语篇与语言的功能

DISCOURSE AND LANGUAGE FUNCTIONS

外语教学与研究出版社
FOREIGN LANGUAGE TEACHING AND RESEARCH PRESS

Discourse and Language Functions

语篇与语言的功能

Editors in Chief: HUANG Guowen（黄国文）
（主编） WANG Zongyan（王宗炎）
Deputy Editors in Chief: CHEN Yongpei（陈永培）
（副主编） YANG Bingjun（杨炳钧）
LIN Zequan（林泽铨）
HUANG Jiayou（黄家佑）

外语教学与研究出版社
FOREIGN LANGUAGE TEACHING AND RESEARCH PRESS

(京)新登字 155 号

图书在版编目(CIP)数据

语篇与语言的功能/黄国文、王宗炎主编. —北京:外语教学与研究出版社,2000

ISBN 7-5600-2668-0

Ⅰ. 语… Ⅱ. 黄… Ⅲ. 英语—语言读物—英、汉 Ⅳ. H319.4

中国版本图书馆 CIP 数据核字(2002)第 031418 号

语篇与语言的功能

主编: 黄国文 王宗炎

* * *

责任编辑: 刘相东

出版发行: 外语教学与研究出版社

社 址: 北京市西三环北路 19 号 (100089)

网 址: http://www.fltrp.com

印 刷: 北京外国语大学印刷厂

开 本: 787×1092 1/16

印 张: 15.25

版 次: 2002 年 6 月第 1 版 2004 年 2 月第 2 次印刷

书 号: ISBN 7-5600-2668-0/H·1383

定 价: 18.90 元

* * *

Foreword

The papers in this volume were selected from a total of 86 presented at the International Conference on Discourse and Language Functions organised by the School of Foreign Languages, Zhongshan (Sun Yat-sen) University (Guangzhou), from the 10th to the 13th of August 1999. This was the first conference on linguistics ever hosted by the School and a major event in our academic history. By the time when the conference was held, Zhongshan University was already one of the most important universities in China focusing on systemic functional linguistics both in teaching and research. The purposes of the conference were to promote academic exchange between scholars at home and abroad, to strengthen our ties with the outside world, to familiarize our friends with the developments of the School, to make the recent happenings and up-to-date information in the field accessible to our teachers and students and to provide a forum for language researchers as well as practitioners on issues related to discourse and language functions.

The conference's 102 participants came from more than 40 universities and colleges in America, Australia, Britain, Canada, Hong Kong, Macau and other Chinese cities. At the conference seven distinguished linguists were invited to be plenary speakers (in alphabetical order): Robin FAWCETT (Cardiff), Peter FRIES (Central Michigan), M. A. K. HALLIDAY (Sydney), Ruqaiya HASAN (Macquarie), HU Zhuanglin (Peking), Christian MATTHIESSEN (Macquarie), and WANG Zongyan (Zhongshan). Glad to have so many non-Chinese linguists here, the Program Committee made very good use of the opportunity by asking them to lead the "Special Interest Group" discussions (in their respective research areas) and to join in the Panel Discussion, apart from giving plenary speeches.

The conference's success depended on the efforts of both the participants and the organisers; on behalf of the Organising Committee, we would like to express our gratitude to all those who helped us.

Many people have helped in bringing this book about, and we would like to take this opportunity to thank them. First and foremost are the contributors to the volume, especially HALLIDAY, FRIES, and MATTHIESSEN. The Lingnan Foundation gave us very generous financial support, which made it possible to invite plenary speakers from different parts of the world and to publish the book; such help is warmly appreciated. We would also like to thank the plenary speakers and the participants who played active roles at the Conference. The Guangzhou English Language Centre and the Chinese Language Centre of Zhongshan University gave the Conference financial support. Many people have helped us by reviewing the papers. Last but not least, we would like to thank the Organising Committee of the Conference. Those whose work and support which made the Conference a great success are gratefully remembered and warmly appreciated.

Huang Guowen

Wang Zongyan

Contents

1

Introduction to Halliday's "Computing Meanings: Some Reflections on Past Experience and Present Prospects"

Christian M. I. M. Matthiessen

Macquarie University, Australia

Professor M. A. K. Halliday was invited to give a plenary speech to PACLING 95, held in April 1995 in Brisbane. The result was "Computing meanings: some reflections on past experience and present prospects".

In this article, Professor Halliday reviews the changing relationship between language and computing in the field that has come to be known as "computational linguistics" or "natural language processing". When Halliday first became involved in this area as a linguist taking part in a machine translation project directed by Margaret Masterman in Cambridge in the mid 1950s, computing language meant computing words; but over the decades, he suggests, computing language has moved closer to the central property of language — meaning. Thus in the 1960s researchers began to try to compute grammatical structures — the patterns by which meaning is realized by wording in the lexicogrammar of a language (although this natural relationship between grammar and meaning was obscured by the formal theories of syntax at the time). This led to the development of grammatical parsers such as Augmented Transition Network parsers and this work continued in the 1970s. During this period the task of computing meaning began to be explored: researchers started to develop semantic interpreters to work with the output from grammatical parsers and systems for representing meaning (logic-based systems and network-based systems, the two gradually being integrated in the hybrid systems of the 1980s onwards). In the 1980s, meaning-based computational linguistics continued to develop. The challenge of representing meaning was better understood and more principled modes of representation were developed. In addition, semantic phenomena extending beyond the domain of grammatical units such as the clause were increasingly taken into account; for example, reference chains and rhetorical relations were modelled in Natural Language Processing (NLP) systems. Moreover, text generation was added to the computational linguistic research agenda, which meant that meaning had to be taken as the point of departure. This was further underlined by the attention given to systems capable of dialogue and, particularly in the 1990s, by the development of multilingual and multimodal generation systems. Another important feature of NLP in the 1990s has been the development of techniques for linking various NLP tasks to information drawn from corpora. This work is based on what Halliday calls the cline of instantiation.

Halliday's own general theory of language, systemic functional theory, and descriptions of Chinese and English have played an important role in certain areas of computational linguistics / natural language processing. The earliest well-known example of an NLP system building on systemic functional theory is Winograd's celebrated SHRDLU system from the beginning of the

1970s. This was followed, later in the 1970s, by one of the pioneering text generation systems, Davey's Proteus system, which was capable of describing games of noughts and crosses. Starting around 1980, systemic functional theory began to be used more widely as a resource of various text generation systems — not only the first of these systems, the Penman system, but also a number of other important contributions such as Terry Patten's SLANG system, Robin Fawcett & Gordon Tucker's COMMUNAL system and the system developed at Kyoto University by John Bateman. Some of these efforts have continued throughout the 1990s; and John Bateman has gone on to develop a general systemic functional generator core, the KPML system, which is available together with a user-friendly workbench. We have carried out systemic functional NLP research in Sydney since 1990, building on the Penman tradition (this has included the development of multilingual and multimodal text generation by Zeng Licheng and others in our group and of computational tools for linguistic research by Wu Canzhong and others in our group). During this period other tasks were added to the research agenda based on Halliday's systemic functional theory, including explorations in parsing by Robert Kasper and Mick O'Donnell and multilingual research by Erich Steiner and Elke Teich.

The research mentioned above can be characterised as "systemic functional computational linguistics" because it is based entirely on systemic functional theory at the level of theoretical design. However, there is also a very significant flow of ideas from Halliday within another area of computational linguistics. Halliday's work was central to Martin Kay's conception of his functional unification grammar (Kay, p.c.); and this in turn influenced Robert Kaplan in his and Joan Bresnan's development of Lexical Functional Grammar (LFG) (stated by Joan Bresnan in a talk at AILA 1999, Waseda University, Tokyo). This suggests that researchers working within LFG — and also within Head-driven Phrase Structure Grammar (HPSG) — would have a great deal to gain by drawing on Halliday's work in particular and systemic functional theory in general.

In my view, future generations will look back on 20th century linguistics and see Halliday as one of a handful of truly outstanding scholars laying the foundations for research focusing on language in the 21st century. Computational linguistics in the 20th century drew mostly on work from formal linguistics. This is easy to understand. Formal accounts of language are much easier to take into the computational realm than functional ones. Halliday's work may be significantly harder to take on board in computational modelling; but it can certainly be done, as has been demonstrated successfully for Halliday's theory of lexicogrammar, and the benefits in future will be much more far-reaching because unlike formal linguists Halliday gives us a comprehensive picture of language in context and, again unlike most formal linguists, he has made the key property of language — the ability to make meaning — central to his theory of language. In fact, Halliday's account also explains why modelling language is such a challenging task: the reason is that it is a fourth-order system — a semiotic or meaning-making system, the most complex kind of system in the hierarchy of systems that he presents in his article. To manage this complexity, we can use the kind of map of language that Halliday sketches in "Computing meanings". This constitutes a holistic approach to language in context based on systems-thinking and provides us with an alternative to the componential approach based on Cartesian Analysis that has dominated formal linguistics and a great deal of computational linguistics.

2

Computing Meanings:

Some Reflections on Past Experience and Present Prospects

M. A. K. Halliday

University of Sydney, Australia

The present paper came into being in a specific academic context. Professor Michio Sugeno, of the Tokyo Institute of Technology, well known for his leadership in fuzzy computing, had become convinced that if computers were going to advance any further towards human-like intelligence they would have to operate in a more human-like manner — which meant, with natural language. He was aware of the experience that had been gained in using systemic functional grammar in natural language processing; particularly the PENMAN text generation project started in 1980 at the Information Sciences Institute of the University of Southern California. In that project, directed by William Mann, Christian Matthiessen had been the principal linguist, and I had been retained from time to time as a consultant.

Professor Sugeno began a regular dialogue with Christian Matthiessen, and he and two of his younger colleagues attended a series of seminars on systemic theory which I gave while teaching in Japan in 1992. Matthiessen and I visited his laboratory, and prepared some comments about the linguistic implications of his work and about the nature of language as a "fuzzy" system. Sugeno was a member of the Organizing Committee of the International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and the Second International Fuzzy Engineering Symposium (FUZZ-IEEE/IFES'95) held in Yokohama on 20-24 March 1995, and Matthiessen and I were invited to participate. I offered a plenary paper, submitted in written form as "Fuzzy grammatics: a systemic functional approach to fuzziness in natural language", which appeared in the Proceedings of the conference; and gave a spoken presentation entitled "On language in relation to fuzzy logic and intelligent computing".

About a month later, the 2nd Conference of the Pacific Association for Computational Linguistics (PACLING 95) was held in Brisbane, and I was asked by the organisers to present a paper. Sugeno had begun to refer to his concept of intelligent computing as "computing with words"; and this seemed a clear formulation of his commitment to natural language. But I felt that "computing with words" was liable to be misinterpreted, and on two counts: that if you were computing with language at the lexicogrammatical level it would be with clauses and sentences rather than with words; but that in any case the relevant interfacing with language would be with the semantics rather than the lexicogrammar. If intelligent computing was computing based on natural language, it would be computing with meanings, not computing with wordings. But, at the same time, it had to be made clear that meanings are just as much phenomena of language as wordings are. Thus I was not wanting to deflect Sugeno's project, but only to interpret it in a way that seemed to me more real, and more accurate as an account of how human beings go about their tasks.

This was the genesis of the present paper. In it I tried to put the issue into a historical context, and also to

make explicit, with the help of illustrations, what the notion of computing meanings would imply in relation to a stratified, metafunctional model of language. The paper was prepared for oral presentation and I have retained it in its original state.

It is the privilege of an old person to be allowed to look back in time — even perhaps in a context such as this, when addressing specialists in computing. Computing is, after all, one of the most future-oriented of all present-day human activities! But it happens to be just forty years since I had my first encounter with computational linguistics. This was in an early project in machine translation, which was the main purpose that was envisaged for linguistic computing at the time. I was Assistant Lecturer in Chinese at Cambridge; and I was delighted at being asked to be a member of the Cambridge Language Research Unit, along with Margaret Masterman, who directed it, R. H. Richens, plant geneticist, and A. F. Parker-Rhodes, whose extensive interests ranged from botany to mathematical statistics. That was a fairly brief excursion, since I left Cambridge shortly afterwards; but since then I have had further encounters from time to time, as the field of linguistic computing has developed and expanded to its present high-energy state. Of these the most notable, for me, were the times I spent working with Sydney Lamb, first at Berkeley then at Yale, in the 1960s, and then later, in the 1980s, working with William Mann and his group at the Information Sciences Institute of the University of Southern California.

Looking back over this period, I seem to see a change of direction taking place roughly every fifteen years. The first turning point, of course, was the idea of computing with language at all; we can date this in round figures from 1950, when the idea of using a computer to translate from one language to another began to be taken seriously. The approach was essentially mathematical, in that machine translation was seen as a problem to be solved by applying to language the same logical methods that had gone into the design of the computer itself. There was, at least among mainstream researchers, no serious concern with language as a distinct type of phenomenon, one which might need to be investigated theoretically in terms of its own systemic traits.

Then in and around the mid-sixties a significant transformation took place. Language came more sharply into focus; it began to be treated as a phenomenon sui generis, with linguistics replacing logic as the theoretical point of departure from which to engage with it. In Russia, and perhaps in Japan, this transformation was fairly gradual; but in North America and western Europe it was more catastrophic, because 1965 was the year in which the U.S. Air Force officially pronounced machine translation a failure and withdrew the funding from the major translation projects. American researchers who continued computing with language carried on their activities under other headings; taking over "artificial intelligence" as a unifying concept, they addressed some more specific computational tasks that were not tied directly to translating, such as parsing, abstracting, question-answering or expert systems and the like. Terry Winograd's major contribution *Understanding Natural Language* belongs to this time; so does the earliest work in computing our own system networks (forming paradigms, implementing realisations and so on) which was done by Alick Henrici at University College London (see Winograd 1972; Henrici 1966 [Halliday & Martin 1981]).

The third turning point came around 1980, when a new generation of computers made it possible to build systems that approached somewhat closer to the complexity of natural language.

Computational linguists now had at their disposal the necessary speed, memory and processing power that made parsing and generating look like more realistic goals. Before that time, computational grammars had remained fairly simple: limited to "toy" domains, syntactically constrained, and accommodating only rather gross distinctions in meaning; whereas one could now begin to conceive of "writing the grammar of a natural language in computable form", with account taken of considerations from linguistic theory. For linguists this meant that the computer now became, for the first time, a tool for linguistic research, a means of finding out new things about language: it made it possible on the one hand to test grammatical descriptions (which were becoming too complex to be tested manually) and on the other hand to build a large-scale corpus — not only to assemble and manage it but to access it and interrogate it from many different angles. Two very large systemic grammars of English for text generation were developed during this period: the Penman "Nigel" grammar built up by Christian Matthiessen under William Mann's direction at I.S.I., and the "Communal" grammar compiled by Robin Fawcett at the University of Wales (Matthiessen & Bateman 1992; Fawcett et al 1993).

Fifteen years from 1980 brings us up to 1995; so are we now moving through another turning point? I suspect we may be, in that the status of language — that is, its general relationship to computing — is once again undergoing a major shift. It seems that language is being relocated at the very centre of the computing process itself. In order to explore this further, I would like to refer explicitly to the notion of "computing meanings" that I used in the title of my talk. But first let me say what I do not mean. I am not referring here to the rather overworked concept of the "information society". It is no doubt true, as we have been being told for the past two decades, that the exchange of goods-&-services is rapidly being overtaken by the exchange of information as the dominant mode of socio-economic activity; and this is certainly relevant, since information is prototypically made of language. But it is relevant as the **context** of the change I am referring to; it is not the nature of the change itself. What I have in mind is what Professor Sugeno calls "intelligent computing", which he defines as computing that is based on natural language. In Sugeno's view, if we want to move forward to the point where the computer becomes truly intelligent, we have to make it function, as people do, through the medium of language. And this critically alters the relationship of language to computing.

Let me then briefly return to the past, and use the concept of "computing meanings" to track the shifting relationship between the two. In the 1950s, there was no direct engagement between them at all; it was taken for granted that translating meant selecting from lists of possible formal equivalents —that it was an engineering job, in other words. The prevailing metaphor was that of a code: as Warren Weaver conceived of it, a Russian text was simply an English text with different coding conventions. There was no conception of language as the systemic resources that lay behind a text hence no need to construct any kind of theoretical model of language. This is not to imply that nothing of value was achieved; on the contrary, there was a great deal of essential analytical work, especially in lexis and morphology: see for example Delavenay's (1960) *Introduction to Machine Translation*. But meaning was taken as given, rather than problematized; this was not yet "computing with meanings".

In the second phase, language emerged as a computable object, needing to be modelled in its own right the concept of "a grammar", a description of a language as written by a linguist, came to

be accepted — or at least tolerated as a necessary nuisance. But such a descriptive grammar had no direct place in the computing process; the computational grammar was purely procedural, a program for parsing strings of words. (There was little text generation during this phase; one of the few such projects was the systemic generator developed in Edinburgh by Anthony Davey (1978).) On the other hand, as implied by the label "computational linguistics" which came into favour during this period, these were computing operations performed on language; the way the strings of words were manipulated was designed to establish what they meant. There had clearly been a move in the direction of computing with meanings.

It was in the third phase, after about 1980, that developments took place which the phrase "computing with meanings" could more accurately describe. By this time computational grammars came to take the form of descriptive, or "declarative", representations, and researchers sought to develop generalized forms of representation suited to their computational needs. It was accepted that these had to accommodate large-scale grammars with reasonable detail and complexity, not just the dedicated and simplified grammars of the earlier phase. The move from procedural to declarative, and the shift in scale and in depth of focus, changed the relationship once again: the value of a piece of wording was, for the first time, being interpreted in terms of **agnation**, its locus in the total meaning potential of the language. It would seem accurate now to characterise computational linguistic operations as operations on meaning.

Perhaps I could gloss this with a story from personal experience. Back in the 1960s, the working assumption was, "If we can't compute your grammar, your grammar must be wrong". This was not, let me make it clear, arrogance on the part of individual computer specialists; simply a conviction that the computer defined the parameters of human understanding. To me it seemed that the constraints set by current technology, and even more those set by current theories of logic, at that particular moment in human history, were quite irrelevant for evaluating models of grammar; this was the main reason why I moved away from the scene, for the second and (as I thought) the final time. So when in 1980 William Mann came to see me — I was working at U. C. Irvine — and asked me to write a systemic grammar of English for his forthcoming text generation project (the "Penman" I mentioned earlier), I challenged him on this point: how much would I have to "simplify" (that is, distort) the grammar in order to make it computable? Bill Mann's answer was, "If I can't compute your grammar, I'll have to learn how". He later added an interesting comment: "I don't always understand why linguists describe things the way they do; but I've come to realise they always have a reason". It was clear then that the relationship of language to computing had moved on.

There is a serious point underlying this little anecdote. Of course we, as grammarians, had to learn to write our descriptions in computable form: that is, to make them fully explicit. This was an important exercise, from which we learnt a great deal. But to make them explicit is not the same demand as to make them **simple**. Language is not simple; it is ferociously complex— perhaps the single most complex phenomenon in nature; and at least some of that complexity had to be accounted for. To take just one example: what was referred to under the general label "constituency" is not a single, undifferentiated type of structure (like a "tree"), but a highly variable array of different meaning-making resources, with highly complex interrelations among them. Unfortunately the linguists themselves had made the problem worse: the prevailing ideology, at least

in America, but also perhaps in western Europe and in Japan, was the structuralist one deriving out of Bloomfield via Chomsky; and this was highly reductionist, in that, in order for natural language to be represented as a formal system, much of the rich variation in meaning had to be idealized out of the picture. But not only was it reductionist — it was also authoritarian: for linguists of this persuasion, the standard response to anyone who disagreed with them was, "Either your grammar is a notational variant of my grammar, or else your grammar is wrong". So computer scientists who ventured across the frontier into linguistics, in that second phase, might reasonably conclude that a natural language could be modelled as a well-formed system conforming to a recognizable mathematical-type logic. This period in mainstream linguistics was an age of syntax, in which a grammar could be reduced to an inventory of well-defined structural forms.

But by the time of what I am calling the third phase, the orientation of linguistics had changed. Much more attention was now being paid to semantics, both by linguists working from within the Chomskyan paradigm and by those who had remained outside it. Thus the concept of "computational linguistics" already implied something closer to "computing meanings"; it implied a system that fully engaged with natural language, parsing and generating text in operational contexts. Machine translation once again figured prominently on the agenda, as the four components of a speech-to-speech translation programme fell into place: recognition + parsing + generation + synthesis were all seen to be possible, and such systems now began to appear on the market — not perhaps doing all that was claimed for them, but performing adequately for a limited range of tasks. The general encompassing term was now "natural language processing".

But these continue to appear as two distinct activities: computing, and language processing. There is still the awkward disjunction between computing in general, and meaning. Computational linguistics, or natural language processing, or generation and parsing, are separate operations from computing in the general sense. And this has some strange consequences. I recently visited Japan, to attend the international conference on fuzzy systems chaired by Professor Sugeno. There were many interesting exhibits, from industry and also from LIFE, the Laboratory for International Fuzzy Engineering, which was just coming to the end of its own life cycle. One of the latter was FLINS, the Fuzzy Natural Language Communication System; this was a question-answering system incorporating an inference engine, which reasoned by taking the special case, assigning it to a general class and inferring a course of action there-from. One very interesting feature of this system was the use of traditional proverbs to state the general proposition — this being precisely the function that proverbs had in our traditional societies (my grandmother had a proverb for every occasion). But what struck me particularly was that their inferencing procedures, which involved both symbolic and fuzzy matching, were totally insulated from their language processing; they had simply bought ready made commercial systems for parsing and generating text. In other words, reasoning and inferencing were being treated as non-linguistic operations — although they are being entirely carried out through the medium of language. Notice what this implies: that there is no semantic relationship — no systematic relationship in meaning — between, say, the proverbial expression *more haste less speed* and (the verbal component of) an instance to which it relates as a general proposition, for example *Don't be in a hurry to overtake; it only slows you down. More haste less speed*

This seems to be a scenario in which linguistics and computing inhabit two different worlds.

Natural language is being seen as something to be separately processed, rather than as an inherent part of the computing process itself. And this is the difference, it seems to me, between the third phase and the subsequent, fourth phase which we may now be entering. With "intelligent computing", the boundary between computing in general and natural language processing rather disappears: all computing could involve operating with natural language. Computing then becomes synonymous with computing meanings.

"Intelligent computing" is an important concept which could have very far-reaching consequences; so I would like to consider it here in a little more detail. As I mentioned earlier, Sugeno defines this as computing based on natural language; he also sometimes expresses it in simplified terms as "computing with words". Lotfi Zadeh, the founder of "fuzzy logic", used this same formulation in his talk at the conference, remarking that computing with words "enhances the ability of machines to mimic the human mind, and points to a flaw in the foundations of science and engineering". He also observed that, while human reasoning is "overwhelmingly expressed through the medium of words", people are only taught to compute with numbers — they are never taught how to compute with words. Commenting on the formulation "computing with words", Zadeh glossed "computing" as "computing and reasoning", and "words" as "strings of words, not very small".

We need to reformulate that last expression, in two steps as follows:

(1) for *words* (or *strings of words*), read *wordings*: words in grammatical structures, i.e. lexicogrammatical strings of any extent;

(2) for *wordings*, in turn, read *meanings*, again in the technical sense; that is, semantic sequences of any given extent.

Note that, in reformulating in this way, we are not questioning Sugeno's concept; we are recasting it in terms of semiotic practice. People reason and infer with **meanings**, not with wordings (they don't store wordings, as is easily demonstrated when they repeat the steps along the way). To put this in more technical terms: reasoning and inferencing are semantic operations. Thus, if they are performed computationally, this will be done on semantic representations. But, at the same time, such semantic representations are related systemically to the lexicogrammatical ones. In other words, the meanings are **construed** in wordings. When people reason through talk, they are actually reasoning with **meanings**; but these meanings are not a separate "cognitive" universe of concepts or ideas — they are patterns of semantic (that is, linguistic) organisation brought about, or "realized", by the wordings. They are reasoning in language, even though not, strictly speaking, in words.

There are in fact two noticeable disjunctions needing to be overcome, in the move into the intelligent computing phase. One is this one between language and knowledge, or rather between meaning and knowing, or meaning and thinking. Natural language processing systems typically operate with one kind of representation for language, a "grammar", and another, very different kind of representation for knowledge — a separate "knowledge base"; and the two different representations have then to be made to interact [Figure 1]. If we reconceptualize the knowledge base as a **meaning base** — as another level in the representation of language — the problem becomes more easily manageable: instead of two systems, one inside language and one outside (the grammatical and the conceptual), there is only one system, language, with two levels of

representation, the grammatical (lexicogrammatical) and the semantic [Figure 2] (cf. Halliday & Matthiessen 1999).

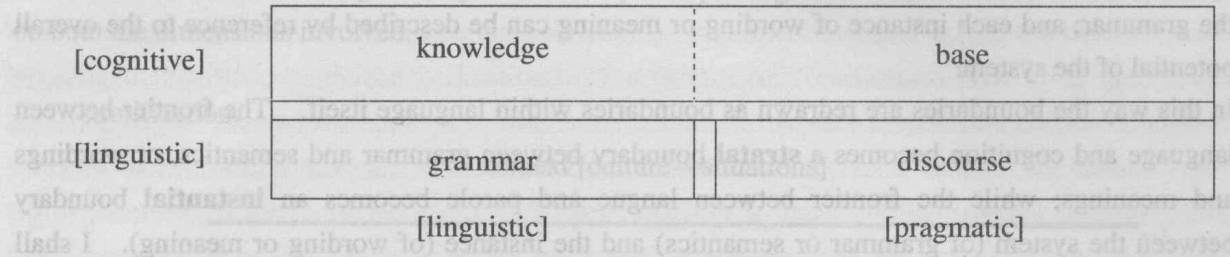

Figure 1: The cognitive model

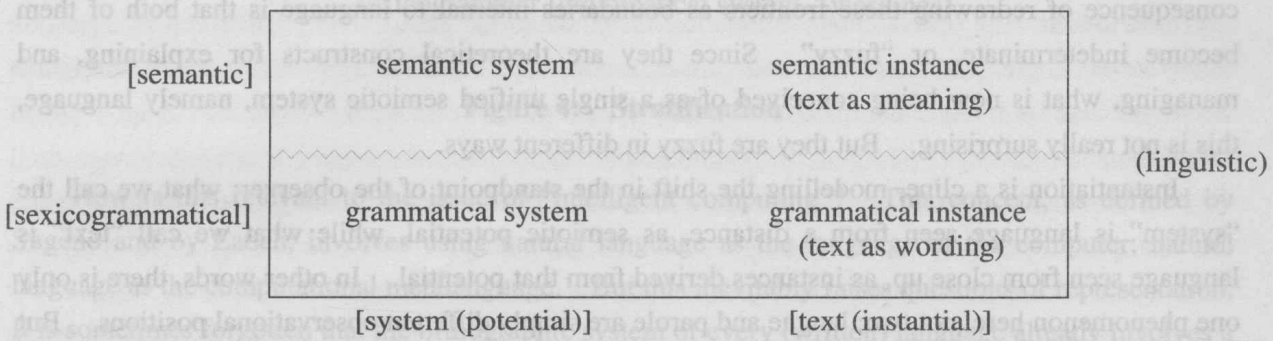

Figure 2: The semantic model

The second disjunction I have in mind is that between the **instance** and the **system** of which it is an instance. The grammar of a natural language is represented **systematically**: conventionally as a set of structures, or, in a systemic grammar, as a network of options — language as semiotic potential. On the other hand, reasoning and inferencing are operations performed on (representations of) **instances**. Now, any instance is meaningful only by virtue of its place in, and its derivation from, the total systemic potential. But in computational reasoning and inferencing, the instances are typically presented as unrelated to the overall system of the language. They are handled as "discourse", with the implication that discourse is process, without any system behind it. If we reconceptualize discourse as "text"—as the instantiation of an underlying system — the problem of reasoning and inferencing may also become more easily manageable [Figures 1 & 2].

Each of these disjunctions corresponds to a major duality in western thinking, enshrined in contemporary philosophy and philosophical linguistics. The first is the duality between language and mind, which is institutionalized in today's dichotomy between linguistics and cognitive science; the second is the duality of langue and parole, which in turn is institutionalized in today's dichotomy between linguistics and pragmatics (See Ellis 1993, esp. chapter 5, and Matthiessen 1993, for relevant discussions.). These dualities are taken over into our intellectual pursuits — such as computational linguistics — without being problematized or even brought to conscious attention. But there are alternative strategies available, as I have remarked. Instead of two systems, one inside language (the grammatical) and one outside language (the conceptual, or cognitive), it is possible to operate with one system, language, having two related levels of representation, the (lexico)grammatical and the semantic; and instead of two classes of phenomena, one of language

(grammar) and one of parole (discourse), it is possible to operate with one phenomenon, language, having two degrees or phases of instantiation, that of system and that of text. The "knowledge base" becomes a "meaning base", which can then be described in a framework that is homologous to the grammar; and each instance of wording or meaning can be described by reference to the overall potential of the system.

In this way the boundaries are redrawn as boundaries within language itself. The frontier between language and cognition becomes a **stratal** boundary between grammar and semantics, or wordings and meanings; while the frontier between langue and parole becomes an **instantial** boundary between the system (of grammar or semantics) and the instance (of wording or meaning). I shall suggest below that the two dimensions of **stratification** and **instantiation** can be used to define a matrix for locating computational linguistic representations (See Figure 7). Meanwhile, a consequence of redrawing these frontiers as boundaries internal to language is that both of them become indeterminate, or "fuzzy". Since they are theoretical constructs for explaining, and managing, what is now being conceived of as a single unified semiotic system, namely language, this is not really surprising. But they are fuzzy in different ways.

Instantiation is a cline, modelling the shift in the standpoint of the observer: what we call the "system" is language seen from a distance, as semiotic potential, while what we call "text" is language seen from close up, as instances derived from that potential. In other words, there is only one phenomenon here, not two; langue and parole are simply different observational positions. But we can also position ourselves at an intermediate point along this cline, moving in from one end or from the other. I think that the critical intermediate concept, for our purposes, is that of **register**, which enables us to model contextual variation in language. Seen from the instantial end of the cline, a register appears as a cluster of similar texts, a **text type**; whereas seen from the systemic end, a register appears as a **sub-system**. Computationally it is possible to exploit these two complementary perspectives [Figure 3].

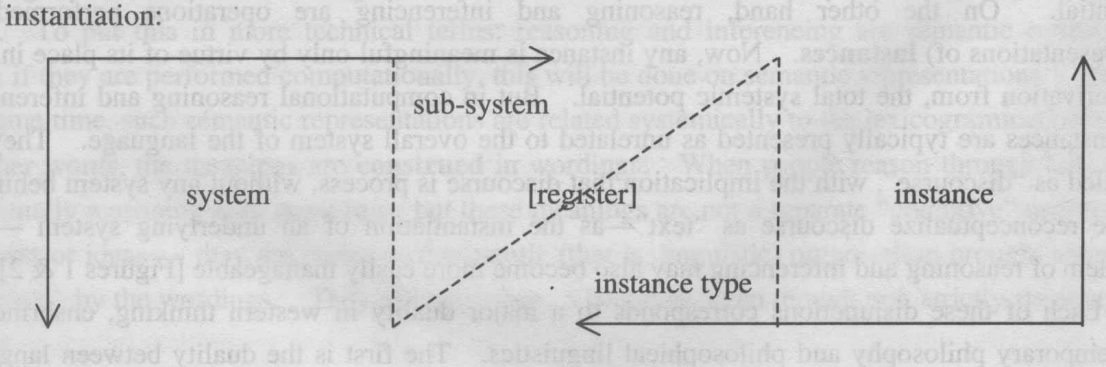

Figure 3: Instantiation

Stratification is a relationship of a different kind. The boundary between semantics and grammar is a true boundary; these are two distinct levels, or, **strata**, within the content plane of natural language, and they have different realms of **agnation** — that is, the organisation of grammatical space differs from the organisation of semantic space. But we can draw this boundary at different places, shifting it "up or down" according to the task in hand. Furthermore, having defined the relationship between semantics and grammar in this way, we are then able to extend the

same concept (known as **realisation**) so as to model the relationship of language to the context of situation, interpreting the context also as a semiotic system [Figure 4]. Thus by deconstructing the two dualities and overcoming the disjunctions they set up, we gain additional freedom of movement on both the dimensions involved.

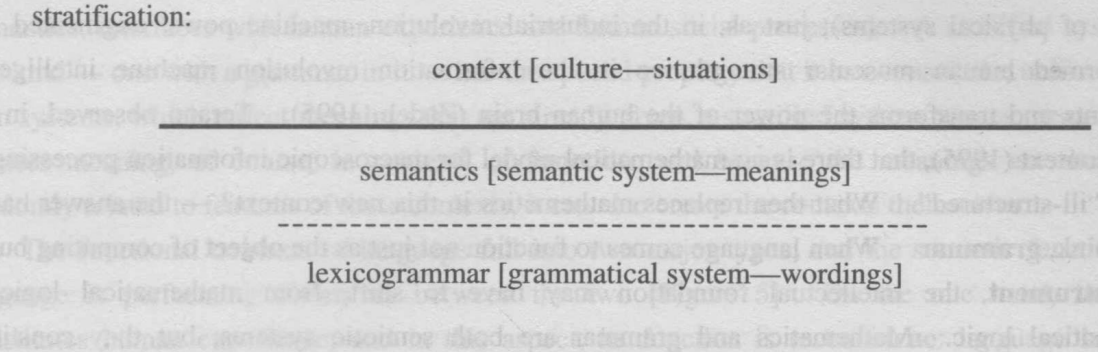

Figure 4: Stratification

How is this relevant to the issue of "intelligent computing"? This concept, as defined by Sugeno and by Zadeh, involves using natural language as the language of the computer; natural language as the computational metalanguage. But this inevitably raises questions of representation. It is sometimes forgotten that the orthographic system of every (written) language already involves a large number of highly complex decisions about meaning — what meanings are to be distinguished, how they are to be interrelated, and so on; to take a very clear example, in the English writing system, if a semantic contrast is realized by the order of elements it will be registered in writing *(you can go / can you go*), whereas if it is realized by intonation or rhythm it is not (thus *you can go* represents a wide range of different kinds and degrees of emphasis, different modalities and so on). Since these decisions were taken a long time ago, without conscious reflection, we tend simply to forget about them; but the differences in meaning that are obscured in this way could be critical to the computational process. Even if we have a speech recognition system which can accept spoken input, however, this does not eliminate the problem. Using natural language as metalanguage does not mean plucking isolated instances out of the air, as can be done with simple command control or question-answering systems. It means locating each instance within a defined realm of meaning potential, from which it derives its value in the given computational context — in other words, relating the instance to the system, or to some specified register or sub-system; and this requires engaging with the language in theoretical terms. We cannot just take language for granted as something we know all about in terms of common sense (which usually means, in practice, in terms of how our teachers used to talk about language in primary school). Intelligent computing will make stringent demands on the full range of complexity of natural languages; otherwise they would be of little use for the purpose. Therefore, it will require theoretical tools for describing and managing that complexity.

In other words, it will require what I have been calling a **grammatics**: a model of language which uses grammar as its underlying logic. Let me recall here Zadeh's reference to the "flaw in the foundations of science and engineering", and ask what Zadeh meant by this remark. The flaw, as Zadeh saw it, was that mathematics ruled, and that events were "quantized rather than granular":