic: that A is deducible from Γ means (for linear elements of Γ) that each such element must be used at least once and can only be used once in the derivation. But this is a troubling notion, especially when one tries to provide an intuitive semantics for Linear Logic, in particular a notion of consequence. After all, these elements are thought of propositions, and according to what conception of proposition is a proposition a (very) finite resource, something that can be used up (and that must be used) in its capacity as a premise for a conclusion? Notice that if we think in terms of the Curry-Howard-DeBruijn interpretation or BHK semantics, propositions are not interpreted as statements, but rather as types, whose elements are proofs or constructions—functions constructively construed—of the type. While it is true that according to the classical extensional notion of function, it makes no more sense to constrain or record how many times a function uses an input than to constrain or record how many times a proposition is used as a premise, this is not so clearly true of a rule for computing that function—an algorithm or program or *process*. Despite this, intuitionistic semantics makes no direct use of this resource-sensitivity. Linear Logic, which can be seen as deriving from an attempt to analyze the Curry-Howard-DeBruijn/BHK semantics for Intuitionism into more fundamental elements, does precisely that. And in this regard, game semantics has shown itself capable of providing some illumination to this oddly troubling feature of Linear Logic.

For our purposes, and at the most abstract level, the two critical features of the game semantics for Linear Logic and for programming languages such as **PCF** are: (i) It is a semantics of *interaction* between Player and Opponent (or Server and Client as in [Blass94]). As Abramsky says, "The key feature of games, by comparison with with the many extant models of computation labeled transition systems, event structures, etc.) is that they provide an *explicit representation of the environment* and hence model interaction in an intrinsic fashion." ([Abramsky97]. Emphasis in the original.) (ii) The basic elements in a game are *moves*, which, of course, are *acts*—and while these are of repeatable types, they are themselves nonrepeatable individuals (or, if you like, act-type occurrences). Talk of resources *can* find a natural home here.

Still, as I noted above, there has been a troubling asymmetry in the work on game semantics, one that it inherits from Lorenzen's dialogue semantics—an asymmetry between Player and Opponent which mirrors the asymmetry between input and out (or server and client, for that matter). This asymmetry is built into Intuitionism and, sure enough, when game semantics was extended to cover Classical Linear Logic, the asymmetry has to be abandoned; the game semantic account of Classical Logic is intrinsically interactive. This allows a view much more in the spirit of real dialogues (or symmetric automata): participants switch roles quite freely and without any *global* scheduling protocol.

For more on this, see e.,g., [AM97, Laird98, BDER97].[13]

There is also that troubling question about the *architecture* of a semantic account for natural languages. The development of dynamic accounts of the semantics of natural languages has represented a step away from the static truth-conditional model.[14] Following up a game-theoretic account will involve pushing a process-oriented semantics even further. And at what cost? Perhaps we should think of the account in terms of games as yielding a theory of the operational semantics of a language that must then be supplemented by a more traditional truth-theoretic account; and that the two accounts must mesh in some fashion akin to that of full abstractness of an operational semantics of a programming language with respect to its denotational semantics.

And then there is the question of termination and winning strategies. If we are interested in proofs, say in Classical Linear Logic, there is bound to be a notion of winning strategies that privileges the Player—again the agent for the candidate theorem. But in real dialogues, as in real interactive systems, there is no winning position and need be no winners (and losers). Or put another way, the winning strategies of interest here, in truly interactive systems as in life, have the form of liveness properties: that is, they ensure that, as Beckett would have, we can keep on keeping on. And I've kept on quite long enough.

9 Summing Up

Aside from the many significant technical results due to him,[15] Johan has contributed immeasurably — immeasurably but enormously — to a number of research programs over the years, sometimes by willing them into existence or nurturing them when in their ungainly infancy. A mark of these research programs has been their interdisciplinary flavor. Johan's interests and his knowledge seemingly know no bounds and he certainly will not brook disciplinary bounds obstructing research progress. More than that, he is actively on the lookout for connections among work in different fields—actively looking to inject new ideas from disparate places into perhaps long isolated research sites.

Over the last decade or so there has been a significant effort to adapt ideas from computer science to thinking about the semantics and pragmatics of natural language. Sure enough, Johan has been at the forefront of this activity, especially fostering and encouraging this trend and suggesting further directions for study and experimentation. Most recently, these directions have included (i) the theory of proofs, especially in its structural and computational guise and (ii) the theory of games, or rather the various theories of games that have been developed in logic and elsewhere.

In this note, I have attempted my own little scouting expedition along some of the paths that Johan has forged. Let me sum up my findings: I do think

[13]Again crudely: this symmetry can be seen to be reflected in the fact that in sequents for Classical Logic, there is a more complete symmetry between antecedent (input) and succeedent (output)—both can be multiple. A bit more deeply, in Classical Logic we have to do with a negation that is more fully a duality, allowing free interchange across $\vdash$ and between Players.

[14]Though, in my opinion, even this step has been partial and not fully thought through.

[15]often to him and one or another of an impressively wide variety of collaborators

that seeing computation as proof search and seeing proof search as model construction present useful ways of looking at *aspects* of extended natural language use. And I think that it is natural to look at dialogues, in particular, through the prism of *one or another* theory of games. As Johan has stressed, there are many such theories and true to Johan's model, it is best to look for connections among them. My modest proposal is that one good place to look is at current work on game theoretic approaches to interactive systems.

10 CONGRATULATIONS

Happy Birthday Johan and may you keep on building bridges!!

References

[JvB 94] Johan van Benthem. *Logic and Argumentation Theory.* Report X-94-05, Institute for Logic, Language and Computation, University of Amsterdam. 1994.

[JvB88] Johan van Benthem. *Games in Logic.* J. Hoepelman, ed., 1988, **Representation and Reasoning.** Niemeyer Verlag, Tübingen, 3-15, 165-168. 1988.

[JvB98] Johan van Benthem. *When are Two Games the Same?* G. Bonanno et al., eds., **Games and Economic Theory**, Proceedings LOFT-3, Torino. 1998.

[AJ94] Samson Abramsky and Radha Jagadeesan. *Games and full completeness for multiplicative linear logic.* **JSL**, 59(2), 543-574. 1994.

[AJM94] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. *Full abstraction for PCF (extended abstract).* Masami Hagiya and John Mitchell, eds., **TACS'94**, Sendai, Japan. Springer-Verlag, LNCS, #789, 1-15. 1994. For a more complete version, see AJM: PCF.

[Blass92] Andreas Blass. *A game semantics for linear logic.* **APAL**, 56, 183-220. 1992.

[Blass94] Andreas Blass. *Is game semantics necessary?* E. Börger et al., eds., **CSL'93**. Springer-Verlag, LNCS, #832. 1994.

[Moortgat97] Michael Moortgat. *Categorial Type Logics.* J. van Benthem and A. ter Meulen, eds., **Handbook of Logic & Language.** The MIT Press, Cambridge MA. 1997.

[Morrill94] Glynn Morrill. **Type Logical Grammar. Categorial Logic of Signs.** Kluwer, Dordrecht. 1994.

[H095] Martin Hyland and Luke Ong. *On Full Abstraction for PCF: I, II and III.* To appear in **Information and Computation**. Available at HO: PCF.

[Stalnaker72] Robert Stalnaker. *Pragmatics.* D. Davidson and G. Harman, eds., **Semantics of Natural Language.** Reidel, Dordrecht. 1972.

[Stalnaker74] Robert Stalnaker. *Pragmatic Presuppositions.* M. Munitz and P. Unger, eds., **Semantics and Philosophy.** New York University Press. 1974.

[Grice58] H. Paul Grice. *Meaning.* **Philosophical Review**, 66, 377-388. 1956.

[Grice67] H. Paul Grice. *Logic and Conversation.* **Studies in the Way of Words.** A revised version of the William James Lectures at Harvard University, 1967. Harvard University Press. 1989.

[AM97] Samson Abramsky and Guy McCusker. *Game Semantics.* Lecture Notes from the Proceedings of Marktoberdorf'97 Summer School. 1997. Available at AM Mdorf.

[Laird98] James Laird. *Full Abstraction for Functional Languages with Control.* 1998. Available at Laird: CC.

[Abramsky97] Samson Abramsky. *Semantics of Interaction: an Introduction to Game Semantics.* A. Pitts and P. Dybjer, eds., **Semantics and Logic of Computation.** Cambridge University Press. 1997.

[Hyland97] Martin Hyland. *Game Semantics.* A. Pitts and P. Dybjer, eds., **Semantics and Logic of Computation.** Cambridge University Press. 1997.

[Felscher86] Walter Felscher. *Dialogues as a Foundation for Intuitionistic Logic.* Dov Gabbay and Franz Guenthner, eds., **Handbook of Philosophical Logic**, Volume III, 341-372. Kluwer, 1986.

[BDER97] Patrick Baillot, Vincent Danos, Thomas Ehrhard, and Laurent Regnier. *Believe it or not, AJM's games model is a model of Classical Linear Logic.* **LICS'97**, 68-75. 1997.

[Girard98] Jean-Yves Girard. *On the meaning of logical rules*, I, II. Available at Girard: LL.

[KohKon:mgfdrt99] Michale Kohlhase and Karsten Konrad. *Model Generation for Discourse Representation Theory.* Technical Report, Dept. of Computer Science, Universität des Saarlandes, Germany. 1999.