ORGANIZATION AND INFERENCE IN A FRAME-LIKE SYSTEM OF COMMON SENSE KNOWLEDGE

Eugene Charniak
Institute for Semantic and Cognitive Studies
Castagnola, Switzerland

I. INTRODUCTION

Rather than write two short papers for the two sessions I participate in (Memory: Organization, and Memory: Reasoning and Inferencing) I have chosen to write one paper of twice the length. My reason for doing so, beyond the fact that it takes less time to write one long paper than two short ones, is that the issues involved are so intertwined as to make separating them an unprofitable task.

My goals have not changed since (Charniak 72). I am still interested in the construction of a computer program which will answer questions about simple narration (e.g. children's stories). More exactly, if one makes the somewhat unrealistic division of the problem into (a) going from natural language to a convenient internal representation, and (b) being able to "reason" about the information in the story in order to answer questions, my interests are clearly in the latter section. I will take it as given that such reasoning requires large amounts of "common sense knowledge" about the topics mentioned in the text, so I will not demonstrate this point. (However it should come out incidentally from the examples used to demonstrate other points.) To reason with this knowledge requires that it be organized, by which I simply mean it must be structured so that the system can get at necessary knowledge when it is needed, but that unnecessary knowledge will not clog the system with the all too familiar "combinatorial explosion". I will start with my current thoughts on organization.

The scheme presented here has been clearly inspired by Minsky's "frames paper" (Minsky 74) so I have called the primary organizational grouping a "frame". It is perhaps indicative of the convergence of ideas reflected in (or perhaps inspired by) Minsky's paper that the overall organization proposed here (although not the details) is quite similar to the independently developed "scripts" of (Schank and Abelson 75).

II. FRAMES, FRAME STATEMENTS, AND FRAME IMAGES

I take a frame to be a static data structure about one stereotyped topic, such as shopping at the supermarket, taking a bath, or piggy banks. Each frame is primarily made up of many statements about the frame topic, called "frame statements" (henceforth abbreviated to FS). These statements are expressed in a suitable semantic representation, although I will simply express them in ordinary English in this paper.

The primary mechanism of understanding a line of a story is to see it as instantiating one or more FS's. So, for example, a particular FS in the shopping at the supermarket frame would be:

(1) SHOPPER obtain use of BASKET

(SHOPPER, BASKET, and in general any part of an FS written in all capitals is a variable. These variables must be restricted so that SHOPPER is probably human, and certainly animate, while BASKET should only be bound to baskets, as opposed to, say, pockets.) This FS would be instantiated by the second line of story (2).

(2) Jack was going to get some things at the supermarket.
The basket he took was the last one left.

Here we assume that part of the second line will be represented by the story statement (SS):

(3) Jack1 obtain use of basket1

(Of course, really both (1) and (3) would be represented in some more abstract internal representation.) Naturally, (3) would be an instantiation of (1), and this fact would be recorded with a special pointer from (3) to (1). I am, of course, making the common distinction between a data base which contains the particular story information, like (3), and a "knowledge base" which contains our generalized real world knowledge, such as the supermarket frame.

The supermarket frame will contain other FS's which refer to (1), such as:

(4) (1) usually occurs before (5)

(5) SHOPPER obtains PURCHASE-ITEMS

Any modification (like (4)) of a particular FS (like (1)) will be assumed true of all SS's which instantiate that FS (like (3)), unless there is evidence to the contrary. Hence, using (4) we could conclude that Jack has not yet finished his shopping in (2). Other modifications of (1) would tell us that Jack was probably already in the supermarket when he obtained the basket, and that he got the basket to use during shopping.

The variable SHOPPER in (1) also appears in (5), and in general a single variable will appear in many FS's. Hence the scope of these variables must be at least that of the frame in which they appear. When an SS instantiates an FS the variables in the FS will be bound. Naturally it is necessary to keep track of such bindings. For example, failure to do so would cause the system to fail to detect the oddness in (6) and (7).

(6) Jack went to the supermarket. He got a cart and started up and down the aisles. Bill took the goods to the checkout counter and left.

(7) Jack went to the supermarket to get a bag of potatoes. After paying for the milk he left.

It is probably a bad idea to actually change the frame to keep track of such bindings. Instead I assume that the frame remains pure, and that the variable bindings are recorded in a separate data structure called a "frame image" (abbreviated FI). For frames which describe some action, like our shopping at supermarket frame, we will create a separate FI for each instance of someone performing the action. So two different people shopping at the same time, or the same person shopping on two different occasions, would require two FI's to record those particulars which distinguish one instance of supermarket shopping from all others.

Much of this information will be stored in the variable bindings (shopper, purchase items, store, shopping cart used, etc.) However the variable bindings do not exhaust the information we wish to store in the FI, for example it will probably prove necessary to have pointers from the FI to some, if not all, of the SS's which instantiate FS's of the frame in question. Of slightly more interest is that the FI of a frame describing an action will keep track of how far the activity has progressed. So, for example, we would find the following story odd:

(8) Jack drove to the supermarket. He got what he needed, and took it to his car. He then got a shopping cart.

We have already said that FS's are modified by time ordering statements, so to note the oddity of (8) it is only necessary to have one or more progress pointers in the FI to the most time-wise advanced FS yet mentioned in the story. Then when new statements are found in the story which instantiate FS's in the frame the program will automatically check to see if these FS's are consistent with the current progress pointer(s). If so the FI progress pointer(s) may be advanced to indicate the new state of progress. If not, as in (8), the oddity should be noted, and, if possible, the story teller questioned about the oddness of the time sequence.

I have not commented so far about how, given a new SS, we locate an FS which it instantiates. In general this is a difficult problem, and I will have little to say about it. Roughly speaking the problem falls into two parts. First, the system must recognize that a given frame is relevant to a particular story. I am assuming that the presence of a key concept in the story will trigger a given frame. (It should be clear however that this is much too simple minded. For example, the scene setting description of a city block as containing a supermarket, bank, tailors, shoe repair shop, etc., should probably not activate the frames for the activities normally done in each.) Secondly, given that one or more frames have been selected as relevant to the story, how does the program find the particular FS which is instantiated by a particular SS? Here I will assume that a list of current frames is kept and frames which have not been used recently are thrown away. To find the particular FS which is instantiated by the SS I will simply assume that all FS's of the recently used frames are checked for a match. A more sophisticated procedure would be to first check FS's which follow the progress pointer. Another improvement would be to have an index for each frame so that it would not be necessary to check all FS's of the frame. This would, in effect, make each frame into a local data base. If a frame has a sub-frame it too will be checked, although it will probably be necessary to put some limit on how deep one should look into sub-frames.

One final note before moving on to more detailed issues. This paper is concerned primarily with the use of frames in the comprehension of simple narration. However, it seems only reasonable to me to assume that whatever knowledge we have built up into frames was done in large part in order to get around in the world, rather than to read stories. I will assume then that the same knowledge structures should be usable for either task, and upon occasion I will make arguments that structuring a frame in a particular way will make it easier to perform actions based on the frame structured knowledge. It seems to me one of the great advantages of frames that they seem capable of being used in multiple ways, something which is not obviously true, for example, of "demons" (Charniak 72).

III. FRAMES AND SUB-FRAMES

In this and the next two sections we will take a closer look at the internal structure of a frame. In particular I will try to show that Minsky's notion that different frames will have "terminals" in common is applicable to the kind of frames I have in mind. Minsky's idea was that frames applied to problems of vision, would store information about what one was likely to see in a certain situation (e.g. a room) and from a certain vantage point (e.g. just having walked in the door). Upon changing vantage points one moved to different frames, but many of the "terminals" of the frame (e.g. right wall, center wall, lamp, etc.) would appear in both views, and hence in both frames. While Minsky used the term "terminal" when discussing scenario frames (his "terminals" very roughly correspond to my "frame statement") he never applied the idea of common terminals between frames to scenario frames. Nor is it clear that frames such as the ones discussed in this paper have anything directly corresponding to the sharing of a wall terminal between two room frames. However, I will try to show that Minsky's notion does come in useful in a somewhat different way.

Let us start by giving a naive outline of some FS's about supermarkets.

(9) a) Goal (SHOPPER own ITEMS)
b) SHOPPER be at SUPERMARKET
c) SHOPPER have use of BASKET
d) do for all ITEM ITEMS
e) SHOPPER at ITEM
f) BASKET at ITEM
g) ITEM in BASKET
h) end
i) SHOPPER at CHECKOUT COUNTER
j) BASKET at CHECKOUT COUNTER
k) SHOPPER pay for ITEMS
l) SHOPPER leave SUPERMARKET

I am assuming in (9) an implicit time ordering from top to bottom. In the actual frame this time ordering would be made explicitly. The reader might also notice that most of the FS's are states which must be achieved at some point in the course of the action. For reasons why I use states rather than actions to express what happens in an action sequence, see (Charniak 75a).

There are countless things missing or wrong with (9), but I wish to concentrate on only one of them, the relation between shopping and using a shopping cart. It should be obvious that one can do shopping without using a cart, although (9) would make it seem that cart usage is an indispensable part of shopping. I will suggest that a good way to gain this flexibility in the supermarket frame is to have a separate cart frame which shares information with the supermarket frame, by having, in effect, some FS's common between the two.

One way to account for the ability to shop with or without a cart (and to understand stories about same) would be to have a second frame for shopping without a cart. However, not only is the idea unsatisfying, since it would require the duplication of the many facts the two activities have in common, but it would also lead to problems in the comprehension of certain types of stories. For example, one could easily imagine a story which starts out with Jack using a cart, the wheel of the cart sticking, and rather than going to get a second cart Jack finishes his shopping without a cart. It seems a priori that such combinations would be extraordinarily hard to account for with two completely separate frames for the two forms of supermarket shopping. (Alternatively, it is common practice in crowded situations to park one's cart in a general vicinity of several items and pick up the individual items without the cart, only to bring them all to the cart at a later point and resume shopping with the cart.)

One possibility for a single frame which handles both kinds of supermarket shopping is:

(10) a) Goal: SHOPPER owns PURCHASE-ITEMS
b) SHOPPER decide if to use a basket. If so, set CART to T
c) If CART then SHOPPER obtain BASKET
d) SHOPPER obtain PURCHASE-ITEMS

method

e) Do for all ITEM PURCHASE-ITEMS
f) SHOPPER decide on next ITEM
g) SHOPPER at ITEM
h) If CART then BASKET also at ITEM
i) SHOPPER hold ITEM
j) If CART then ITEM in BASKET
k) End
l) SHOPPER at CHECK-OUT COUNTER
m) If CART then BASKET at CHECK-OUT COUNTER
n) SHOPPER pay for PURCHASE-ITEMS
o) SHOPPER leave SUPERMARKET

Now one problem with (10) is that to my eye the constant repetitions of "If CART then ..." give it a rather ad hoc appearance. And while it was this ugliness which gave me the initial impetus to find a better representation, there are more concrete problems with (10). One major one is the lack of any information about why these various actions are to be performed and why in this particular order. This lack comes out most strongly when we consider stories where something goes slightly wrong in the course of the shopping. For example:

(11) Jack was shopping at the supermarket. After getting a few things he returned to his cart only to find that some prankster had taken everything out of it and put the things on the floor. After putting the things in the cart Jack finished his shopping, but was not able to find out who had done it.

Question: Why did Jack put the groceries which were on the floor into his cart?

The question here is so simple that one might be at a loss to know what sort of answer is desired, but one certainly knows that the items were put back in the cart so Jack could continue his shopping. The point here is that frame (10) does not give any explanation for Jack's action in (11). Naturally we would not expect any supermarket frame to explicitly take into account strange situations like (11), but this is not necessary to be able to answer the qustion in (11). All that is needed is an understanding that the purpose of a cart is to transport goods from place to place in the supermarket and that to do this the goods must be in the cart. Hence, in this situation, if, as is most likely, Jack still wants to transport the goods elsewhere he should once again put the items in the cart. But (10) does not give this information. It tells us to put the items in the cart, but not why or for how long.

I should point out that if (11) seems like an extraordinarily odd story on which to base any conclusions, there are much more normal ones which make the same point. For example:

(12) Jack was shopping at the supermarket. After getting a few items the wheel of his cart stuck. He got a second cart and finished his shopping.

Question: Why did Jack transfer his groceries to a second cart?

The story does not say that Jack transferred his groceries, and to infer that he did require essentially the same reasoning process required to understand why Jack put the groceries back in the cart in (11).

To handle stories like (11) and (12) we must therefore put two pieces of information into (10) which are not there at present. First, that in those cases like lines (10h) and (10m) where we have the basket going along with the person, the reason is to keep the previously collected items with one. Second, that to carry something with a cart requires that it be in the cart. We will indicate the first of these by:

(13) SHOPPER at ITEM
side condition - DONE at ITEM also
method - suggested
cart-carry
(SHOPPER,BASKET,DONE,ITEM)

Here DONE is a list of those items already collected. Cart-carry is a frame (and hence a sub-frame of the supermarket frame) describing the use of a cart for carrying things. (I am introducing a bit of terminology (method-suggested) from (Charniak 75a); I will assume that it is reasonably self-explanatory. Consult (Charniak 75a) for some explanation and justification.) The FS's in (13) replace lines (10g) and (10h) and differ from them in two respects. First (13) formulates the goal in a manner neutral with respect to using a basket or not, with only a suggestion that a basket be used. This is obviously necessary if we are to handle stories where the person does not use a basket. Secondly, it assumes the existence of a separate cart-carry frame in which we store information about using carts to carry things. We will see other advantages of this move later, but at the moment we can at least note that if one were to ask "Why does Jack use a basket?" two answers (at least) would be possible - "to do shopping", or "to carry his groceries". (13) allows for both of these answers, whereas (10g) and (10h) only allow for the former, since there is no separate "carry" level.

The second piece of information we needed to handle stories (11) and (12) was that to carry something with a basket it is necessary that the thing be in the basket. The most natural place to put such information would be in our newly created cart-carry frame where it would be some sort of a pre-requisite (or more precisely a "strict" (as opposed, for example, to "suggested") substate to use the terminology of (Charniak 75a)). By creating the cart-carry frame and locating information about the action within it we also circumvent the need to duplicate this information elsewhere. For example, an expanded supermarket frame would include the fact that in some circumstances it is permitted (and suggested) that one uses one's basket to take the groceries to one's car. By stating this as suggesting cart-carry (SHOPPER, BASKET, PURCHASE-ITEMS, CAR) we no longer need an instruction to put the groceries into the basket again before setting out.

IV. SHARING FRAME STATEMENTS BETWEEN FRAMES

So far then I have argued that frames must be able to reference sub-frames, and in particular, the supermarket frame needs some sub-frame like cart-carry. There is nothing exceedingly strange in this, but the next step will perhaps be a bit more interesting. Here I will suggest that some of the frame statements in our supermarket frame be shared with the cart-carry frame. To see the reasons for this, let us start by noting that the failure to allow for common FS's will lead to some curious redundancies in our frame. One of these occurs in the DO loop of (10) which handles the collection of the PURCHASE-ITEMS. With our latest changes, this portion of (10) (lines (e) through (k)) looks like:

(14)
a) Do for all ITEM PURCHASE-ITEMS
b) SHOPPER choose next ITEM
PURCHASE-ITEMS - DONE
c) SHOPPER at ITEM
d) side-condition DONE at ITEM also
e) method - suggested
f) cart-carry
(SHOPPER,BASKET,DONE,ITEM)
g) SHOPPER hold ITEM
h) If CART then ITEM in BASKET
i) DONE <- DONE + ITEM
j) End

The redundancy is this: we have already stated that cart-carry has a strict sub-state that the things to be carried must be in the basket. But in (14) we further specify in line (h) that ITEM is to be in BASKET, which is simply a special case.

Once again my initial reason for being concerned with this is that I find such redundancy unappealing, but again there seem to be more solid reasons for doing away with it. In particular consider the following story:

(15) Jack was doing some shopping at the supermarket using a shopping cart. The last thing he got was a package of gum, which he picked up right at the check-out counter. Jack put the gum and everything else down on the counter.

Question: Why didn't Jack put the gum in his basket?

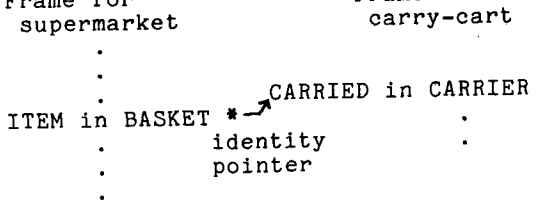
Again the question seems silly, but again we all know that there would be no reason to put the gum into the cart simply because Jack already has everything at the check-out counter. But (14) as stated does not allow us to make this inference. It simply says to put each ITEM into the BASKET and since no reasons are given there is no way to see that in the particular case of the gum in (15) there is no reason to put the gum into the basket.

My solution to this problem is to see line (14h) as shared between the supermarket frame and the cart-carry frame. Looked at in this light, the reason for obeying (14h) is then the same as the reason for having items in the basket as stated in the cart-carry frame - if you wish to use the cart to carry an item it must be in the cart. Since Jack has no reason for carrying the gum in the cart he has no reason to put it in the cart. (To be a bit more precise, Jack might have a reason to put the gum in the cart, namely using the cart to carry the groceries to his car afterwards, but this only occurs after putting the groceries on the check-out counter, hence does not count as a reason for doing it at the particular point in time we are discussing.)

Now when I suggest that the two frames share an FS, I do not mean that the one FS physically appears in both frames, although this would be perfectly possible in a list processing language like LISP. There are, however, several good reasons for not implementing FS sharing by physical identity. For one thing it would mean that different frames would have to have the same variables, which at the very least would create a major debugging problem. From a theoretical point of view it seems likely that such an attempt will run into trouble because two or more statements in one frame will share the same FS in a second frame. If the two FS's in frame one have different variables there would be no way for the FS in frame two to be identical to both, and hence could not be physically the same. (However I do not have a clear-cut example of this happening.)

So I will not assume that FS (14h) is physically identical to the corresponding FS in cart-carry, but rather that there is a pointer from (14h) to the FS in cart-carry which says that (14h) should be considered to be the same FS. That is, we would have an arrangement somewhat like:

(16)



Here I have created two new variables for the cart-carry frame, CARRIED which specifies what is carried, and CARRIER which is the cart used to carry. Naturally, to actually use the cart-carry information for understanding the ITEM in BASKET line of the supermarket frame it will be necessary to see that ITEM corresponds to CARRIED, etc. (It may also be necessary to see that SHOPPER in supermarket corresponds to the variable for the actor in cart-carry, and this would require more formalism, but I am not sure it is necessary.) Finally, note that there seems to be no reason to have a corresponding pointer from the FS in cart-carry to ITEM in BASKET, since there is no need to know about supermarkets in order to use a basket.

Now if all of this seems eminently reasonable to you, feel free to skip the next section. But for those of you to whom this seems a strange sort of data structure, what follows is an attempt to justify it by considering one alternative and showing how the shared FS proposal is superior.

Beyond the "ugliness" of the redundancy, the only argument we gave for replacing (14h) with (16) was the story (15) where Jack did not put the gum into the shopping cart. One way to solve both the redundancy, and the problem spotlighted by (15), would be to simply remove (14h) from the supermarket frame. This clearly solves the redundancy problem, and it also solves (15) since now the only FS to the effect that the goods should be in the basket appears in cart-carry, and since Jack has no reason for carrying the gum in the cart he has no reason to put the gum into the cart.

The argument against this possibility must start from the recognition that it has some counter intuitive properties. Intuitively one sees putting ITEM into BASKET as the last state of collecting ITEM. With this new solution this is no longer the case. Instead, putting ITEM into BASKET is a result of wanting to move to the next ITEM. (To distinguish we will call the item one has just obtained ITEM and the item one is going to obtain next ITEM .) Hence, putting ITEM into the basket is not the last thing of the N'th cycle, but one of the first of the N+1'st. This is againt my intuition and makes me immediately suspect it.

Furthermore, this counter intuitiveness seems to have more substantive implications. For example:

(17) Jack was shopping at the supermarket. After getting a basket he went to the milk counter and picked up a carton of milk. He then thought about what to get next.

Question: Did Jack put the milk in the basket?

Answer: I would assume so.

The shared FS model would allow for this answer since deciding on ITEM occurs after putting ITEM into the basket. But by deleting (14h) we remove this information so there would be no way to answer the question other than "I don't know". (Of course, "I don't know" is also an acceptable answer, but our model must allow for the various alternative answers people can give.)

A second argument against deleting (14h) comes from using our supermarket frame in actually doing shopping. By deleting (14h) we would be saying in effect that one puts ITEM into the basket when checking to see that all of DONE is in the basket when going to get ITEM . Computationally it seems horribly inefficient to bother to check on all of DONE each time (and in fact

I am sure that people do not do it).

Finally by deleting (14h) we make it difficult to account for "mistakes" that people make. For example, suppose we had the variation of (15) where Jack does put the gum into his cart only to immediately take it out again. We then ask the question:

(18) Question: Why did Jack put the gum into the cart, since he only had to immediately take it out again?

Answer: Well, I suppose one normally puts things into the cart immediately after picking them up.

Such considerations lead me to reject the alternative of deleting (14h).

V. A BETTER LOOKING SUPERMARKET FRAME

So far I have argued for the shared FS model primarily on the basis of its ability to handle stories where mistakes occurred, but it is also the case that it allows us to solve one of the "aesthetic" problems mentioned earlier, namely the constant repetition of "If CART then..." in (10). What we find is that all occurrences of this phrase in (10) can be replaced by either an explicit call to cart-carry (as in (13)) or an identity pointer to an FS in cart-carry, as in (16). To give another example of this, consider line (10c), repeated here:

(10)c) If CART then SHOPPER have use of BASKET

This is clearly another example of a pre-requisite of cart-carry, and hence should be considered shared with cart-carry in the same way that ITEM is BASKET is shared. Furthermore, we can now remove the "If CART then" portion of (10c) by assuming the eminently reasonable convention that a shared node in frame-1 which has a pointer to frame-2 is only applicable to the action in rame-1 if frame-2 is activated in the sense that we have created a frame image for frame-2. When performing the action in real life this means that upon deciding to use a cart one sets up a cart-carry image and this in turn makes the vaious FS's in the supermarket frame dealing with carts relevant to one's activities. While reading a story the general rule will be that any SS instantiating an FS which is shared with cart-carry will be sufficient to create an FI for cart-carry. With this convention (10c) becomes:

(19) SHOPPER have use of BASKET *
↓
identity pointer to cart-carry

Furthermore, since I have argued that these pointers are needed on independent grounds, we have received this simplification for free.

With both this simplification, and the use of cart-carry, our supermarket frame now looks like:

(20)
a) Goal: SHOPPER owns PURCHASE-ITEMS
b) SHOPPER decide if to use basket, if so set up cart-carry FI
c) SHOPPER obtain BASKET *cart-carry
d) SHOPPER obtain PURCHASE-ITEMS
e) | method - suggested
f) ↳Do for all ITEM PURCHASE-ITEMS
g) SHOPPER choose ITEM PURCHASE-ITEMS - DONE
h) SHOPPER at ITEM
i) | side-condition DONE at ITEM also
j) | method - suggested
k) ↳ cart-carry (SHOPPER, BASKET, DONE, ITEM)
l) SHOPPER hold ITEM
m) ITEM in BASKET *cart-carry
n) DONE <- DONE + ITEM
o) End
p) SHOPPER at CHECK-OUT-COUNTER
q) side-condition PURCHASE-ITEMS at CHECK-OUT-COUNTER also
r) | method - suggested
s) ↳cart-carry (SHOPPER, BASKET, PURCHASE-ITEMS, CHECK-OUT-COUNTER)
t) SHOPPER pay for PURCHASE-ITEMS
u) SHOPPER leave SUPERMARKET

I have here adopted the convention of indicating an identity pointer to another frame by a "*" followed by the name of the second frame.

In spite of the "simplificatons" introduced, (20) is considerably longer than (10). But on the other hand, (20) shows much more of the structure of shopping at supermarkets than does (10). So to point out only one way where (20) is superior, it, but not (10), states that it is necessary to get one's groceries to the checkout counter, whether or not one uses a cart. Of course, (20) still does not contain more than a fraction of our knowledge of supermarkets, and in fact many of its particulars are clearly wrong, but it's a start.

VI. INFERENCE ON FRAME BASED KNOWLEDGE

So far I have been discussing the organization of knowledge and have suggested that a large portion of it is stored in frames which connect up to other frames by either sub-frame relations or identity pointers. But a quick look back will reveal that at the same time several issues of inference have crept in. For example near the very beginning we stated that if an SS instantiates an FS then any modification of the FS within the frame is true of the SS also unless there is explicit information to the contrary. This, you will remember, allowed the system to conclude that when Jack got the shopping cart he was most likely already at the supermarket, but had yet to begin the actual act of collecting the groceries. Or again, when I mentioned that an FI had one or more progress pointers, I implicitly assumed that one could infer what actions had already taken

place using the time sequence information in the frame plus the progress pointer in the FI.

This is, of course, as it should be. One cannot, or at least should not, discuss structure independently of use and the use of frames is to allow us to make inferences about the stories we read. Nevertheless, there remain many issues of inference left untouched by the previous discussion and I will cover one or two of them here.

In (Charniak 72) I argued that some inferencing had to be done as the story was read (i.e. at "read time") rather than after a question was asked ("question time"). I think it is fair to say that this is now generally accepted, and the question now is how much inference is done at read time, and of what sort. I will not argue the point here (for a more up-to-date presentation of the issue see (Charniak 75b)), but rather make the rather strong assumption that one has not "understood" the text unless one has made a certain, as yet undefined, class of inferences which enable one to tie the text together into a coherent whole.

Given this assumption we are immediately confronted with a pressing problem. In principle one can draw an infinite number of inferences from a given body of text, and even in practice the total number would be prohibitively large. Some way then is needed to distinguish those inferences which need be made from those which do not. It is my impression that the system outlined previously in this paper has several nice properties in this regard.

For one thing, many inferences which would have to be made in a "demon" system (Charniak 72) need not be made in the system we have just outlined. To again take the example of Jack getting a shopping cart, I pointed out that the supermarket frame allowed the system to make several inferences about the statement, like why he did it, and that he had yet to start the shopping, but I left it vague as to whether these inferences should actually be made at read time, or only if a qustion was asked. In fact, there seems to be little reason for actually making most of these inferences at read time. Since the SS will have a pointer to the FS it instantiates, we may assume that a standard tactic for answering questions about a particular SS, like why the action was performed, or where, or when, would be to look in the frame for the answer.* To put this slightly differently, when an SS instantiates as FS it is not necessary to put into the data base instantiations of all the modifications of the FS.

Looked at in this light, it is interesting to ask under what circumstances one would want to instantiate an FS and put it in the data base. This is, after all, a large class of potential inferences and some restrictions on them would be at least a start on the problem of which inferences need be made at read time. The best answer I currently have to this question is summed up in the following rule.

(21) The Dual Usage Rule: If X is an FS in an active frame (one which has an FI) then X will only appear instantiated in the data base if it has two purposes.

Some typical purposes are:

a) appearing in an active frame
b) appearing in the semantic representation of the text
c) updating older statements in the data base

(this list will surely be expanded)

So the typical example of an FS which appears instantiated in the data base is one which appeared in the semantic representation of the text and was found to instantiate some FS. A more intersting case is exhibited by:

(22) Jack was going to Bill's birthday party. He thought Bill would like a kite. Jack then went to the store.

In (22) we should expect Jack to be buying a kite at the store. But consider the statement

(23) Jack1 own kite1

On one hand we can only predict that this is likely to occur using information from the birthday frame (perhaps in conjunction with the given subframe). On the other hand, this statement serves as the goal statement in the store frame. To enable (23) to act as the link between frames in this fashion it must appear explicitly in the data base with frame pointers to two diferent frame statements in two different frames. That is to say, (23) appears in two active frames, and hence is a justified instantiation.

It seems to me that (21) is a fairly strong rule, and it will be intersting to see if it can be maintained. Of course, given that the list of uses can be expanded, it is not completely clear what would serve as a counter exmple to the rule, but it seems to me that the general intent should be clear. For example, (21) would prohibit many of the inferences made by Rieger's system (Rieger 74). To give only one example, given statement (24) Rieger's system would infer statements like (25)-(28) none of which would qualify by the standard set up by (21).

(24) Jack told Mary that Bill wants a book
(25) John believes that Bill wants a book
(26) Mary now knows that Bill wants a book

---

*I might point out in passing that the use of the modifying information in such a manner is a major reason why frames as I describe them look like "data" rather than "program". Traditionally programs have the property that they are not meaningful "locally" whereas we want to be able to use a modifying FS to answer a question about an SS without going through the entire frame. Hence my description of frames at the begining of section II as "a static data structure."

(assuming that the representation of "tell" is not something ike "cause to know by word of mouth")
(27) Bill might get himself a book
(28) John may want Mary to give Bill a book

(This is one of Rieger's own examples.)

VII. COMPARISON WITH THE DEMON APPROACH

To get some perspective in frames as presented here let me compare them to the demon based system of my Ph.D. thesis (Charniak 72). On one hand this will enable me to explain why I have given up on the latter, while at the same time pointing out what I see as the advantages of the former. For those of you not familiar with (Charniak 72) a paragraph of summary is in order.

The point of the demon model was that many lines which had little significance in themselves took on greater significance in context, a prime example being "There was no sound" in the context of a child shaking his piggy-bank. To account for such situations th model associated with each "topic concept" (e.g. piggy-bank, or supermarket) a "base routine" which was a program which set up "demons" which would lie in wait for lines like "There was no sound" (which would be the "pattern" of the demon). Should the line occur, the demon would in effect say, "I know what this line means", and proceed to put statements in the data base which explicated the significance of the line (e.g. a statement like "This line implies there is nothing in the piggy bank"). We can draw the parallel between base routines and frames on one hand, and demons and frame statements on the other. I will come back to the significance of this parallel later, for the moment just take it as a helpful analogy.

One problem with the demon model is that in some cases one is forced into ad hoc formulations of facts due to the theoretical machinery seemingly not being suited to the problem.

Consider a fact like:

(29) Umbrellas are used to keep rain off one's head.

To fit such a fact into the model just presented, we would most naturally treat it as follows:

(30) Base routine which activates demon: Possibility of rain.

Pattern: Person gets umbrella.

Program: If person might be caught in rain he got the umbrella to prevent getting wet.

This will work quite well for stories like:

(31) It looked like rain. Jack got his umbrella.

It would even be possible, using extra mechanisms in the model, to handle a story like:

(32) As Jack was leaving the house he heard on the radio that it might rain. He went to the closet.

If asked why he did this we would respond that he was probably getting an umbrella. (It would be also possible to answer that he was getting his raincoat, but this is not important since "raincoat" would also have information connecting it to rain.) The extra mechanism which is needed here is the ability to put together the "expectation" of getting an umbrella, with our knowledge of where umbrellas are normally kept to conclude that he is going to get his umbrella in spite of the fact that the word "umbrella" was never mentioned in (32).

The trouble with this solution is that it would not account for the following story:

(33) Jack began to worry when he realized that everyone on the street was carrying an umbrella.

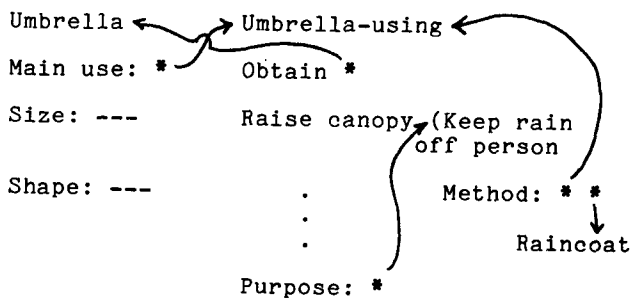
Question: What was Jack worrying about?

Answer: That it might rain, and he was without an umbrella.

While it is intuitively clear that a fact like (29) comes into play here, the formulation in (30) as vague as it is, is incapable of accounting for (33). The problem is that since (33) never mentioned rain, the demon expressed in (30) would never have been activated. To put this in terms of pointers, the fact in (30) only allows a pointer from "rain" to "umbrella", it does not allow a pointer from "umbrella" to "rain" and hence cannot be used to help us conclude in (33) that the problem is rain. It would be possible to put in a second demon which would handle (33), but this is the sort of ad hoc solution one wants to avoid.

Within the frames model proposed in this paper it is not hard to think up several solutions to this problem. My particular favourite looks like:

(34)



Here Umbrella and Umbrella-using would be frames, while (Keep rain off person) might be a frame, although more likely it is an FS in the Rain frame or some such. The point

here is that what appears to be a problem in the demon model is quite straight forward in the frame model.

Let us now return to the analogy made earlier between frame statements and demons. This analogy works so well because frame statements accomplish precisely what demons were designed to accomplish -- assign significance to a line due to the context it is in. So we notice how lines like (35)-(38) in the context of supermarket will instantiate FS's in the supermarket frame and hence (35)-(38) will be given more significance than they have out of context.

(35) Jack got a cart
(36) Jack picked up a carton of milk
(37) Jack walked further down the aisle
(38) Jack walked to the front of the store. He put the groceries on the counter.

What is interesting in this comparison is that one demon usually has a minimum of three or four statements, whereas obviously a single FS is only one statement. FS's seem then to have a considerable conciseness to them, at least when compared to demons. The reason for this is not hard to see. Demons being idependent facts must bind their own variables, and much of the size of a demon is due to checks to make sure that the variable bindings are correct (e.g. BASKET must be a basket, and not a carton of milk). These same things must be checked in a frame, but since the scope of the variable is the entire frame, rather than a single FS, the overhead, so to speak, is shared. Furthermore, the inferences about a given FS are stored implicitly in the structure of the frame, whereas they had to be stated explicitly in the demon. So a second advantage of the frames approach over demons is the conceptual economy one obtains in the expression of facts.

The analogy between FS's and demons also points to a third way in which the frames approach seems superior. One problem which bothers many people (including myself) about the demon approach is that it seemingly calls for large numbers of demons to be activated every time a given topic is mentioned in the story, although it is unlikely that more than a small fraction of the demons will ever be used. There are two possible reasons why people feel this is a problem. One is that so many active demons might make it hard to locate those demons which really should apply. Frames do not help with this problem since there will be equal numbers of FS's.

To see the second reason why activating large numbers of demons is problematic, note that if it took no time at all to set up a demon, setting up many of them would seem less bad. But of course it does take time to set up a demon, and it becomes a problem to justify this computation in light of the unlikeliness of the demon ever being used. Frames do offer a potential solution to this second problem because with frames, rather than supermarket activating many demons, we need only create a frame image for one frame (i.e. supermarket). This would take much less time, and hence would be better, but it should be noted that we pay a price. In particular most of the work involved in setting up a demon is to index our storage of active demons so that retrieving the ones needed will be reasonably easy. By comparison looking through frames to find matching FS's promises to be a time consuming task unless we do something similar. This is what I meant earlier when I said that perhaps each frame would have its own index to its contents. On the other hand, the approach presented here allows one to trade more time for locating an FS in return for less time to set up a new topic (frame), and the spectre of all those never to be used demons makes me inclined to accept this trade.

Finally, the frames presented here have no problem handing time relations between FS's as we saw earlier in the paper. The same cannot be said of demons. We saw earlier how we might use a progress pointer to allow the program to notice actions which were out of sequence. What could be the equivalent in the demon model of the progress pointer? For one thing, where would such a pointer be stored? Short of giving every demon a pointer to the progress pointer, an inelegant solution at best, it is not clear what one could do. Furthermore, where would the time ordering information be stored? Notice that time ordering information is much more complex than a simple string, or even lattice which indicates the time orderings of actions. For example some time orderings are "strict" in the sense that one cannot possibly do things any other way, while others are "suggested" in the sense that it is a good idea to do the actions in a given order, but possible to do them some other way, while yet others are "regulatory" in the sense that it is possible, but illegal to do the actions in the opposite order (Charniak 75a). In the frames model one can store time ordering statements in the frame along with the rest. It is by no means obvious what to do in the demon model. This is not to say that one could not do it, but rather that having done it one would be left with something of little resemblance to the original demon model, and even less aesthetic appeal.

REFERENCES

Charniak, E., Toward a model of children's story comprehension. AI-TR266, MIT Artificial Intelligence Laboratory, 1972.

Charniak, E., A partial taxonomy of knowledge about actions. Paper submitted to the Fourth International Joint Conference on Artificial Intelligence, 1975a.

Charniak, E., Inference and knowledge. Course notes for the Tutorial on Computational Semantics, Institute for Semantic and Cognitive Studies, 1975b.

Minsky, M., A framework for representing knowledge. AI-Memo 306, MIT Artificial

Intelligence Laboratory, 1974.

Reiger, C.J., Conceptual Memory. Unpublished Ph.D. Thesis, Stanford University, 1974.

Schank R.C. and Abelson, R.P. Scripts, plans and knowledge. Paper submitted to the Fourth International Joint Conference on Artificial Intelligence, 1975.