Integrating Natural Language Processing and Knowledge Based Processing*

Rebecca Passonneau and **Carl Weir** and **Tim Finin** and **Martha Palmer**
Unisys Corporation
The Center for Advanced Information Technology
Paoli, Pennsylvania

Abstract

A central problem in text-understanding research is the indeterminacy of natural language. Two related issues that arise in confronting this problem are the need to make complex interactions possible among the system components that search for cues, and the need to control the amount of reasoning that is done once cues have been discovered. We identify a key difficulty in enabling true interaction among system components and we propose an architectural framework that minimizes this difficulty. A concrete example of a reasoning task encountered in an actual text-understanding application is used to motivate the design principles of our framework.

Introduction

The central problem confronting text-understanding research is what in recent years has been characterized as the *economy* of natural language [Barwise and Perry, 1983]. Natural languages possess an economy of expression because humans take advantage of situational cues in conveying whatever information they intend to partially encode as a natural language utterance. For a text-understanding system to approach human competence in coping with natural language, it must be capable of exploiting the same linguistic cues and general knowledge that humans exploit. Two related issues that arise in efforts to endow systems with this capability are the need to make complex interactions possible among the processing components, or *knowledge sources*, that search for cues, and the need to control how much reasoning is done with the cues once they are discovered.

In the following section, a conflict between two different processing methodologies is identified as the key difficulty in enabling complex interactions among the processing components of text understanding systems. An architecture is proposed that minimizes this problem, but that places a burden on system designers to insure that the underlying representation language used as a communication medium among the system's processing components is rich enough to capture the information content of an input text at a sufficient level of detail to be useful for the application-specific tasks the system is serving. We propose six design principles addressing the issue of how expressive the underlying representation language needs to be, and how the reasoning processes that manipulate expressions in this language should be controlled.

In order to convey as concretely as possible the depth of understanding that text understanding systems are expected to achieve, we have selected a particularly instructive example from a recent evaluation effort in which participating text-understanding systems performed a summarization task. Our discussion of this example will motivate our six design principles and indicate how our proposed framework permits evolutionary progress towards more reliable text analysis.

Component Interaction

The flow of control in most text-understanding systems generally consists of an initial phase of syntactic parsing and semantic interpretation that results in a logical form which is then translated into a less expressive, unambiguous representation serving as input to whatever general reasoning modules the system has access to. Although there are many variations on this general theme, systems that rely on a careful syntactic analysis of textual input typically enumerate a set of unambiguous or partially ambiguous (*least commitment*) parses for a given utterance that are semantically interpreted, and the first coherent interpretation is passed on to the knowledge representation and reasoning component as the literal information content of the utterance. In systems possessing such architectures, the interaction between linguistic processing and general reasoning is minimal, since the general reasoning mechanisms are not defined over the data structures used in linguistic processing.

A similar lack of interaction between linguistic processing and general reasoning is present in text-understanding systems that are less dependent on a careful syntactic analysis of textual input. SCISOR is

*This work was partially supported by DARPA Contract N00014-85-C-0012.

an example of such a system [Rau and Jacobs, 1988]. When SCISOR processes a sentence in a text, it first tries to derive a full parse using a chart-based parsing algorithm. If this is not possible, a partial parse can be used to instantiate event descriptions based on expectations of how a given type of event is structured, including script-based intuitions about likely orderings of events in a typical scenario. Although this use of a backup, expectation-based processing strategy is a promising technique for handling gaps in coverage, it doesn't actually result in true interaction between the parser and the reasoning mechanisms used to analyze situation structure. Other well documented text-understanding systems that are heavily dependent on knowledge-based techniques exhibit difficulty in taking advantage of linguistic cues. In Hirst's Absity system, e.g., the use of marker-passing as the principal reasoning mechanism makes it difficult to accommodate syntactic cues in lexical disambiguation [Hirst, 1986].

Efficient linguistic processing is based upon a generate-and-test search methodology. In contrast, general reasoning techniques involve the creation and maintenance of persistent, complex, data structures. A generate-and-test methodology is effective if the cost of creating data-structures representing hypotheses is relatively cheap, which is not the case with the sorts of data structures that must be created and maintained when doing general reasoning. Consequently, there is a fundamental conflict between efficient linguistic processing techniques and general reasoning techniques that results in a natural tendency to separate the two forms of processing [Passonneau *et al.*, 1989].

There is a growing realization that although a strict separation between linguistic processing and general reasoning makes for a modular, efficient system architecture, it is ultimately untenable because it doesn't allow processing decisions to be postponed until adequate information is available to make well-motivated choices [Allen, 1989]. Efforts to allow for the postponement of such decisions through the use of canonical structures intended to capture multiple interpretations have been problematic. For example, Wittenberg and Barnett have observed that the use of canonical structures in the Lucy system developed at MCC resulted in abandonment of both compositional interpretation and modularity in the declarative representation of information [Wittenburg and Barnett, 1988].

We conclude that current text-understanding systems are not properly designed for the sort of interaction among components that is required to cope with the indeterminacy of natural language. An obvious framework for achieving this sort of interaction is a blackboard architecture in which individual processing components communicate with one another via a common language that they use to post and examine facts in some globally accessible data-structure. A few natural language processing systems already make limited use of a blackboard architecture. Thus, the manner in which polaroid words in Graeme Hirst's Absity semantic intepreter communicate with one another is via a blackboard structure [Hirst, 1986]. The discourse component in the Lucy system developed at MCC exhibits a blackboard style of interaction among a number of sub-components that are used to perform reference resolution [Rich and LuperFoy, 1988]. We believe text-understanding systems should incorporate a blackboard architecture in which all components communicate via a common language over which the system's available reasoning mechanisms have been defined, and that this language be expressive enough to capture all of the information that the various system components are capable of contributing to the text-understanding task.

Six Design Principles

Unfortunately, it is currently impossible to design a text-understanding system in which all components communicate via a common language with great expressive power over which sophisticated reasoning mechanisms have been defined. The knowledge representation and reasoning techniques currently available are simply not up to the task of capturing the nuances of meaning in natural language. James Allen has suggested that this inadequacy is the principal reason that data structures commonly referred to as representations of logical form are created; they capture more of the expressiveness of natural language than do current knowledge representation formalisms with well-defined inference mechanisms [Allen, 1989].

Given this state of affairs, one can either abandon the design of text understanding systems based on the existence of a communication medium of the sort that is required, or one can pursue the design of such systems with the hope that appropriate representation languages will be developed in the near future. We recommend the latter choice for two reasons. First, experience has shown that research and development efforts based on the old enumeration paradigm have begun to show diminishing returns. Second, there is a growing recognition in the knowledge representation and reasoning community that general purpose knowledge representation systems need to be built that incorporate more expressive languages, even if doing so requires the abandonment of completeness [Doyle and Patil, 1989].

We propose a design methodology based on the following six principles:

1. A capability for data-driven reasoning.
   The system should be endowed with the intelligence to know when available data suggests that a particular line of reasoning would be worth pursuing.
2. A capability for constraint reasoning.
   The system should be capable of propagating constraints on an interpretation and reasoning about them.

3. **An MRL with adequate expressive power.**
   In selecting a knowledge representation and reasoning component, the expressive power of the MRL should take precedence over completeness and worst-case time complexity.
4. **A single MRL.**
   The system's knowledge representation and reasoning component should provide a single meaning representation language (MRL) which may be used as a medium of communication for all system components. This language should serve both domain-specific and application-specific representation and reasoning tasks.
5. **A capability for delayed reasoning.**
   The system should be able to postpone making a decision if insufficient information is available to make a reasonable choice.
6. **A capability for demand-driven reasoning.**
   The system must be intelligent enough to know that certain decisions simply do not need to be made.

In the following sections, we provide a detailed example of a reasoning problem encountered in a text understanding task and illustrate how the design principles we have proposed make it possible to properly confront it. To follow our discussion, it is necessary to have a basic understanding of the knowledge representation and reasoning component used in the KERNEL text understanding system that we are currently building.

The knowledge representation and reasoning component in KERNEL is based on the tripartite model popularized by Brachman, Fikes, and Levesque in the KRYPTON system [Brachman *et al.*, 1985]. 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 asking queries about the current state of such representations [Weir, 1988]. PKR does not possess all the expressive power ultimately needed for text understanding, but it does provide adequate access to the two knowledge representation and reasoning modules that KERNEL currently uses, as shown in Fig. 1.

Concept hierarchies are currently defined in KERNEL using a KL-ONE style representation language called KNET [Matuszek, 1987]. Assertions are expressed in terms of facts posted to the database of a forward-chaining system called Pfc [Finin *et al.*, 1989]. It is the forward-chaining database maintained by Pfc that serves as a blackboard structure in KERNEL. Pfc is built on top of Prolog and provides justification-based truth maintenance for expressing and reasoning about instances of concepts. Moreover, Pfc is able to manipulate two fundamentally different types of rules: *eager* rules and *persistent* rules. An eager rule is a typical forward-chaining rule that provides KERNEL with the capability to do data-driven reasoning. A persistent rule is a rule that is posted to the forward-chaining database, but that remains inert until a consumer appears for one of its conclusions. When a consumer appears, an instance of the rule is instantiated as an eager rule to satisfy that consumer. The use of persistent rules helps control the amount of reasoning engaged in by the system.

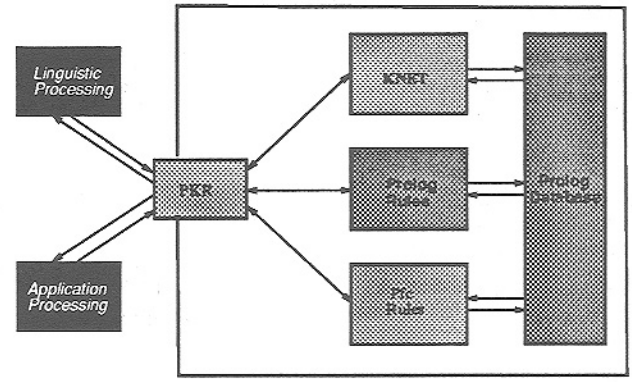

Figure 1: Kernel's current knowledge representation and reasoning system has four components: PKR provides an abstract interface; KNET is a terminological representation system, Prolog is used for some backward chaining, and Pfc provides a more flexible reasoning component with an integrated truth maintenance system.

Archetypal NL Reasoning Problem

As part of an effort to develop evaluation metrics for text understanding systems, NOSC sponsored a workshop in which participating systems trained and were tested on a summarization task for a corpus of military messages [Sundheim, 1989]. The texts consisted of OPREP (OPerational REPort) messages describing naval sightings of surface, subsurface and airborne vessels. Fig. 2 illustrates a sample text and the target output. To demonstrate understanding, each participating system had to recognize the events mentioned in a message and determine whether they fell into one of 5 critical types of events. At most, two such events were to be identified per message: the most important such event initiated by the friendly forces and, similarly, the most important one initiated by the hostile forces. Generating one template per reported event and correctly filling the 10 slots of each template constituted successful understanding. In cases where two templates were generated, the system was to determine the correct temporal ordering. Fig. 2 illustrates a message with two event templates where the correct temporal ordering is not provided explicitly in the message, but instead must be inferred from the message content.

The critical sentence of the sample message is:

friendly CAP a/c splashed hostile tu-16 proceeding

Narrative text from message

Friendly CAP A/C splashed hostile TU-16 proceeding inbound to Enterprise at 35nm. Last hostile acft in vicinity. Air warning red weapons tight. Remaining alert for additional attacks.

Paraphrase of narrative text

A friendly combat air patrol aircraft shot down a hostile TU-16 aircraft thirty-five nautical miles away from the carrier Enterprise. The hostile aircraft had been proceeding towards the carrier at the time of the attack. It was the last hostile aircraft in the area. All friendly forces should remain at the highest level of alert, but are not given permission to fire their weapons.

Filled in Database templates

Slot	Template 1	Template 2
Message number	DEV-GP1-N09722-002B	DEV-GP1-N09722-002B
Category of action	track	attack
Category of force initiating action	hostile	friendly
Platform category of force initiating action	air	air
Platform category of force acted upon	surf	air
Most specific description of force initiating action	tu-16	f-14a
Most specific description of force acted upon	uss enterprise cvn-65	tu-16
Weapons or other instruments used in action	no data	no data
Location of action	no data	[at 35nm]
Time of action	no data	no data
Consequences of action	response by opposing force	damage or loss to object

Figure 2: This figure shows an example of the narrative free text portion of one of the Oprep messages, a paraphrase of the intended meaning, and the properly filled database templates which represent the meaning.

inbound to enterprise at 35nm.[1]

In this domain, an aircraft proceeding towards an opposing carrier constitutes a *track* event, thus licensing the first template shown above. An aircraft that gets splashed has been shot down into the ocean, which counts as an *attack* event. Note that the *track* template is correctly ordered prior to the *attack* template.[2] An accurate analysis of the sentence shown above leads to the correct ordering of the two templates demanded by the application, but requires close cooperation between linguistic and knowledge-based processing. We describe below how this sentence illustrates the need to simultaneously make use of local linguistic information and global contextual information. But note that the information required by the application cannot predict in general the degree to which such reasoning will be required. For messages with more explicit temporal information, linguistic analysis alone may be sufficient to provide the correct template fills.

Given that the template filling task requires temporal information, the system should provide its best guess regarding the temporal order of the *proceed* and *splash* events even in the absence of explicit assertions. But since the system may have access to many different kinds of knowledge, the need to control deep reasoning with respect to specific goals arises. For example, for this task, the system should not attempt to infer the tu-16's location of origin because it is irrelevant. As we describe the inference problem in more detail, it should be clear both that the original sentence does not explicitly order the two relevant events, and also that inferring the temporal order depends on multiple knowledge sources.

The *proceeding* event, a temporally situated occurrence involving the referent of the noun phrase *hostile tu16*, is mentioned in a post-modifier with no explicit temporal information, i.e., no tense and no temporal locatives. Given the sentence structure, there are three possible temporal locations for the event. These three possibilities are that the reference time of the event is the same as the matrix clause reference time specified by the simple past tense, a different past time, or the present time (utterance time, or here, message composition time). Graphic representations of three illustrative sentences are shown in Figs. 3–5.[3]

Given that the linguistic structure permits three tem-

[1] See Fig. 2 for a paraphrase.

[2] The Kernel system generates the template output shown in Fig. 2, relying on a combination of linguistic processing, deep reasoning, and application-specific heuristics; however, we have not fully implemented the architecture proposed here.

[3] A Reichenbachian interpretation of tense involves three temporal indices, one of which, ST, represents the time at which a speech or text event occurs [Reichenbach, 1947]. Since in these examples event time (ET) and reference time (RT) are identical, only ST and RT are used.

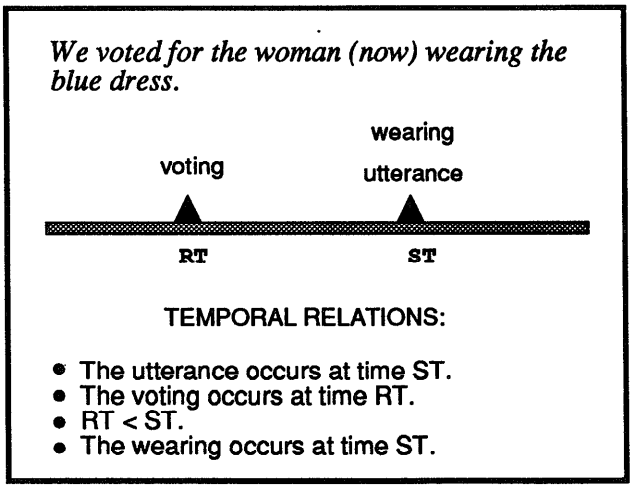

Figure 3: The first possibility for the reference time of the modifier is that it is the same as the time when the sentence or utterance is produced (ST).

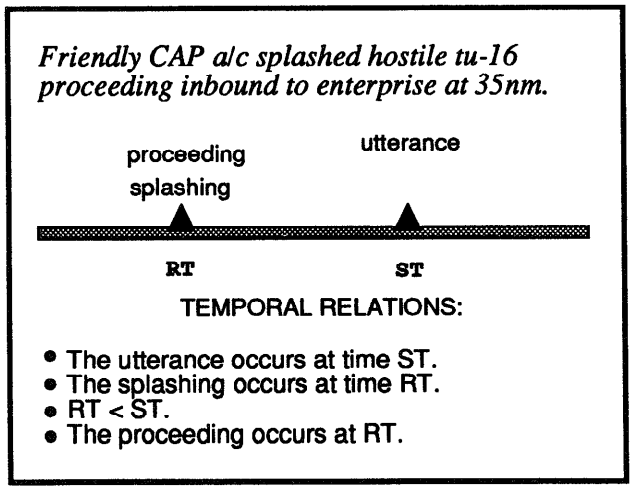

Figure 4: The second possibility for the reference time of the modifier is that it is the same as the time specified by the matrix clause tense (RT).

poral interpretations for the time of the *proceeding*, the next step is to examine what factors favor one interpretation over another. The three inference rules presented in Figs. 6, 7 and 8 illustrate a reliance on multiple knowledge sources. Note that the rules use the following symbols and relations[4]:

- S_1 – the event or situation mentioned in the matrix clause
- S_2 – the event or situation mentioned in the postmodifier
- S_3 – the situational context at the time of the utterance
- holds(S, RT) – true if event or situation S holds at time RT
- salient(E, S, ST) – true if entity E is salient in situational context S at time ST
- consistent(E, S_1, S_2, RT) – true if what is predicated of entity E in situation S_1, where S_2 holds at RT, is consistent with the assumption that S_1 also holds at RT

In all three rules, the first 5 clauses are the same, and depend on the local linguistic structure. They make reference to syntactic relations like *matrix clause* or semantic correlates thereof, such as the event or situation evoked by a clause, the specification of a reference time for that event or situation, e.g., a known RT_1 for the tensed matrix clause and an unknown RT_2 for the untensed reduced relative. Similarly, the first 5 clauses of all three rules make reference to the consistency of the *-ing* predicate with what is known about the modified entity at various other known times. In general, the three rules depend on both context-independent semantic interpretation and context-dependent pragmatic processes such as determing the reference time of the matrix clause, determining the referent of the noun phrase, and determining the salience and consistency of entities in the discourse context. While there is no space here to discuss salience and consistency in detail, salience of an entity can be taken as a discourse property that involves the notions of local and global *focus* [Grosz and Sidner, 1986]. Consistency involves a combination of lexically driven inference (i.e., what facts about X and Y follow from *splash(X, Y)*; likewise for *proceed(X, Y)*), general world knowledge about times and situations, and an evaluation of what is known about the relevant entity and situations at a particular reference time.

The three inference rules capture a general reasoning process that can be described as follows: when resolving the referent E, attempt to find a known time to assign the situation S_2 that is the same or prior to ST by insuring the consistency of S_2 with everything else that is known about E for that time. The question is when and how to execute this reasoning process, and more crucially, whether it can be performed as a simple sequential process. We believe it cannot be performed as a sequence of distinct steps in distinct semantic and pragmatic processing stages for the following reasons. If the reasoning is done incrementally, as a semantic interpretation for each phrase is arrived at, then all the relevant local syntactic and semantic information specified in the first 5 clauses of the inference rules in Figs. 6-8 will indeed be available. But the problem here would be to handle the interdependence between the two pragmatic processes of resolving the referent of the noun phrase and finding the temporal location of its modifier. There is a cirularity in that knowing the referent of the noun phrase might eliminate certain temporal locations for the situation predicated of that

[4]Terminological note: an event is a specialization of a situation; a situational context for an utterance or text is also a specialization of a situation.

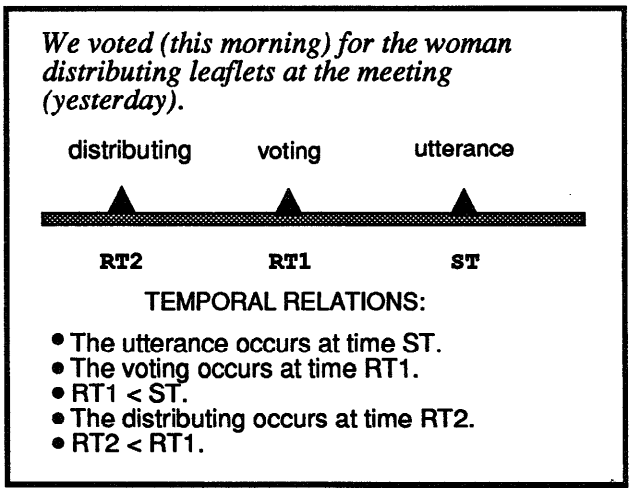

Figure 5: The third possibility for the reference time of the modifier (RT2) is that it is prior to the utterance time (ST) but distinct from the reference time of the matrix clause (RT1). The figure illustrates the case where RT2 precedes, rather than follows, RT1.

referent in the postmodifier; but, knowing the temporal location of the situation in the postmodifier might help determine the referent of the noun phrase. This argues for a solution which circumvents the problem of having to order the two pragmatic processes with respect to one another. We show in the following section how the six principles outlined in the section preceding this one provide this feature, as well as other advantages.

An alternative to performing the reasoning incrementally would be to postpone it until a semantic interpretation for the whole sentence has been performed. Here there are two potential problems. The first is that if the linguistic input has alternative analyses, then being able to reason from the current discourse context might help choose among them. In contrast, if the reasoning is carried out after the semantic interpretation of the sentence, then the global discourse information may be available but the output of linguistic processing contains representations of entities, situations and times in a form quite remote from surface linguistic structure. The reasoning mechanism would need access to the local linguistic information specified in the rules above, i.e., that there was a particular syntactic configuration and consequent semantic relations. Again, the solution we propose in the following section circumvents this difficulty. In addition, our solution maximizes the ability of the system to accurately solve for the unknown reference time, and thus to perform the required application task as accurately as possible.

We voted for the woman wearing the blue dress.

a) $holds(S_1, RT_1)$
b) $RT_1 < ST$
c) $holds(S_2, RT_2)$
d) $unknown(RT_2)$
e) $salient(E, S_3, ST)$
f) $consistent(E, S_2, S_3, ST)$
$\Rightarrow$
$RT_2 = \text{ST}$.

Figure 6: This rule handles cases where the RT of the post-modifier (RT_2) and ST are the same.

Friendly cap a/c splashed hostile tu-16 proceeding inbound to enterprise

a)–d) same as above
e) $not(salient(E, S_3, RT_1))$
f) $consistent(E, S_2, S_1, RT_1)$
$\Rightarrow$
$RT_2 = RT_1$.

Figure 7: This rule handles cases where RT of the matrix clause (RT_1) and RT of the post-modifier (RT_2) are the same.

We voted for the woman passing out leaflets at Tom's party yesterday

a)–d) same as above
e) $not(salient(E, S_3, RT_1))$
f) $not(consistent(E, S_2, S_1, RT_1))$
$\Rightarrow$
$RT_2 < \text{ST}$,
$RT_2 \neq RT_1$.

Figure 8: This rule handles cases in which RT of the post-modifier (RT_2) is distinct from ST and RT of the matrix clause (RT_1).

Improved Integration

The six principles we have proposed comprise a package in which maximum benefit of each principle follows from implementing the whole. Below we use the example described in the previous section to illustrate how this combination of principles would circumvent many of the processing difficulties posed by this example.

We assume that when the modifier in the example sentence is semantically interpreted, the available data consists of the full parse tree and a partial semantic analysis. We also assume that in general, during lin-

guistic processing of a sentence, the status in the discourse model of entities referenced in the same sentence may or may not be available. Principle 1, data driven reasoning, dictates that elaboration of the MRL representation of a sentence be data-driven. We have shown that the reference time of the postmodifier in the sample sentence is constrained by the three inference rules provided in Figs. 6-8, irrespective of what other data may or may not have been already derived about the entities and situations mentioned in the sentence. We therefore conclude that the knowledge represented in the three inference rules ought to be included in the output of linguistic processing. KERNEL currently enforces data-driven reasoning in that facts derived from linguistic processing are posted to PKR during linguistic processing and the KR&R modules must then reason from this data.

Principle 2 asserts that the MRL expressions representing the output of linguistic processing must express no more nor less than can be justified by the linguistic input. Thus in cases where the linguistic input is indeterminate, as in the example sentence, the output of linguistic processing should preserve this indeterminacy. At present, we can reason with constraint rules such as those shown in Figs. 6-8, but we do not have a concise means of reasoning about constraints as first class objects. To return to our example, it might prove useful in some circumstances to reason about the *proceeding* and the *splash* events in a way that takes into account the constraints on the reference time of the *proceeding*, e.g., that $RT_2 \leq ST$, prior to being able to resolve the constraints.

Principle 3 embodies our belief that the expressive power of the MRL should take precedence over the ability to reason efficiently over MRL expressions (cf. [Allen, 1989] [Doyle and Patil, 1989]). The relation of linguistically derived temporal information to temporal reasoning is a good example of the potential need to sacrifice completeness, since temporal interpretation of textual input generally results in partially ordered times at best [Allen, 1983].

Principle 4 requires that the information be represented in a common MRL so that any other processing components can potentially contribute to or reason over the representation. The semantic and pragmatic interpretation procedures in KERNEL currently use PKR to determine how to express the facts it derives, thus PKR provides access to whatever KR&R components are available while insulating the semantic and pragmatic interpreters from dependence on a particular KR framework. It is only necessary that for each query from KERNEL there be an equivalent MRL expression. In this case, PKR insures that all facts about entities, times, situations and temporal relations that are derived from the sample sentence are expressed in MRL expressions that can mediate between the linguistic modules and KR&R resources.

By restricting the responsibility of the linguistic modules to that of posting linguistically justifiable conclusions, the time at which deep reasoning takes place is open. For example, it can be postponed until contingent information is more likely to be available, as noted in Principle 5. For efficiency reasons, we currently postpone invocation of PFC until after linguistic processing of an input sentence. At this point, PFC would have access to the maximum set of facts pertaining to the determination of RT_2 in the sample sentence. According to our model, if the data support a specific conclusion about the identity of RT_2, that conclusion should be derived. If not, the representation of RT_2 should remain expressed in terms of constraints.

The final principle, demand-driven reasoning, complements data-driven reasoning in a way which integrates well with application requirements. If we take our example sentence in isolation from its discourse context, there is insufficient linguistic data to equate RT_2 with either RT_1 or ST; there is also insufficient data to unequivocally distinguish it from RT_1 or ST. A particular discourse context and the additional data it provides might justify more specific conclusions about RT_2. For example, if the discourse context definitively supports the conclusion that "not(salient(E,S_3,RT_1))" (cf. clause e) of the rules in Figs. 7-8), then perhaps the mere presence of this extra data should trigger the reasoning process that would eliminate ST as a possible value of RT_2. On the other hand, it is not necessarily useful to derive all justifiable conclusions from a particular set of data. Even in a discourse context which could further constrain RT_2, it may be preferable for the additional reasoning to be suspended until there is an explicit demand. We speculate that such a demand could originate either from other application-independent processing or from application-dependent tasks. In our example, the application should drive the search for relevant temporal information.

Conclusion

We see a fundamental conflict between linguistic processing strategies and knowledge-based reasoning processes. In adapting KERNEL to a specific application task, we confronted a tradeoff between our long term goals of designing a general purpose, application independent NL system and the specific requirements of the appplication task. Although we were able to satisfy the immediate demands of the application without changing the fundamental architecture of the system, we felt that it should be possible in the long run to achieve a better balance between application-specific post-processing modules and the underlying text understanding system. We have described our recent efforts to achieve a better solution and our current views regarding the promise of closer integration of general purpose reasoning, linguistic processing, and application oriented reasoning. It is our conviction that long-

term research goals concerning the development of practical applications of text understanding systems must focus on the problem of integrating multiple knowledge sources as cleanly as possible.

Acknowledgements

The authors would like to thank James Allen for his valuable comments on a previous draft, as well as two anonymous reviewers. We also thank Rich Fritzson and Dave Matusek for help and inspiration.

References

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

[Allen, 1989] 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.*

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

[Brachman *et al.*, 1985] 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.

[Doyle and Patil, 1989] Jon Doyle and Ramesh S. Patil. Two dogmas of knowledge representation. Technical Report MIT/LCS/TM-387.b, MIT Laboratory for Computer Science, Cambridge, MA., September 1989.

[Finin *et al.*, 1989] T. Finin, R. Fritzson, and D. Matuzsek. Adding forward chaining and truth maintenance to prolog. In *CAIA-89*, pages 123–130, March 1989.

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

[Hirst, 1986] Graeme Hirst. *Semantic Interpretation and the resolution of ambiguity.* Studies in Natural Language Processing. Cambridge University Press, Cambridge, 1986.

[Matuszek, 1987] 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.

[Passonneau *et al.*, 1989] Rebecca Passonneau, Carl Weir, and Tim Finin. Interfacing natural language processing and knowledge-based processing in Pundit, 1989. Darpa Spoken Language/Natural Language Meeting.

[Rau and Jacobs, 1988] Lisa F. Rau and Paul S. Jacobs. Integrating top-down and bottom-up strategies in a text processing system. In *Second Conference on Applied Natural Language Processing*, pages 129–135, Austin, Texas, February 1988. Association for Computational Linguistics.

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

[Rich and LuperFoy, 1988] Elaine Rich and Susann LuperFoy. An architecture for anaphora resolution. In *Proceedings of the second conference on applied natural language processing*, pages 18–24, Austin, TX, February 1988. Association for Computational Linguistics.

[Sundheim, 1989] B. M. Sundheim. Plans for a task-oriented evaluation of natural language understanding systems. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 197–202. Morgan Kaufman Publishers, Inc., Philadelphia, PA, 1989.

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

[Wittenburg and Barnett, 1988] Kent Wittenburg and Jim Barnett. Canonical representation in NLP system design: a critical evaluation. In *Proceedings of the second conference on applied natural language processing*, pages 253–259, Austin, TX, February 1988. Association for Computational Linguistics.