1 Introduction

The most common methodological approach for natural language processing involves finding a system of representation and operations that allows sentences to be treated as well formed expressions in some formal language. Often this is accomplished by developing a set of phrase structure rules that describe the syntax of a natural language, and a set of predicates that capture the semantic relations [31, 1, 19]. This kind of approach has proven to be particularly successful for question answering systems which can take advantage of correspondences between natural language and set theory [8]. It has consequently become a popular method for AI researchers, perhaps because the touchstone of machine intelligence — the Turing test [27] — lies within the domain of question answering.

Unfortunately, the Turing test has distorted the development of natural language processing by placing undue emphasis on the degree to which a computer can respond to questions at the expense of more creative uses of language. While many projects have addressed more sublime aspects of language use [1, 15, 17], a conspicuous [illegible] nearly all natural language processors has been their dependence on human initiative to stimulate a response. In order for the machine to perform a less passive role in language exchange, it must be able to actualize broader aspects of discourse like the ability to instigate a natural language exchange, follow a topic of discourse, and understand and initiate the logical conclusion of an exchange. Furthermore, its ability to support these aspects can only be properly assessed if it can do so without human intervention, provocation, or control.

This paper outlines TAILOR, a research effort designed to investigate these broader aspects of natural language through the process of telling stories. To generate story text, a computer has to initiate a discourse by introducing an opening setting and cast of characters; develop and unveil a plot that provides continuity for the discourse, and bring the story to a suitable climax and denouement [19]. While a computer system aimed at satisfying these requirements must necessarily be concerned with the problems of sentence form and content, the principal difficulty of text generation addressed by TAILOR is the arrangement of natural language expressions into a logical and coherent sequence [4].

In accordance with a speech act theory of natural language [6, 23, 21, 9] TAILOR regards the sentences of a story as deliberate acts by a speaker. In this way it is able to treat the coherency problem as the more general problem of action planning. Furthermore, by taking advantage of an analogy between story telling and game playing, TAILOR solves many of the difficulties associated with text generation by using well understood techniques of automated game playing. By mapping story elements onto game elements, the sentences of a story are generated in a strategic series similar to the sequence of moves for a game. This endows the resulting discourse with a plot that follows a consistent path from an initial situation through to a logical conclusion.

2 Text generation

All automatic text generating systems must confront three primary difficulties: 1) deciding what information is to be expressed in the text, 2) finding a suitable sequence for presenting this information, and 3) transforming the information into well formed sentences of a particular language [11, 13]. The first of these is essentially a problem of extracting relevant information fragments from a knowledge base [20, 36]. Interactive systems most often accomplish this by looking for clues within the input expressions. Using simple principles of set manipulation, these clues are regarded as a kind of mathematical description of the subject of discussion [9, 12, 25, 33]. For example, the underlying principle of a set theoretic question-answering system is that the answer to a question will be an element of the power set for the database contents. Key words from the query are applied as set operations for filtering out the relevant data items. Nouns and adjectives construct subsets, verbs and prepositions establish relations between subset elements, conjunctions perform union and intersection on the subsets, and determiners establish quantification. The resulting subset (possibly empty) will contain the candidate answers to the query [8].

For non-interactive systems, deciding what information is to be conveyed in the text is much less straightforward. Such systems require a mechanism for maintaining a consistent and focussed topic of discourse while guarding against lapses into monotony [5]. An effective content mechanism for a story generating system can be designed which imitates the psychological phenomenon of a mental picture—an image of the story's current situation maintained in the mind of the speaker. As events occur in the story, their continuity and plausibility are checked against the mental picture before the image is permanently altered with the appropriate consequent effects. The image often contains details that are not explicitly stated in the discourse, but which play an important role in controlling contextual fidelity [26].

A computer program for generating story text can simulate this mental picture with a data structure that contains a full description of the current story situation [18]. Information within this structure is updated as the story events take place. A frame-based analysis of this context model allows those data elements that undergo change to be regarded as salient story features by the text generator [31].

2.1 Production rules and creativity

The transformation of data changes into sentences is achieved by production rules. Story events and data objects are matched with appropriate lexical entries, using thematic and grammatical specifications to ensure such things as tense and number agreement. The entries are subsequently instantiated into specific syntactic prescriptions from an arbitrarily complex phrase-structure grammar. Further embellishment operations may also be applied to perform operations like pronoun substitution [10].

Though such rules are able to generate readable text from changes to the context model, they alone will not produce text with a high degree of semantic interest. Clearly the "creativity" of the story generator will ultimately depend on the mechanism responsible for bringing about the changes to the story situation. The ordering of the expressions generated must reflect reasoned sequences of events within the story, and follow a logical presentation of germane information.

TAILOR uses production rules and a context model to generate text that reflects salient story points [29]. The order of presentation for this information is determined by sequencing story events using a problem-solving approach in which characters are expected to satisfy goals [30]. The characters choose events that will help them achieve their objectives, and as the events occur they bring about changes to the story's context model. Text is generated as a report of these events, and a coherent plot emerges as a by-product of the characters' planning processes.

The underlying design of the planning mechanism rests on an analogy between story telling and game playing. Before the function of this mechanism can be properly explained it is necessary to identify how each aspect of a story has its equivalent in a game.

3 The story game infrastructure

Wittgenstein observed that when people engage in discourse they do so by playing what he called the *language game* [32, 14, 3]. He postulated that people issue utterances and follow topics of discourse in manners that support their conversational objectives. They accomplish this by constructing and modifying plans for speech acts in just the same way as they play games.

Wittgenstein's metaphor works particularly well with respect to narratives. The beginning of a story is similar to an opening game position prior to play. For both games and stories there is a series of events between the beginning and the end which lead from the initial state to the final one. In a game these events are the moves of the players. In a story they are the events that cause transition from one situation to another. The participants in the game have their counterparts in the characters within the story, and the playing area is the environment in which all events of the story occur and all character strategies either succeed or fail.

3.1 The characters

In keeping with the tradition of story-telling programs [16, 22], the characters of TAILOR-made stories are personified animals (though this tradition probably derives from the amount of story-ness animals can lend to a text). Character types are defined by predicates which outline the respective attributes of a few arctic animals. Figure 1 shows three of the character predicates and how they define a type of animal by including its eating habits, abode, and a set that specifies its methods of locomotion.

```
animal(polarbear,carnivorous,cave,{walk,swim}).
animal(arctictern,omnivorous,nest,{fly}).
animal(caribou,herbivorous,copse,{walk}).
```

Figure 1: Some of the predicates that define character types.

TAILOR creates a principal character (protagonist) from the database of character types, and assigns that character a name. As with games, where each participant has at least one principal objective that must be achieved, the protagonist is provided with an initial predicament from which he will derive an implicit primary objective. Thus, if *Horace the hungry caribou* were chosen, his implicit goal would be to become *not hungry.* To satisfy this objective, the character must construct and modify a plan of action that promises to bring him to a successful goal state. Just as the game ends when a player's primary objective is obtained or a point is reached where its satisfaction becomes impossible, so the story ends when the protagonist arrives at a goal state, or at a non-goal state from which the primary objective is unattainable—a *tragedy*, as it were. The story text is generated as a recounting of the successes and failures experienced by the protagonist as his plan is executed.

3.2 The story space

TAILOR views a story as a path through a state space beginning in an initial state and ending in a final one, with intermediate situations corresponding to states that are subject to certain ordering restrictions. A *story space* is a graph whose nodes are situations and whose edges are events that allows transition from one node to another. Each story generated corresponds to one particular path through this story space network.

One function of the story space is to provide a collection of discrete locations in which events may occur—the setting. A small portion of the Canadian Arctic frontier is sustained by TAILOR and contains 22 distinct locations. Figure 2 shows how locations are defined with a predicate that includes a set specifying their contents, and a set of travel modes which permit access to them. Paths are not explicitly defined between locations. Instead the locomotive abilities of the characters determine what places are accessible from any given location.

The cango rule defined in Figure 2 establishes this connectivity relation. The rule is expressed as a logical statement, where the notation

```
a :- b, c.
```

has the declarative meaning

$$b \wedge c \Rightarrow a,$$

```
location(iceberg,{ice,fish},{swim,fly}).
location(tundra,{moss,berries},{walk,fly}).
location(shore,{rocks,twigs,fish},{swim,fly}).
location(village,{fire,inuit},{walk,fly}).
location(hill,{berries,twigs},{walk,fly}).
location(marsh,{moss},{walk,fly}).

cango(Agent,Dest):- animal(Agent,_,_,Modes),
                    location(Dest,_,Access),
                    intersect(Modes,Access).
```

Figure 2: Some location predicates and the connectivity rule.

a, b and c being logic predicates with the appropriate number of arguments. Names that begin with lower case letters are constants, and those beginning with upper case letters are variables. Free variables are implicitly existentially quantified, and a '_' is used as a "don't care" variable. Thus the cango condition is true for an Agent wanting to go to location Dest if the Modes of locomotion for the Agent have a non-empty intersection with the Access modes for that location. Because the eating habits and abode of the animal and the contents of the location are not relevant for this rule they are left as "don't care" variables.

After the protagonist has been created and assigned an initial predicament, he is placed at a randomly chosen location from which to begin his search for a solution. A portion of this solution will involve moving from one location to another. But the story space network also includes states wherein something other than the location is changed. Some items within a particular location can be moved to another, changing the current location's contents and inducing a change of state. Moreover, the execution of an action that neither affects the contents of the current location nor moves the protagonist to another location may still bring about a change in the story situation. For instance, when the character looks for something in a particular location the situation is changed abstractly from a novel environment to a familiar one. Thus, state changes are effected through execution of the protagonist's plan. The plan itself is a list of *primitive actions* whose occurrence will cause changes, whether subtle or conspicuous, to the current situation.

3.3 The primitive actions

The concept of a primitive action derives from Schank's theory of conceptual dependency. This proposes that all specific actions can be generalized to one of a handful of basic primitives according to their characteristic similarities [21]. For example, the acts of throwing, picking up, and walking are all related in that they reflect the

```
primitive(travel,Agent,Action,Dest):-
                        cango(Agent,Dest),
                        location(Dest,_,Access),
                        animal(Agent,_,_,Modes),
                        member(Action,Access),
                        member(Action,Modes).
```

Figure 3: The *travel* primitive.

physical movement of something from one place to another. Conceptual dependency theory generalizes these actions as specific instances of physical transfer—the PTRANS primitive. This allows the entities involved to be viewed as sharing the same dependency relations (i.e. some agent moved something from somewhere to somewhere else). Abstract movements, like the transfer of control or possession, are generalized under the ATRANS primitive, and the transfer of knowledge, information, and other mental concepts are reduced to MTRANS—mental transference. Further primitives are defined for eating and drinking (INGEST), hearing and seeing (ATTEND), and several other general categories of activity, each of which is intended to permit meaningful inferences to be made.

Like conceptual dependency theory, TAILOR embodies a set of primitives that will erase the subtle differences between actions that might otherwise be considered distinct according to more superficial characteristics. For example, the characters of TAILOR stories move from location to location by flying, walking, or swimming; but each of these forms of transportation is merely a specific instance of the more general action of travelling. Any action, when bound to the current story situation, constitutes an event that causes a transition from that situation to a new one. To prevent the story from slipping into redundancy, each event must effect an entirely different transition. The reduction of a specific action to its primitive correlative is, therefore, essential if the generator is to prevent the occurrence of an event that is fundamentally the same as some prior event.

The primitive actions of TAILOR are grouped into five abstract categories: 1) `travel`, for getting about, 2) `speak`, for asking or telling, 3) `seek`, for looking and finding, 4) `need`, for recognizing deficiencies, and 5) `employ`, for using items found in the story environment. For any primitive to become a candidate event for the story it is necessary to determine whether or not sufficient conditions exist in the current situation to permit that action to be executed. Each primitive is defined as a predicate which evaluates these preconditions and (if satisfiable) returns a specific action. For example, the `travel` primitive in Figure 3 is permissible if the `Agent` is able to reach the location `Dest`. Once the `cango` rule is satisfied, a specific `Action` from the location's `Access` modes—one that is also included in the animal's `Modes` of locomotion—is returned. Since any number of primitive actions may be permissible

```
initialplan(hungry,[need,look,find,eat]).

primitive(need,_,need,_).

primitive(eat,Agent,eat,Locale):-
                     location(Locale,Contents,_),
                     animal(Agent,Diet,_,_),
                     eats(Diet,Contents).

eats(herbivorous,Foods):- member(moss,Foods).
```

Figure 4: The *hungry* plan and some essential information.

for a given situation, heuristics are used to create a plan of events that promises to bring the protagonist closer to a solution state.

4 Planning stories

At the time the protagonist is assigned an initial predicament he is also supplied with a minimal plan for achieving his goal. As was previously mentioned (section 3.2), the plan is a list of primitive actions which, when realized as story events, will satisfy the protagonist's primary objective. The plan is *minimal* in that it assumes ideal circumstances for its execution. However, though some actions of the initial plan are trivially effectible, there are others whose preconditions are not satisfied. For example, Figure 4 shows that the initial plan for *hungry* begins with the primitive *need* which has no preconditions and is consequently always executable (Note: the notation `[...]` denotes an ordered sequence of items). The final action of the initial plan, however, is the action *eat* which requires that the current location of the protagonist contain food agreeable to his diet. If this requirement cannot be immediately satisfied then the plan must be modified by the addition of further actions (perhaps `travel`) whose execution will correct the deficit.

4.1 Modifying an existing plan

When "game plans" fail to bring about desired results for a player, she must be able to revise her plan in search of an alternative solution. Typically, there will be several options available, requiring the player to have a number of heuristics that can be applied to select the most promising plan of action. Similarly, the characters of TAILOR must use heuristics to select an appropriate series of primitives from a variety of possible plan modifications.

```
evaluate(_,Agent,Locale,Value):-
                (setof(Friend,animal(Friend,_,_,_),Helpers),
                 remove(Agent,Helpers,Help),
                 location(Locale,Contents,_),
                 intersect(Contents,Help),
                 Value = 7) ; Value = 0.

evaluate(Problem,Agent,Locale,Value):-
                 animal(Agent,_,Home,_),
                 assoc3(Problem,Home,Remedy),
                 location(Locale,Contents,_),
                 findall(Val,eval3(Contents,Remedy,Val),Vals),
                 sum(Vals,Value).

assoc3(homeless,nest,twigs).

eval3(Items,Item,Val):- (member(Item,Items), Val = 9) ; Val = 0.
```

Figure 5: Two of the evaluation functions.

TAILOR uses a set of evaluation functions to search the story space for a solution path. These functions are applied to all possible next states which could result from the execution of each permissible primitive action for the current situation. The functions derive a numerical value for each possible next state by assessing the respective values of the state contents. The sum of these values is assumed to reflect the degree to which a character believes a particular action could be a relevant part of a solution. For example, the first `evaluate` function expressed in Figure 5 assesses the value of a state according to whether or not there is someone at that location who could be asked a question. The set of location contents is checked against the set of all characters (excluding the `Agent` doing the assessment) to see if they have anything in common. If they do, the state's overall value is increased by 7; otherwise no value is added (the semicolon denotes logical "or"). The second `evaluate` function in Figure 5 assesses how much help each of the location's contents will offer to the `Agent` for solving his `Problem` by rating each item according to its association with a `Remedy`. Once all evaluation results have been calculated for each permissible action, the highest valued next state is treated as the one that the `Agent` believes to be most desirable.

The evaluation process is repeated for each permissible action from all reachable next states; thus endowing the `Agent` with a degree of foresight. Since an exhaustive search of the story space is computationally intractable, the evaluation is only performed for states which lie within a fixed distance from the current situation. The

```
PLAN:[need,look,find,eat]
STORY BOARD:[]

PLAN:[look,find,eat]
STORY BOARD:[<shore,horace,need,moss>]

PLAN:[travel,look,find,eat]                              plan is modified
STORY BOARD:[<shore,horace,need,moss>]

PLAN:[look,find,eat]
STORY BOARD:[<shore,horace,need,moss>,<shore,horace,walk,tundra>]

  etc., etc., . . .
```

Figure 6: Plan primitives are transformed into story events.

sequence of events which will bring the story to the highest valued state is then added to the plan and the next primitive on the list is executed. The evaluation process is repeated after each event to permit a plan that had previously been regarded as optimum to be devalued in the light of new information.

4.2 The story board

Each event of a story is represented as a data structure containing the specific action, the agent who performed it, the location from which it occurred, and the direct object for the action. When a primitive action is executed it is removed from the plan list and its corresponding event information is recorded in the *story board* list. The story board serves two purposes. It is the record of all story events 1) from which the text is ultimately generated, and 2) against which all permissible events can be checked for uniqueness. Figure 6 shows how each element of a plan list is transformed into an event data structure and added to the end of the storyboard.

Given that TAILOR regards a story as any path through the story space network, it is essential to put in place a mechanism that safeguards against infinite loops. This is accomplished by insisting that each event of the story be unique. That is, for any action to be considered a candidate event for the current situation of the story, it cannot have occurred previously under the same circumstances. Uniqueness is evaluated by constructing an event data structure corresponding to the action and checking that the structure does not appear in the story board list. This means that at least one of the structure's data fields (either the action under consideration, the agent who would perform it, the location from which it would occur, or its direct object) must be different from those of all prior events in order for that action to qualify as a candidate transition. The story ends either when the plan list is empty (which indicates that the protagonist has satisfied his primary objective) or when the

```
sentence     ->     sub-phrase,verb-part,obj-phrase
                ||  sub-phrase,verb,proper-noun,wh-phrase
                ||  sub-phrase,verb,proper-noun,decl-phrase

sub-phrase   ->     proper-noun
                ||  pronoun

verb-part    ->     verb
                ||  verb,particle

obj-phrase   ->     noun
                ||  quantifier,noun
                ||  det,noun,prep-phrase

prep-phrase ->      prep,det,locative

wh-phrase    ->     interrogative,decl-phrase

decl-phrase ->      declarative,aux,obj-phrase
```

Figure 7: The phrase structure grammar of TAILOR.

story arrives at a situation for which there is no unique, permissible event (which indicates that the protagonist failed to reach a goal state).

4.3 Generating story text from the story board

After all events have been determined, the text-generating mechanism transforms the completed story board into a corresponding sequence of simple,[1] past tense sentences. Each event data item is removed from the story board and its contents are paired with appropriate words from the lexicon. The words are then mapped on to a syntactic structure in accordance with certain grammatical constraints—such as number and person agreement, and thematic roles. Figure 7 outlines a syntactic description of the phrase structures available in TAILOR.

The selection of a syntactic structure is constrained by the action being expressed—in particular, by the valency requirements[2] of the correlative verb. For example, a syntactic structure which permits the use of a *wh-phrase* can only be used with the *speak* primitive, and more specifically with the verb *asked*. Instances where more than

[1]The term *simple* is used to denote a class of sentences which have a single subject and a single predicate with no subordinate or coordinate clauses.

[2]The valency requirements for a verb are its transitivity and thematic constraints.

Events	*Sentences*
	`once upon a time there`
	`was a caribou named wally.`
	`wally was cold.`
`[iceberg,wally,need,fire]`	`he needed a fire.`
`[iceberg,wally,walk,tundra]`	`he walked to the tundra.`
`[tundra,wally,look,fire]`	`wally looked for a fire.`
`[tundra,wally,find,fire]`	`he found no fire.`
`[tundra,wally,walk,village]`	`wally walked to the village.`
`[village,wally,look,fire]`	`he looked for a fire.`
`[village,wally,find,fire]`	`he found a fire.`
`[village,wally,warm,fire]`	`wally warmed himself by the fire.`
...	`and lived happily ever after!`

Figure 8: The one-to-one correspondence between events and sentences.

one syntactic pattern is permissible result in a random selection from the possible alternatives (e.g. where a pronoun or proper noun can be used to refer to the subject of the sentence). The resulting text reflects a one-to-one relationship between each event of the story board and each sentence in the body of the story—as the example of Figure 8 exemplifies. The exceptions to this process are the first two sentences, which are produced during the character and setting selection procedure, and the final one, which is generated in accordance with the state of the plan list at the story's conclusion.

4.4 Introducing conflict into the plot

So far we have examined how TAILOR uses its planning mechanism for generating stories of the simplest kind—single character stories. But automated game playing systems are most often concerned with issues of strategy development in multi-player games. Techniques like the state-evaluation heuristics described in section 4.1 were developed as methods for plan construction in the face of opposition. In fact, one might argue that the most interesting aspect of games in general is having to develop a strategy for success in a situation where someone at cross purposes is both able and eager to interfere with one's objective.

Most stories use a similar notion of conflict to develop plots with greater interest and higher levels of intricacy. In keeping with the game analogy, TAILOR brings conflict to its stories by introducing a character into the story space who will interact with the story's hero in ways which will impede the hero's search for a solution path.[3]

[3]The analogy being expressed here is between stories and a particular class of games called *zero-sum perfect-information* games (of which chess and tic-tac-toe are good examples).

The antagonist arrives in the story at the location of the current situation only after several events have transpired. To facilitate the interaction of the antagonist and the protagonist, the story planner allows the characters to take turns at selecting primitive actions for the story events. Both characters use the same heuristic functions for evaluating next states, but the antagonist does so with the purpose of directing the story to low-valued situations. For this reason, when the protagonist is searching the story space for an optimum plan of action from the current situation, he must bear in mind that the next event will be determined by his opponent and, thus, will most probably not be in his best interest.

If the purpose of the protagonist is to select a sequence of events that will maximize his chances of success, then the purpose of the antagonist is to select events which will minimize those chances. Therefore, when the protagonist searches for a maximum valued reachable state, he must take into account that the antagonist will ensure that every other event will cause transition to a minimum valued situation. This so called *minimax* principle is a standard technique used in computer game-playing systems [28] for preventing the planning mechanism from overlooking long term consequences for short term benefits.

To ensure that the characters interact, it is sometimes necessary for the antagonist to take several consecutive turns. Since he has no means of keeping and modifying a plan of his own, he must make static evaluations at each turn. By examining the protagonist's plan, he tries to ascertain which of the primitive actions available to him will guide the story to a least favourable situation for the hero. If the antagonist is unable to bring the story to a state of lower value than the current situation, and if there is a primitive action available to him whose occurrence would constitute a unique event, he is allowed to initiate another consecutive story event.

In principle, all primitive actions are executable by either character, providing the relevant preconditions are satisfied. In practice, however, there are some actions which are of use only to one character or the other. For example, the mental action of *concealing* usually precedes the abstract action of *telling* a lie. Since such nefarious actions are always perceived to reduce the hero's chances of finding a solution, there is no situation in which the protagonist will select such actions. Good guys never lie! Furthermore, some primitive actions are explicitly defined to give higher or lower next state values. For example, the action *ask* is considered to be the best way of finding a solution path and is consequently often chosen when its preconditions are satisfiable. Figure 9 shows an example of a two-character story generated by TAILOR.

5 Conclusions

This research demonstrates a system that plans speech acts and provides a means for modifying its plans as the discourse unfolds. Furthermore, it shows how a story can be generated by sustaining a microcosm that realizes the propositions of the discourse. It illustrates how the story-telling process can capitalize on the richness

```
once upon a time there was an arctictern named truman.
truman was homeless. truman needed a nest. he flew to
the shore. truman looked for some twigs. truman found
no twigs. he flew to the tundra. he met a polarbear
named horace. truman asked horace where there were some
twigs. horace concealed the twigs. horace told truman
there were some twigs on the iceberg. truman flew to the
iceberg. he looked for some twigs. he found no twigs.
horace walked to the shore. he swam to the iceberg.
horace looked for some meat. he found some meat. he ate
truman. truman died.
```

Figure 9: A two-character tale of conflict.

of the environment, in terms of pre-prepared mental pictures of locations, to enrich the stories themselves by adding relevant detail to them. It provides a method of creating text that follows a logical sequence of natural language expressions from the initiation of a discourse to a reasonable conclusion—without external stimulation or human intervention.

There are a number of fairly obvious ways to improve the quality of the text the system generates. These range from the introduction of simple lexical rules for capitalization, through the connection of sentences by conjunctions, both coordinating (e.g. "and" and "but") and subordinating (e.g. "if", "when", "because"), to the use of more elaborate phrase-structure rules to generate more sophisticated sentence structures. However, such improvements would be relatively superficial.

A more serious criticism of the system is that no clear distinction is made between the story itself and the telling of it. TAILOR first generates a simulated event sequence within the story space and then reads it out item by item. The story slavishly relates the events in the order in which they occur. This has particularly serious consequences in multi-character stories, for the actions of each character are interleaved even when they do not interact. A focus-of-attention mechanism would separate the event sequence from the story and allow the story-teller more latitude to separate parallel event sequences, suppress irrelevant actions, and even to apply rhetorical tricks like concealing crucial events until long after they have occurred.

The Turing test is a game between two players, a person and a computer, that takes all initiative away from the players and places it in the hands of an interrogator. In contrast, Wittgenstein's idea of a language game focuses attention on the initiative required to plan and execute speech acts to serve one's goals, and the central role that this plays in our interactions. Following this view, the methods that TAILOR employs for creating, modifying and merging speech act plans for discourse participants—which, though illustrated here in a severely limited context, are quite general—may have application in other language domains.

Acknowledgements

We gratefully acknowledge the stimulating research environment provided by the Knowledge Science Lab at the University of Calgary. This research is supported by the Natural Sciences and Engineering Research Council of Canada.

References

[1] D. Appelt. Planning english referring expressions. *Artificial Intelligence*, 26:1–33, 1985.

[2] J. L. Austin. *How to do things with words.* Oxford University Press, 1962.

[3] G. P. Baker and P. M. S. Hacker. *Wittgenstein: Understanding and meaning.* Basil Blackwell Publisher Ltd., Oxford, 1980.

[4] M. Bratman. *Intentions, Plans, and Practical Reason.* Harvard University Press, Cambridge, Massachussetts, 1986.

[5] Bertram C. Bruce. Generation as a social action. In *Theoretical Issues in Natural Language Processing-1*, pages 64–67. Association for Computational Linguistics, 1975. Urbana-Champaign.

[6] P. Cohen and C. R. Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.

[7] J. A. Fodor. *The Language of Thought.* Harvard University Press, Cambridge, Massachussetts, 1975.

[8] R. Frost and J. Launchbury. Constructing natural language interpreters in a lazy functional language. *The Computer Journal*, 32(2):108–121, 1989.

[9] B. Green, A. Wolf, C. Chomsky, and K. Laugherty. Baseball: An automatic question answerer. In E. Feigenbaum and J. Feldman, editors, *Computers and Thought*, pages 207–216. McGraw-Hill, New York, 1963.

[10] Ralph Grishman. *Computational Linguistics.* Cambridge University Press, Cambridge, 1986.

[11] Barbara J. Grosz, Karen Spark Jones, and Bonnie Lynn Webber, editors. *Readings in Natural Language Processing.* Morgan Kaufmann Publishers Inc., Los Altos, Calif., 1986.

[12] J.. Hoepelman. On questions. In F.. Kiefer, editor, *Questions and Answers.* D. Reidel Publishing Co., Dordrecht, 1983.

[13] J. J. Katz and P. M. Postal. *An Integrated Theory of Linguistic Descriptions.* The MIT Press, Cambridge, Massachussetts, 1964.

[14] Saul A. Kripke. *Wittgenstein on Rules and Private Language.* Harvard University Press, Cambridge, MA, 1982.

[15] K. McKeown. Discourse strategies for generating natural language-text. *Artificial Intelligence*, 27:1–42, 1985.

[16] J. R. Meehan. Tale-spin, an interactive program that writes stories. In *Proceedings Fifth International Joint Conference on Artificial Intelligence*, pages 91–98, Cambridge, Mass., August 1977.

[17] R. C. Parkison, K. M. Colby, and W. S. Faught. Conversational language comprehension using integrated pattern-matching and parsing. *Artificial Intelligence*, 9:111–134, 1977.

[18] Frank Rose. *Into the heart of the mind.* Vintage Books, New York, 1985.

[19] D. E. Rumelhart. Notes on a schema for stories. In D. G. Bobrow and A. Collins, editors, *Studies in Cognitive Science*, pages 122–137. Academic Press, New York, 1975.

[20] G. Salton. *Automatic text processing.* Addison-Wesley, Reading, Massachussetts, 1989.

[21] R. Schank and R. Abelson. *Scripts, Plans, Goals and Understanding.* Lawrence Erlbaum Assoc., Hillsdale, NJ, 1977.

[22] R.C. Schank and C.K. Riesbeck. *Inside computer understanding: five programs plus miniatures.* Lawrence Erlbaum Associates, 1981.

[23] J. R. Searle. *Speech Acts: An Essay on the Philosophy of Language.* Cambridge University Press, Cambridge, 1969.

[24] J. R. Searle. A taxonomy of illocutionary acts. In K. Gunderson, editor, *Language, mind, and knowledge.* University of Minnesota Press, 1976.

[25] R. F. Simmons. Natural language question-answering systems:1969. *Communications of the ACM*, 13:15–30, 1969.

[26] P. Thorndyke. Cognitive structures in comprehension and memory of narrative discourse. *Cognitive Psychology*, 9:88–110, 1977.

[27] A. M. Turing. Computing machinery and intelligence. *Mind*, 59:433–460, October 1950.

[28] J. von Neumann and O. Morgenstern. *Theory of games and economic behaviour*. Princeton University Press, Princeton, New Jersey, 1944.

[29] Robert Wilensky. Points: A theory of the structure of stories in memory. In W. Lehnert and M. Rengle, editors, *Strategies for Natural Language Processing*, pages 345–374. Lawrence Erlbaum and Assoc., 1982.

[30] Terry Winograd. *A Procedural Model of Language Understanding*, pages 152–186. W. H. Freeman and Company, New York, 1973. Schank, R. and Colby, K, eds.

[31] Patrick Henry Winston. *Artificial Intelligence*. Addison-Wesley, Reading Mass., 1984.

[32] L. Wittgenstein. *Philosophical Investigations*. Basil Blackwell Ltd., Oxford, UK, 1953.

[33] W. A. Woods. Procedural semantics for a question answering system. In *1968 AFIPS Conference Proceedings*, volume 33, pages 457–471, 1968.

[34] W. A. Woods. Transition network grammars for natural language analysis. *CACM*, 3(10):591–606, 1970.

[35] W. A. Woods. Semantics and quantification in natural language question answering. *Advances in Computers*, 17:2–64, 1978. Academic Press, New York.

[36] W. A. Woods, R. M. Kaplan, and B. Nash-Webber. The lunar sciences natural language information system. Final report 2378, Bolt, Beranek and Newman, Cambridge, Mass., 1972.