The KERNEL Text Understanding System*

Martha Palmer
National University of Singapore

Carl Weir
Paramax Systems Corporation

Rebecca J. Passonneau
Columbia University

Tim Finin
The University of Maryland Baltimore County

April, 1992

Abstract

This article describes KERNEL, a text understanding system developed at the Unisys Center for Advanced Information Technology. KERNEL's design is motivated by the need to make complex interactions possible among system modules, and to control the amount of reasoning done by those modules. We will explain how Kernel's architecture meets these needs, and how the architectures of similar systems compare in achieving the same goal.

*This work was partially supported by DARPA Contract N00014-85-C-0012. Paramax Systems Corporation is wholly owned by Unisys.

Contact Information

Please address all correspondence to Tim Finin.

Martha Palmer
Department of Information Systems and Computer Science
National University of Singapore
10 Kent Ridge Crescent
Singapore 0511
email: mpalmer@iscs.nus.sg
..-65-475-1655, ..-65-738-1169 (fax)

Carl Weir
Paramax Systems Corporation
Valley Forge Labs R&D
70 East Swedesford Road
P.O. Box 517
Paoli PA 10301
email: weir@prc.unisys.com
215-648-2369, 215-648-2288 (fax)

Rebecca J. Passonneau
Department of Computer Science
Columbia University
500 West 120th St.
New York, NY 10027
email: becky@cs.columbia.edu
212-854-8089 (fax)

Tim Finin
Department of Computer Science
University of Maryland, Baltimore County
5401 Wilkens Ave.
Baltimore MD 21228
email: finin@cs.umbc.edu
410-455-3522, 410-455-3969 (fax)

Contents

1 Introduction

Natural language systems require information from several different sources to correctly analyze text. These sources include the lexicon, the grammar, the discourse context and common sense knowledge about the world. Consensus on the best architecture for supporting a constructive melding of these different knowledge sources has yet to be achieved. Some of the outstanding questions that need to be answered include:

- What common processing tasks do linguistic modules such as syntax, semantics, and pragmatics share with each other and with knowledge representation and reasoning that they need to cooperate on?
- What concepts do they share that facilitate communication?
- Do they make decisions at the same choice points?

In this paper we discuss these questions with respect to KERNEL, a text understanding system that combines a standard serial architecture for syntax/semantics interaction with closely integrated semantic and pragmatic processing. KERNEL, which is implemented in Prolog, is intended for use in generating database records from free narrative text [51]. Most of the processing components used in KERNEL were initially developed for use in the PUNDIT text processing system [26]. KERNEL differs from PUNDIT in having greater reasoning capabilities. We will begin by describing KERNEL's processing components and then discuss how the system's control structure facilitates the handling of particular linguistic phenomena such as nominalization, implicit arguments of verbs, and implicit references to times exhibited by past tense. The KERNEL control structure will then be compared to control structures in a number of systems that have been applied to the some of the same message corpora PROTEUS [21, 19, 20], TACITUS [31, 34] and CANDIDE [55, 54]. The examples we will use will principally come from three different message corpora from Navy ships: the CASREP domain, messages reporting equipment problems; the MUCK I domain, messages of ship sightings, and the MUCK II domain, messages of sightings of surface, subsurface and airborne vessels.[1]

In any natural language system the multiple knowledge sources, both linguistic and reasoning, share the task of producing a semantic representation for each linguistic unit, and integrating it into the system's model of the discourse context. We will argue in favour of an architecture in which syntactic, semantic and pragmatic tasks are segregated into separate

[1]In recent years, ONR and DARPA have sponsored conferences designed to evaluate and compare current message understanding technology. We refer to the message corpora used in the first two Message Understanding Conferences as MUCK I and MUCK II.

processing modules, but we will demonstrate the need for inter-module communication. We will also argue that many linguistic phenomena requiring interaction with knowledge representation and reasoning (KR&R) can be channeled through lexical semantics, rather than allowing syntax and pragmatics to deal with KR&R on an individual basis.

In §2 we concentrate on providing the details of how KERNEL produces a semantic representation for a linguistic unit through semantic interpretation of a canonical syntactic representation that explicitly represents grammatical relations like subject and object. By linguistic unit we mean a single predicating expression and its arguments. The decision that a particular phrase refers to an entity that can serve as a given argument of a predicating expression depends on pragmatic knowledge about the specific entities that have been referred to in a text, and on domain and world knowledge about the types of entities in the world. We describe the pragmatics modules in §3, followed in §4 by a discussion of the interface between *kernel*'s semantic and pragmatics modules and knowledge representation and reasoning modules. Then in §5 we use a particular message to exemplify how the semantic representation of each linguistic unit is dynamically integrated with the discourse context, illustrating the interaction of the three modules in the recovery of implicit information.

2 Producing a semantic representation

Here we describe KERNEL's syntax and semantics modules, illustrating their functionality with the example from the MUCK I domain in Figure 1.[2]

2.1 An overview of KERNEL

KERNEL performs natural language analysis in two stages: syntactic *parsing*, which has limited access to shallow semantic constraints for parse disambiguation, and integrated *semantic* and *pragmatic* processing, which has constrained access via a single interface—PKR—to external knowledge sources (cf. §4). PKR makes it possible to perform non-linguistic reasoning for various semantic and pragmatic tasks independently of any particular knowledge representation formalism. The grammatical relations specified by the parser serve as input to four separate modules which interactively perform semantic and pragmatic analysis: clause analysis (cf. §2.3.1), noun phrase analysis (cf. §2.3.2), reference resolution (cf. §3.1), and temporal analysis (cf. §3.2). As each linguistic unit is processed, the resulting representations are incrementally added to a temporary representation of the evolving discourse

[2]Figure 1 shows an actual MUCK I message from one of the several message domains that KERNEL has been applied to. The header information regarding the reporting and enemy platforms are excerpted from the actual headers. The paraphrase is not part of KERNEL's output or processing; it is included for the convenience of the reader not familiar with the naval sublanguage used in RAINFORM text.

Enemy platform: SUBMARINE
Reporting platform: VIRGINIA

VISUAL SIGHTING OF PERISCOPE FOLLOWED BY ATTACK WITH ASROC
AND TORPEDO. WENT SINKER. LOOSEFOOT 722/723 CONTINUE SEARCH.

Paraphrase:

> *Visual sighting of periscope [of submarine] [by VIRGINIA] followed by attack [by VIRGINIA] [on submarine] with anti-submarine rocket and torpedos. [submarine] went sinker, i.e, submerged. LOOSEFOOT 722 and LOOSEFOOT 723 i.e., helicopters, continue [their] search [for submarine].*

Figure 1: This Naval Rainform message is drawn from a corpus for the MUCK I domain, one of several that Kernel has been applied to. The header information regarding the reporting and enemy platforms are excerpted from the actual headers The paraphrase is not a result of Kernel's processing, but is included for the reader's convenience.

context. The clause analysis module controls KERNEL's semantic and pragmatic interpretation process. This module first attempts to associate the grammatical relations from the syntactic input with argument positions in a conceptual representation corresponding to the lexical entry of the current predicating expression, e.g., the matrix verb. Before the instantiation of an argument by a syntactic constituent can take place, the syntactic constituent must itself be semantically and pragmatically interpreted by the noun phrase analysis module working with the reference resolution module. Queries to PKR test semantic class constraints on the arguments of predicating expressions. The instantiated conceptual representations produced by clause analysis have correlate discourse referents that correspond to situations, and the time analysis module posts facts about the temporal and aspectual relations that exist among such situations [53]. After a sentence has been fully processed, the referents and relations produced by semantic and pragmatic analysis are added to the Integrated Discourse Representation. If a particular application is directed towards completion of a task such as filling database relations in a frame representation, control is then passed to the KR&R module for completion of this task.

2.2 Syntactic processing in KERNEL

Syntactic processing in KERNEL yields two parallel representations of a sentence: one is a detailed surface parse tree, and the other is a regularized structure called an *intermediate syntactic representation*, or ISR for short. ISRs are canonical representations of surface structure parse trees. Certain constituents of the parse tree serve as arguments in grammatical relations in the ISR. Thus, in an active sentence the subject will serve as an argument to a `subject` predicate, the object will serve as an argument to an `object` and so on for each grammatical role type. The ISR representation of predicate argument relations thus resembles *f-structures* in lexical functional grammar (LFG). *F-structures* consist of sets of attribute value pairs, including grammatical functions like subject and object, whose values are the lexical and morphological formatives from a phrase structure parse [6]. As in LFG, the ISR is the input the semantic component. In addition to providing a specification of available grammatical role represents tense and aspect information conveyed by verbal inflectional morphology as sentence operators. Finally, certain attachment ambiguities are reduced in the ISR: in particular, compound noun expressions are given a flat branching structure.

The grammar formalism currently used in KERNEL is called Restriction Grammar [30]. Restriction grammars consist of a set of context-free BNF definitions augmented by operations called *restrictions* that are used to enforce well-formedness constraints, and in some cases to apply optimization strategies for preventing unnecessary structure building. A *meta-rule* formalism is used to extend KERNEL's grammar to include rules and restrictions for processing a full range of coordinate structures and *wh*-constructions [27, 28]. Other syntactic phenomena treated by the current English grammar include questions, imperatives, sentence adjuncts, relative clauses, and a wide variety of nominal structures, including compound nouns, nominalized verbs, embedded clauses, and sentence fragments [41].

The compositional construction of the ISR for a clause is accomplished by associating each Restriction grammar rule with a corresponding rule that indicates how to construct the ISR for a parent node from the ISRs of its children [29]. The sentences from Figure 1 illustrate the information conveyed by the ISR, starting with the last sentence which is structurally the most straightforward. The pretty-printed ISR shown below has three types of elements: OPS, for temporal operators derived from the tense and aspect inflectional morphology on the verb, VERB, for the matrix verb of the clause, and a list of the syntactic arguments of the verb with a grammatical role predicate identifying each argument.[3] Note that the ISR representation of the coordinate noun phrase LOOSEFOOT 722/723[4] lists the conjunction *and*, followed by the individual conjuncts in which the head noun (*loosefoot*)

[3]The pretty printed ISRs shown here obscure certain syntactic details not relevant to the present discussion.

[4]The slash character '/' is a domain specific spelling of the conjunction *and*; *loosefoot* is a class of helicopter, of which the *722* and *723* are distinct types; the caret is a connective produced by lexical analysis of the input string.

and grammatical number (**singular**) of the noun phrase have been identified.

(1) LOOSEFOOT 722/723 CONTINUE SEARCH.

```
OPS:     present
VERB:    continue
SUBJ:    and (noun loosefoot^722 (sing), noun loosefoot^723 (sing))
OBJ:     search
```

The next sentence from Figure 1 shown here is a subjectless tensed clause. It exemplifies the most common of the five fragment types typical of message text (cf. [41] for discussion of the five types). The ISR constant **elided** fills the missing subject position and an appropriate filler for the subject argument of the verb will eventually be suggested by semantic and pragmatic processing.[5] To *go sinker* is treated as an idiomatic expression meaning to *submerge*. The morphological marking of tense on the first word in the idiomatic phrase is extracted and represented in the ISR as the operator **past**:

(2) WENT SINKER.

```
OPS:    past
VERB    go_sinker
SUBJ:   elided
```

The first sentence, shown below, is the most complicated. First the passive is regularized, placing the grammatical subject, *visual sighting of periscope* in the object position. The ISR constant **passive** is placed in subject position. The lack of tense is represented by another ISR constant, **untensed**. Since **by-pp's** are not always the logical subject in

[5]The other four fragment types are handled similarly, with the ISR supplying some of the missing information. A zero-copula fragment such as *disk bad* has the null verb replaced with tenseless *be*, as in *disk [be] bad.* An isolated noun phrase is given existential treatment, so that *failure of sac* becomes equivalent to *There was [a] failure of [the] sac*. The same treatment extends to fragments where both the subject and verb are missing. The isolated complement of an elided *be* verb, e.g., *inoperative*, becomes *elided [be] inoperative*, while the predicate *repairing engine* becomes *[elided] [be] repairing engine* [41].

passive sentences, the **by-pp** is left in its original position.

```
(3) VISUAL SIGHTING OF PERISCOPE FOLLOWED BY ATTACK WITH ASROC
    AND TORPEDO.

    OPS:    untensed
    VERB:   follow
    SUBJ:   passive
    OBJ:    gerund:  sight
               L-MOD: adj:  visual
               R_MOD: pp:   of
                            periscope (sing)
    PP:      by
               attack (sing)
               R_MOD: pp:  with
                           and (asroc (sing), torpedo (sing))
```

2.2.1 Syntax/semantics interaction

In the current version of Kernel semantically anomalous parses are filtered by a distinct module, SPQR [35], that uses interactively acquired word co-occurrence patterns. Queries about the semantic validity of partial parses are passed to SPQR at major phrase boundaries.[6] Unfortunately, since there is no integration between SPQR and semantic analysis, porting to new domains involves a fair amount of duplication of effort. In addition, SPQR is fairly rigid, and cannot automatically generalize about linguistic phenomena such as transitivity/intransitivity alternations. The semantic interpreter and the Lexical Conceptual Clauses described in §2.3.1 are not used to constrain the parse because of the disparity between the data structures used by the Restriction Grammar and those used for semantic interpretation.[7]

2.3 Lexical semantic interpretation

There are two distinct modules of the semantic interpretation process, clause analysis and noun phrase analysis. These are not wholly separated, since clause analysis is handled by a general algorithm for the interpretation of predicating expressions that applies to

[6] In one study, we found SPQR cut the average number of parses found per sentence from 4.66 to 1.45 [35].

[7] The semantic interpreter and its lexical rules are intended to comprise a modular system that could be linked with any grammar formalism and syntactic parsing mechanism. Adapting the semantic interpreter to perform SPQR's function would have been a one-time task, since the structures built by Restriction Grammar differ markedly from more widely known grammar formalisms.

noun phrases whose head nouns take arguments, e.g., derived nominalizations. In addition, there is a tight interleaving between the semantic and pragmatic phases of clause and noun phrase analysis. Processing a clause always involves processing the noun phrases that are arguments to the verb, and a noun phrase might include a relative clause as a modifier which would require clause analysis. Mutually recursive calls between the two analysis processes are dependent on the structures encountered in the ISR. This section first describes each module separately, and then gives examples of how they communicate with each other.

2.3.1 Clause analysis

The quest to fill in the arguments of the matrix verb provides the driving force for the clause analysis process. The lexical semantic structure of a verb is represented in the style of Jackendovian Lexical Conceptual Structures [32], [45]. Our formalism is somewhat different from Jackendoff's, however, as we use declarative logical representations expressed as Prolog clauses which are then executed during the semantic interpretation process. We refer to our verb representations as Lexical Conceptual Clauses (LCC's) but we see them as being close to Lexical Conceptual Structures in spirit. This is principally because of the emphasis we place on thematic roles, or relations, as components of conceptual structure, in accordance with Jackendoff [32].

The clause analysis implementation has two separate, but interrelated components: the interpreter that performs the execution of the semantic analysis process; and the Lexical Conceptual Clauses (LCC's) whose structure and content controls the semantic analysis process for each domain-specific lexical item. The only information the semantic interpreter has about the lexical item it is processing besides the word stem from the ISR is its part of speech. If the lexical item has an LCC, the semantic analysis process is begun, during which the interpreter follows the structure and content of the LCC. Each LCC must contain all of the information about a given lexical item that is necessary for semantic analysis. The result of the semantic analysis is a set of partially instantiated semantic predicates similar to a frame representation, a representational device typical of much work in lexical semantics [32, 45, 39]. To produce this representation, the semantic components share access to a domain model. The semantic components are designed to work independently of any particular model or representation language by relying on a single interface to all KR&R sources, as described in §4.

The clause analysis module makes specific requests of the other semantic and pragmatic modules at well-defined points. Specifically, noun phrase analysis and then reference resolution are called with a request for a discourse referent for a particular syntactic constituent every time clause analysis attempts to bind the constituent to the argument of a predicating expression. Time analysis is called after an LCC is instantiated to interpret the temporal

```
attack :- attackP(actor(A), theme(T), w_Instrument(I))
```

Mapping Rules:

```
Actor(A)          :-    subject (A) / predP(actor(X), Y)
Theme(T)          :-    obj(T)/ predP(X, theme(Y))
Theme(T)          :-    pp(on, T)/ attackP(X,theme(T),Z)
W_Instrument(I)   :-    pp(with,I) / impact_predP(X,Y, w_instrument(I))
```

Semantic Class Restrictions:

```
actor(A)          :-    class(platform_group, A) / X
theme (T)         :-    class(platform_group, A) / attackP(X,theme(T),Z)
w_instrument(I)   :-    class(weapon,I) / attackP(X,Y,w_instrument(I))
```

Figure 2: Lexical Conceptual Clause for ATTACK, with mapping rules and semantic class restrictions

operators in the ISR, as described in §3.2. LCCs are described in detail below.

Lexical Conceptual Clauses. The LCC rules that are used to bind the argument slots of conceptual predicates can be illustrated using the verb *attack*. Figure 2 shows the LCC rules that account for the following usages of *attack*:

intransitive: *Barsuk attacked.*

simple transitive: *Barsuk attacked Virginia.*

transitive plus with-pp: *Virginia attacked Barsuk with asroc and torpedo.*

transitive phrasal verb: *Texas attacked successfully on Adm Golovko with guns.*

The LCC for *attack* indicates that an Actor attacks a Theme with a weapon used as an Instrument.[8] The clause-analysis algorithm begins by finding the LCC associated with

[8]The thematic role names are used simply for purposes of clarity of exposition. The argument positions could also be labelled actant1, actant2, and actant3, but this makes it harder to keep track of which argument is being referred to. The difficulties inherent in finding a consensus on thematic roles has been documented in several places, including [48] and [15].

the verb. Then a single pass is made through the verb arguments, attempting to fill each one in turn. First, mapping rules are applied to select a syntactic constituent to fill a given argument slot. If the semantic properties of the referent of the constituent satisfy the semantic class restrictions, then the argument is filled, and the next argument can be considered. If there are no suitable syntactic constituents and the role is classed as *obligatory*, failure results immediately and backtracking occurs, possibly to an alternative lexical entry for the predicating expression. If there are no suitable syntactic constituents and the role is classed as *essential*, reference resolution is called to deduce a filler from the context. Finally, if the role is non-essential and non-obligatory, it is left unfilled and the system moves on to the next role. Before giving examples of the functioning of the clause analysis algorithm and a more complete discussion of the use of the obligatory and essential classifications, let us examine the mapping rules themselves.

Mapping rules. The mapping rules reflect Fillmore's intuitions about syntactic correspondences to semantic arguments first embodied in the notion of case [16, 45]. Some of these correspondences are quite general, while others may be specific to a semantic class or to individual lexical items. The W-Instrument in the *attack* LCC is an example of the former. It is a special case of the more general, classic Instrument role. The more general Instrument can always be introduced by either the **subject** as in *The hammer broke the vase*, (or a **by-pp**, if the sentence is passive), or a **with-pp**, as in *John broke the vase with a hammer*, which we conventionally represent (cf. Palmer [47]) as:

```
(4) break : - causeP(agent(A),
                      useP(instrument(I),
                           separate_into_piecesP(patient(P))).
```

The position of the thematic role in the LCC helps capture Fillmore's original intuitions about how cases are filled. He specified that if present the Agent would always fill the **subject** position, but if not present that position would be filled by the Instrument, if present. If the neither the Agent nor the Instrument are present, then the Patient can fill the **subject** position.[9] Notice that with *break*, the thematic roles occur in the following order, from left to right: Agent, Instrument, Patient. The clause analysis algorithm will try to fill the Agent first, then the Instrument and finally the Patient. The first mapping rules it will try for Agent, Instrument or Patient are general rules specifying that the **subject**

[9]Note that the Instrument can only be introduced by a **with-pp** if the Agent is the **subject**, and not if the Patient is the **subject**: * *The window broke with a bat.* The application of the **instrument(I)** $\leftarrow$ **with-pp(I)** rule must be constrained to a context where the Agent has already been filled. The ways in which the first thematic roles are filled place context-sensitive restrictions on how the remaining roles can be filled. These have to be captured by the predicate environment.

can fill that argument, i.e., **agent(A)** $\leftarrow$ **subject(A)**, **instrument(I)** $\leftarrow$ **subject(I)**, **patient(P)** $\leftarrow$ **subject(P)**, thus implementing Fillmore's specifications exactly.

However, this only hold for verbs that have what can be called classic Instruments, such as *break* or *shatter*, i.e, *John broke the vase, The hammer broke the vase, The vase broke.* Included in this class are Intermediaries which were introduced in [45] and that occur in **contact** verbs such as *connect*, and **support** verbs such as *hang.*[10] The Intermediary, often a *string*, acts as a classic Instrument that effects an indirect **contact** relation, and that can occur in the **subject** position [46].

Turning back to our *attack* W-Instrument, note that it is clearly distinguished from the classic Instrument, in that it can only appear as a **with-pp**, and not as a **subject.**[11] However, in spite of the syntactic difference, W-Instruments share with Instruments the conceptual property of being intermediaries that effect indirect contacts between two participants. The aim is to capture the syntactic differences without losing the conceptual similarities. *Attack* belongs to the **impact** class of verbs, which in this domain includes *attack and hit.* Both of these verbs can take weapons as W-Instruments (e.g., *hit with missiles, Virginia attacked Barsuk with asroc and torpedo*), and share the same mapping rules. The weapons can also appear as noun modifiers in the nominalized verbs (e.g., *sustained torpedo hit, heavy bomb attack*).

Mapping rules are general to a domain or specific to a particular verb or verb class, depending on the predicate environment. For example, the Actor mapping rule in Figure 2 is a general rule that can be applied to every Actor role that is the first argument of an LCC in this domain. This is indicated by the general "pred" relationship on the right hand side of the "/" with an Actor first argument. Similarly, the first Theme mapping rule can be applied whenever the Theme is the second argument of a predicate, as indicated by the predicate environment.[12] As mentioned above, the W-Instrument can only be filled by the

[10]The presence of the Intermediary distinguishes *connect* from *attach*, a similar **contact** verb. In *A particle is attached to the end of a string,* there is a direct **contact** between the *particle* and the *end of the string* implying that they are at the same **location**. In *A particle is connected to a particle by a string,* or *A string connects two particles*, the direct **contact** is between each *particle* and each *end of the string,* with the corresponding *location* implications. The *particles* themselves are in indirect **contact** with each other, by virtue of the string. This method of representing **contact** verbs with the corresponding **location** implications has also been recently adopted by [33].

[11]In accordance with Dowty's view of prototype Agents and prototype Patients, the W-Instrument could be characterized as having two contributing properties of a prototype Agent; 1) causing a change of state in another participant and 2) moving [15]. It also has two contributing properties of a prototype Patient, in that it probably undergoes a change of state itself and is causally affected by another participant. This would give it a fairly indeterminate status as neither a proper Agent, nor a proper Patient, which might explain why it is relegated to a third place position.

[12]This use of a sortal hierarchy to further specify the range of the mapping rules was implemented for a pulley word problem domain, (c.f., Palmer [45]), but was not used explicitly in the message domains in

with-pp. This is specified here by both the predicate environment which associates this rule with **impact** verbs, and the position in the LCC. Finally, there can be a verb-specific rule such as the rule for allowing an **on- pp** to indicate the Theme in sentences like *Texas attacked successfully on Adm Golovko with guns.* As indicated by the predicate environment, this somewhat odd usage would only apply to this domain specific use of *attack*.

To summarize, the ordering of the roles within an LCC and its predicate environment play equally important parts in constraining the application of mapping rules. Roles are filled in the order in which they appear in an LCC, which reflects the syntactic precedence of the possible fillers. This captures Fillmore's original intuition regarding the precedence ordering of *Agent* > *Instrument* > *Patient.* As we can see with *attack*, there can be other types of Instrument which cannot occur in the **subject** position, which have to be handled as specific to a particular verb class as indicated by the predicate environment and an alternative ordering of the roles.[13]

Semantic class restrictions. Semantic class restrictions are expressed in terms of a domain model. They vary more from verb to verb than the mapping rules do, although there are occasionally domain-specific general ones. For example, in the MUCK I message domain there is a general semantic class restriction on all Actors as well as Agents that they must be **platform_groups**. As already mentioned, the W-Instruments must be of type **weapon.** A Theme must also be a **platform_group**. The procedures which check semantic class restrictions must have access not only to the domain model, but also to the current discourse context, since they may have to check semantic properties of referents already bound to other thematic roles.

2.3.2 Noun phrase analysis

The main task of noun phrase analysis is to associate the indicated modifers with the head noun and pass control to reference resolution for determination of a likely discourse referent. Many head nouns are considered to be predicating expressions, and receive a treatment similar to that of clauses. The basic approach described above for clause analysis handles predicating expressions in a full range of syntactic environments, including noun phrases and modifiers. This section focuses on the differences in the interaction between the semantic and pragmatic modules occasioned by each different type of predicating expression, and

order to save processing time [46].

[13]For a detailed discussion of the theoretical aspects of the implementation of predicate environments, including the details of how the context sensitivities are preserved, and the effect of **Intermediaries**, see [45], [46]. In linguistics the most similar approach is currently termed Linking Theory, and a discussion of its status and unresolved issues can be found in Jackendoff [33].

discusses nominalizations in detail (cf. also Dahl [12]).[14] Some predicating expressions of different syntactic categories have similar LCC structures, such as the verbs *fail* and *monitor* versus the related nominalizations *failure* and *monitoring*, or verb/deverbal nouns such as *crack*.[15] Distinct mapping rules for the paired nouns and verbs reflect the syntactic category distinctions. Each multiply categorized predicating expression requires a customized version of the semantic analysis algorithm, as well as special LCC rules.

2.3.3 A distinct mode of operation for nominalizations

The distinct phases of semantic and pragmatic interpretation required for syntactically distinct incarnations of a lexical stem are triggered by recognition of its syntactic position in the ISR. The semantic analysis interpreter operates in any one of several modes depending on the syntactic position of a predicating expression. The mode determines which optional steps in the algorithm will be performed and in what manner. The mode also determines which set of syntactic mapping rules is relevant, and whether or not unfilled obligatory roles should cause failure.

Nominalizations are processed very similarly to their related verbs, in that they share the same LCC and semantic class restrictions. As would be expected, however, given that they are different parts of speech, they have different mapping rules [12]. Roles that appear as **subjects** of clauses will tend to appear as possessive determiners or **of-pp**'s. For example, *the Barsuk attacked* becomes *the attack of the Barsuk* or *Barsuk's attack*. Also, whereas the Theme may be a direct **object** for the clause, it is likely to appear as an **on-pp** or an **of-pp** in the noun phrase, as in *the attack on the Virginia, the attack of the Virginia*. A W-Instrument can also appear as a noun-modifier, as in *bomb attack* or as a *with-pp*. Note that, in the nominalized form, oblique roles such as the W-Instrument can share the same *pp* mapping rule that applied to the verb.

There are also differences in the control of the algorithm. Each argument in the LCC is still filled in turn, from left to right, but there are now two stages instead of one. Since modifiers are never obligatory for noun phrases, none of the thematic roles associated with a nominalization can be syntactically obligatory so they cannot cause failure. Secondly, because nominalizations may occur in anaphoric noun phrases, there are two separate role-filling stages in the algorithm instead of just one. The first pass is for filling roles with syntactically available constituents. Essential roles are left unfilled. If a nominalization is

[14] There are two types of nominalizations, 1) nominalizations which are formed productively, gerunds such as *monitoring*, and 2) nominalizations which are formed derivationally, such as *failure*.

[15] Although Chomsky's [7] seminal work on nominalizations points out the semantic differences between nominalizations and derivationally related verbs, using identical LCC structures was adequate for the message domains KERNEL has been applied to. The semantic differences between verbs and related deverbal nouns noted in Clark and Clark [8] were also irrelevant in our domains.

being used anaphorically, some of its roles may have been filled when the event it refers to was first mentioned. Thus after the first pass through semantic analysis, reference resolution is called to look for an antecedent referent. The anaphoric reference to the event via the nominalization automatically inherits previously mentioned or inferred role fillers as a by-product of reference resolution. For example, the clause *Texas attacked on Adm Golovko with guns and Virginia attacked on Barsuk with torpedoes* might be succeeded by *Gun attack was successful but torpedo attack failed.* In the interpretation of the second clause, two *attack* representations would be produced, one with a *gun* W-Instrument and one with a *torpedo* W-Instrument. During reference resolution, the *gun attack* is identified with the *Texas attack*, and the Actor and Theme roles are filled with the *Texas* and the *Golovko* roles respectfully, since they unify with the Actor and Theme roles from the *gun attack* in the first sentence. The *torpedo attack* unifies as well, and inherits the *Virginia* Actor and *Barsuk* Theme. After reference resolution, a second role-filling pass is made, where unfilled roles may yet be filled pragmatically with default values.

Since nominalizations have lexical aspect, the procedures in temporal analysis for computing temporal structure and for interpreting temporal adverbs apply to nominalizations, as described in §5.2. Since noun phrases never have tense but can be modified by locative temporal adverbs (e.g., *visual sighting at 1100 hours followed by attack with asrocs*), noun phrases headed by nominalizations undergo a modified version of the procedure for computing temporal location. Lexical declarations also identify verbs like *follow* and *continue* that provide further temporal information about their arguments.

2.4 Semantic interpretation of an example clause

The sentence *Texas attacked successfully on Adm Golovko with guns.* has the following ISR. We will see how the clause analysis process uses the LCC for *attack* and the mapping rules from Figure 2 to instantiate the argument slots of **attackP**.

```
(5) OPS:   past
     VERB   attack
     SUBJ:  Texas (sing)
     PP:    on
            Golovko (sing)
     PP:    with
            gun  (sing)
```

The first verb argument to be filled is the Actor. The **actor(A)** $\leftarrow$ **subject(A)** rule is applied.[16] Noun phrase analysis is called to produce a representation for *Texas*, the **subject**

[16] In a Prolog implementation, the execution of this statement causes the A argument of Actor(A) to be

of the clause. It recognizes *Texas* as a proper name of an instance of a **platform_group**, and in turn calls reference resolution (described in more detail in the following section) to assign a unique identifier—**texas**—to the referent of the noun *Texas*. (Reference resolution assumes that proper names are already unique identifiers.) For each discourse referent, the type and the identifier are represented in an **id** relation, e.g., **id(texas, texas)**. The Actor argument is instantiated with **texas**, yielding **actor(texas)**, and the semantic class restriction **platform_group** is applied. Since this succeeds, the argument is now filled and the interpretation process moves on to the second argument, the theme. The first mapping rule tries to map the **object** to the theme, but since there is no **object** this rule fails. The second rule, mapping the **object** of an **on-pp**, is then applied. The Golovko satisfies the semantic class constraint of being a **platform_group**, and the second argument is successfully filled. Our representation is now **attackP(actor(texas), theme(golovko), instrument(I))**. There is a **with-pp** available to fill the **instrument**, and its object satisfies the semantic class constraint of being a **weapon**, yielding the following representations for the Integrated Discourse Representation:

```
(6) id(texas, texas)
     id(golovko,golovko)
     id(gun,gun1)
     attackP(actor(texas), theme(golovko), instrument(gun1))
```

Note that reference resolution produced a unique referent, **gun1**, of type *gun* for the object of the **with-pp**.

2.5 Summary

This section has given the details of the production of the semantic representation. The algorithm for semantic analysis of verb phrases extended naturally to other predicating expressions such as nominalizations and participial modifiers. The different types of predicating expressions required variations in their interaction with syntax as well as with reference resolution and time analysis. The implementation consisted of a single interpreter that controlled the interaction between semantics and pragmatics for all predicating expressions, and which was tailored to the different requirements of the different types of predicating expressions. The folowing section describes the two pragmatics modules, reference resolution and temporal analysis. This is followed by a section giving an extended example of how the system integrates the semantic representations it produces into the discourse context.

---

instantiated with the A argument of subject(A). In the KERNEL implementation, this mapping is slightly less direct to allow for different types of referents for different types of noun phrases. However, for the purposes of this paper it is sufficient to think of it as an immediate instantiation.

3 Pragmatic analysis

As described in section §2, the goal of the lexical semantic interpretation process in KERNEL is to develop a conceptual representation of the semantic relations between predicates and their arguments. These conceptual representations are one component of complex objects, referred to as *situations*, whose other key component is the temporal information about when and how these relations have been asserted to occur. The three types of discourse entities represented in KERNEL are thus the situations that have been referred to, the times at which they occur, and the other types of discourse entities referred to by the arguments of predicates. The goal of the pragmatic modules is to instantiate these three types of discourse entities. Satisfying this goal involves cooperation among the semantic and pragmatic knowledge sources and procedures. For example, the referent of a referential noun phrase is assumed to be a specific discourse entity of a particular semantic type (cf. §2.3.2. Lexical semantic analysis of the head of a noun phrase generally yields its semantic type, although for *one*-anaphora and definite pronouns, the semantic type of the referent is determined by the semantics of the antecedent noun phrase. A focussing algorithm [9] determines whether a noun phrase is anaphoric, and if so, controls the search for the relevant discourse entity in the evolving discourse model. The generate-and-test strategy used by the focusing procedure is constrained by other knowledge sources, such as the domain model, the lexical semantic constraints associated with the argument position of the governing predicate, or the semantics and pragmatics of modifiers in the noun phrase itself. The domain model specifies the types of objects and relations that occur in the domain, thus can be used to support the inference that a newly introduced entity, e.g., *the periscope*, stands in a part/whole relation to a previously mentioned entity, e.g., *the submarine*. Control of the process of instantiating the discourse entities referred to by referential noun phrases resides with the reference resolution module, as described in §3.1.

As noted by Davidson [13], clauses share certain properties with referential noun phrases that suggest they also evoke discourse entities. Clauses serve as arguments to adverbs, verbs, and even nouns, as in *the fact that Matilda won the race*. The events they evoke can be anaphorically referenced in a subsequent sentence, as in 7).[17]

(7) The Clintons addressed the national TV audience. *It* helped his campaign.

In the spirit of Davidson's proposal, KERNEL explicitly represents the denotations of clauses as discourse referents.[18] As with the interpretation of referential noun phrases, the

[17] Such examples rarely occur in KERNEL's message domains.

[18] Subsequent to Davidson, many distinct logical representations that include a term for the referents of

instantiation of the discourse referents of clauses requires multiple semantic and pragmatic knowledge sources and procedures. The discourse entities corresponding to clauses, referred to as *situations*, are typed as states, processes and transition events, depending in part on the lexical aspect of the matrix verb. Initially, aspectual information was represented as part of the verb's lexical semantic decomposition [53]. In later implementations, the concepts corresponding to individual verbs and their aspectual classes were represented in a KL-ONE style knowledge base in order to take advantage of subsumption (e.g., **process isa situation**) and inheritance for reasoning about situations and their temporal structure [51]. Each typed situation entity consists of a conceptual relation derived from semantic analysis of the verb with its arguments, and a temporal argument representing the time for which the situation is asserted to hold. The temporal structure of a situation is derived from aspectual elements such as the lexical or grammatical aspect or grammatical aspect of the verb, and corresopnds to a particular type of temporal argument. The temporal location of a situation derives from the interpretation of tense and relational adverbs such as *before* and *after*, and constrains the specific temporal argument of a situation. Control of the process of deriving representations of situations and their temporal relations resides with the temporal analysis module, as described in §3.2.

In this section we describe the two pragmatics modules in more detail. Then in §5 we can complete the discussion of our example MUCK I message from the previous section, illustrating the cooperation among the various components. This will include an explanation of the recovery of implicit information. For example, in a sentence fragment such as *went sinker*, recovering the implicit argument of *go sinker* involves recognizing the missing syntactic subject, the thematic role it would have filled, and finding a discourse referent to fill that role. For such a simple past tense sentence with no temporal adverbs, recovering the implicit time when the *go sinker* event occcurred depends on finding a previously mentioned time in the specific discourse context that the event can be related to.

3.1 Reference resolution

Referents of noun phrases have a status similar to that of indeterminates in Situation Semantics in that they are place-holders for entities defined in a domain model [4]. When a noun phrase is suggested as an instantiation for an argument of a predicating expression, an attempt is always made to find its referent. At this point, semantic constraints on the referent—both those associated with the noun phrases modifiers and those associated with

clauses have been proposed. Like Barwise and Perry [4], and in contrast to Davidson [13], we do not treat the entity introduced by a clause as an argument of the verb. Moore [44] has suggested that elements of both approaches are required to handle certain kinds of adverbial modification, but we do not address these issues.

the thematic role of the Lexical Conceptual Clause—are available. It is the job of the reference resolution module to propose an appropriate referent. This will be tested against the semantic class constraints. If it fails the constraints, reference resolution is asked to propose an alternative referent, and this process continues until a referent is found that satisfies the semantic class constraints, or reference resolution runs out of alternatives.

KERNEL's reference resolution module is able to find referents for the following types of constructions:

- Pronouns (including zeroes, such as the unexpressed subject in *Replaced engine*) and *one*-anaphora, using a syntax-based focusing algorithm [9].
- Definite and indefinite noun phrases, as well as noun phrases without determiners found in telegraphic-style messages.
- Implicit associates such as *engine* and *pressure* in *Sac failure due to loss of oil pressure*,[19] where it is important to express the fact that the oil under consideration is the oil in the engine, not just any oil [9, 10].
- Conjoined noun phrases, where if the types of the individual conjuncts are different, the type of the conjoined set is the most specific supertype that is a generalization of each conjunct.
- Nominal references to events and situations first mentioned in clauses [12], such as *failure* in *Sac failed. Failure occurred during engine start.*
- Referents not mentioned explicitly [49], such as the investigated item in *Investigation revealed adequate lube oil.*

The reference resolution module maintains a list of referents in a *focus list* that is ordered on the basis of saliency [60],[9], [23]. The current implementation of the focusing algorithm considers the entire previous utterance to be the preferred potential focus. A previously mentioned pronoun would receive second preference, the direct object of the previous utterance would be in the third position, and the subject would be fourth. Any referents mentioned in prepositional phrases would be last. For a discussion of alternative strategies for ordering the focus list and related literature, see [11]. Using this strategy, after processing one of our example clauses, *Texas attacked successfully on Adm Golovko with gun,* the

[19] In the CASREP domain, a *sac* is a starting-air-compressor

focus list would contain the following discourse entities:

```
[[attack1], [texas], [golovko], [gun1]]
```

Golovko rather than the **on-pp**, i.e., *Texas attacked Adm Golovko successfully with gun*, the focus list would put the Adm Golovko in a more prominent position:

```
[[attack1], [golovko], [texas], [gun1]]
```

Referents for pronominal expressions (pronouns and elided elements) are selected from this list, as are referents for definite noun phrases. Domain-specific default antecedents may be established, and these are always attempted first for elided subjects. The domain-specific default antecedent is usually the message originator.[20] For example, in analyzing the sentence fragment *Replaced engine* in the CASREP domain, the default antecedent of the elided agent will be a referent denoting an abstract entity referred to as the *ship's force*. If the defaults fail the semantic class constraints, the focus list is examined.

3.2 Temporal analysis

Two issues in natural language understanding of tense and other temporal expressions demonstrate the need for close cooperation between natural language semantic and pragmatic processing, supported by general reasoning capabilities. First, there are numerous semantic and pragmatic interdependencies within and above the level of individual sentences, as we discuss elsewhere [51] [53] [52], and briefly review below. Second, committing to a specific temporal interpretation often requires general reasoning and a rich domain model. Here we characterize our approach to temporal analysis primarily in terms of the interaction between semantic and pragmatic modules, with knowledge representation and reasoning services provided through the medium of the PKR interface (cf. §4).[21] Given KERNEL's system design and representation of temporal information, the task of integrating with other knowledge tools, for example, to propagate temporal relations (e.g., Allen [1]) or to compute defeasible inferences,[22] would be a straightforward operation.

The semantic and pragmatic complexities of tense interpretation can be illustrated with the simple present tense. Identifying distinct uses of present depends in part on factors as diverse as the discourse intentions of the speaker or writer, or the lexical aspect of the

[20] Approximately 95% of the zero-subjects in our corpus of messages refer to the message originator.

[21] For example, PKR provides access to knowledge about the temporal aspect of verb meanings organized in a frame-based subsumption hierarchy.

[22] Cf. Lascarides and Oberlander [37] on the role of defeasible inference for tense understanding in discourse.

tensed verb. One use of present tense sentences, typical of directive discourse, is to refer to activities one needs to carry out in order to accomplish some goal, as in the following excerpt from a task dialogue (from Grosz and Sidner [23]):

(8) First, you remove the flywheel.

A present tense sentence can also be used to refer to a generic truth or definitional fact, as in the following excerpt from an explanation text (from Paris [50]):

(9) The telephone is a device that transmits soundwaves.

Or a present tense sentence can refer to a specific situation that is asserted to be true at the time the text is produced, as in the following CASREP-like sentence:

(10) The oil pressure is low.

The previous example contrasts with the use of present tense illustrated in 11), which will be discussed momentarily.

(11) The air pressure drops.

The examples in 8-10) are only some of many uses of present noted by Leech [38]. He presents a similarly broad range of uses of the simple past, and of the other components of complex tenses, such as perfect (as in present perfect, e.g., *the pressure has dropped*; or past perfect, e.g., *the pressure had dropped*). For a system to distinguish reliably between examples like 8)-10) would require, among other things, recognition of the distinct discourse goals of instruction versus explanation versus report text, which KERNEL does not do. On the other hand, the difference in interpretation of present tense in 10) versus 11) can be handled partly in terms of the meanings of individual words within the two sentences. In KERNEL, the interpretation of tense was designed to recognize such lexical semantic properties that constrain the interpretation of tense and temporal adverbs.

Despite the fact that both 10) and 11) are in the simple present tense, only the former refers to a specific present situation, in fact a state situation, whose type is directly derived from the stative aspect of the predicate *be low*. To parallel 10), the sentence in 11) with the event verb *drop* should refer to a specific present event. Intead, it would commonly be interpreted as referring to a generic state in which a **dropP** type event is asserted to have the property of occurring.[23] The difference between 10) and 11) can be traced largely to the difference in the aspectual meaning of the two predicates *be low* versus *drop*. Aspectual meaning here refers to the semantic component of lexical items that contributes to the determination of the temporal structure of the referent. With verbs whose lexical aspect is non-stative (e.g., *drop*), the present progressive would typically be used instead of the simple present in order to refer to a specific drop event (or other event) that occurs at the time the sentence is produced (e.g., *the air pressure is dropping*).[24] As noted in the introductory remarks of this section (§3), we assume that sentences like those in 10) and 11) evoke discourse referents that are situations. Lexical aspect helps determine whether the situation being referred to is a state (e.g., 10) and 12a), a process (e.g., 12b) or a transition event (e.g., 12c), which in turn constrains the interpretation of tense and temporal adverbs.

In KERNEL, as in most other systems or computational approaches to tense understanding (cf. [24] [63] [61] [57]), the simplifying assumption is made that tense is restricted to the uses found in simple narrative texts or reports, as in 10, 11, and 12. The basic algorithm for intra-sentential temporal analysis and its coverage have been presented elsewhere [53] [52]. Here a brief overview of this work will be given in order to explain how the situations referred to in simple sentences such as 12a-b) are computed and represented. Then we will discuss the implementation of inter-sentential temporal reasoning and its evolution from the algorithm for intra-sentential analysis.

An individual sentence may contain a simple or complex tense, and may have one or more tensed clauses. Here, tense is used to refer to past or present inflectional morphology on a verb. Future is assumed to be a modality that has present (*will*) and past (*would*) forms analogous to other inflected verbs, although the inflectional paradigms of modal verbs are otherwise restricted.[25] The auxiliary verbs *be* and *have* of the so-called complex tenses, such as the past progressive (e.g., *was* in *was dropping*) or present perfect (e.g., *has* in *has dropped*), are the tense-bearing elements of their verb phrases. The interpretation of tense is sensitive to certain temporal semantic properties of the verb, such as lexical aspect. In the case of the simple sentences illustrated in 12a-c), the input to temporal analysis consists of a partially interpreted ISR; cf. §2.2) That is, the verb and its arguments have been replaced by the conceptual representation (LCC) produced by the semantic interpreter and pragmatics modules, but the ISR retains the original ordered temporal operators produced by morpho-

---

[23]The aspectual classification of lexical items in KERNEL distinguishes situations into states and events, events into transition events and processes, and processes into bounded and unspecified processes [53] [52]. Note the typographical conventions observed here: italics for lexical items and English phrases; boldface for knowledge base concepts.

[24]This observation holds except in contexts having the special properties of sports news casts, e.g., *he steps up to the plate; he swings; he hits a home run.*

[25]E.g., modal verbs are uninflected for person and number, do not occur in the progressive or perfect forms, nor as heads of infinitival or gerundive phrases.

syntactic analysis of the verb phrase. Partially interpreted ISR's for the sentences 12a-c) are illustrated in 12a'-c').[26] The LCC for the verb phrase *be aloft* plus its subject argument *loosefoot* is **aloftP(theme([loosefoot3]))**. This representation remains in the scope of the ISR operator PAST, the temporal operator corresponding to the simple past tense, as shown in 13a).[27]

(12) a. *loosefoot was aloft.*
a'. `[PAST [aloftP(theme([loosefoot3]))]]`

b. *sighted periscope.*
b'. `[PAST [sightP(experiencer([loosefoot3]),theme([periscope9]))]]`
IDR relation: `partP([periscope9],[submarine14])`

c. *went sinker.*
c'. `[PAST [go_sinkerP(actor([submarine14]))]]`

The key characteristic of KERNEL's temporal analysis is a division of labor between the analysis of how situations evolve in time (their *aktionsart*), referred to here as temporal structure, and how situations are located in time, referred to as their temporal location. The former pertains to the number of intervals over which a situation is asserted to hold and properties of these intervals, such as whether they are stative or dynamic, and whether they have implicit endpoints. Temporal location pertains to the temporal ordering relations between a given situation and other known times, such as the time a report is produced, clock and calendar times mentioned in the sentence, or the times of other situations mentioned in the text. The semantic interpretation of tense in KERNEL is a modification of Reichenbach's [56] approach, which is based on relations of precedence or simultaneity among three temporal indices: speech time, event time and reference time. For our purposes, the discussion will be restricted to the two indices of speech time—the time a text or utterance is produced—and reference time. Here, reference time will be used to refer to the time of occurrence of the situation in the scope of the tense (Reichenbach's event time), as well as the anaphoric index for the interpretation of tense and inter-sentential temporal reference (Reichenbach's reference time). The representations of speech time, reference time, and

[26] As in §2, the ISRs are pretty-printed, obscuring certain irrelvant details.

[27] The ISR operators for complex tense forms with progressive or perfect are PROG and PERF. Sentence fragments with no tensed verb are represented as syntactically complete sentences with the special ISR constant, **untensed**, indicating the absence of tense. Because tense must appear on the first auxiliary or main verb if at all, and because perfect progressive is a possible complex tense (e.g., *had been dropping*) wheareas progressive perfect is not, the possible combinations of temporal operators in the ISR derived from a single tensed verb can be represented as: (PAST | PRESENT | NULL) X (PERF | NULL) X (PROG | NULL).

their interrelations comprise a point-based representation of time. However, the representation of temporal structure makes use of an interval-based representation, in particular, Allen's **meets** relation (cf. [1]), as noted below. Reference time plays a role in linking the two aspects of temporal interpretation, thereby yielding a mixed interval-based and point- based representation. The discussion of 12) illustrates how specifying distinct relations among reference time and the different situation types makes it possible to represent distinct interpretations of past for the various situation types using only a single semantic rule for past tense.

The interpretation of each simple past sentence in 12) results in distinct temporal structures due to the differences in lexical aspect between the predicates *be aloft, sight* and *go sinker*. The single rule for past specifies, in essence, that the reference time precedes the speech time.[28] However, each situation type has a distinct relation between its reference time and its full temporal structure. As representations of discourse referents for situations and times are computed, and the relations among them, they are added to the evolving representation of the discourse context, referred to here as the Integrated Discourse Representation (IDR). In 12a), the conceptual predicate **aloftP** is stative, thus the output situation is of type state, as indicated in the IDR excerpt shown in 13a). In effect, **aloftP** is a specializaton of **stateP**.[29] Situations are represented here as 3-place relations among the discourse referent index for the situation, the conceptual relation derived from the verb and its arguments, and the time for which the situation holds. The reference time of a situation is always of type **moment**. It is either the time argument of the situation, or it stands in a specified relation to the time argument. For a stative situation, the reference time is necessarily within its interval time argument, as illustrated by the relation

[28] The actual rule also specifies that event time and reference time coincide, which is immaterial to the present discussion.

[29] In earlier implementations, lexical aspect was represented as part of the LCC of a verb. Later, aspectual information was retrieved via the PKR interface from KNET taxonmies of situation types. All conceptual predicates represented in the knowledge base are indicated here by the affix - P.

includesP([period34],[moment34]) in 13a).[30]

(13) **a.** *loosefoot was aloft.*

situation:	`stateP([aloftP4], aloftP(theme([loosefoot3])), [period4])`
rt:	`reference_timeP([aloftP4], [moment4])`
rt relation to situation:	`includesP([period4], [moment4])`
rt relation to st:	`precedesP([moment4], speech_time)`
rt relation to other rts:	*none*

b. *sighted periscope.*

situation:	`processP([sightP9], sightP(experiencer([loosefoot3]), theme([periscope9])), [period6])`
rt:	`reference_timeP([sightP9], [moment6])`
rt relation to situation:	`hasP([period6], [moment6])`
rt relation to st:	`precedesP([moment6], speech_time)` `coincideP([moment6], [moment4])`

c. *went sinker.*

situation:	`transition_eventP([go_sinkerP13], go_sinkerP(actor([submarine14])), [moment8])`
rt:	`reference_timeP([go_sinkerP13], [moment8])`
rt relation to situation:	*identity with time argument*
rt relation to st:	`precedesP([moment8], speech_time)`
rt relation to other rts:	`precedesP([moment6], [moment8])`

Temporal analysis of 12b) and 12c) is analogous to that for 12a). Since **sightP** is of type process, the situation in 13b) is of type process.[31] Its reference time has an unspecified relation to its interval, represented by the relation **hasP**.[32] For 12c) a complex situation of type **transition_eventP** is created. A transition event implies the existence of two simple situations: a process leading up to the transition event, and the resulting state, each

[30] The symbols rt and st stand for reference time and speech time.

[31] Note that the periscope mentioned in 12b) is necessarily part of a submarine in this domain. This would be represented by a **haspartP** relation in the IDR relations for this sentence. Since a submarine is a type of entity that can go sinker, but periscope is not, the submarine evoked by the reference to the periscope then becomes the referent of the zero-pronominal subject of 12c). Cf. §3.1

[32] For full discussion of the various possible situation types and temporal structures, cf. [53].

with an associated interval.[33] The reference time of the transition event is the moment corresponding to the juncture of these two intervals; i.e., it is both the endpoint of the initial process and the onset of the resulting state. Due to the three distinct temporal structures of the situations in 12a-c), the full temporal structure of each situation in 13) has a distinct temporal relation to the speech time even though in all cases, past tense simply places the reference time before the speech time (cf., e.g., **precedesP([moment8], speech_time)** in 13c). For example, since the reference time of a state is entirely within the interval over which it holds, a state is assumed to extend indefinitely into the past and future of its reference time, in the absence of knowledge to the contrary. A past state could potentially be inferred to extend up to the present. In contrast, the reference time of the transition event in 12c) terminates a process of becoming submerged and initiates a resulting state of being submerged. The transition event situation is entirely in the past. However, the resulting state, whose onset is the reference time of the transition event, extends indefinitely into the future. The consequences of explicitly inferring the two phases of a transition event are particularly obvious in contexts with temporal adverbs that further specify a reference time. For example, if 12c) contained the temporal adverbial *at 5 o'clock*, KERNEL would explicitly represent that the submarine became submerged as of 5 o'clock but not before, and that the submarine remained in a submerged state for an unknown duration thereafter. While originally developed to handle such intra-sentential inferences, the representations presented here also support inferences about the temporal order among situations in different sentences.

A situation mentioned in one sentence of a text is often interpreted as occurring after a situation mentioned in the preceding sentence. In such cases, the sentence order is isomorphic with the temporal order of the situations they mention. This seems to be the default for event sentences in narrative text (cf. 12b-c). Other possible relations can be inferred, such as inclusion or overlap (cf. 12a-b); or the linear order of sentences may reverse the temporal order. However, in the report texts dealt with by KERNEL, the two most frequent possibilities, exemplified in 12), depend largely on the kinds of differences in temporal structure that KERNEL recognizes. States are generally interpreted as overlapping with or including a preceding or following event. The natural interpretation of 12a-b) is that the loosefoot helicopter was still aloft when it sighted the periscope. In contrast, a non-stative situation is generally inferred to occur after a previously mentioned non- stative (process or transition event). For example, the **go_sinkerP** transition event mentioned in 12c) is assumed to follow the **sightP** process mentioned in 12b). In other words, temporal progression of non-statives mentioned in distinct sentences is essentially isomorphic to

[33]The initial process and consequent state are not shown here. The term transition event is borrowed from Leech [38]; cf. Moens and Steedman [43] for a similar tri-partite structure of events.

the sentence order. KERNEL applies a simple algorithm based on these observations that determines inter-sentential temporal reference using the reference times of situations.[34]

The temporal inference that the **go_sinkerP** event mentioned in 12c) follows the **sightP** event mentioned in 12b) illustrates the anaphoric properties of reference time. Webber [63] has argued that reference times in successive sentences function analogously to anaphoric noun phrases, and that the same focussing mechanisms proposed for handling definite noun phrases (cf. e.g., Sidner [59]) apply to reference times. In discourses having a hierarchical segment structure, two sentences that are adjacent in a text may actually be parts of distinct segments. Webber's [63] temporal focus heuristics address the problem of relating reference times within and across distinct segments. Since the reports analyzed by KERNEL have no segmental structure, the algorithm for intersentential temporal reference was designed on the assumption that the reference time of a given sentence always serves as the antecedent for the reference time of the next sentence. The temporal relation between a reference time and its antecedent was operationalized along lines similar to the *Temporal Discourse Interpretation Principle* (TDIP) proposed by Dowty [14], which encodes the observation that time progressives in narrative text.

For sentences without temporal adverbials, the TDIP says that the reference time of a sentence S_i should be interpreted as *a time which immediately follows the reference time of the previous sentence* S_{i-1} (Dowty, [14]). However, the TDIP applies to all sentences, including statives. Dowty [14] believes that the discourse rules of English should *not make reference to the aspectual class of lexical verbs*. As a consequence, his proposal requires that the inference illustrated in 12a-b), namely that the event mentioned in 12b) occurs before the previously mentioned state ends, be handled by a commonsense reasoning mechanism that is independent of sentence level processing or rules for discourse anaphora. In KERNEL, a variant of the TDIP that is sensitive to distinct types of situations is used for computing different relations between a reference time and its antecedent, depending on the situation types involved. As shown in above, the rule has two parts: i) the reference time for a non-stative situation occurs immediately after its antecedent reference time, if the antecedent

[34] For complex sentences with multiple reference times, a single reference time will serve as the reference time for the whole sentence.

situation is also non-stative; ii) otherwise, the reference times co-occur.[35]

> **Discourse Tense Rule:** Where rt_i is the reference time of a situation entity Σ_i referred to by a sentence S_i, and rt_{i+1} is the reference time of a situation entity Σ_{i+1} referred to by an immediately following sentence S_{i+1}, then
>
> 1. if Σ_i and Σ_{i+1} are non-stative, then rt_{i+1} occurs immediately after rt_i;
> 2. else, rt_{i+1} co-occurs with rt_i.

There are two motivations for building the state/non-state distinction into the discourse rule for tense given above[36] First, the type of reasoning Dowty [14] attempts to avoid can be performed without determining the aspect of individual lexical items, and in fact, is the same type of reasoning required for anaphoric processing in general. That is, an antecedent and an anaphoric expression may evoke the same entity in the discourse model. Or, as Webber points out, they may evoke distinct entities that are inferentially linked, where the appropriate inferential relation *follows in large part from the 'ontology' of the specified entities*. Since the ontological type of the situations evoked by sentences is represented in KERNEL's IDR, and since subsumption relations among distinct types of situations are available via PKR, the discourse tense rule depends directly on world knowledge and the discourse model, rather than on individual lexical items.

The second motivation for making the discourse tense sensitive to the distinction between stative and non-stative situations has to do with the sorts of inferences required for understanding other texts like 12). It is easy to construct a text in which a state mentioned in one sentence is inferred to terminate upon the occcurrence of an event mentioned in a following

[35] In a collaborative effort with Megumi Kameyam and Massimo Poesio, Passonneau is currently developing a temporal centering algorithm for anaphoric uses of tense that accommodates all three possibilities.

[36] Hinrichs [25] has a similar pair conditions of conditions on his tense rule, distinguishing between event-type references (Vendler's [62] accomplishments and achievements) and the other *aktionsarten*. Hinrichs' rule applies prior to compositional semantic interpretation, during the construction of Discourse Representation Structures that handle various kinds of anaphoric and co-indexing relations. As noted by Dowty [14], there is a paradox in Hinrichs' proposal in that the aspectual distinctions that feed Hinrichs' rule cannot be computed prior to compositional semantic analysis. Hinrichs proposal is based on the assumption, disputed by Dowty [14], that intra-sentential and inter-sentential tense reference can be handled by the same mechanism.

sentence:

(14) **a.** The helicopter was aloft.
b. It was shot down.

An interpretation of 14) in which the helicopter is aloft until it is shot down is consistent with the discourse tense rule in that this reading requires the reference time of the shooting in 14b) to co-occur with some time T in which the helicopter is aloft. The reference time of being aloft is such a time, because it co-occurs with the reference time of the shooting by application of clause ii) of the discourse tense rule. Nothing in the temporal representation would block the inference, based on causal reasoning, that the reference time of being aloft also happens to be the termination of being aloft as a consequence of the shooting. While texts like 14) are easy to construct and find, it should be noted if the strict version of the TDIP is correct, there ought to be texts analogous to 12a-b) and 14) in which a state mentioned in one sentence is inferred to terminate prior to an event mentioned in the immediately following sentence. Such texts never occurred in KERNEL's report corpora, and may not exist.

For the two reasons just outlined, *kernel*'s discourse tense rule is preferable to Dowty's TDIP. Application of this rule results in the relations shown in italics in 13). For example, the reference time for the **aloftP** state evoked by 12a) is **[moment4]**. By clause ii) of the rule, this reference time (**[moment4]**) coincides with the reference time of the subsequently mentioned **sightP** process of 12b) (**[moment6]**); this relation is shown in 13b). Clause i) of the rule applies to the reference times in 12c) (**[moment8]**) and 12b) (**[moment6]**) because the respective situations are both non-statives. This yields the **precedesP** relation shown in 13c). Thus application of the discourse tense rule yields the inference that the **[sightP9]** event preceded **[go_sinkerP13]** event. Given the indeterminate relation of **[sightP9]** to its reference time, the time at which the submarine went below the surface could have occurred during, at the end of, or properly after the sighting of the periscope.

4 Knowledge representation and reasoning

Knowledge representation and reasoning in KERNEL is loosely based on the tripartite model popularized by Brachman, Fikes, and Levesque in the KRYPTON system [5]. The key feature in this architecture is the use of an interface language to insulate other processing components from the implementation details of the knowledge representation and reasoning modules. This interface language, called PKR in KERNEL, serves as a protocol for asserting what to include in representations of the information content of texts, and for

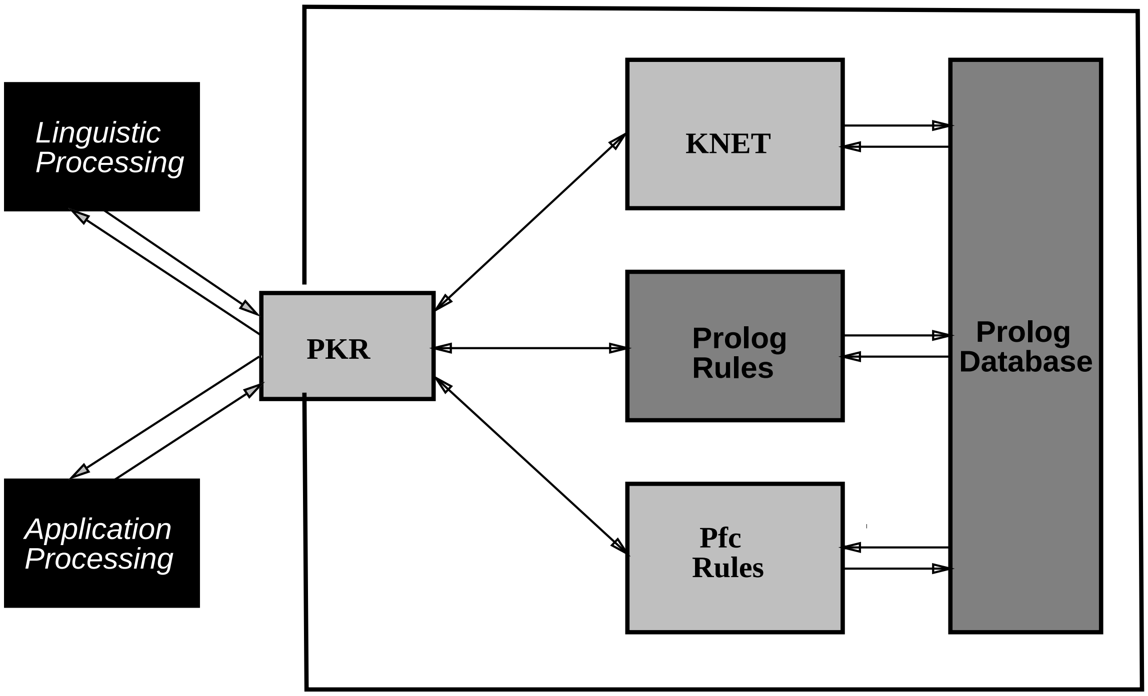

Figure 3: Kernel's current knowledge representation and reasoning system has four components: PKR provides an abstract interface; KNET is a terminological representation system, Prolog is used for some backward chaining, and Pfc provides a more flexible reasoning component with an integrated truth maintenance system.

asking queries about the current state of such representations [64].

The existing PKR protocol does not possess the expressive power needed for full text understanding, but it does provide adequate access to the knowledge representation and reasoning modules that KERNEL currently uses, shown in Fig. 3.

4.1 Concept definition

Concept hierarchies are defined in KERNEL using a representation language called KNET [18, 42]. This language was initially developed for use in a machine configuration system [58] and has more recently been used as the basis for a maintenance expert system.

KERNEL knowledge bases written in the KNET formalism do not typically take advantage of KNET's full functionality: primarily concepts (frames) are defined with only a few roles (slots), if any, specified. These concepts model the types of objects that natural language expressions may denote. As in other knowledge representation formalisms, concepts in KNET form a lattice. The most desirable features that KNET contributes are the easy specification and maintenance of concept definitions within a subsumption lattice and the efficient multiple inheritance of of roles and role constraints within this lattice. A problem with using KNET is that searching the transitive closure of the concept hierarchy is slow compared to implementations in terms of Prolog clauses. This is unfortunate because KERNEL's semantic interpretation algorithm frequently searches this hierarchy in order to determine that selectional constraints are satisfied. Consequently, in KERNEL the transitive closure of the concept hierarchy defined in KNET is compiled out — this results in very fast checks for constraint satisfaction, but at the cost of a significant increase in program space.[37]

4.2 Mapping lexical items onto concept definitions

Closely tied with the activity of building and maintaining a knowledge base is establishing the relationships between lexical items and concept definitions. In KERNEL, instances of the three-place predicate `denotes_concept` are used to define such relations. The three arguments of this predicate are the root form of a lexical item, a syntactic category, and a concept name. In the following example, the lexical item *Atlanta* is asserted to denote the concept `atlanta_C` whenever it functions as a proper name:

(15) `denotes_concept(atlanta,proper,atlanta_C).`

Note that the only properties of a lexical item used to distinguish it from all other lexical items is its root form and its syntactic category. It is assumed that if two lexical items share both of these properties, then it is reasonable to expect them to denote the same type of object. Presumably the appropriate granularity for assigning denotations will be determined through further empirical efforts.

It is possible to express lexical ambiguity by providing a lexical item with more than one concept mapping, which amounts to allowing two different `denotes_concept` clauses to have the same first and second arguments, but a different third argument. Common polysemous verbs like *be*, *get*, and *go* are assigned multiple mappings of this sort in KERNEL.

4.3 Reasoning about relations among concepts

In KERNEL, reasoning about relationships among concepts is done in terms of facts posted to the database of a forward-chaining system called Pfc [17]. Pfc is built on top of Prolog.

[37]Selectional constraint satisfaction is not as time critical in non-NLP applications such as maintenance expert systems.

Consequently, Pfc inference rules and the facts derived through their activation are Prolog terms asserted to the Prolog database. Pfc rules are of the following form:

(16) $A_1, \ldots, A_n \rightarrow C_1, \ldots, C_n$

where $A_1, \ldots, A_n$ on the left-hand side (LHS) of the arrow are *antecedent* clauses that express conditions on the firing of the rule and $C_1, \ldots, C_n$ on the right-hand side (RHS) of the arrow are *consequent* clauses that result in some action being performed, should the conditions of the LHS be satisfied.

The default action in Pfc is for the consequent clauses $C_1, \ldots, C_n$ to be asserted to Pfc's factbase (which is a subset of the Prolog database). However, clauses delimited by curly braces ({ }) are expressions that directly call Prolog. In this way, the efficient backtracking capabilities of Prolog can be utilized whenever they are appropriate. In KERNEL such escapes have been used, for example, to consult large databases of relatively static knowledge stored elsewhere.

Pfc makes use of a justification-based truth maintenance system (TMS). This TMS system permits the application of default reasoning, which is perhaps the most significant Pfc capability that has been taken advantage of in KERNEL. Default reasoning involves the inference of a given conclusion in the absence of any information suggesting a more suitable conclusion. Should a more suitable conclusion arise at some point in the analysis of a text, then the *default* conclusion will be retracted by the TMS.

Consider the following MUCK II sentence:

(17) *FRIENDLY CAP A/C SPLASHED HOSTILE TU-16 PROCEEDING INBOUND TO ENTERPRISE AT 35NM.*

KERNEL's knowledge representation and reasoning component is clever enough to know that the CAP A/C (combat air patrol aircraft) are associated with the Enterprise, since it can determine that the Enterprise is an aircraft carrier, and that aircraft carriers have combat air patrol aircraft assigned to them. This type of information may be stored in the concept definition for the Enterprise, where Enterprise_C is taken to be a rigid designator for a certain air craft carrier. However, an additional fact that can be encoded is that the combat air patrol aircraft assigned to the Enterprise are generally F-14s. Although this sort of *characteristic* information cannot be encoded in the concept hierarchy, at least not in the particular KB system currently used in KERNEL (KNET), it can be encoded as a Pfc rule. The basic idea is to assert that unless there is evidence to the contrary, assume that a CAP associated with the Enterprise is an F-14.

```
(18) cap(A),
     carrier_task_force(A,enterprise)
     \+(aircraft_type(A,Type) , {Type/==f_14})
     ==>
     aircraft_type(A,f_14).
```

This rule asserts that if some object A is a CAP, and is part of the Enterprise's carrier task force, then if no statement has been made about the type of aircraft A, except perhaps an assertion that A is an F-14, then assume that A is an F-14.

Note the use of the negated conjunction `\+(aircraft_type(A,Type),{Type/==f_14})` as one of the conditions on the left hand side of the rule prevents a situation in which an `aircraft_type` fact would be asserted, causing the TMS system to remove the justification for asserting the fact, thus causing the unfortunate effect of retracting the derived fact, thus causing the rule to fire again and assert the fact again, and so on. The braces around the term $Type/ == f_14$ inform the Pfc compiler that this is a simple call to Prolog and shouldn't invoke the TMS system.

Default reasoning of this sort is a handy technique in most text-processing applications, since such applications are generally already constrained to a relatively narrow domain in which characteristic relations among objects are common. A problem with default reasoning is that it takes advantage of the TMS's ability to retract facts posted to the factbase. Keeping track of facts and their justifications is a computationally expensive enterprise. If there are a large numbers of interdepencies among default values within a given application, then the TMS may begin to thrash in asserting and retracting activities.

5 Integration with discourse context

The message introduced in Figure 1 illustrates our point that much of the cooperation among linguistic and KR&R processing required for adequate understanding is facilitated by rich lexical semantic knowledge. The message exemplifies two types of implicit arguments, syntactically implicit and semantically implicit. Both types require cooperation among the semantic module and the reference resolution module. The message also illustrates the role of aspectual information for intersentential temporal reasoning, whose general mechanism was presented in §3.2. As noted in §2.3, MUCK I messages like the one in Figure 1, repeated here as Figure 4 or ease of exposition, include free text preceded by headers indicating the Reporting and Enemy platforms. The actual platforms are rarely mentioned explicitly, but are often referred to implicitly. The methods for indentifying and resolving implicit references are discussed in the following section (§5.1). The message can also be used to

Enemy platform: SUBMARINE
Reporting platform: VIRGINIA

VISUAL SIGHTING OF PERISCOPE FOLLOWED BY ATTACK WITH
ASROC AND TORPEDO. WENT SINKER. LOOSEFOOT 722/723
CONTINUE SEARCH

Paraphrase:

> *Visual sighting of periscope [of submarine] [by VIRGINIA] followed by attack [by VIRGINIA] [on submarine] with anti-submarine rocket and torpedos. [submarine] went sinker, i.e, submerged. LOOSEFOOT 722 and LOOSEFOOT 723 i.e., helicopters, continue [their] search [for submarine].*

Figure 4: This message from the MUCK I domain exemplifies both syntactically and semantically implicit arguments and requires cooperation among linguistic and KR&R processing required for adequate understanding.

illustrate the resolution of intersentential temporal reference described in §3.2.

5.1 Making Implicit Information Explicit

We have isolated two types of implicit information: syntactically implicit (i.e., missing syntactic constituents), and semantically implicit (i.e., unfilled semantic roles). The syntactic and semantic modules recognize implicit references to discourse entities so that reference resolution can be asked to determine them. In essence, part of the task of making implicit references explicit relies on the existing functionality of reference resolution. However, the manner in which reference resolution contributes to the resolution of implicit references differs for the two types of implicit information.

In order to recognize implicit references, syntax and semantics must each distinguish between optional and obligatory information. If syntactically obligatory arguments are missing, as in the case of sentence fragments (cf. §2.2), they are assumed to be implicit references. Syntactically elided arguments are recognized during the parse, and represented explicitly in the ISR by special constants (e.g., **elided**). An elided subject, for example, is treated as a zero-pronominal reference to an entity that will fill the same semantic role that would be filled by an overt subject NP. Adjuncts, and in many contexts, prepositional phrases, are syntactically optional. Thus, prepositional phrases with *by* and *with* can introduce references to entities filling Instrument or Agent roles, respectively. But such prepositional phrases are syntactically optional. The distinction between syntactically obligatory and optional constituents is quite general, and ports easily from domain to domain.

The assignment of thematic roles to obligatory, essential, and optional categories is more domain specific, and the same verb could have its thematic roles assigned differently in different domains. An obligatory role must be filled by a syntactic constituent via a mapping rule. An unfilled obligatory thematic role causes failure and backtracking to find an alternative set of mappings for that LCC, or an alternative LCC for that verb. If neither is available, the syntactic parser can be asked for another parse. Essential roles differ in that instead of causing failure, the lack of a role filler is assumed to represent a semantically implicit reference. When the semantic interpreter recognizes that no syntactic constituents are available to fill an essential role, it makes a special call to reference resolution. Reference resolution uses the semantic class restriction on the role to find a referent, somewhat analogous to the process of finding the referent of a noun phrase with no article, where the type is known. The referent may or may not have already been added to the discourse context (via explicit or implicit reference).

Essential thematic roles In order to present the details of the recovery of essential thematic roles, it is necessary to embed the discussion in a step-by-step description of how the entire message is processed. The first sentence in the text (cf. 4.) provides several examples of implicit semantic information. First, the headers are parsed and interpreted, resulting in the addition of entities representing the two platforms, the Virginia and the submarine, to the IDR. In fact, they are added to the focus list because in the RAINFORM message corpus that the example is drawn from, it can invariably be assumed that the Reporting_platform is the topic of the report and that the Enemy_platform is salient to the discourse. For our example message, the first sentence is thus processed in the context of an IDR containing the following information:

Focus list: `[[virginia], [submarine1]]`

The first sentence and its ISR are:

(19) VISUAL SIGHTING OF PERISCOPE FOLLOWED BY ATTACK WITH ASROC AND TORPEDO

```
OPS:    untensed
VERB:   follow
SUBJ:   passive
OBJ:    gerund:  sight
           L-MOD: adj:  visual
           R_MOD: pp:   of
                              periscope (sing)
PP:      by
           attack (sing)
           R_MOD: pp:  with
                              and (asroc (sing), torpedo (sing))
```

The LCC of the matrix verb, *follow*, is the following :

(20) *follow* ← `followP(theme(T),proposition(P))`

Following the order of roles in the LCC for *follow*, the semantic interpreter attempts to fill the Theme role first. The passive marker in the subject position allows semantics to quickly pick up the **by-pp** as a possible filler. Before the semantic class constraint on the Theme of *follow* can be applied, noun phrase analysis and reference resolution are called to interpret the noun phrase object of *by*, whose head noun is *attack*. Since *attack* is a nominalization, and since nominalizations have predicate/argument structure, the noun phrase analyser recursively calls the full semantic interpretation process that was called for *follow* (cf. §2.3.2). The LCC for *attack* is retrieved, and the argument instantiation process begins again with:

(21) *attack* ← `attackP(actor(A), theme(T), W_instrument(I))`

In general, nominalizations have distinct mapping rules from the corresponding verb form. For example, Actor and Theme roles are filled by noun-modifiers and **of-pp**'s rather than **subjects** or **objects**. Here, there are no consituents to potentially map to Actor and Theme. Although these are essential roles, during this pass the semantic interpreter will leave them unfilled and go on to the W-Instrument role. Here, the rule that maps a W-

Instrument to a **with-pp** happens to be the same as the verb mapping rule. The referent of the conjunction *asroc and torpedo* is instantiated as the filler of W_Instrument.[38] There are no previous mentions of attacks to help fill in the Actor and Theme roles, but since they are classed as essential roles they cannot be left unfilled. With nominalizations, it is only after reference resolution has searched for a previous mention of the referent that it is then asked to fill any remaining unfilled essential roles. Using the focus list produced by processing the message headers, reference resolution suggests that the Actor be filled by the reporting platform: **[virginia]**. It is accepted as the role filler because it satisfies the semantic class constraint of being a **platform_group**, so the semantic interpreter moves on to the Theme. Reference resolution then proposes that the Theme be filled by the Enemy_platform—**[submarine1]**—which satisfies the same semantic class constraint on the Theme that it be filled by a platform. In this fashion, the system produces the following instantiated LCC for the conceptual representation of the *attack*:

```
attackP(actor(virginia),theme([submarine1]),w_instrument([projectiles1]))
```

At this stage time analysis is called. As described in §5.2, the noun phrase headed by *attack* is recognized as referring to a process, and the following situation representation is produced:

```
processP([attack1],
         attackP(actor(virginia),
                 theme([submarine1]),
                 w_instrument([projectiles1]))
         period([attack1]))
```

Now that a referent, **[attack1]**, has been created for the noun phrase *attack with asroc and torpedo*, it can be used to instantiate the Theme role of *follow*.

The second role to be filled for *follow* is the Proposition role, and the likely filler according to the mapping rules is the subject noun phrase *visual sighting of periscope*. Here the head noun is the gerund form, *sighting*. Gerunds with noun phrase arguments[39] are handled much like nominalizations.[40] As with nominalizations, the mapping rules for nominal gerunds

[38] As noted above in §3.1, the semantic type of the referent of a conjunction is taken to be the first supertype that includes the types of each conjunct, which in this case is **projectile**.

[39] Prepositional phrases and possessive determiners occur with nominal gerunds, as in *they reported the virginia's sighting of the periscope*, whereas verbal gerunds have subject and object arguments analogous to verbs, as in *they observed the virginia sighting the periscope*.

[40] The only difference between nominalizations and nominal gerunds pertains to the left hand side of LCC rules. The left hand side of an LCC rule for a nominalization is the nominalization itself, which for a case

differs from those for verbs. The LCC for the lexical stem *sight* is[41]:

(22) *sight* ← `sightP(experiencer(E),theme(T),t_instrument(I))`

The Experiencer in the nominalized form can be filled by a possessive. Since there are none, the role is left unfilled in the first pass. The Theme can be filled by an **of-pp** so *periscope* is sent off to be analyzed. In this domain, a *periscope* is recognized by reference resolution to imply the existence of an associated referent, namely the **submarine** of which it must be a part. Consequently, reference resolution searches the discourse context for a previously mentioned periscope and a previously mentioned submarine [11]. After finding the submarine in the focus list, a **haspartP([submarine1],[periscope1])** relationship is added to the discourse model. The referent for the *periscope*, **[periscope1]**, fills the Theme role. The T_Instrument can be filled by a modifier, i.e., *visual*, which satisfies the semantic constraint of being of type **sensor**. Now that the first pass at role filling is finished, we have a partially instantiated LCC:

(23) `sightP(experiencer(E),theme([periscope1]),t_instrument(visual))`

A previous mention of a *sighting* is searched for, but not found. During the second pass, an essential role, the Experiencer still needs to be filled, so reference resolution is called with the semantic type **platform_group**. The **virginia** is still the first item in the focus list, and it satisfies this constraint, so our final representation is:

(24)
`sightP(experiencer([virginia]),theme([periscope1]),t_instrument([visual]))`

Time analysis treats a *sighting* as an process, and produces the following situation repre-

like *bombardment* (as opposed to *attack*) differs from the related verb (*bombard*). However, as shown above, the ISR of a nominal gerund represents the head noun as a lexical stem, stripped of the *ing* affix. In effect, the lexical unit available to the semantic interpreter is identical to that for any form of the related verb. Consequently, the very same LCC rule applies to the nominal gerund *sighting* as would apply to any form of the verb *sight*. It would thus be possible to provide distinct LCC rules for nominalizations and the related verb stems, a feature which could be exploited to capture the relative unpredictability in conceptual structure of nominalizations in contrast to gerunds. As noted above in §2.3.2, for the domain discussed here we opted to use the same LCC rules for nominalizations and their related verbs in order to take advantage of Prolog unification for filling in the roles of a nominalization if an antecedent clause could be found.

[41]The T_instrument, a tool instrument, such as *radar, telescope* or *visual*, is syntactically similar to the W_instrument, in that it cannot appear in the subject position.

sentation:

```
(25) process([sight1],
             sightP(experiencer([virginia]),
                    theme([periscope1]),
                    t_instrument([visual])),
             period([sight1]))
```

A referent for *sighting* event now being available, *[sight1]*, it can be used to instantiate the Proposition role of **followP**:

```
(26) followP(theme([attack1]),proposition([sight1]))
```

Elided syntactic constituents. The second sentence of the sample message illustrates the resolution of syntactically implicit information. The ISR for the sentence fragment, *went sinker* (cf. discussion of fragments in §2.2) is:

```
(27) OPS:   past
     VERB   go_sinker
     SUBJ:  elided
```

It is the task of semantics to assign a likely thematic role to the elided subject. This begins by retrieving the LCC for *go_sinker*:

```
(28) go_sinker ← submergedP(actor(A))
```

The first mapping rule for the Actor, `actor(A) ← subject(P)` indicates that the elided **subject** potentially fills the Actor role. The semantic interpreter thus asks reference resolution to instantiate the elided subject with a discourse entity. Reference resolution treats elided constituents very similarly to definite pronouns (cf. [41]). As a result of processing the first sentence, a number of new entities have been added to the focus list and it has been reordered. However, the first entity in the list that satisfies the semantic class constraint of being a **submerging_platform** is **[submarine1]**. Thus the final representation is:

```
(29) event([sinker1],
           submerged(actor([submarine1])),
           moment([sinker1]))
```

The preceding discussion illustrates in detail the cooperation among the semantic and pragmatic modules for filling in the two types of implicit arguments illustrated in the first two sentences. We have seen how semantically implicit information can be inferred even in the absence of syntactic cues by relying on the notion of essential roles. We have also seen that the semantic and pragmatic interpretation of syntactically implicit constituents is entirely analogous to the interpretation of actual constituents. It remains to briefly illustrate how temporal relations among events mentioned in distinct sentences can be inferred.

5.2 Instantiating Situations and Times

Temporal analysis of the example message illustrates the partial parallelism between noun phrases headed by nominalizations and clauses, as briefly noted in §3.2. Since nominalizations have lexical aspect, the procedures in temporal analysis for computing temporal structure and for interpreting temporal adverbs apply to nominalizations. Since noun phrases never have tense but can be modified by locative temporal adverbs (e.g., *visual sighting bf at 1100 hours followed by attack with asrocs*), noun phrases headed by nominalizations undergo a modified version of the procedure for computing temporal location. The aspectual component of temporal analysis queries PKR for the aspectual properties of the predicates of the LCC rules for *attack* and *sighting*, which are **attackP** and **sightP**. As a result, it selects the appropriate situation type and temporal structure as illustrated above. Since there are no temporal adverbial modifiers within either noun phrase, no further temporal analysis is performed. Lexical declarations in the knowledge based also identify predicates like **followP** and **continueP** that provide further temporal information about their arguments. In this example, the relative temporal location of the *attack* and the *sighting* mentioned in the first sentence of the example is given by the meaning of **followP**, thus the **sightP** process evoked by the subject precedes the **attackP** process evoked by the prepositional object. The temporal location of the two processes relative to the report time is given by the past tense inflection on the verb *follow*, thus both events occur prior to the report time. The reference time of the first sentence is taken to be the reference time of the first argument of the LCC representation, namely that of the attack.[42] The *attack* mentioned in the first sentence and the *go_sinker* situation mentioned in the subsequent sentence are both events, thus the first clause of the discourse tense rule given in §3.2 applies. As a result, the *go_sinker* event is inferred to follow the **attackP** event.

[42] The choice of reference time for such clauses should in principle be more flexible, but the implementation reflects the existing control structure and data structures available to the temporal analysis component. A more sophisticated treatment of such clauses would take a variety of discourse factors into account, such as the surface order of arguments, and would allow for alternative assignments of reference time, depending on the relation of a sentence to the discourse as a whole.

6 Issues requiring increased inter-module communication

What is striking about the solutions for the different linguistic phenomena as presented here is their simplicity. To a large degree, they make use of pre-existing techniques and modules. Extending the grammar coverage to include fragment types only required the addition of a single grammar rule for each fragment type. The rest of the grammar and the parser stayed basically the same. Extending the semantic interpreter from handling verbs to additional categories of predicating expressions did not change its basic structure. Shared processing techniques for the different predicating expressions such as verbs and nominalizations corresponded directly to their linguistic similarities. The differences in the interpretation process reflect the differences in their respective linguistic status, such as different mapping rules for the same lexical stem as in different parts of speech. Such categorial differences, which are reprsented explicitly and implicitly in the ISR, cause distinct interpretation procedures reflected declaratively in the different "modes" of the semantic interpreter. Extending reference resolution to the recovery of implicit information did not involve adding a major new module, but instead involved determining the correct contexts for calling the existing module. The lesson we derive from this is that explicit use of the available linguistic information for handling new phenomena can simplify the computational task rather than complicating it. This is only true when the relevant linguistic information is available to the modules when they require it, hence the importance of global data structures, such as the ISR.

We return to the questions we raised at the beginning of the paper. In examining *Which concepts do the modules share that facilitate communication?*, we have found that all of the modules need access to the semantic representations derived from LCCs, including which arguments are bound by the verb, and to the discourse entities. We maintain these representations in global data structures thus simplifying the interaction among modules. In asking *Whether or not the modules make decisions at the same choice points?*, the answer is that the choice points are basically provided by the syntactic structure. The syntactic phrase boundaries served as the primary choice points for interaction between syntax and semantics, or between semantics and discourse analysis. The similarity in choice points has been confirmed by the generality of the semantic interpretation algorithm where the same underlying framework can be used to control the processing of several different modules across several different types of predicating expressions. Using the binding of the predicate arguments to control the timing of finding referents has proven to be quite effective, for implicit entities as well as explicit entities.

There are other processing decisions, especially those that involve some form of syntactic or semantic ambiguity, that are best handled by a more flexible access to other linguistic and contextual information than is provided in KERNEL. Classic examples of syntactic ambiguities that can be resolved by access to a semantic module are prepositional phrase attachment and identification of part of speech. In KERNEL these are handled by predetermining the correct co-occurrence patterns for SPQR. An obvious extension would be to make more use of the semantic interpretation process for parse disambiguation, especially for prepositional phrase attachment. The semantic interpreter could quickly determine whether a particular prepositional phrase was a likely verb argument. It could not, however, recognize semantic arguments of the verb that appear as sentence adjuncts. This brings to light a limitation of the semantic analysis algorithm as currently implemented: its reliance on local syntactic structure. At any one time it only considers as potential arguments those entities which are within the scope of the syntactic phrase being analyzed. For instance, many types of events can take optional time, manner and place adjuncts that are not strictly considered part of the verb's subcategorization frame [33], [36]. Although certain types of temporal adverbs are handled, KERNEL does not currently have a principled method for handling the full range of adjuncts and quasi-arguments.

Raising the question of the utility of lexical semantic interpretation for parse disambiguation, raises the related question of whether pragmatic interpretation would also prove helpful. There are particular parsing ambiguities that are best resolved by access to the discourse context. It is traditionally the responsibility of the parser to assign the sentence type, but this cannot always be done on syntactic information alone. Certain clauses consisting of subject-less tensed verbs can often be either sentence fragments or imperatives:

(30) put new filter in starting air compressor

In the telegraphic style typical of our report domains, the preceding example could be a command to replace a filter, or a report that the filter has been replaced. Isolated noun phrases provide an additional example of the need for pragmatic input, since they can either be treated existentially or as answers to questions.

(31) clogged filter

That is, the above could be interpreted as an observation that one of the conditions *possibly* relevant to the events reported in a text is that *there was a clogged filter*. Or, in a report field that constitutes an implicit question regarding the causes of a machine failure, the same isolated noun phrase could be interpreted as supplying the questioned argument, *the cause of sac failure was a clogged filter.* A discourse module that can reason about relations among explicit and implicit speech acts, such as requests for information, could resolve these ambiguities under certain circumstances. First, the discourse module must be capable of recognizing which interpretation is favoured by the discourse context. Then it must make this information available to the parser at the appropriate choice point. Given the current KERNEL control structure, syntax never communicates with pragmatics directly, and certainly not during the parsing process. Syntactic information is passed to pragmatics via the ISR, but there are no choice points where syntax is encouraged to query the discourse context about the suitability of a particular sentence type. Nor is there an appropriate common vocabulary captured in a global data structure that can be used to pass this type of information back and forth. One of KERNEL's applications had a structured message format consisting of a series of answers to questions. An application dependent discourse module made extensive use of this knowledge for interpreting isolated noun phrases as response fragments by recognizing that the noun phrase replaces the questioned element in the question, thus yielding the semantic and pragmatic analysis of a full assertion. However, this was handled as a special case of the application, and there was no general mechanism for making discourse expectations available to the parser [3].

Any system under development will have gaps in its coverage similar to the ones that have just been mentioned, but we feel that what is required for KERNEL's next stage is more than just an increase in linguistic coverage. It also needs a more flexible control structure to allow that information to be made more widely available. At the moment semantics and pragmatics are highly interactive, processing each syntactic unit in tandem, passing control back and forth through mutually recursive calls. There should also be the same level of interaction with the syntactic parser, where the parser can pass control to semantics or pragmatics, along with partial parse information, at particular choice points. This will require a more flexible control structure for the individual modules, as well as new data structures for encapsulating the relevant discourse and situation information.

In the next section we will examine control structures in other text understanding systems, comparing them to KERNEL and keeping in mind this requirement for more flexible interaction between modules. We will discuss a particular example where KERNEL's almost deterministic control structure for reference resolution needs to be overridden, and discuss the benefits of an alternative control structure.

7 Comparisons with other systems

In this section we compare KERNEL to three other systems: PROTEUS, TACITUS, and CANDIDE. All four systems have similar goals in their emphasis on the utility of linguistic information and the desirability of building broad-coverage, general-purpose systems. We focus on comparing the different control structures for semantic and pragmatic analysis, and emphasize the following points which characterize KERNEL, namely:

- Communication between modules via shared data structures.
- A preference for localized processing whenever possible.

- Customized interaction between semantics and pragmatics for different types of predicating expressions.

We also point out some difficulties with the current KERNEL control structure, in that it can only override its preference for localized processing on a case by case basis. The CANDIDE system provides an example of an alternative control structure which is more flexible, and which, while it performs localized processing whenever enough data is available, can also automatically delay processing when there isn't enough data.

7.1 PROTEUS

PROTEUS [43] [20] is a text processing system developed at NYU under a DARPA contract jointly held with Unisys. PROTEUS and KERNEL, at that time in its PUNDIT incarnation, were built in parallel, with much trading back and forth of approaches, and consequently there are far more similarities in basic control structure between the two systems than there are differences. Like KERNEL, PROTEUS has the standard serial architecture, with separate modules for syntax, semantics and pragmatics. PROTEUS also uses a grammar based on the Linguistic String Grammar, although the parser is quite different. PROTEUS uses a Lisp implementation of a chart parser which proved especially useful for producing partial parses from fragmentary text. The ISR used by PUNDIT was originally based on work done by Mark Gavron at NYU which was also incorporated into PROTEUS. The approach to verb semantics for both systems is based on Palmer's Inference- driven Semantics, although the NYU semantic interpreter is implemented in Lisp, and is a simpler version of the KERNEL version. The PROTEUS verb representations tend to be richer and more complex than the KERNEL representations, and provided important inference information for MUCK II. PROTEUS emphasized common-sense reasoning from the beginning of the DARPA contract, whereas PUNDIT initially focussed on reference resolution and temporal analysis and careful integration of semantic and pragmatic processing. The systems are currently evening out these differences, with PROTEUS adding temporal analysis and the recent evolution of PUNDIT into KERNEL. PROTEUS has also made further progress than KERNEL in robustness achieved through the relaxation of semantic constraints on verbs which was especially useful for MUCK II [22].

7.2 TACITUS

TACITUS [31] is the text processing system developed at SRI under the same DARPA program that funded KERNEL. It has stressed the encoding of deep, general common sense

[43] PROTEUS is an acronym for PROtotype TExt Understanding System.

and domain knowledge as predicate calculus axioms that can be reasoned about using abductive inference. Abductive reasoning is identified with the notion of determining the best explanation for a given state of affairs. In practice within the TACITUS system, abductive reasoning involves assigning a cost to inferences that are made in determining the information conveyed by a piece of text. Chains of inferences that require more assumptions to be made will entail greater costs than chains of inferences that require fewer assumptions. Assumptions must be made when parts of an expression cannot be otherwise derived. The 'best explanation' is the one identified with the chain of inferences with the lowest cost. This approach is designed for handling ambiguities. It is particularly useful for difficult problems in component interaction such as the the inherent difficulty in determining when semantic constraints should be relaxed, as described in Section 6. If semantic constraints are being used to prune parses for a sentence such as *Shipyard replaced engine*, and they fail, it is appropriate to relax them and allow *shipyard* to be coerced into *shipyard worker*. However, given *temperature believed contributor to engine failure*, the desired behavior is the rejection of the active parse based on the failure of semantic constraints, and a search for an alternative parse. By following alternative inference chains, and assigning respective costs, TACITUS can eventually determine the best course in both of these cases.

KERNEL and TACITUS are the two opposite ends of the spectrum in terms of control of interaction between components. KERNEL has a tightly controlled paradigm of interaction between modular components (syntax, semantics, pragmatics, knowledge representation); TACITUS is aimed towards a uniform representation of all of the information as axioms that serve as input to a general, abductive theorem prover. The general approach embodied in KERNEL has been to make decisions as deterministically as possible, and make special provisions to allow more flexible control only when it has been demonstrated that the deterministic solution fails. Since the deterministic solution succeeds most of the time, it makes the KERNEL implementation especially efficient. We are committed to determining exactly what linguistic information is useful for what types of processing decisions, and making sure it is available at the necessary points. However, as described in the next section, we are faced with having to predefine special cases that will need more alternative processing, and would benefit from a more general control structure. On the other hand, one of the inherent problems in the use of abduction in TACITUS is its tendency towards explosiveness with the resulting high computational cost. To control the abduction in TACITUS it will be necessary to make use of heuristics based on the same kind of linguistic information that KERNEL is already using. The only way to find the heuristics is to experiment with tighter control structures in the way that KERNEL does. Eventually KERNEL and TACITUS will draw closer together. KERNEL's current almost deterministic control strategy (with backtracking for semantically anomalous parse, ambiguities, metonymy, etc.) will evolve into a set of heuristics for controlling a more flexible control structure. The general approach of TACITUS will eventually make use of a set of heuristics tagged to particular categories of linguistics knowledge (syntax, semantics, pragmatics).

In spite of claims made about the generality of the TACITUS approach, and how different types of information can all be applied at any point in time, in fact TACITUS is itself still fairly dependent on a standard serial architecture at the crucial junction between semantics and pragmatics, with the classic conflict between representation of linguistic information and representation of knowledge base information [2]. TACITUS uses the Dialogic system to produce its syntactic and semantic representation, without any reference to pragmatic and common sense information during the parse. This logical representation is passed on to the theorem prover for validation, and this is where the pragmatics, etc. come into play. In order to use syntactic and semantic information for reference resolution, which is done by the theorem prover, TACITUS is faced with the issue of having to pass the linguistic information used by Dialogic along to the theorem prover in the form of axioms. It is not surprising that reference resolution in TACITUS relies very heavily on pragmatics rather than syntax and semantics. The question is whether or not that is the most efficient and natural way to resolve references.

One of the aims behind the implementation of KERNEL has been the discovery of exactly those points in the processing that can most benefit from interaction, while concentrating on segregating knowledge sources in as modular a fashion as possible for portability purposes. We also see the need for powerful knowledge representation and reasoning capabilities, and agree that abduction has an important role to play. Pfc supports abduction through incorporation of a tightly integrated *truth maintenance system* and offers one kind of abductive reasoner in its *conflict resolution mechanism* [17]. However, our intention is to limit the application of abduction to those areas which cannot be resolved using other techniques.

7.3 CANDIDE

CANDIDE, like KERNEL, has a standard pipeline architecture for syntax/semantics interaction. CANDIDE's parser is more noncommittal than KERNEL's, and produces analysis trees that are neutral with respect to context-dependent attachment decisions. For example, all prepositional phrases are attached low and right, compound nominals are bracketed to the right, and quantifiers are left in place. CANDIDE's semantic and pragmatic interpretation component traverses the syntactic representation proposed by its parser, building an interpretation for each syntactic constituent represented in the structure after visiting and interpreting all of its subconstituents. The representations have two parts: a component referred to as the `sense`, which is intended to represent whatever information is in the context in which it happened to be uttered; and a representation referred to as the `assumption list` that contains a list of constraints on how the sense of the constituent may be extended on the basis of whatever inferences happen to be made in the discourse situation. For a noun, these constraints include determiner information and modifier information. Semantic rules determine the interpretation of a constituent by computing its sense based on the senses in the interpretations of its subconstituents, and by taking the union of the lists of assumptions of its subconstituents, possibly adding additional constraints to the resulting set.

After a semantic rule has been executed to determine the representation for a given constituent, the interpreter may examine the list of assumptions in the representation and decide to evaluate, i.e., discharge, some. The process of selecting assumptions for evaluation is, in principle, non-deterministic, although in practice there are guiding heuristics. In general the assumptions are discharged as soon as possible, in other words, as soon as enough semantic information is available. Additional heuristics impose syntactic boundaries on how long the discharge of assumptions can be delayed. Some can be delayed until the end of a noun phrase, some until the end of a clause and some until the end of a sentence. The discharge of an assumption is likely to affect the sense of the constituent in addition to making changes in the representation of the discourse context. From a practical point of view, it is during the discharge of these assumptions that a referent for a noun is determined. Many of the assumptions consist of the semantic class restrictions, and, as in KERNEL, they must fit the proposed referent. The ability of the interpreter to discharge assumptions in different orders is made critical use of in expressing ambiguities like variations in quantifier scope. This represents the most important difference in control structure from KERNEL, since KERNEL takes the opposite stance, that assumptions, (constraints), are discharged immediately. There is no general mechanism for delaying this discharge, although in special cases the normal control structure can be overridden [49].

CANDIDE's ability to allow the availability of semantic information to determine when referents are instantiated will be especially useful for interaction with syntactic parsers. This can best be illustrated by discussing the difficulties that arise due to KERNEL's more deterministic control structure.

We have experimented with a version of KERNEL which has integrated the syntactic and semantic processing, with the semantic interpreter being called at the end of every verb phrase and noun phrase. Initially this version simply used semantic constraints to reject semantically anomalous parses, but a more recent version performed the full semantic and pragmatic analysis. It is in the interaction with syntax that the strong underlying assumptions of the default control structure became clearly apparent. For instance, it was assumed that all relevant information from the noun phrase and the verb phrase would be available at the point where the noun phrase was being mapped to a verb argument. This assumption is only valid if the parse has been completed. If semantics is being called during the parse there will be cases when the relevant information might not have yet found its way into the discourse context. For instance, in the following sentence, the *one* refers to the second of the *two pumps*. *After installation of two pumps, pressure failure in the second one occurred.* As explained below, this causes a problem for KERNEL even after the parse has been completed. Trying to deal with incomplete parse information leads to many more examples of this type.

With KERNEL's current control structure, the subordinate clause, *after installation of two pumps*, has to have been completely processed, including time analysis, before the discourse context is updated with the two pumps. Time analysis prefers to process matrix clauses first to simplify the ordering of the time relations, which would mean that the *pumps* would not yet be in the discourse context when the *one* phrase is first encountered. This causes a problem for KERNEL but would not trouble CANDIDE. CANDIDE would in effect propagate forward the semantic constraint associated with the verb argument that the *one* should fill, i.e., that it should be of sort `device`. Then when the *pumps* finally reach the discourse context the assumptions associated with the *one* can be discharged, and the correct referent can be found.

It is possible to override KERNEL's default control structure in a particular context and warn reference resolution that it will need to wait [49]. Reference resolution could be notified that in the case of a pronoun it should wait until after the parse is complete. This would solve this particular problem in a fashion similar to CANDIDE's solution, by holding the semantic constraint information until a more suitable time. The same technique of overriding the default control structure could be used in the case of noun phrases scoped by quantifiers, which again would allow a solution similar to CANDIDE's. KERNEL could also choose to delay reference resolution of all nouns where the discourse context might indicate a non-specific reading, assuming that such a context could be recognized, and so on. The problem is that each of these circumstances has to be explicitly defined ahead of time as a special case. There is no guarantee that it will be possible to predetermine all of the special cases. At this point the appeal of an architecture like CANDIDE's that simply allows reference resolution to delay itself indefinitely depending on the context becomes apparent, and would be a preferable solution for KERNEL. The default control structure could still be retained as one of the primary heuristics for guiding the nondeterministic control strategy, but it would no longer have to be explicitly overridden for predefined special cases. This would allow KERNEL to enjoy the benefits of Candide's more flexible control structure without losing the capability of semantics/pragmatics interaction that has allowed successful handling of phenomena such as nominalizations, the recovery of implicit information, and time analysis.

7.4 Summary

We see all of these systems as moving towards a control structure that is flexible enough to allow for different modes of interaction, and yet which can still prefer simple, localized processing whenever possible. Of the systems we have discussed, CANDIDE and TACITUS have the most flexible control structures while PROTEUS and KERNEL make the most use of localized processing whenever possible. The differing goals of these four systems and KERNEL are not in conflict, but rather are aimed at different aspects of the same approach, and complement each other in the areas in which they can provide results. TACITUS explores the usefulness of powerful, open-ended reasoning capabilities, and provides invaluable insights into the questions of which difficult problems can be solved by this capability, what types of inference chains provide the solutions, and at what points will the reasoning become uncontrollable. KERNEL explores the use of linguistic cues to solve particular problems as efficiently as possible, and provides data on when and where those cues are insufficient, and how they need to be supplemented by either other types of linguistic information or reasoning capabilities. CANDIDE, if extended to greater depth of coverage, will provide insights into both the power and limits of a compositional approach and formal methods of representation.

8 Future directions for Kernel

We have discussed the KERNEL analysis process as being performed in stages, first parsing, and then an integrated semantic and pragmatic analysis, which controls access to further calls to reference resolution and temporal analysis. As strongly as we have been arguing in favor of a more flexible control structure, we still see basic benefits in having separate modules for distinctive categories of linguistic information, and a basic flow of control that begins with syntax and ends with knowledge representation and reasoning. This is a natural flow since for most analysis decisions it provides the relevant information at the stage where it is needed. Many parsing decisions can be made locally, looking only at information available from the phrase itself, and considering only syntactic properties. Most semantic decisions can also be made locally, but are made much more efficiently with access to all of the pertinent syntactic data about the item in question, as well as its semantic properties. Pragmatic decisions for the most part are dependent on syntactic and semantic input, as well as local pragmatic properties and the discourse context. It would be fruitless in the worst cases, and inefficient in even the easiest cases, to attempt to make decisions about temporal analysis without knowing the tense and aspect of the clause, or which adverbial phrases might be relevant.

We are not suggesting that this flow of control should be reversed, but rather that it should be extended to allow more communication between the modules, i.e., allowing access to sentence level discourse concerns during the parsing process, especially when they can be helpful in parse disambiguation. Clauses that consist of subject-less tensed verbs can be either sentence fragments or imperatives. It is traditionally the responsibility of the parser to assign the sentence type, but this cannot always be done on syntactic information alone. Isolated noun phrases provide an additional example of the need for pragmatic input, since they can either be treated existentially or as answers to questions. An informed discourse component can quickly resolve these issues, but only if its knowledge can be brought to bear during the parsing process. In all of the systems we have discussed, reference resolution is several stages away from syntactic parsing. Even with KERNEL's integrated semantic/pragmatic processing, it is only after a semantic representation is produced that it is fit into the discourse context. The reference resolution module plays a very passive role, never doing more than answering questions raised by semantics during the course of producing the semantic representation. Discourse information needs to play a more aggressive role in the analysis process which cannot be achieved simply by performing full semantic/pragmatic analysis on every partial parse. What we would like to see in our next version of KERNEL is a more top-down approach to discourse analysis that can operate in parallel with the parser, and which the parser can communicate with on a "need to know" basis. In order to achieve this, syntax and discourse need a common vocabulary for communicating about discourse concerns, in the same way that they now have a common vocabulary for communicating about implicit information. [44]

We see lexical semantics as potentially playing the same role with respect to communication between linguistic processing and KR&R that the ISR currently plays between syntax, semantics and pragmatics, i.e., as a shared data structure that is a key means of cross-component communication. In the way that the ISR uses grammatical roles such as **subject** and **object** and classifications such as *elided* to communicate important syntactic properties to semantics and pragmatics, lexical semantics can provide a global representation of semantic information that can be equally meaningful to linguistics and to KR&R. This requires a common terminology, which is currently best approximated by thematic roles such as Agent and Patient. For these roles to be meaningful they must have precise syntactic properties as well as word-independent conceptual properties of relevance to KR&R. [45] It is not necessary that these properties should be universal, but simply that they each apply to a class of more than one verb. Our different classes of Instruments, classic Instruments, W_Instruments, and T_Instruments are an example of the types of generalizations we can expect.

[44] Cf. Grosz and Sidner, [23], on discourse structure.

[45] Cf. Dowty, [15], for a discussion of principles for relating a thematic role's denotations to its syntactic properties.

A principled approach aimed at broader coverage would begin with Levin's verb taxonomy [40], attempting to use her verb classes as the appropriate categories for the mapping rule predicate environments. More general frameworks for situations that can allow for manner and place arguments are also needed [36]. The appropriate home for this type of information is a verb and situation taxonomy which would be implemented using the KR&R facility. This would strengthen the ties between semantics and KR&R, and allow even more reasoning capability to be accessed through the use of lexical conceptual representations and semantic class constraints. Filling arguments to predicating expressions is currently a key point of interaction between linguistic processing and KR&R in every system that we have mentioned. The KR&R capability is used to apply semantic constraints to potential argument fillers, either rejecting or accepting the filler on the basis of its semantic suitability. One of the reasons this particular method of accessing KR&R has been so successful is that it provides a carefully controlled environment within which the reasoning takes place, with respect to a very specific goal. We need to ensure that other goals we pass to KR&R, whether they be about parse pruning issues or pragmatic issues, are equally well-defined and limited in scope, so that we can continue to constrain the reasoning capability.

Acknowledgements

The authors wish to acknowledge the other co-developers of KERNEL and its predecessor, PUNDIT: Lynette Hirschman, Deborah Dahl, John Dowding, Francois Lang, Marcia Linebarger, Lew Norton and Cathy Ball. We would also like to thank Martha Pollack for clarification of certain points in the CANDIDE papers, Jerry Hobbs for many useful discussions on control issues and lexical semantics and the anonymous referees for many general comments on both the content and presentation of this paper.

References

[1] James Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–43, 1983.

[2] James Allen. Natural language, knowledge representation and logical form, November 1989. Paper delivered at BBN symposium, *Natural Language Processing: Language and Action in the World.*

[3] Catherine N. Ball. Analyzing explicitly-structured discourse in a limited domain: Trouble and failure reports. In *Proceedings of the DARPA Speech and Natural Language Workshop*, Philadelphia, PA, February 1989.

[4] J. Barwise and J. Perry. *Situations and Attitudes.* Bradford Books, MIT Press, Cambridge, Mass., 1983.

[5] R. J. Brachman, R. E. Fikes, and H. J. Levesque. Krypton: A functional approach to knowledge representation. In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, pages 411–430. Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1985.

[6] Joan W. Bresnan and Ronald M. Kaplan. Lexical-functional grammar: a formal system for grammatical representation. In Joan W. Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, Mass., 1982.

[7] Noam Chomsky. Remarks on nominalization. In *Studies on Semantics in Generative Grammar*, pages 11–61. Mouton, The Hague, 1972.

[8] Eve V. Clark and Herbert H. Clark. When nouns surface as verbs. *Language*, 55:1979, 767-811.

[9] Deborah A. Dahl. Focusing and reference resolution in PUNDIT. In *Proceedings of the 5th National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986.

[10] Deborah A. Dahl. Determiners, entities, and contexts. In *Proceedings of TINLAP-3*, Las Cruces, NM, January 1987.

[11] Deborah A. Dahl and Catherine N. Ball. Reference resolution in PUNDIT. In P. Saint-Dizier and S. Szpakowicz, editors, *Logic and logic grammars for language processing.* Ellis Horwood Limited, 1990.

[12] Deborah A. Dahl, Martha S. Palmer, and Rebecca J. Passonneau. Nominalizations in PUNDIT. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, Stanford University, Stanford, CA, July 1987.

[13] Donald Davidson. The logical form of action sentences. In Nicholas Rescher, editor, *The logic of decision and action*, pages 81–95. University of Pittsburgh Press, 1967.

[14] David Dowty. The effects of aspectual class on the temporal structure of discourse: Semantics or pragmatics. *Linguistics and Philosophy*, 9:37–61, 1986.

[15] David Dowty. Thematic proto-roles and argument selection. *Language*, 67:1990, 547-619.

[16] C. Fillmore. The case for case. In Bach and Harms, editors, *Universals in Linguistic Theory*, pages 1–88. Holt, Reinhart, and Winston, New York, 1980.

[17] Tim Finin, Rich Fritzson, and Dave Matuzsek. Adding forward chaining and truth maintenance to Prolog. In *Fifth IEEE Conference on Artificial Intelligence Application*, pages 123–130, March 1989.

[18] Michael Freeman, Lynette Hirschman, Donald McKay, and Martha Palmer. KNET: A logic-based associative network framework for expert systems. Technical Memo 12, SDC—A Burroughs Company, P.O. Box 517, Paoli, PA 19301, September 1983.

[19] R. Grishman, T. Ksiezyk, and N.T. Nhan. Model-based analysis of messages about equipment. Computer Science Department Technical Report 236, New York University, 1986. PROTEUS Project Memorandum 2.

[20] Ralph Grishman. Proteus report. In *Proceedings of the DARPA Speech and Natural Language Workshop*, February 1989.

[21] Ralph Grishman and Lynette Hirschman. PROTEUS and PUNDIT: Research in text understanding. *Computational Linguistics*, 12(2):141–45, 1986.

[22] Ralph Grishman and John Sterling. Preference semantics for message understanding. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 71–74, Cape Cod, Massachusetts, 1989.

[23] Barbara J. Grosz and Candace L. Sidner. Attention, intentions and the structure of discourse. *Computational Linguistics*, 1986.

[24] Mary P. Harper and Eugene Charniak. Time and tense in English. In *Proceedings of the 24th Annual Meeting of the ACL*, pages 3–9, New York, 1986.

[25] Erhard Hincrichs. Temporal anaphora in discourses of english. *Linguistics and Philosophy*, 9:63–82, 1986.

[26] L. Hirschman, M. Palmer, J. Dowding, D. Dahl, M. Linebarger, R. Passonneau, F.-M. Lang, C. Ball, and C. Weir. The PUNDIT natural-language processing system. In *AI Systems in Government Conf.* Computer Society of the IEEE, March 1989.

[27] Lynette Hirschman. Conjunction in meta-restriction grammar. *Journal of Logic Programming*, 4:299–328, 1986.

[28] Lynette Hirschman. A meta-rule treatment for English *wh*-constructions. In *Proceedings of META88 Workshop on Meta-Programming in Logic Programming.* University of Bristol, Bristol, 1988.

[29] Lynette Hirschman and John Dowding. Restriction grammar: A logic grammar. In P. Saint-Dizier and S. Szpakowicz, editors, *Logic and Logic Grammars for Language Processing*, pages 141–167. Ellis Horwood, 1990.

[30] Lynette Hirschman and Karl Puder. Restriction grammar: A Prolog implementation. In D.H.D. Warren and M. VanCaneghem, editors, *Logic Programming and its Applications*, pages 244–261. Ablex Publishing Corp., Norwood, N.J., 1986.

[31] Jerry R. Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. Interpretation as abduction. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, NY, June 1988. A longer, unpublished version of this paper exists.

[32] R.S. Jackendoff. *Semantic interpretation in generative grammar.* MIT Press, Cambridge, Mass., 1972.

[33] R.S. Jackendoff. *Semantic structures.* MIT Press, Cambridge, Mass., 1990.

[34] Tomasz Ksiezyk and Ralph Grishman. Equipment simulation for language understanding. Technical Report 11, New York University, September 1987. PROTEUS Project Memorandum.

[35] François-Michel Lang and Lynette Hirschman. Improved portability and parsing through interactive acquisition of semantic information. In *Proceedings of the Second Conference on Applied Natural Language Processing*, Austin, TX, February 1988.

[36] Richard K. Larson. Implicit arguments in situation semantics. *Linguistics and Philosophy*, 11(2):131–168, May 1988.

[37] Alex Lascarides and Jon Oberlander. Temporal coherence and defeasible knowledge. *Theoretical Linguistics*, 1992. To appear.

[38] Geoffrey N. Leech. *Meaning and the English Verb.* Longman, London, 1987. 2nd Edition.

[39] B. Levin. Lexical semantics in review: An introduction. Lexicon Project Working Papers 1, MIT Center for Cognitive Science, 1985.

[40] Beth Levin. Verbal diathesis, 1988. ms.

[41] Marcia C. Linebarger, Deborah A. Dahl, Lynette Hirschman, and Rebecca J. Passonneau. Sentence fragments regular structures. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, NY, June 1988.

[42] David L. Matuszek. K-Pack: A programmer's interface to KNET. Technical Memo 61, Unisys Corporation, P.O. Box 517, Paoli, PA 19301, October 1987.

[43] Mark Moens and Mark J. Steedman. Temporal ontology and temporal reference. *Computational Linguistics*, 14:15–28, 1988.

[44] Robert C. Moore. Events, situations and adverbs. In R. Weischedel and M. Bates, editors, *Challenges in Natural Language Processing.* Cambridge University Press, To appear.

[45] M. Palmer. A case for rule-driven semantic analysis. In *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics*, Stanford, CA, June 1981.

[46] Martha Palmer. *Driving Semantics for a Limited Domain.* PhD thesis, University of Edinburgh, 1985.

[47] Martha Palmer. Customizing verb definitions for specific semantic domains. *Machine Translation*, 5, 1990.

[48] Martha Palmer. *Semantic Processing for Finite Domains.* Cambridge University Press, Cambridge, England, 1990.

[49] Martha S. Palmer, Deborah A. Dahl, Rebecca J. [Schiffman] Passonneau, Lynette Hirschman, Marcia Linebarger, and John Dowding. Recovering implicit information. In *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, Columbia University, New York, August 1986.

[50] Cécile L. Paris. Tailoring object descriptions to a user's level of expertise. *Computational Linguistics*, pages 64–78, 1988.

[51] Rebecca Passonneau, Carl Weir, Tim Finin, and Martha Palmer. Integrating natural language processing and knowledge based processing, 1990. Eighth National Conference on Artifical Intelligence, AAAI 90.

[52] Rebecca J. Passonneau. Situations and intervals. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 16–24, 1987.

[53] Rebecca J. Passonneau. A computational model of the semantics of tense and aspect. *Computational Linguistics*, 14(2):44–60, 1988.

[54] Fernando C. N. Pereira and Martha E. Pollack. Incremental interpretation. Technical Note 490, SRI International, Menlo Park, CA., May 1990.

[55] Martha E. Pollack and Fernando C. N. Pereira. An integrated framework for semantic and pragmatic interpretation. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, pages 75–86. Association for Computational Linguistics, 1988.

[56] Hans Reichenbach. *Elements of Symbolic Logic*. The Free Press, New York, 1947.

[57] Lehnart K. Schubert and Chung Hee Hwang. Picking reference events from tense trees: A formal, implementable theory of English tense-aspect semantics. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 34–41, Hidden Valley, PA, June 1990.

[58] David B. Searls and Lewis M. Norton. Logic-based configuration with a semantic network. *The Journal of Logic Programming*, 8:53–73, 1990.

[59] Candace L. Sidner. Focusing in the comprehension of definite anaphora. In M. Brady and R. C. Berwick, editors, *Computational Models of Discourse*, pages 267–330. The MIT Press, Cambridge, Massachusetts, 1983.

[60] C.L. Sidner. *Towards a computational theory of definite anaphora comprehension in English discourse*. PhD thesis, MIT, 1979.

[61] Fei Song and Robin Cohen. Tense interpretation in the context of narrative. In *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 131–136, 1991.

[62] Zeno Vendler. Verbs and times. *Linguistics in Philosophy*, pages 97–121, 1967.

[63] Bonnie Lynn Webber. Tense as discourse anaphor. *Computational Linguistics*, 14, 1988.

[64] Carl Weir. Knowledge representation in Pundit. Technical Report PRC-LBS-8914, Unisys, Paoli, PA, October 1988.