CHAPTER 1

# Introduction to Ontological Semantics

This chapter introduces and briefly outlines ontological semantics. After placing the theory in the context of its predecessors in this general part, the chapter goes into the following sections:

- Section 1.1 places the theory in a model of language-communication situation, focusing on the division of labor between two intelligent agents: the discourse producer and the discourse consumer.
- Section 1.2 contains the initial sketch of ontological semantics. This is perhaps the only section that is essential to read before skipping to part II, though sections 1.5 and 1.6 may also turn out to provide some useful preparation for such a reader.
- Section 1.3 mentions briefly the relations of ontological semantics to the non-semantic components of an NLP system. The only other place in the book where this issue is touched on again is section 8.1.
- Section 1.4 addresses the alternative architectures for an NLP system, such as the stratified model, the flat model, and the constraint-satisfaction model, the latter being largely adoped in ontological semantics.
- Section 1.5 characterizes in more detail the functions (but not the actual processing—see chapter 8 for that) of the dynamic sources of ontological semantics, the analyzer and the generator. Of these two, the former gets the most attention in the book (see also the book's conclusion).
- Section 1.6 discusses the function (but not the substance—see chapter 7 for that) of the static sources in ontological semantics: the ontology, the lexicon(s), and their derivatives.
- Section 1.7 explains the role and integration of microtheories in ontological semantics. The microtheories introduce the polymethodological nature of this enterprise and remove it from the realm of the (single) method-driven theories.
- Section 1.8 traces the development of ontological semantics over the last decade and a half and places the various earlier contributions to it both chronologically and in the context of the pertinent applications. This section is largely of interest only to

readers who have kept themselves informed over the years about the activities of the ontologial semantics community.

*Ontological semantics* is a theory of meaning in natural language and an approach to natural language processing (NLP) that uses a constructed world model, or ontology, as the central resource for extracting and representing the meaning of natural language texts and for reasoning about knowledge derived from texts. This approach also makes it possible to generate natural language texts based on representations of their meaning. The architecture of an archetypal implementation of ontological semantics comprises, at the most coarse-grain level of description,

- A set of static knowledge sources, namely, an *ontology*, a *fact repository*, a *lexicon* connecting an ontology with a natural language, and an *onomasticon* or lexicon of names (one lexicon and one onomasticon are needed for each language)
- *Knowledge-representation languages* for specifying meaning structures, ontologies, and lexicons
- A set of processing modules—at the least, a semantic *analyzer* and a semantic *text generator*

Ontological semantics directly supports such applications as machine translation of natural languages, information extraction, text summarization, question answering, advice giving, collaborative work of networks of human and software agents, and so on. For applications other than machine translation, a reasoning module is added that manipulates meaning representations produced by the analyzer to generate additional meanings that can be recorded in the fact repository and/or serve as inputs to text generation for human consumption.

Any large, practical, multilingual computational linguistic application requires many knowledge and processing modules integrated in a single architecture and control environment. For maximum output quality, such comprehensive systems must have knowledge about speech situations, goal-directed communicative actions, rules of semantic and pragmatic inference over symbolic representations of discourse meanings, and knowledge of syntactic, morphological, and phonological/graphological properties of particular languages. Heuristic methods, extensive descriptive work on building world models, lexicons, and grammars, as well as a sound computational architecture are crucial to the success of this overall paradigm. Ontological semantics is responsible for a large subset of these capabilities.

The above generalized application architecture also includes an "ecological," a morphological, and a syntactic component, both in the analysis and in the generation processes. In applications, such components have usually been developed quite independently of the central ontological semantic component, though the knowledge required for them was often (though not in every implementation) integrated in the overall system lexicons. Thus, for instance, grammar formalisms have remained out-side the immediate scope of theoretical work in the ontological semantic model, and indeed several different grammar formalisms have been used to support analysis and generation in the different implementations. Due to this state of affairs, we do not include grammar formalisms and actual rule sets in the core knowledge sources of the model. The interaction between the ontological semantic processing and the rest of the processing takes place in actual implementations through the specification of the content of the input structures to the semantic analyzer and of the output structure of the semantics-based sentence-planner module of the generator.

Our theoretical work in semantics is devoted to developing a general semantic theory that is detailed and formal enough to support natural language processing by computer. Therefore, issues of text-meaning representation, semantic (and pragmatic) processing, the nature of background knowledge required for this processing, and the process of its acquisition are among the central topics of our effort. Ontological semantics shares the commitment to these foundational issues with a number of approaches to processing meaning in artificial intelligence, among them conceptual dependency, preference semantics, procedural semantics, and related approaches (e.g., Schank 1975; Schank and Abelson 1977; Schank and Riesbeck 1981; Wilensky 1983; Wilks 1975a, 1975b, 1977, 1982; Charniak and Wilks 1976; Woods 1975, 1981; Lehnert and Ringle 1982; Waltz 1982; Charniak 1983b; Hirst 1987). Moreover, the influences go beyond purely computational contributions back to cognitive psychology and cognitive science (Miller and Johnson-Laird 1976; Fodor, Beaver, and Garrett 1974; see also Norman 1980). The foundational issues in this research paradigm, in fact, transcend natural language processing. They include the study of other perceptors (e.g., speech, vision) and effectors (e.g., robotic movement, speech synthesis) as well as reasoning (e.g., general problem solving, abductive reasoning, uncertainty, and many other issues). Newell and Simon (1972) provide an influential formulation of the overall paradigm that underlies all the above-mentioned work as well as many contributions that were not mentioned (see also Newell, Shaw, and Simon 1958; Miller, Galanter, and Pribram 1960; Newell 1973; Newell and Simon 1961, 1976; McCarthy and Hayes 1969; McCarthy 1977). This paradigm certainly underlies ontological semantics.

What sets this knowledge-based paradigm apart is the reliance on the glass-box rather than black-box approach to modeling understanding. In other words, these theories do not attempt to account for meaning in terms of fully observable (though, interestingly, not necessarily correctly understood!) phenomena, namely, pairs of inputs and outputs (stimuli and responses; see section 3.4.1) to a language processor, understood as a black box. Instead they aspire to come up with hypotheses about what processes and what knowledge is needed in order to recreate the human ability to process language using computers. This is done by modeling the contents of the black box, necessarily using notions that are not directly observable.

Ontological semantics subscribes to a version of this tenet, the so-called weak AI thesis (see section 2.4.2.2), which avoids the claim that computer programs directly model human semantic capacity. Instead, this hypothesis suggests functional equivalence—that is, that computer programs can attain human-quality results, though not using the exact methods that humans use.

The tenets of ontological semantics overlap with the tenets of semantic theories developed within the generative paradigm in linguistics (Fodor 1977; see also section 3.5). There are also important differences, along at least the following two dimensions:

- The purview of the theory (ontological semantics includes all of the following: lexical and compositional semantics, pragmatics, reasoning)
- The degree to which the theory has been actually both developed and implemented through language description and computer-system construction

A number of differences exist between the mandates of general semantic theory and semantic theory for NLP. In what follows, we suggest a number of points of such difference (this list is an extension of the discussion in Nirenburg and Raskin 1986; see also Raskin 1990—cf. chapter 4).

While it is agreed that both general and NLP-related theories must be formal, the nature of the formalisms can be quite different because different types of reasoning must be supported. A general linguistic theory must ensure a complete and equal grain-size coverage of every phenomenon in the language; an NLP-related theory must be sufficiently flexible and robust to adapt to the purposes of any application. The ultimate criterion of validity for a general linguistic theory is explanatory adequacy; for an NLP-related theory, it is the success of the intended applications. A general linguistic theory can avoid complete descriptions of phenomena once a general principle or method has been established: a small number of clarification examples will suffice for its purposes. In NLP, the entire set of phenomena present in the sublanguages of applications must be covered exhaustively. A general linguistic theory has to be—and, actually, has occasionally been (see, for instance, Raskin 1985a, 1985b)—concerned about the boundary between linguistic and encyclopedic knowledge. This distinction is more spurious in NLP-oriented semantic theories because in order to make semantic (and pragmatic) decisions, a system must have access equally to both types of data (Raskin 2000).

A general linguistic theory can be method driven—that is, seek ways of applying a description technique developed for one phenomenon in the description of additional phenomena (this reflects the predominant view that generalization is the main methodology in building linguistic theories). But an NLP-related theory should be task driven—which means that adequacy and efficiency of description take precedence over generalization (Nirenburg and Raskin 1999).

The research program of ontological semantics is shared to a large degree by the work of Lenat and associates on CYC (e.g., Lenat 1995; Lenat and Guha 1990; see also Mahesh et al. 1996b) and by John Sowa's knowledge-representation efforts (e.g., Sowa 2000). While considerable differences exist in the types of knowledge represented, size and coverage of various knowledge resources, intended applications, the nature of systems implemented on the basis of each approach, and so on, Lenat's and Sowa's work joins ontological semantics in assuming that the task of extracting and formally representing knowledge about the world and language is a necessary condition for attaining truly intelligent computer systems.

## 1.1 A Model of Language-Communication Situation for Ontological Semantic Theory

Ontological semantics, as a mentalist approach to building NLP-related language-processing theories, is centered around the metaphor of the model of an intelligent agent.[1] An NLP-related theory must account for such properties of intelligent agents as goal- and plan-directed activity, of which language activity is a part—verbal actions, together with perceptual, mental, and physical actions, comprise the effector inventory of an intelligent agent. Such a theory must also take into account the knowledge of the agent's attitudes to the entities in the world model as well as to remembered instances of events and objects in its own episodic memory. Not only are these attitudes often the subject of a discourse, but they influence the form of discourse on other topics.

Building nontrivial natural language processing systems that manipulate meaning is best done using the metaphor of modeling intelligent agents immersed in a language-communication situation. In other words, we prefer to ground our meaning-representation theory on cognitive premises rather than on purely logical ones. In the most basic and simplified terms, we define our model of an intelligent agent as follows. An intelligent agent is a member of a society of intelligent agents. The agent's actions are goal directed. It is capable of perception, internal symbol manipulation, and action. Its actions can be physical, mental, or communicative. The communicative actions are used for communicating with other agents. An agent's perceptual mechanism is a model of the perceptual mechanism of humans. The peculiarities of the perception and action sides of the agent are less central to a discussion of ontological semantics, so we will concentrate on the agent's resident knowledge and the processing environment for the treatment of natural language.

We model the communication situation as follows. It involves at least two intelligent agents—a discourse (text, speech) producer and a discourse consumer. The communication situation also involves the discourse itself—in our case, a text. More precisely (though this is not a crucial distinction from the standpoint of text processing), discourse producer and consumer are roles played by intelligent agents, because each agent can play any of these roles at different times. The message conveyed by a text can be viewed as an action that the discourse consumer perceives as a step in a discourse producer's plan to achieve one of his or her active goals.[2] These plans take into account the knowledge the producer has (or assumes it has) about the target audience. A theory of discourse goals must, therefore, follow the prior introduction of a model of a participant in a language-communication situation.

### 1.1.1 Relevant Components of an Intelligent Agent's Model

The following components in an agent's model are relevant for its language-processing ability:[3]

- Knowledge about the world, which we find useful to subdivide into:
  - An ontology, which contains knowledge about types of things (objects, processes, properties, intentions) in the world
  - A fact repository, an episodic memory module containing knowledge about instances (tokens) of the above types and about their combinations; a marked recursive subtype of this knowledge is a set of mental models of other agents (see, for instance, Ballim and Wilks 1991, for an analysis of the "artificial believers"), complete with their own components—these models can be markedly different from the "host" model
- Knowledge of natural language(s), including, for each language:
  - Ecological, phonological, morphological, syntactic, and prosodic constraints
  - Semantic-interpretation and semantic-realization rules and constraints, formulated as mappings between lexical units of the language and elements of the world model of the producer
  - Pragmatics and discourse-related rules that map between modes of speech and interagent situations, on the one hand, and syntactic and lexical elements of the meaning-representation language, on the other
- Emotional states that influence the "slant" of discourse generated by an agent (Picard 2000)
- An agenda of active goal and plan instances (the intentional plane of an agent)

### 1.1.2 Goals and Operation of the Discourse Producer

The discourse-producer goals will be formulated in terms of these different components. Thus, a producer may want to achieve the following types of interagent communicative goals:

1. Modify the discourse consumer's ontology—for example, by giving a definition of a concept

2. Modify the discourse consumer's episodic memory—for example, by stating a fact, describing an object, or relating an event
3. Modify the discourse consumer's model of the producer—for example, by expressing its attitude toward some fact (e.g., *Unfortunately, Peter will come too*)
4. Modify the discourse consumer's attitudes toward facts of the world
5. Modify the discourse consumer's agenda—for example, by threatening, giving an order, or asking a question
6. Modify the discourse consumer's emotional state

A discourse producer can achieve these goals by choosing not only what to say, but also how to say things. Usually, one element of discourse will achieve several goals at the same time. For instance, if the producer has any authority over the hearer, the fact of simply stating its own opinion about a fact (a goal of type 3) may very well affect the hearer's opinions, thus achieving a goal of type 4 (e.g., Wilensky 1983). Goal types are represented in the world model of an agent as postconditions (effects) of complex events (see Carlson and Nirenburg 1990 for the description of the formalism and the motivation behind it; cf. section 7.1.5—see also Moreno Ortiz, Raskin, and Nirenburg 2002).

The producer's processing during generation can be sketched as follows. Given an input stimulus, the producer will activate a goal, choose a rhetorical plan to realize that goal, and generate a text. This is done with the help of its knowledge about the world, about the consumer, about the target language (at both the sentence and the discourse level), and about the relevant pragmatic constraints.

### 1.1.3 Operation of the Discourse Consumer

The discourse consumer's processing during analysis can be very roughly sketched as follows. Given an input text, the consumer must first attempt to match the lexical units comprising the text, through the mediation of a special lexicon, with elements of the consumer's model of the world. To facilitate this, it will have to analyze syntactic dependencies among these units and determine the boundaries of syntactic constituents. The next step is filtering out unacceptable candidate readings through the use of selectional restrictions, collocations, and special heuristics, stored in the lexicon. The consumer must then also resolve the problems of coreference by finding referents for pronouns, other deictic lexical units, and elliptical constructions. Furthermore, information on text cohesion and producer attitudes has to be determined, as well as, in some applications, the goals and plans that lead the producer to produce the text under analysis.

Many additional processes are involved in interpretation. A semantic theory for natural language processing must also account for their interaction in a computational model—that is, the overall architecture and control of the semantic and pragmatic interpretation process. Control considerations, we believe, must be an integral part of semantic theories for natural language processing, of which ontological semantics is an example. However, many of the current semantic theories, notably those relying on unification as the main processing method, essentially relinquish control over control. A whole dimension of modeling is thus dispensed with, leading to reduction in the expressive power of a theory and extra constraints on building applications. Why not accept unification as one of a number of possible control structures? And, for every processing module, choose a control structure most responsive to the peculiarities of the phenomenon treated? In AI, there is a long tradition of looking for the most appropriate representation of a problem, which will "suggest" the most appropriate algorithm for processing it. It is clear that different representations must be preferred for different problems (see, e.g., Newell and Simon 1972). Adopting a single type of representation and a single control method for all tasks means putting method before phenomena.

## 1.2 Ontological Semantics: An Initial Sketch

Like any semantic theory for natural language processing, ontological semantics must account for the processes of generating and manipulating text meaning. An accepted general method of doing this is to describe the meanings of words and, separately, specify the rules for combining word meanings into meanings of sentences and, further, texts. Hence the division of semantics into lexical (word) semantics and compositional (sentence) semantics. Semantics for NLP must also address issues connected with the meaning-related activities in both natural language understanding and generation by a computer. While the semantic processing for these two tasks is different in nature—for instance, understanding centrally involves resolution of ambiguity while generation deals with resolution of synonymy for lexical selection—the knowledge bases, knowledge-representation approaches, and the underlying system architecture and control structures for analysis and generation can be, to a realistic degree, shared. This view is a departure from our earlier views (Nirenburg and Raskin 1987a, 1987c), brought about by practical experience in description and implementation of nontoy applications.

In ontological semantics, the meaning representation of a text is derived through

- Establishing the lexical meanings of individual words and phrases comprising the text
- Disambiguating these meanings
- Combining these meanings into a semantic dependency structure covering
  - The propositional semantic content, including causal, temporal, and other relations among individual statements

  - The attitudes of the speaker toward the propositional content
  - The parameters of the speech situation
- Filling any gaps in the structure based on the knowledge instantiated in the structure as well as on ontological knowledge

It is clear from the above description that ontological semantics incorporates information that in some approaches (e.g., Lascarides 1995; Asher and Lascarides 1995) has been delegated to pragmatics.

The final result of the process of text understanding may include some information not overtly present in the source text. For instance, it may include results of reasoning by the consumer, aimed at filling in elements required in the representation but not directly obtainable from the source text. It may also involve reconstructing the agenda of rhetorical goals and plans of the producer active at the time of text production and connecting its elements to chunks of meaning representation.

Early AI-related natural language understanding approaches were criticized for not paying attention to the halting condition on meaning representation (a criticism of the same kind as Weinreich's attack on Katz and Fodor—see section 9.3.5). The criticism was justified to the extent that these approaches did not make a very clear distinction between the information directly present in the text and information retrieved from the understander's background knowledge about the entities mentioned in the text. This criticism is valid when the program must apply all possible inferences to the results of the initial representation of text meaning and not when a clear objective is present, such as resolution of ambiguity relative to a given set of static knowledge sources, beyond which no more processing is required.

It follows that text meaning is, in this view, a combination of

- The information directly conveyed in the NL input
- The (agent-dependent and context-dependent) ellipsis-removing (lacuna-filling) information that makes the input self-sufficient for the computer program to process
- Pointers to any background information that might be brought to bear on the understanding of the current discourse
- Records about the discourse in the discourse participants' fact repository

Additionally, text understanding in this approach includes detecting and representing a text component as an element of a script/plan (in Schank-Abelson-Cullingford-Wilensky's terms—see Schank and Abelson 1977; Cullingford 1981; Wilensky 1983; see also section 7.1.5) or determining which of the producer goals are furthered by the utterance of this text component. We stop the analysis process when, relative to a given ontology, we can find no more producer goals/plans that can be furthered by uttering the sentence. But first we extract the propositional meaning of an utterance using our knowledge about selectional restrictions and collocations among lexical units. If some semantic constraints are violated, we turn on metonymy, metaphor, and other "unexpected" input-treatment means. After the propositional meaning is obtained, we actually proceed to determine the role of this utterance in script/plan/goal processing. In doing so, we extract speech-act information, covert attitude meanings, and eventually irony, lying, and so on. The extant implementations of ontological semantics make no claim about including all these features.

There is a tempting belief among applied computational semanticists that, in a practical application such as MT, the halting condition on representing the meaning of an input text can, in many cases, be less involved than the general one. The reason for this belief is the observation that, when a target-language text is generated from such a limited representation, one can, in many cases, expect the consumer to understand it by completing the understanding process given only partial information. Unfortunately, since, without human involvement, there is no way of knowing whether the complete understanding is, in fact, recoverable by humans, it is, in the general case, impossible to posit a shallower (and hence more attainable) level of understanding. To stretch the point further, humans can indeed correctly guess the meaning of many ungrammatical, fragmentary, and otherwise irregular texts—for example, Charniak's (1983a, 159) example of "lecture, student, confusion, question." This, however, does not mean that an automatic analyzer, without specially designed extensions, will be capable of assigning meanings to such fragments—their semantic complexity is of the same order as that of "regular" text.

## 1.3 Ontological Semantics and Nonsemantic NLP Processors

Ontological semantics takes care of only a part, albeit a crucial part, of the operation of the major dynamic knowledge sources in NLP: the analyzer and the generator. These processors also rely on syntactic, morphological, and ecological information about a particular language. Syntactic processing establishes the boundaries and nesting of phrases in the text and the dependency structures at the clause and sentence levels by manipulating knowledge about word order and grammatical meanings carried by lexical items. Morphological processing establishes the grammatical meanings carried by individual words, which helps the syntactic processor decide on types of grammatical agreement among the words in the sentence, which, in turn, provides heuristics for determining syntactic dependencies and phrase boundaries. The "ecology" of a language (Donald Walker's term) includes information about punctuation and spelling conventions, representation of proper names, dates, numbers, and so on.

Historically, the integration of all these steps of processing into a single theory and system has been carried out in a variety of ways. Thus, the meaning-text model (Apresyan, Mel'čuk, and Zholkovsky 1969, 1973; Mel'čuk 1974, 1979) dealt with most of these levels of processing and representation, sometimes at a finer-grain size than necessary for ontological semantics. However, that approach did not focus on semantic representation, and its computational applications (e.g., Kittredge, Iordanskaja, and Polguère 1988) did not address semantics at all, concentrating instead on deep and surface syntax and morphology. Conceptual dependency (Schank 1975) did concentrate on semantic representations but neglected to consider syntax or morphology as a separate concern: most of the application programs based on conceptual dependency (and all the early ones) simply incorporated a modicum of treatment of syntax and morphology in a single processor (e.g., Riesbeck 1975; Cullingford 1981). Ontological semantics, while concentrating on meaning, enters into a well-defined relationship with syntactic, morphological, and ecological processing in any application.

The most immediate and important element supporting the relations between ontological semantics and the nonsemantic components of an NLP system is the content of those zones of the ontological semantic lexicon entry that support the process of linking syntactic and semantic dependencies (see section 7.3). Specifically, what is linked is the syntactic dependency and the semantic dependency on clause and phrase heads. This essentially covers all words in a language that take syntactic arguments, which suggests that their meanings are predicates taking semantic arguments. The dynamic knowledge sources use this information to create and/or manipulate a text-meaning representation (TMR). The dynamic knowledge sources, however, also use morphological, syntactic, and other nonsemantic information in their operation.

## 1.4 Architectures for Comprehensive NLP Applications

The ideal state of affairs in NLP applications (as in all the other complex multi-module software systems) is when each component produces a single, correct result for each element of input. For example, a morphological analyzer can produce a single citation form with a single set of inflectional forms for a given input word. Thus, given the English *lain*, it produces "*lie*; Verb, Intransitive, past participle; 'be prostrate,'" while disambiguating it at the same time from "*lie* 'to make an untrue statement with intent to deceive.'"

Unfortunately, this state of affairs does not always hold. For example, given the Russian *myla* as input, a Russian morphological analyzer will (correctly!) produce three candidate outputs:

1. *mylo* 'soap'; Noun, Neuter, Genitive, Singular
2. *mylo* 'soap'; Noun, Neuter, Nominative, Plural
3. *myt* 'to wash'; Verb, Transitive, Past, Feminine

In context, only one of the multiple outputs will be appropriate. Conversely, the English morphological analyzer will (correctly!) fail to produce a candidate for the

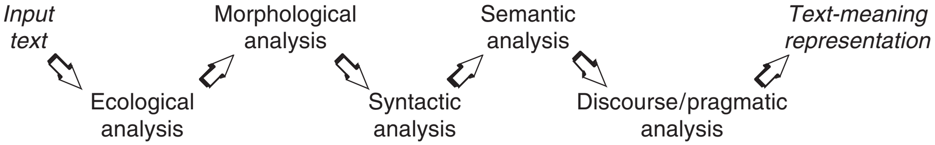

**Figure 1.1**
Stratified model I: Analysis. A schematic view of a traditional pipelined architecture for the analysis module of a comprehensive NLP system (e.g., an MT system). Results of each processing stage are used as input to the next processing stage in the order of application.

input string *mylo*, because it is not a word in the English language. Or, to use another example, a standard semantic analyzer for English will not be able to interpret the English phrase *kill the project* if the lexicon entry for *kill* (reasonably) lists its meaning as something like "cause not to be alive." Indeed, because projects are not living beings, the combination does not work.

The history of NLP can be viewed as the fight against these two outcomes: underspecification—that is, being unable to cut the number of candidate solutions down to exactly one—and failure to produce even a single candidate solution, due to overconstraining or incompleteness of static knowledge sources. The big problem is that it is difficult, if at all possible, to develop static knowledge sources (lexicons, grammars, and so on) with information that is correct in all contexts that can be attested in running text. Selecting an appropriate computational architecture is one of the methods of dealing with these difficulties as well as of improving the efficiency of the overall process.

### 1.4.1 The Stratified Model

The most widely used NLP system architecture conforms to the stratified model (see figures 1.1 and 1.2): the task is modularized, and the modules are run on a text one by one, in their entirety, with the cumulative results of the earlier modules serving as inputs to the later modules. This architecture has been a step forward compared to the early architectures, which were not modular in that they heaped all the processing knowledge together rather indiscriminately. (See, for instance, the early MT systems or the early AI NLP systems, such as Margie (Schank 1975).) One of the reasons for introducing the modularity is the difficulty of acquiring static knowledge sources for an "integral" system. Indeed, each of the analysis stages—the original ones of morphology, syntax, and semantics and the later additions of pragmatics, discourse, and, eventually, ecology—was (and still is) a complex problem that is difficult to study even in isolation, let alone taking into account its connections with other language analysis problems.

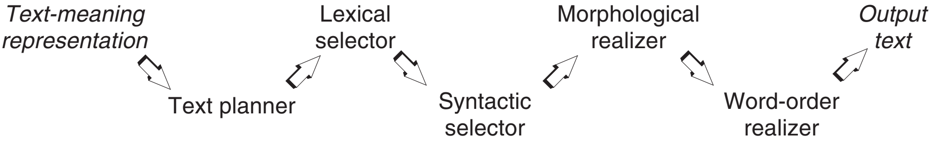

**Figure 1.2**
Stratified model II: Generation. A schematic view of a traditional pipelined architecture for the generation module of a comprehensive NLP system (e.g., an MT system). Results of each processing stage are used as input to the next processing stage in the order of application.

It is clear that this architecture was designed for processing without specification, overconstraining, or knowledge lacunae. Indeed, it presupposes that each module can successfully complete its processing before the later modules take over. While it was not clear what could be done architecturally to counteract possible overconstraining—or other reasons for a failure to find a solution for an element of input, such as lack of necessary background knowledge—modifications were introduced to the architecture to deal with underspecification.

The most prominent deficiency of the strictly pipelined architecture is the systematic insufficiency of knowledge within a single module for disambiguating among several output candidates. To try to alleviate this problem, the basic architecture can be modified by allowing underspecification of the outputs of individual modules, with the exception of the last one. Underspecification, then, essentially, amounts to postponing decisions of a particular module by allowing it to produce, instead of a single solution, a set of candidate solutions and subsequently using information obtained through the operation of later modules to filter this set (or these sets, if several instances of underspecification occurred). Figure 1.3 illustrates this kind of architecture for the case of text analysis.

### 1.4.2 The "Flat" Model

The stratified architecture of language processing is, in many ways, constraining. Thus, even in the model with feedback, such as that of figure 1.3, no use is made of the fact that findings of each of the modules can contribute to text-meaning specification directly, not necessarily through the operation of other (later) modules. In addition, the individual results from any module can contribute to determining more than one text-meaning element. Conversely, a particular combination of clues from a variety of sources may make possible the determination of a text-meaning element. None of the above is directly facilitated by the stratificational architecture. Underspecification may be difficult to implement efficiently because, in the simplest case, it necessitates carrying along possibly enormous amounts of intermediate data.

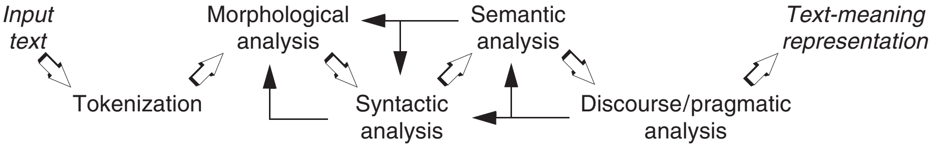

**Figure 1.3**
Stratified model modified: A schematic view of an enhanced pipelined architecture for the analysis module of a comprehensive NLP system (e.g., an MT system). Thin arrows represent knowledge from later modules that is used to disambiguate results of a prior module.

A "flat" architectural model (see figure 1.4) represents a swing of the pendulum back from pipelining but not back to the lack of modularity. In the flat module, all processing modules operate simultaneously, without waiting for the results of an "earlier" module—for example, the semantic analyzer does not wait until the syntactic analyzer finishes with an input element before starting to work on the latter. Of course, in isolation, the analyzer modules will not be able to complete their processing. However, they will succeed partially. For instance, morphologically uninflected words will be found in the lexicon and the set of their senses instantiated by the semantic processing module irrespective of the results of the syntactic analyzer. If only one sense is recorded for a word in the lexicon, this sense becomes a strong constraint that is used to constrain further the realization choices of other text components.

### 1.4.3 Toward Constraint-Satisfaction Architectures

One cannot rely on the partial successes of some modules in an unqualified manner. There are many real-world obstacles for a constraint-satisfaction process of this kind. First of all, lexicons can often be incorrect. In particular they may

- Contain fewer senses for a word (or a phrase) than necessary for a task; this state of affairs may cause the compositional semantic process of deriving text-meaning representation to fail because of overconstraining—the process may find no candidates that match the constraints specified in the meanings of TMR components with which they must combine; for example, if in a lexicon only the furniture sense is listed for the word *table*, the process will fail on the input *The two last rows of the table had to be deleted*
- Contain more senses for a word (or a phrase) than sufficient for a task; dictionaries compiled for human use typically contain more senses than should be included in the lexicon of an NLP system; thus, for instance, *Longman's Dictionary of Contemporary English* lists eleven senses of *bank*; should an NLP system use such a dictionary, it will have to be equipped with the means of disambiguating among all these eleven senses, which makes computation quite complex (see section 9.3.5)

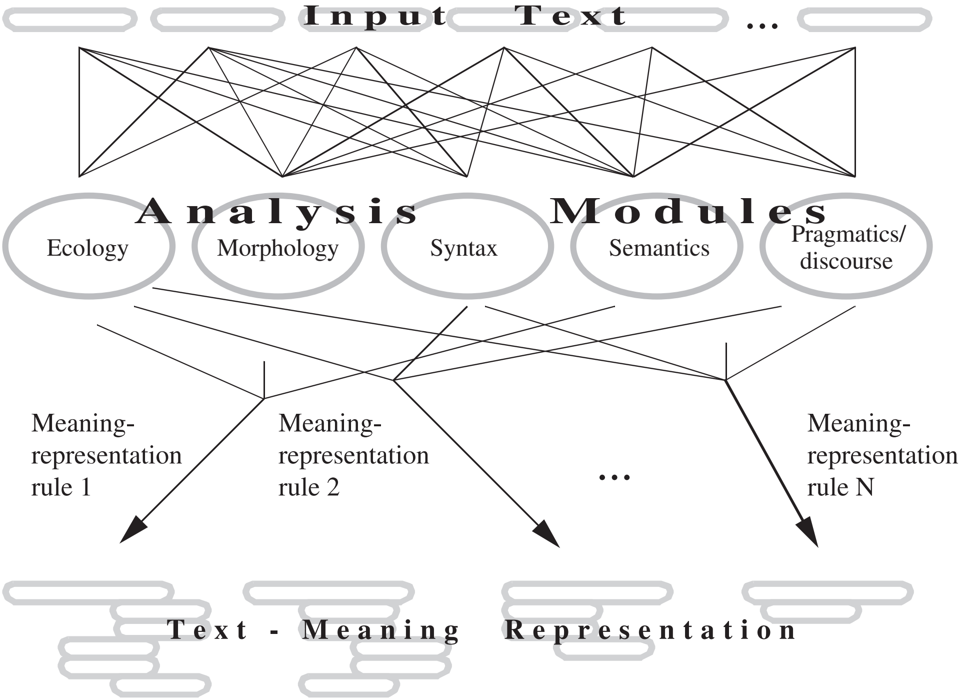
Input Text
...
Analysis Modules
Ecology
Morphology
Syntax
Semantics
Pragmatics/ discourse
Meaning-representation rule 1
Meaning-representation rule 2
...
Meaning-representation rule N
Text - Meaning Representation

**Figure 1.4**
In a "flat" model (illustrated here for the case of text analysis), all modules operate and record partial results simultaneously. An intrinsic ordering remains because "later" modules often need results of "prior" modules to produce results or to disambiguate among candidate analyses. However, partial results are still possible even if an earlier module fails on an element of input. The results of the individual module operations provide clues for the left-hand sides of meaning-representation rules. Robustness of the system is further enhanced if the rules are allowed to "fire" even if not all of the terms in their left-hand sides are bound (naturally, this relaxation must be carefully controlled).

- Incorrectly interpret the senses or provide incorrect, that is, too relaxed or too strict, constraints on co-occurrence

While the deficiencies of the lexicon are real and omnipresent in all real-size applications, much more serious difficulties arise from the preponderance in natural language texts, even nonartistic, expository ones, of nonliteral language—metaphors, metonymies, and other tropes. In terms of the basic compositional-semantic processing mode, nonliteral language leads to violations of co-occurrence constraints. Indeed, you do not really crush your opponent in an argument, or have the orchestra play the composer Bach (e.g., Ballim, Wilks, and Barnden 1991; Martin 1992; see also section 8.4.2).

One deficiency of the flat model, as sketched above, is that it does not benefit from the intermediate results of its processing, namely, from the availability of the nascent text-meaning representation. In fact, intermediate results of analysis—that is, elements of the nascent TMR—can provide reliable clues for the analyzer and must be allowed as constraints in the left-hand sides of the text-meaning representation rules. Thus, these rules can draw on the entire set of knowledge sources in comprehensive NLP processing: the lexicons, the ontology, the fact repository, the text-meaning representation, and the results of ecological, morphological, and syntactic processing. Pragmatics and discourse-related issues are folded in the semantic processing in current implementations of ontological semantics. This, however, is not essential from the theoretical point of view: a single theory covering all these issues can be implemented in more than one application module.

The modified flat model can be realized in practice using the so-called blackboard architecture (e.g., Erman et al. 1980; Hayes-Roth 1985), in which a public data structure, a blackboard, is used to store the results of each processing module in the system. This is one way to implement each module's access to the results of every other module (dynamic knowledge source) in the system. The blackboard also contains control and triggering mechanisms to activate certain processes once an item is posted on the blackboard. The actual control in blackboard systems usually uses the agenda mechanism. An agenda is a queue of knowledge-source instantiations (KSIs), each corresponding roughly to a rule—that is, a situation-action pair, where the situation is a combination of constraints. When all its constraints hold, a KSI can "fire" and produce some output to be posted on the blackboard. It is clear that manipulating the positioning of the various KSIs on the agenda (or using multiple-queue agendas) is, in this environment, the best method to improve the control behavior of the system. In fact, a significant amount of scholarship has been devoted to developing intelligent control strategies for blackboard systems, resulting in implementations of metalevel rules and control heuristics.

A different approach to the realization of the modified flat model consists in attempting to represent the entire problem as an interconnected graph of individual choices with constraints imposed on the co-occurrence of local solutions. A method has been developed of avoiding the need for manually constructed control heuristics once the above representation of the problem is achieved (Beale 1997). This method combines the idea of applying rules (KSIs) from any processing module, as soon as all the constraints necessary for their application are established, with the idea of underspecification. The KSIs will then be able to produce partial solutions in the absence of some knowledge elements necessary for producing a single result. As a result, an implicit ordering of KSIs is established automatically through the availability of constraints.

All the control methods specified above rely on two crucial assumptions about the constraints:

- All constraints are binary—that is, they either hold or do not hold.
- In a rule (a KSI), all constraints in the left-hand side must hold before the KSI can fire.

In reality, some constraints are "hard"—that is, inviolable (e.g., the English word *slowest* can only be a superlative adjective, while *uranium* refers exclusively to a chemical element). Some other constraints are "soft" or gradable (e.g., the constraint on the filler of the empty slot in the context *the city of* ______ may well be specified as "name of a city"; however, phrases like *the city of Charlemagne* or *the city of light* are quite acceptable, too—cf. McCawley's (1968) *They named their son something outlandish*).

An extension to the control structure may allow a KSI to fire at a certain stage in the process even if not all of the clauses among its conditions are bound or if one of these constraints only partially satisfies the condition. This requires making the processing architecture more complicated in two (interconnected) ways: first, by introducing a confidence measure for all decisions, and second, by developing procedures for relaxing the constraints based, among other things, on the confidence values of the knowledge used to make decisions.

The relaxation of constraints and the relaxation of constraint applications are evoked when the process detects an instance of overconstraining or an instance of residual underconstraining after all the modules have finished their processing. At this point, the general approach reaches its limit, for a given set of static knowledge sources. This means that finding an appropriate output in such a case can be entrusted to a completely different method, not inherently connected with the spirit of the main approach. In the case of residual lexical ambiguity, for example, many systems resort to selecting an arbitrary—usually the first—sense in the lexicon entry. Alternatively, a word that is more frequent in a corpus may be selected. All such solutions are, in a way, similar to tossing a coin. Such solutions are quite acceptable when a system based on an explanatory theory fails—not necessarily due to theoretical deficiencies but often because of the low quality of some elements in the static knowledge sources of the system. Indeed, more sophisticated corpus-based statistical techniques can be developed and used in these "emergency" cases. We believe that this is the best strategy for tightly coupled "hybridization" of NLP systems—that is, for using knowledge-oriented and corpus-based techniques in a single computational environment. Loosely coupled hybridization involves merging the results of the operation of rule-based and corpus-based systems on the same input. (See Nirenburg et al. 1994 and Frederking and Nirenburg 1994.)

## 1.5 The Major Dynamic Knowledge Sources in Ontological Semantics

The interplay of semantic and nonsemantic knowledge sources, as suggested in our general approach to NLP, is not, strictly speaking, necessary for the specification of the ontological semantic theory. But we believe that the division of labor and the application architecture we suggest are the most mutually beneficial for each module in an NLP system, because knowledge from a variety of modules must be included in the discovery procedures for the semantic processes. We also believe that it is not appropriate to omit references to syntax, morphology, and ecology while developing a semantic theory for the support of comprehensive NLP applications. It follows that the knowledge sources in our approach transcend purely semantic concerns. The following summarizes the components of the basic dynamic knowledge sources in our model.

### 1.5.1 The Analyzer

A comprehensive text analyzer consists of

- A tokenizer that treats ecological issues such as all special characters and strings, numbers, symbols, differences in fonts, alphabets, and encodings as well as, if needed, word boundaries (this would be an issue for languages such as Chinese)
- A morphological analyzer that deals with the separation of lexical and grammatical morphemes and establishing the meanings of the latter
- A semantic analyzer that, depending on the concrete NLP application, can contain different submodules, including:
  - A lexical disambiguator that selects the appropriate word sense from the list of senses enumerated in a lexicon entry
  - A semantic dependency builder that constructs the meanings of clauses
  - A discourse-level dependency builder that constructs the meanings of texts
  - A module that manages the background knowledge necessary for the understanding of the content of the text; this module centrally involves processing reference and coreference
  - A module that determines the goals and plans of the speaker, the hearer, and the protagonists of the text
  - A module that tracks the attitudes of the speaker toward the content of the text
  - A module that determines the parameters (indices) of the speech situation—that is, the time, place, identity and properties of the speaker and hearer, and so on
  - A module that determines the style of the text

### 1.5.2 The Generator

Text generators vary significantly, depending on the application. A major difference is the type of input expected by the generator, which, in turn, determines the kind of generation result possible. If the input to generation is a text-meaning representation, then the most natural generation task would be to construct a text whose meaning is similar to that of the input, in its entirety (e.g., for machine translation) or partially (e.g., for text summarization). If the input to generation is a set of knowledge structures corresponding to the state of a world, the generator is probably incorporated in a reasoning system and may be called on to create a text that analyzes the state of affairs for a human user. One kind of task that the generator may perform is to express the output in the form of the response to a human query. If, for example, the input is in the form of formatted numerical data, the generator is typically called on to present this data as a text (e.g., Kittredge, Polguère, and Goldberg 1986). If the input is a picture, the generator is typically required to describe it (e.g., McDonald and Conklin 1982). Text generators can include the following modules:

- A content-specification module that determines what must be said. The operation of this module, in its most general formulation, results in the specification of meaning of the text to be generated. The content-specification module sometimes includes
  - A communicative-function specification module that decides to include certain information based on the purposes of the communication
  - An interpersonal-function module that determines how much of the input can be assumed to be already known by the hearer
- A text-structure module that organizes the text meaning by organizing the input into sentences and clauses and ordering them
- A lexical selection module that takes into account not only the semantic dependencies in the target language but also idiosyncratic relationships such as collocation
- A syntactic structure selection module
- A morphological realizer for individual words
- The clause- and word-level linearizer

### 1.5.3 World-Knowledge Maintenance and Reasoning Module

In the framework of ontological semantics, world knowledge is contained in several static knowledge sources: the ontology, the lexicons, and the fact repository (see chapter 7). World knowledge is necessary for lexical and referential disambiguation, including establishing coreference relations and resolving ellipsis as well as for establishing and maintaining connectivity of the discourse and adherence of the text to the text producer's goals and plans.

Different applications use the static knowledge sources differently. While analysis and generation of texts are basic processes used in any application of ontological semantics, some applications require additional processes. In MT, analysis and generation account for most of the system processing because an MT system does not always need to use as much world knowledge as such applications as information extraction (IE) or question answering (QA). This is because the human consumer of MT is expected to fill in any implicit knowledge present in the output text, thus allowing some expressions that are potentially vague and/or ambiguous in the original text to "carry" over to the target text. Thus, while *good book* is potentially ambiguous in that it can mean a book good to read or a well-manufactured book (or any number of other things; see also Raskin and Nirenburg 1998 and Pustejovsky 1995 as well as sections 7.3, 8.4.4, and 9.3.5), the text producer's meaning is not ambiguous in any given instance. And the text consumer, due to the fact that it shares the same basic world knowledge with the producer, can readily recreate the intended meaning. Of course, errors of miscommunication happen, but they are much rarer than successful understanding, as is readily proved by the fact that miscommunication errors are regular subjects of amusing anecdotes. More scientifically (though less amusingly), this finding is sustained by the statistics of error rates in communication gathered by researchers in linguistic error analysis (Fromkin 1973).

In most cases, languages seem to be universally lenient with respect to being able to render vagueness and ambiguity, defined in this sense, either within a language or across languages. For example, in translation, one can in most cases retain deictic (*here*, *now*, *this*, and so on) or referential indices (*he*, *them*, *the same*, and so forth). MT can gloss over these cases unless an indexical mismatch occurs, as for instance, when a source language (say, English) does not have grammatical gender while the target language (say, Hebrew) does, forcing a choice of forms in the translation: the English *them* should be translated into Hebrew as *otam* (masc.) or *otan* (fem.), as required.

To make a decision in a case like the above, one must actually resolve referential ambiguity in the source text. In applications other than MT, this capability is much more necessary, because there is no expectation, for example, in information extraction, that the results of input text processing will be observed and further disambiguated by a human. The background world knowledge is the single most important basis for the disambiguation task. The more knowledge in the fact repository about remembered event and object instances, the higher the chances of the analyzer finding the quantum of information required for disambiguation. The above means that the prototypical ontological semantic system is a learning system. To enhance the quality of future processing, the results of successful text analysis are not only output in accordance with the requirements of a particular application but are also recorded and multiply indexed in the fact repository.

While MT can "go easy" on world knowledge, it still must extract and represent in the TMR every bit of information present in the input text. The situation with IE is different: it does rely on stored world knowledge, not only on the analysis of inputs, to help fill templates, but it does not typically pay a penalty for missing a particular bit of information in the input. This is because there is a realistic expectation that if that bit is important, it will appear in some other part of the input text stream where it would be captured. In other words, the grain size of the TMR varies somewhat depending on the particular application.

## 1.6 The Static Knowledge Sources

The static knowledge sources of a comprehensive NLP system include the following:

- An ontology, a view of the intelligent agent's world, including knowledge about types of things in the world; the ontology consists of
  - A model of the physical world
  - A model of discourse participants ("self" and others), including knowledge of the participants' goals and static attitudes toward elements of the ontology and toward remembered instances of ontological objects
  - Knowledge about the language-communication situation
- A fact repository containing remembered instances of events and objects; the fact repository can be updated in two ways: either as a result of the operation of a text analyzer, when the facts (event and object instances) mentioned in an input text are recorded, or directly through human acquisition
- A lexicon and an onomasticon for each of the natural languages in the system; the lexicon contains the union of types of information required for analysis and generation;[4] the information in entries for polysemic lexical items includes knowledge supporting lexical disambiguation; the same type of information is used to resolve synonymy in lexical selection during generation; the entries also include information for the use by the syntactic, morphological, and ecological dynamic knowledge sources
- A text-meaning representation formalism
- Knowledge for semantic processing (analysis and generation), including
  - Structural mappings relating syntactic and semantic dependency structures
  - Knowledge for treatment of reference (anaphora, deixis, ellipsis)
  - Knowledge supporting treatment of nonliteral input (including metaphor and metonymy)
  - Text-structure planning rules
  - Knowledge about both representation (in analysis) and realization (in generation) of discourse and pragmatic phenomena, including cohesion, textual relations, producer attitudes, and so on

## 1.7 The Concept of Microtheories

Decades of research and development in natural language processing have at least taught the practitioners that it is futile to expect that a single comprehensive theory can be developed to account for all the phenomena in the field. A realistic alternative may be to develop a society of *microtheories* responsible for manageable-size chunks of the overall set of phenomena. These components may be circumscribed on the basis of a variety of approaches. There may be microtheories devoted to language in general or particular languages; to parts of speech, syntactic constructions, semantic and pragmatic phenomena, or any other linguistic category; to world knowledge (ontological) phenomena underlying semantic descriptions; and to any of the processes involved in analysis and generation of language by computer.

Examples of microtheories include those of Spanish prepositions, of negation, of passive, of aspect, of speech acts, of reification of properties, of semantic-dependency building, and many others. The working hypothesis here is that it is possible to combine all these, sometimes overlapping, microtheories into a single computational system that accounts for a totality of language phenomena for which it is supposed to serve as a model. The number of microtheories, as described above, can be, of course, very high. In practice, it is necessary to determine which subset of such microtheories is the most appropriate for a particular task. At present, there is no formal mechanism for doing this, and simple common sense is used to keep the number of microtheories and overlaps among them to a possible minimum.

The microtheory approach facilitates the incorporation of fruitful ideas found in linguistics, computational linguistics, cognitive science, AI, philosophy of language, and corpus linguistics. Most linguistic descriptions are, in fact, microtheories, because they deal with fragments of the overall set of language phenomena. The difficulty of combining two linguistic descriptions to form a coordinated single description of the union of the phenomena covered by each individual description is well known and stems from differences in the premises, formats, and purpose. This creates the need to integrate the microtheories by providing a computational architecture that allows the joint operation of all the processing modules based on these microtheories in a particular NLP application.

The integration of microtheories can be carried out in several flat architectural models, for instance, using a blackboard system or a system similar to Hunter-Gatherer (Beale 1997). The nature of the process of adapting a microtheory to the formalism and control conditions of a computational system is illustrated in Pustejovsky and Nirenburg 1988 with the example of the microtheory of aspect and in Raskin and Nirenburg 1998 with the example of the microtheory of adjectives.

From the standpoint of processing architecture, an analysis-related microtheory is thus defined as a set of rules whose right-hand sides are instructions for filling a

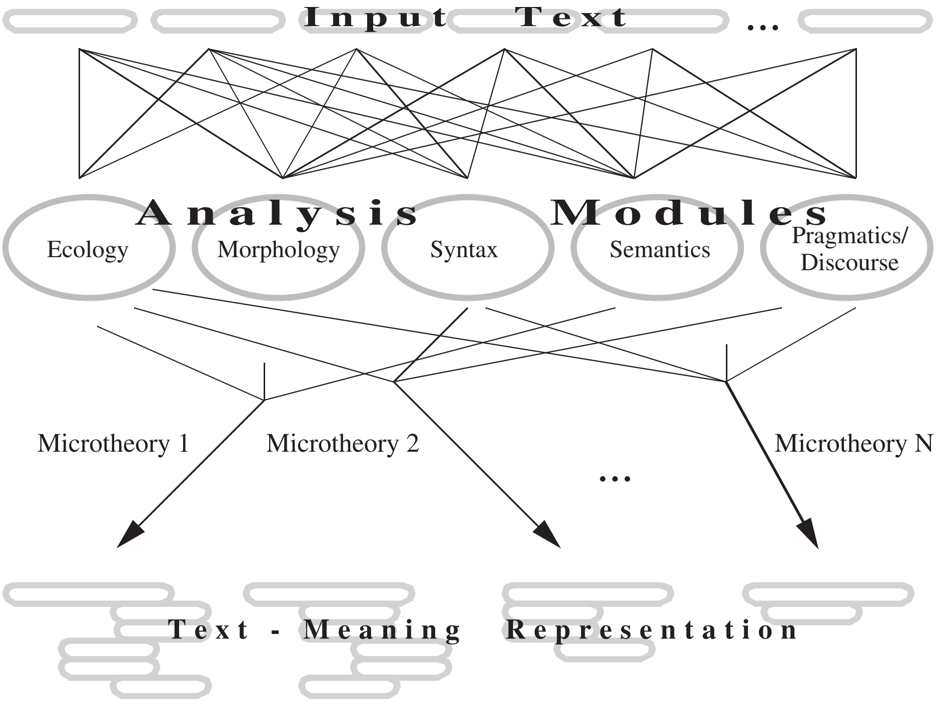

**Figure 1.5**
When meaning-representation rules are bunched according to a single principle, they become realizations of a microtheory.

particular slot in the TMR representation of a text. Figure 1.5 illustrates an architecture for combining a variety of microtheories. For instance, there might be a rule for each possible value of the PHASE slot in an ASPECT frame in a TMR. The left-hand sides of such rules contain a Boolean formula of a set of conditions for assigning a particular value of PHASE to an input clause derived from a variety of knowledge sources—the nascent TMR, morphology, syntax, semantics, pragmatics, or discourse information. Microtheories supporting generation include rules whose left-hand sides contain a Boolean formula of TMR values and prior lexicalization (and other text planning) decisions and whose right-hand sides include instructions for further lexical selection and other appropriate generation decisions.

## 1.8 Historical Record of Ontological Semantic Work

The ontological semantics project has been developed for almost two decades. Our understanding of the issues, the metalanguage, and applications have evolved over

**Table 1.1**
Ontological semantics-related projects

| Project name | Content | Dates | Key developers | References |
|---|---|---|---|---|
| Translator | Knowledge-based MT; original formulation | 1984–1986 | Sergei Nirenburg<br>Victor Raskin<br>Allen B. Tucker | Nirenburg, Raskin, and Tucker 1986 |
| Poplar | Modeling intelligent agents | 1983–1986 | Sergei Nirenburg<br>James H. Reynolds<br>Irene Nirenburg | Nirenburg, Nirenburg, and Reynolds 1985; see also Nirenburg and Lesser 1986 |
| KBMT-89 | Medium-scale KBMT, Japanese-English | 1987–1989 | Sergei Nirenburg<br>Jaime G. Carbonell<br>Masaru Tomita<br>Lori Levin | Nirenburg et al. 1991; Goodman and Nirenburg 1991 |
| Ontos | Ontological modeling and the original acquisition and maintenance toolkit | 1988–1990 | Sergei Nirenburg<br>Ira Monarch<br>Todd Kaufmann<br>Lynn Carlson | Monarch and Nirenburg 1987, 1988; Carlson and Nirenburg 1990 |
| SMEARR | Extension of Ontos; mapping of linguistic semantics into computational semantics | 1988–1991 | Victor Raskin<br>Salvatore Attardo<br>Donalee H. Attardo<br>Manfred Stede | Raskin, Attardo, and Attardo 1994a, 1994b |
| KIWI | Using human expertise to help semantic analysis | 1989–1991 | Ralf Brown<br>Sergei Nirenburg | Brown and Nirenburg 1990 |

| | | | | |
|---|---|---|---|---|
| Dionysus (including DIANA and DIOGENES) | An umbrella project including morphological, syntactic, semantic analysis; text generation and ontological and lexical knowledge acquisition | 1989–1992 | Sergei Nirenburg<br>Ted Gibson<br>Lynn Carlson<br>Ralf Brown<br>Eric Nyberg<br>Christine Defrise<br>Stephen Beale<br>Boyan Onyshkevych<br>Ingrid Meyer | Monarch, Nirenburg, and Mitamura 1989; Nirenburg 1989a, 1989b; Nirenburg, Lesser, and Nyberg 1989; Nirenburg and Nyberg 1989; Defrise and Nirenburg 1990a, 1990b; Nirenburg and Goodman 1990; Onyshkevych and Nirenburg 1991; Nirenburg and Levin 1991; Meyer, Onyshkevych, and Carlson 1990 |
| Pangloss | Another KBMT application, Spanish-English; hybrid system including elements of ontological semantics | 1990–1995 | Jaime G. Carbonell<br>Sergei Nirenburg<br>Yorick Wilks<br>Eduard Hovy<br>David Farwell<br>Stephen Helmreich | Nirenburg 1994; Farwell et al. 1994 |
| Mikrokosmos | Large-scale KBMT; Spanish, English, Japanese; first comprehensive implementation of ontological semantics | 1993–1999 | Sergei Nirenburg<br>Victor Raskin<br>Kavi Mahesh<br>Stephen Beale<br>Evelyne Viegas<br>Boyan Onyshkevych | Beale, Nirenburg, and Mahesh 1995; Mahesh and Nirenburg 1995; Mahesh, Nirenburg, and Beale 1997; Mahesh et al. 1997; Nirenburg, Raskin, and Onyshkevych 1995; Onyshkevych and Nirenburg 1995; Raskin and Nirenburg 1995; Mahesh 1996; Nirenburg et al. 1996; Beale 1997 |
| Savona | A mixed human-computer agent network for generating reports about emerging crises | 1997–1998 | Sergei Nirenburg<br>James Cowie<br>Stephen Beale | Nirenburg 1998b |
| MINDS | Intelligent information extraction | 1998–2000 | James Cowie<br>William Ogden<br>Sergei Nirenburg<br>Eugene Ludovik | Ludovik et al. 1999; Cowie et al. 2000a, 2000b; Cowie and Nirenburg 2000 |

**Table 1.1**
(continued)

| Project name | Content | Dates | Key developers | References |
|---|---|---|---|---|
| Expedition | Semiautomatic environment for configuring MT systems and language knowledge to support them | 1997–2001 | Sergei Nirenburg<br>Victor Raskin<br>Marjorie McShane<br>Ron Zacharski<br>James Cowie<br>Rémi Zajac<br>Svetlana Sheremetyeva | Nirenburg 1998a; Nirenburg and Raskin 1998; Oflazer and Nirenburg 1999; Sheremetyeva and Nirenburg 2000a, 2000b; Oflazer, McShane, and Nirenburg 2001 |
| CAMBIO | Mikrokosmos-lite | 1999–2001 | Sergei Nirenburg<br>Spencer B. Koehler | Nirenburg 2000a |
| CREST | Question answering | 1999–2000 | Sergei Nirenburg<br>James Cowie<br>Spencer B. Koehler<br>Eugene Ludovik<br>Victor Raskin | Nirenburg 2000b |
| MOQA | Question answering | 2002–2004 | Sergei Nirenburg<br>James Cowie<br>Tanya Korelsky<br>Marjorie McShane<br>Stephen Beale | |

this time. Earlier publications may present an outdated view on a number of points of theory and implementation. For the record, table 1.1 shows which NLP projects the various earlier publications on ontological semantics should be attributed to. In this book, we will refer to three implementations of ontological semantics: Dionysus, Mikrokosmos, and CAMBIO/CREST. These implementations represent the major stages in the development of the approach, dating from, roughly, 1992, 1996, and 2000.