TALE-SPIN, AN INTERACTIVE PROGRAM THAT WRITES STORIES

James R. Meehan
Dept. of Information and Computer Science
University of California, Irvine
Irvine, California 92717

INTRODUCTION

In a global view, TALL-SPIN is a program that simulates rational behavior by characters in the world. The simulator-has three active components. There is a problem solver that, given a goal, produces other goals (subgoals) and actual events. There is an assertion mechanism that takes an event and adds it to "memory": trie static description oi each Knowledge area, what the physical world looks liKe at that instant, what social relationships exist between characters at that instant, ana so on. Finally there is an mlerence mecnanisin that produces trie consequences of an event. when an event is asserted, its consequences are computed and asserted, tneir consequences are computed and asserted, and so on. One kind ot consequence is a goal. Goals are asserted, not by adding them to memory, but rather by calling the problem solver, which completes the cycle. This approach to storytelling is very different from the work of Klein, as recently di scussed [2].

Sample output from TALL-SPIN:

ONCL UPON A TIME GEORGE ANT LIVED NEAR A PATCH OF GROUND. THERE WAS A NEST IN AN ASH TREE. WILMA BIRD LIVED IN THE NEST. THERE WAS SOME WATER IN A RIVER. WILMA KNEW THAT THE WATER WAS IN THE RIVER. GEORGE KNEW THAT THE WATER WAS IN THE RIVER. ONE DAY WILMA WAS VERY THIRSTY. WILMA WANTED TO GET NLAR SOME wATtR. wILNA FLEW FROM HER NEST ACROSS A MEADOW THROUGH A VALLbY TO THE RIVER. wILMA DRANK THE WATER. WILMA WAMM 'T THIRSTY ANY MORE.

GLGRGB WAS VERY THIRSTY. GEORGE WANTED TO GET NEAR SOME wATER. GEURGE WALKED FROM HIS PATCH OF GROUND ACROSS THE MEADOW ThKOUGH THE VALLEY TO A RIVER BANK. GEORGE FELL INTO THE WATER. GbuKGE WANTED lu GET NEAR THE VALLEY. GEORGE COuLuN'I bET NEAR THE VALLEY. GEuRGE WANTED TO GET NEAR ThL MLADUW. GLUKGL COULDN'T GET NEAR THE i^EADGw. WILMA WANTED GEORGE TO GET NEAR THE MEADOW. WJLMA WANTLD TO GET NEAR GEORGE. WILMA GRABBED GEURGE WITH HER CLAW. wILMA TOuK GEORGE FROM THE RIVER THROUGH THE VALLLV TO THE MEADOW. GEORGE WAS DEVO'IED TO WILMA. GEORGE OwED EVERYTHING TO WILMA. WILMA LET GO OF GEORGb. GEORGE FELL TO THE MEADOW. THE END.

The user created the characters and the river, and then gave each character a problem to solve — thirst. No information regarding the social relationship between George and Wilma was necessary: all characters are automatically motivated to rescue anyone who they know is in danger of death. Both George and wilma knew that he would have drowned had he stayed in the water. George's sudden devotion to wilma is a consequence of being rescued by her.

THE *Nt^LD* FOR KNO^LEDGE£

SOME MIS-SPUN TALES

"Correct" stories are part of this project's goal, but they're not always trie most interesting thing for people to read a successful solution often makes the problem look easy. To illustrate tiie need for knowledge in generating "correct" stories, 1 present here some stories, produced in TALE-SPIN's early days, that came out "wrong" because of some lack of knowledge, often of the most mundane variety and usually taken for granted. Since these don't concern the English generator, I'll use my own translat ions.

(1) "One day Joe Bear was hungry. He asked his friend Irving Bird where some honey was. Irving told him there was a beehive in the oak tree. Joe threatened to hit Irving if he didn't tell him where some honey was."

Joe has not understood that Irving really has answered his question, albeit indirectly. You've got to know about beehives in order to understand that the answer is acceptable. Also, it's polite to give some details when you answer a question. ("Do you know what time it is?" "Yes.")

(2) "One day Joe Bear was hungry. He asked his friend Irving Bird where some honey was. Irving told him there

was a beehive in the oak tree. Joe walked to the oak tree. he ate the beehive."

A further refinement is to unscramble an acceptable answer in the proper fasnion, remembering a little better what the original question was.

All tne action used to focus on a single character, other characters could respond to him only in very limited ways, when they were asked direct questions, tor example. There was no concept of "noticing": It people walK into your room, they needn't always announce their presence. You see them. The following story was an attempt to produce "The Ant and the Dove," one of the Aesop fables.

(3) "henry Ant was thirsty. He walked over to the river bank where his good friend Bill Bird was sitting. Henry slipped and fell in the river. He was unable to call for help. He drowned."

That wasn^t supposed to happen. Falling into the river was deliberately introduced to cause the central "problem" of the story. Had Henry been able to call to Bill for help, Bill would have saved him, but I had just added a rule that said that being in water prevents speech, which seemed reasonable. Since Bill was not asked a direct question, he didn't notice his friend drowning in the river.

Here are some rules that were in TALE-SPIN when the next horror occurred. If A moves B to location C, we can infer not only that B is in location C, but that A is also. If you're in a river, you want to get out, because you'll drown if you don't. If you have legs, you might be able to swim out. With wings, you might be able to fly away. With friends, you can ask for help. These sound reasonable. However, when I represented "X fell" as "gravity moved X," I got this story:

(4) "Henry Ant was thirsty. He walked over to the river bank where his good friend Bill Bird was sitting. Henry slipped and fell in the river. Gravity [having neither legs, wings, nor friends] drowned."

It didn't know enough about gravity, obviously. But adding knowledge can produce unexpected results. After characters became aware of their environment, noticing people and things around them, this story appeared:

(5) "Once upon a time there was a dishonest fox and a vain crow. One day the crow was sitting in his tree, holding a piece of cheese in his mouth. He noticed that he was holding the piece of cheese. he became hungry, and swallowed the cheese. The fox walked over to the crow. The end."

That was supposed to have been "The Fox and the Crow," of course. The fox was going to trick the crow out of the cheese, but when he got there, there was no cheese, because the crow "noticed" that he had some food in his mouth, from which the program inferred that he'd be hungry. (The fix was to assert at the beginning that the crow had eaten recently, so that even when he noticed the cheese, he didn't become hungry.)

(6) "Joe Bear was hungry. tie asked Irving Bird where some honey was. Irving refused to tell him, so Joe offered to bring nim a worm if he'd tell him where some honey was. Irving agreed. But Joe didn't know where any worms were, so he asKed Irving, who refused to say. So Joe offered to bring him a worm if he'd tell him where a worm was. Irving agreed. But Joe didn't know where any worms were, so he asked Irving, who refused to say. So Joe offered to bring him a worm if he'd tell him where a worm was...."

Lesson: don't put a goal on the stack if it's already there. Try something else. If there isn't anything else, you can't achieve that goal.

Here are some more rules. If you're hungry and you see some food, you'll want to eat it. If you're trying to get some food and you fail, you get sick. If you want some object, try bargaining with the object's owner. Innocuous, right?

(7) "One day Henry Crow sat in his tree, holding a piece of cheese in his mouth, when up came Bill Fox. Bill saw the cheese and was hungry. [Hunger is now on his goal stack.] He said, 'Henry, I like your singing very much. won't you please sing for me?' Henry, flattered by this compliment, began to sing. The cheese fell to the ground. Bill Fox saw the cheese on the ground and was very hungry. [Hunger is about to be added to his goal stack.] He became ill. [Because hunger was already on his goal stack, he couldn't get the food, so he gets sick.] Henry Crow saw the cheese on the ground, and he became hungry, but he knew that he owned the cheese. He felt pretty honest with himself, so he decided not to trick himself into giving up the cheese. He wasn't trying to deceive himself, either, nor did he feel competitive with himself. But he did dominate himself, and was very familiar with himself, so he asked himself for the cheese. He trusted himself, but he remembered

tnat ne was also in a position of dominance over himself, so he refused to give himself the cheese. He couldn't think of a good reason why he should give himself the cheese [if he did that, he'd lose the cheese], so he offered to bring himself a worm if he'd give himself the cheese. That sounded okay, but he didn't know where any worms were. So he said to himself, 'Henry, do you know where any worms are?' But of course, he didn't, so he .... [And so on]"

WHAT DOES TALE-SPIN KNOW?

TALE-SPIN Is a problem solver, top-down and goal-directed. Its output may be regarded as a trace through problem-solving procedures. Problems are associated with a particular area of knowledge, or what I call a problem domain, which is defined by a set of representational primitives, a set of goalstates or problems expressed in terms of those primitives, and procedures tor solving tnose problems.

PLANS

The most common of the domains in TALE-SPIN uses the primitives oi Roger Schank's Conceptual Dependency [5], and its problems and procedures come from a theory of plans by Roger Schank and Robert Abelson [6]. TALE-SPIN uses these plans to generate stories; Wilensky's PAM system [8] uses them for understanding stories.

The first kind of plan is call a delta-act, and they are defined in terms of a particular goalstate. DELTA-PROX(X,Y,Z) is the delta-act that is invoked when X wants Y to be near ("prox") Z. Similarly, DELTA-CONTROL(X,Y,Z) means X wants Y to control Z, DELIA-KNOW(X,Q) means that X wants to find out the answer to the question Q, and TELL(X,Y,Z) means that X wants Y to know Z. There are "negative" counterparts to these: DELTA-NEG-PROX(X,¥,Z) means that X wants Y to get away from Z, and so on. Each delta-act consists of a number of techniques called planboxes tor achieving the desired goal, and an algorithm for deciding the order in which to try the planboxes. The planboxes have a variety of pre- and post-conditions on them, and if successful, they eventually produce an event of some sort.

Here's what the first planbox in DELTA-PROX looks like:

```
Planbox 1: X tries to move Y to Z
  preconditions:
    X is self-movable
    If X is different from Y,
       then DPROX (X, X, Y)
            and DO-GRASP (X, Y)
    DKNOW (X, where is Z?)
    DKNOw (X, where is X?)
    DLINK (X, loc (Z))
  act: DO-PTRANS (X, Y, loc (Z))
  postcondition:
    Is Y really at Z? (DKNOW could have
       goofed)
  postact: If X is different from Y,
              then DO-NEG-GRASP (X, Y)
```

X will try to move Y to Z. If X is not the same as Y, then X must get himself near Y, and then somehow grab on to Y, so that Y will be carried along.

A precondition for Planbox 1 is that X be able to move himself. If that isn't the case, or if the planbox fails, we try another planbox.

The next precondition on Planbox 1 is that X find out where Z is. If Z is a fixed physical location, then nothing needs to be done; we assume that everyone knows where places are. If Z is not a location (Z might be a person, for instance), then we use the delta-act for finding out information, DELTA-KNOW.

The next precondition is that X find out where he himself is. X must know where he is in order to go anywhere. Including this test allows the possibility that X does not know where he is and must therefore find out. Thus, we can tell stories (solve problems) in which characters are lost.

The next precondition is that X be able to figure out a route to Z. The solution to this problem uses the knowledge about the physical world, which I discuss below in the section on MAPS.

Finally, we use DO-PTRANS to effect the actual movement, and if X had picked up Y, he now lets Y go. The DKNOW postcondition allows for the possibility that the DKNOW precondition may have been satisfied unfairly. In one of TALE-SPIN 's stories, Arthur Bear asks George Bird where some honey is. George lies, and Arthur believes him. As far as the first DKNOw was concerned, the precondition was satisfied: Arthur thinks he knows where the honey is. when he gets there, however, he discovers that the honey actually isn't there, so he goes back to do another DKNOW, hopefully more successful.

Other planboxes in DELTA-PROX include: X gets Y to move himself to Z; X gets Z to move himself to Y; X gets a third party to move Y to Z; X gets a third party to move Z to Y. Other planboxes, not yet implemented, include: X gets Y and Z to be in some common location? X gets some "system" to move Y to Z (gravity, the Post Office, whatever 's

appropriate).

Closely related to the delta-acts is the PERSUADE package, a handy collection of planboxes used by many delta-acts. These include simple requests, proposing a good reason, bargaining, and threatening.

It's easy to see how the plans relate to each other as subgoals. In order to get something (DELTA-CONTROL), you may need to find out where it is (DELTA-KNOW), which may require that you go ask someone (DELTA-PROX and TELL) who may bargain with you (PERSUADE), which may cause you to get sometning for him (DELTA-CONTRuL), and so on.

SIGMA-STATES: THE bUDILY NEEDS

Hunger, thirst, rest, an<5 sex are tour physical needs that are primitives in this domain. One of the ways in which these goals are most unlike the plans is that they arise as goals every so often, spontaneously. They call the plans as subgoals but are not themselves subgoals of any plan or of each other. For example, an obvious planbox under SIGMA-HUNGER requires the use of DELTA-CONTROL — for obtaining food — but for what end would becoming hungry be a means?

TALE-SPIN knows what kinds of food different characters eat, but it assumes that water is sufficient to quench anyone's thirst. (In some ways, it's still a very sparse world.) People sleep only at home, but they may "fool around" with whomever they please. (In other ways, it's not so sparse.)

The sigma-states are the top-level goals TALE-SPIN works on. After the user has chosen the initial setting and selected a main character, ne can choose one of the sigma-states as the first goal for that cnaracter. Are tnese stories, then, about nunger, thirst, and so on? Not necessarily. As we'll see in the section called STORIES, the focus or point of a story may be entirely separate from the first goal.

RELATIONSHIPS

TALE-SPIN characterizes relationships between people in terms of competition, dominance, familiarity, affection, trust, deceit, and indebtedness. It's perhaps harder in this domain than in any other to find a good set of primitives that are both necessary and sufficient. The relationships are used in two ways. First, they serve as preconditions to many of the planboxes, particularly in PERSUADE. Before you ask someone to do something for you, you have to be near him (DELTA-PROX) and so on, but you must also consider how you relate on a personal level. If you feel that you're competing with the other person, or if there's a substantial imbalance of power between you, or great differences in social class, or hatred, and so on, you won't ask. Second, they appear as consequences to many actions. When someone ATRANSes a gift to you, not only do you POSSess it, you probably feel some affection towards that person as well.

What about the problem-solving procedures in this domain? Well, they exist, certainly. "John wanted Mary to like him" — sounds okay. The difference is that they don't seem to exist as goals, but on^y as subgoals, the exact opposite of the sigma-states. Furthermore, there's a lot less certainty about them, while there are virtually guaranteed procedures for John to get from New York to London, there are no algorithms for John to get Mary to like him, which says no more than that the "physics" of the real world is simpler — or better understood — than the "physics" of the mind.

PERSONALITIES

To describe character traits, TALE-SPIN attributes degrees of kindness, vanity, honesty, and intelligence to its characters. These traits are used in a "solo" environment, having the least dependence, among the domains we've seen so far, on anything else in the world. If John has been asked to do a favor, and there are no good, "external" reasons for or against it, he may do it if he's a kind person. Compliments to a vain person will produce a greater reaction than to a modest person, as we'll see in "The Fox and the Crow." Dishonesty is a precondition to the "steal" planbox in DELTA-CONTROL. Insulting someone's intelligence produces a marked change in affection. All these reactions are part of TALE-SPIN'S inference mechanism.

These traits are undoubtedly useful but seem to be only mildly different from relationships, until you think about procedures for achieving personality traits. There aren't any, at least none that succeed over anything but long stretches of time, too long for them to be of use as subgoals. Otherwise, the following scenario would make sense:

"Mary needed a new dress and wanted to get some cash from her husband John. 'How can I do that?' she wondered. 'Oh sure, I'll just make him dishonest so he'll rob a bank. Gee, I better make him smart enough not to get caught. Okay, that'll work. John?'"

MAPS^ PHYSICAL SPACE

How far Is It from the 34th Street door

ot the Empire State Building to the north entrance to the Golden bate Bridge? think aoout that for a second. Now, how wide is tne continental USA? if you tnougnt about the answer to tne first question, chances are, you already thougnt about the answer to the second question before you read it.

Physical space is perhaps the easiest domain to model badly (that is, without any psychological validity), because our knowledge of mathematics tells us that we can represent points in 3-space with X-Y-Z coordinates and calculate distance and direction with some simple equations. Rather than do that, however, I've designed a representation that enables us to answer the same questions but also accounts for some phenomena such as the one just described.

I define a map to be the representation of a group of contiguous regions (submaps), a fixed-size picture, if you will, where all maps have approximately tne same resolution. My map of California includes a submap of Soutnern California, which nas a suomap for Irvine, whicn has a suomap for UCI, and so on.

The contents of a map depend on experience, having lived most of my life in Connecticut, my map for that state has some very detailed submaps. Never having been to Utah, however, I imagine it to be a vaguely rectangular area, perfectly blank, without so much as a single city on the map. (Of course, I might recognize that a given city is in Utah, but that's a different issue.)

TALE-SPIN uses maps to solve the problem of finding routes from one place to another. There are two kinds of route problems, inter-map and intra-map. To solve the inter-map problem, it finds the lowest-level map that includes both the origin and the destination as submaps, or submaps of submaps, and so on. Then it solves the intra-map problem of getting from one of the submaps of that map to the otner. Solving the intra-map problem requires that we represent adjacency, so that we can find the path from one to the otner. Crossing tne border between a map and one ot its adjacent maps requires that we mark points of intersection; I call tnem doors. The relationship of a street corner to a street is much like the relationship of a door to a room each is a way of getting from one region to an adjacent one — so both of them are "doors" in the map system.

Every door has two conceptual images, one for each region to which it is attached. People often use different terms to refer to a single door, and they know that doors are not symmetric, as in the case of a door locked from the outside but not the inside. Knowledge about doors can be incomplete — not knowing where it leads, for example. And sometimes we don't even realize that two doors are actually one and the same; street intersections approached along different streets may look entirely different.

In TALE-SPIN, each door has two names that are marked as being "equivalent"; each name represents a map. Every map points to the super-map to which it belongs and to a list of its submaps as well.

TALE-SPIN also uses an abstraction of maps called blueprints, which differ from maps by having a property of minimality. The map for a particular office building may have 17 submaps representing various floors of offices, but the blueprint for the generic office building contains only one sub-blueprint for each different kind of floor, say, one for basements, one for the ground floor, and one for office floors. Maps are instantiations of blueprints, and TALE-SPIN uses blueprints while interacting with the user to construct the maps for all physical spaces.

The goal structure, for example, started with the classical stack of goals. There are stories which require nothing more complicated. But by the end of the project, TALE-SPIN used a mechanism that was less elegant but was adequate for a larger class of stories.

(The goal structure is not entirely new. I freely and openly used ideas from systems like GPS [4], STRIPS [1], and HACKER [7]. But some of their issues do not arise in TALE-SPIN, and vice versa, making direct comparison difficult. While I struggle with, say, the diversity of representation, they work on triangle tables.)

The simple goalstack proves inadequate when we insist that no more effort be expended on achieving a subgoal than is allotted to achieving the goal. In particular, we must avoid trying to achieve a subgoal which is identical to the goal. A case where this was not done is the sixth mis-spun tale above.

Since there is more than one problem-solving character in a story, we need separate goal lists for each of them.

when a goal is successfully achieved, it is removed from the goal list. It a goal that was on the list tails, it is marked but not removed, lest it be tried again and again, leading to another mis-spun tale.

The current goal system is a tar cry from the simple stack, but there's more to come. Part of the ongoing research at UC1 on this project is to extend the capabilities of the goal calculus in a number of ways. First, the goal structure must indicate the various kinds of relations between goals: X is the subgoal of Y; X, ¥, and Z are concurrent goals; X, Y, and Z must be achieved in that order; X is subdivided into A, B, and C (for example, allocating portions of a sum of money to individual purchases); and so on.

Second, there are new goal processes: recognizing when a goal has been superseded; recognizing when a goal can safely be abandoned ("Joe Bear was on his way to ask Irving Bird where to find some honey when he suddenly came across a blueberry bush"); recognizing when a goal should be re-tried, in light of new information (when "mildly hungry" turns into "desperate for food").

Third, tnere is recovery, which is different from both achievement and preservation. Some expenditures, such as cash, can often be recovered; it you buy an item in a department store, and then decide (or find out) that you don't need it (for some other goal), you may be able to return it and get your money back, or part of it, or some equivalent. Time expenditures, on the other hand, are non-recover able.

This new work is related to knowledge about the nature of process, which we are also interested in modeling, and which, not surprisingly, begins to sound like ideas from automatic programming.

STORIES

When Is a text a story, and when is it simply a narration of rational events? what's a story about, and how can you tell?

> (1) One day Joe Bear was famished. He closed his eyes for three seconds. he wasn't hungry any more. The end.
> (2) One day Joe Bear was famished. There was a jar of honey right next to him. He ate the honey. The end.

TALE-SPIN starts a story by creating a problem for the main character, drawn from the four sigma-states mentioned earlier. Many subproblems may be encountered on the way, and there can be side-effects that are problems that need solving as well. The degree of difficulty is in part determined by the user. In specifying how well one character gets along with another, he can make the problem harder to solve, which also makes the story longer. In fact, I have observed that people who run the program usually make the problem very hard; they find the resulting "Trials and Tribulations" story more "interesting" (their word) than "Sweetness and Light" stories where all the characters like each other and do favors at the least suggestion.

But that approach confines the point of the story to the literal level. To go higher than that, we need to understand what story points are and how to model them.

I chose to work on the Aesop fables as examples of simple stories whose points, or morals, are not about animals,

but are, instead, concerned with general lessons, notions from a higher domain. To produce a story with a given moral, we must first understand the correspondence between the moral and the real-world level. For example, the moral of "The Fox and the Crow" is never trust flatter-ers ers. The analysis is as follows:

"Never do X" means that if you do X, then something "bad" will happen. "A flatters B" means that A says something "nice" to B, but is insincere, doing it for some ulterior motive. Since a consequence of saying something nice to B is that B will become more kindly disposed towards A, then it's reasonable to assume that B's kind disposition towards A will enable something to happen which is "good" for A. Putting this all together, we predict that A has some goal which requires that B be kindly disposed towards A, so A says something nice to B, B reacts accordingly, and something happens that causes A to achieve his goal and also causes B to sutler. Since transfer of ownership is an event that is simultaneously good for tne recipient and Dad for the unwilling donor, we know that DELTA-COwTROL can be the main problem, when the program operates in this mode, it sets some of the initial parameters of the world in such a way that when characters behave reasonably in that environment -- and reasonable behavior is what the simulator does — the desired story will result. In other words, to make a certain point in a story, it figures out in advance what some of the world model has to look like, how the stage must be set before the characters arr ive.

THE ENGLISH GENERATOR

As a problem cTomain, this one is unique in that it provides no information to the other domains, although it uses their information extensively. For example, in order to be able to say "Joe Bear r_e-t_urned to the cave" instead of "Joe Bear went to the cave," the generator needs access to the memory to check whether Joe Bear has been in that cave before. when describing someone's trip, it uses the MAP*system 's knowledge and expresses the route in detail; e.g., "Tom walked from the ground by the redwood tree across the meadow through the valley across a meadow to the patch of ground." But apart from making use of all the domains in order to find a better way to express something, the generator is very straightforward. It doesn't even use a grammar, given such a rich set of meaning representations to start with. Interested readers should consult the thesis [3] for details.

MORE STORIES

Here are two stories generated by TALE-SPIN, I present here a translation done by hand, for ease of reading. All the events in the story were produced by the program; only the English is mine.

The Fox and the Crow

"Once upon a time, there was a dishonest fox named Hernry who lived in a cave, and a vain and trusting crow named Joe who 1ived in an elm tree. Joe had gotten a piece of cheese and was holding it in his mouth. One day, Henry walked from his cave, across the meadow to the elm tree. He saw Joe Crow and the cheese and became hungry. He decided that he might get the cheese if Joe Crow spoke, so he told Joe that he liked his singing very much and wanted to hear him sing. Joe was very pleased with Henry and began to sing. The cheese fell out of his mouth, down to the ground. Henry picked up the cheese and told Joe Crow that he was stupid. Joe was angry, and didn't trust Henry anymore. Henry returned to his cave.

Joe Bear and Jack Bear

"Once upon a time, there were two bears named Jack and Joe, and a bee named Sam. Jack was very friendly with Sam but very competitive with Joe, who was a dishonest bear. One day, Jack was hungry. He knew that Sam Bee had some honey and that he might be able to persuade Sam to give him some. He walked from his cave, down the mountain trail, across the valley, over the bridge, to the oak tree where Sam Bee lived. He asked Sam for some honey. Sam gave him some. Then Joe Bear walked over to the oak tree and saw Jack Bear holding the honey. He thought that he might get the honey if Jack put it down, so he told him that he didn't think Jack could run very fast. Jack accepted this challenge and decided to run. He put down the honey and ran over the bridge and across the valley. Joe picked up the honey and went home."

CONCLUSIONS

Storytelling is a natural"" language processing activity that requires many kinds of knowledge, not simply large quantities of it. It's not a limited-domain task, and rather than try to make all the knowledge look the same, I take the opposite position — use the primitives, problems, and procedures that seem most appropriate to the domain. The control structure itself, the goal calculus, is a second-order problem domain that integrates various forms of knowledge and simulates goal-driven characters, avoiding the use of such "formal" techniques as backtracking. With this knowledge, we can write simple, reasonable stories; without it we qet some bizarre tales.

ACKNOWLEDGEMENTS

This work is part of the author's dissertation at Yale University [3] and was supported in part by the Advanced Research Projects Agency of the Department of Defense and monitored under the Office of Naval Research under contract N00014-75-C-1111. I would like to thank my advisor, Roger Schank, and the members of the Yale AI Project.

REFERENCES

1. Richard E. Fikes and Nils J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. Artificial intelligence 2:189-208, 1971".

2. Sheldon Klein and James R. Meehan. AI Forum. SIGART Newsletter (61):2-4, Feoruary 1977.

3. James K. Meehan. The Metanovel: Vvriting. stories Computer,~PhD dissertation, Yale University, 1976. Computer Science Department Research Report 74, Yale University, 1976.

4. Allen Newell and Herbert A. Simon. Human problem Solving Prentice Hall, 1972.

5. Roger C. Schank. Conceptual Inf_ormatL2[n] Processing. American Elsevier, New~York, 1975.

6. Roger C. Schank and Robert P. Abelson. Scripts, plans, and knowledge. In Erik Sandewall, conference committee chairman, Proceedings of the 4_th International Joint Conference on ArtIfIclal Intelligence^ MIT AI Lab, Cambridge, Massachusetts, 1975, 151-158.

7. Gerald Jay Sussman. A Computationall Model of Skill Acquistion dissertation, MIT, 1973.

8. Robert . wilensky. Using plans to understand stories. In Olin G. Johnson, general chairman, Proceedin^ of the Annual Conferences ACM, New York, 1976, 4b-50.