```
*(SPRINT (TRY WATERMAN))
((TRACE
1 ((CHOP1 (AGT (PEASANT (DET INDEF)))     ***** INPUT *****
          (AE TREE) (LOC (WOODS (BY LAKE))))
2  (DROP1 (AGT HE) (OBJ (AXE (POSSBY PEASANT))))
3  (FALL1 (OBJ AXE) (TNS PAST) (DEST WATER))
4  (DIVE1 (AGT HE) (DEST LAKE))
5  (FIND1 (OBJ (AXE (MOD (AND PRECIOUS ONLY)))))
6  (SWIM1 (AGT HE) (DEST (LAKE (MOD BOTTOM))))
7  (SEE1 (PERCEPTOF HE) (OBJ AXE) (TV NOT)))
8 ((((CHOP1 (AE TREE)                      ****** OUTPUT *****
            (LOC (WOODS (BY LAKE)))
            (AGT (PEASANT (DET INDEF)))
            (INSTR (AXE (POSSBY PEASANT)))
9           (COORD (HOLD246 (AGT (PEASANT (DET INDEF)))
                            (OBJ (AXE (POSSBY PEASANT))))))
10   (DROP1 (AGT (PEASANT (DET INDEF)))
            (OBJ (AXE (POSSBY PEASANT)))
11          (ANTE (HOLD246 (AGT (PEASANT (DET INDEF)))
                           (OBJ (AXE (POSSBY PEASANT)))))
12          (CONSQ (FALL1 (TNS PAST)
                          (OBJ (AXE (POSSBY PEASANT))) (DEST WATER))))
13   (SEEK403 (AGT PERSON)
              (OBJ (AXE (MOD (AND PRECIOUS ONLY))))
14            (COORD (TRAVEL256 (DEST (LAKE (MOD BOTTOM)))
                                (LOC LAKE)
                                (AGT PERSON)
15                        (SC (TRAVEL2#254
                               (AGT PERSON)
                               (DEST (LAKE (MOD BOTTOM)))
                               (LOC LAKE)
16                             (SUBSEQ (DIVE1 (AGT PERSON)
                                              (DEST LAKE)))
17                             (SUBSEQ (SWIM1 (AGT PERSON)
                                              (DEST (LAKE (MOD BOTTOM)))
                                              (LOC LAKE)))))))
18            (SEQ (FIND1 (AGT PERSON)
                          (TV NOT)
                          (OBJ (AXE (MOD (AND PRECIOUS ONLY))))))
```

Figure 4-2: The Peasant and the Waterman
14 event/state concepts

```
19                      (SUBSEQ
                         (PERCEIVE429
                          (PERCEPTOF PERSON)
                          (TV NOT)
                          (THM (AXE (MOD (AND PRECIOUS ONLY))))
20                        (SC (SEE1 (OBJ AXE)
                                    (PERCEPTOF PERSON)
                                    (TV NOT)
                                    (THM (AXE (MOD (AND PRECIOUS ONLY)
                                          ))))))))        ))
21  (FOLLOW271 (AGT PERSON)
               (OBJ (AXE (POSSBY PEASANT)))
22             (COORD (TRAVEL256 (DEST (LAKE (MOD BOTTOM)))
                                 (LOC LAKE)
                                 (AGT PERSON)))
23             (COORD (MOVE1#360
                       (OBJ (AXE (POSSBY PEASANT)))
                       (DEST WATER)
24                     (SC (DESCEND278
                              (OBJ (AXE (POSSBY PEASANT)))
                              (DEST WATER)
25                            (SC (FALL1 (TNS PAST)
                                    (OBJ (AXE (POSSBY PEASANT)))
                                    (DEST WATER)))))))))))))
```

Figure 4-2 concluded

to say that one is a detour path and another a normal path would be misleading, because both sets of concepts are related in the dictionary.

Also consider the path that TRACE finds between the 'falling' (line 16) and the 'diving' (line 16).

1) When the axe 'fell' it 'descended'. (line 25)

2) 'Descending' is a kind of 'moving'. (line 23)

3) If a person 'follows' an object the object is moving. (line 23)

4) Another part of 'following' is that the person doing the 'following' is 'travelling'. (line 22)

5) A kind of 'travelling' is 'travelling in water'. (line 15)

6) A part of 'travelling in water' is 'diving into water. (line 16)

TRACE found this connection on its own. The concept of 'following' was derived from a different piece of text ("The Fox and the Wren"). In order to find this connection TRY-PATH rejects a number of other paths suggested by FIND-PATH. One of the paths that TRY-PATH rejects is a path from divel to falll.

1) 'Diving' is a part of 'travelling in water'.

2) 'Travelling in water' is a kind of 'travelling'.

3) A part of 'travelling' is 'moving'.

4) A kind of 'moving' is 'descending'.

5) A kind of 'descending' is 'falling'.

TRY-PATH rejects this path because the agent of the 'diving' becomes

the agent of 'travelling', which will not match, but must match, the object of 'falling', i.e. he and axe do not match.

Another path that TRY-PATH rejects when it tries to connect the 'diving' to a previous schema is the path from 'dropping' to 'holding'.

1) An antecedent of 'dropping' is the object dropped must be 'held'.
2) 'Holding' is a part of 'carrying'.
3) Another part of 'carrying' is 'travelling'.
4) A kind of 'travelling' is 'travelling in water'.
5) A part of 'travelling in water' is 'diving'.

The connection between 'dropping' and 'holding' is a part of the path which connects 'chopping' (line 8) to 'dropping' (line 9). When TRY-PATH tested this path it marked the 'chopping' and its subpart 'holding' as being terminated. The 'chopping' and 'holding' are 'terminated' and not 'completed' because the 'chopping' event was interrupted before it reached its normal conclusion. We differentiate between 'terminated' and 'completed' events because a 'terminated' event can be re-initiated. For a trio of sentence like:

> The peasant chopped wood.
> He dropped the axe.
> He picked it up and continued chopping.

The second 'chopping' event is a continuation of the first one, so to have marked the first 'chopping' as 'completed' would have be an error.

Because the 'holding' has been 'terminated' TRY-PATH rejects the the path from 'diving' to 'holding' (i.e. a 'carrying' has not occurred).

Notice the 'seeking' event that TRACE constructs. The peasant sought the axe (line 13), but failed to find it (line 18). He sought the axe by 'travelling' to the bottom of the lake (line 14), by 'diving' into the water (line 16) and 'swimming' (line 17). He never found the axe because he did not 'perceive' it (line 19) (i.e he did not 'see' it (line 20)).

4.3 The Wishing Ring

This example comes from "The Wishing Ring".

> John labored in the fields. Never did he grow enough to feed his family. Never did he have an extra penny. Never did he eat at the village inn. One day he was plowing in the fields.

For this sample of text TRACE constructs a schema composed of four subschemas: 'eating', 'feeding', 'selling', and 'laboring'. The central subschema is the 'laboring' (line 13).

1) John 'labors' in the field by 'farming'. (line 14)

2) Part of 'farming' is 'growing', and John does not 'grow' enough to 'feed' his family. (line 17)

3) Another part of 'farming' is 'plowing'. (line 18)

4) A consequent of 'farming' is that John 'has' (owns) a crop. (line 15)

5) If John 'owns' a crop than he can 'have' physical (line 16)

```
*(SPRINT (TRY WISH))
((TRACE
1 ((LABOR1 (AGT JOHN) (LOC FIELD))          ***** INPUT *****.
2  (GROW1 (AGT HE) (TV NOT) (QTY ENOUGH) (FOR FEED1))
3  (FEED1 (AGT HE) (AE (FAMILY (POSSBY HE))))
4  (HAVE1 (AGT HE) (OBJ (PENNY MOD EXTRA)) (TM EVER))
5  (EAT1 (AGT HE) (LOC (INN MOD VILLAGE)) (TM NEVER))
6  (PLOW1 (AGT HE) (LOC (FIELD (POSSBY HE)))))
7 ((((EAT1 (TM NEVER)                       ***** OUTPUT *****
           (AGT JOHN)
           (LOC (INN MOD VILLAGE))
8          (PREC (HAVE1 (TM EVER) (AGT JOHN)
                        (OBJ (PENNY MOD EXTRA)))))
9    (FEED1 (AGT HE)
            (AE (FAMILY (POSSBY HE)))
            (OBJ FOOD)
            (ANTE (HAVE2#487 (AGT JOHN) (OBJ CROP))))
10   (SELL506 (AGT JOHN)
11            (ANTE (HAVE519 (OBJ CROP) (AGT JOHN)
                     (SC (HAVE2#487 (AGT JOHN) (OBJ CROP)))))
12            (CONSQ (HAVE1 (TM EVER) (AGT JOHN)
                            (OBJ (PENNY MOD EXTRA)))))
13   (LABOR1 (AGT JOHN)
            (LOC (FIELD (POSSBY HE)))
14          (SC (FARM486 (AE CROP)
                  (AGT JOHN)
                  (LOC (FIELD (POSSBY HE)))
15                (CONSQ (HAVE488 (AGT JOHN)
                                  (OBJ CROP)
16                        (SC (HAVE2#487 (AGT JOHN)
                                         (OBJ CROP)))
                                  ))
17                (SUBSEQ (GROW1 (TV NOT)
                                 (QTY ENOUGH)
                                 (FOR FEED1)
                                 (AGT JOHN)
                                 (AE CROP)
                                 (LOC (FIELD (POSSBY HE)))))
18                (SUBSEQ (PLOW1 (AGT JOHN)
                                 (LOC (FIELD (POSSBY HE))))))
              )))))))
```

Figure 4-3: The Wishing Ring - 12 event/state concepts

possession of it.

What is interesting about the 'laboring' subschema that TRACE constructs is that it gathers together events that occur over a range of time (i.e."Never did he grow enough to feed his family") with events occurring at a particular time (i.e. "One day he was plowing in the fields"). From TRACE's vantage point the text, in either case, is using the exact same event/state concept, and is therefore coherent. This example indicates another way in which event concepts can be broken up in text; as an introduction an event concept may be described over a range of time, followed by a description of a part of the event at some particular time.

'Having an extra penny' (line 12) is connected to the 'laboring' subschema, because John can 'have' money by 'selling' (line 10) his crop. To 'sell' his crop John must 'have' it (lines 11 and 16).

'Feeding his family' (line 9) is connected to 'laboring', because to 'feed' his family John must 'have' food (lines 9 and 16).

'Eating at an Inn' is related to 'laboring' via 'selling'. To 'eat' at an Inn (line 7) John must 'have' an extra penny (line 8). John can 'have' an extra penny (line 12) if he 'sells' his crop (line 10).

Also notice that when TRACE connected the 'feeding' to the 'laboring' it created a 'having'. Consequently it was possible for TRACE to mark 'having an extra penny' as a continuation of 'having' food. TRY-PATH rejects this path because the objects of the two 'havings' do not match (i.e. MATCH(extra penny, food) = false).

4.4 The Restaurant Story

In chapter two we discussed NEXUS' representation of "The Restaurant Story", here we show TRACE's processing of it. For the reader's convenience the text of the story is repeated.

> John went to a restaurant. The hostess seated John. The hostess gave John a menu. The waiter came to the table. John ordered a lobster. He was served quickly. He left a large tip. He left the restaurant.

Figure 4-4 show TRACE's handling of the text. "The Restaurant Story" represents a fuller exploration of the concept of eating at a restaurant than the one touched upon in "The Wishing Ring". To find the coherence of the text in "The Wishing Ring" example TRACE needed the knowledge that to eat at an inn John would need money. To build a representation of the restaurant story TRACE needs more detailed knowledge about eating at restaurants and inns.

Notice that TRACE needs one tree to represent the text of "The Restaurant Story". To 'eat' at a restaurant (line 9) John needed to 'travel' (line 10) to the restaurant. John 'travelled' to

```
*(SPRINT (TRY REST))
((TRACE
1 ((GO1 (AGT JOHN) (DEST RESTAURANT))    ***** INPUT *****
2  (SEAT1 (BEN JOHN) (AGT HOSTESS))
3  (GIVE1 (REC JOHN) (AGT HOSTESS) (OBJ MENU))
4  (COME1 (AGT WAITER) (DEST TABLE))
5  (ORDER1 (AGT JOHN) (OBJ LOBSTER))
6  (SERVE1 (BEN HE))
7  (LEAVE2#1 (AGT HE) (OBJ TIP1#))
8  (LEAVE1 (AGT HE) (SOURCE RESTAURANT)))
9 ((((EAT651 (OBJ LOBSTER)                ***** OUTPUT *****
             (AGT JOHN)
             (LOC RESTAURANT)
10      (ANTE (TRAVEL654 (AGT JOHN)
                  (DEST RESTAURANT)
11                (SUBSEQ (MOVE1#652 (OBJ JOHN)
                                (DEST RESTAURANT)
12                              (SC (GO1 (AGT JOHN)
                                         (OBJ JOHN)
                                         (DEST RESTAURANT))))))
               )
13      (SUBSEQ (ORDER701 (REC WAITER)
                  (INSTR MENU)
                  (AGT JOHN)
                  (OBJ LOBSTER)
14                (PREC (TRAVEL688 (DEST TABLE)
                                (AGT WAITER)
15                              (SC (COME1 (AGT WAITER)
                                           (DEST TABLE)))))
16                (CONTINUATION (ORDER1 (AGT JOHN) (OBJ LOBSTER))))
               )
17             (SEQ (TIP728 (AGT JOHN)
                        (REC WAITER)
18                      (SC (GIVE727 (AGT JOHN)
                                (REC WAITER)
                                (OBJ TIP1#)
19                              (SC (LEAVE2#1 (AGT JOHN)
                                              (OBJ TIP1#))))))
               )
```

Figure 4-4: The Restaurant Story - 16 concepts

```
20          (SUBSEQ (SEAT1 (LOC RESTAURANT)
                        (BEN JOHN)
                        (AGT HOSTESS)
                        (SUBSEQ (GIVE1 (REC JOHN) (OBJ MENU)
                                       (AGT HOSTESS)))
                    ))
21          (SEQ (DEPART755 (AGT JOHN)
                            (SOURCE RESTAURANT)
22                          (SC (LEAVE1 (AGT JOHN)
                                        (SOURCE RESTAURANT)))))
23          (SUBSEQ (SERVE1 (OBJ LOBSTER) (BEN JOHN)
                            (AGT WAITER)))))))))
```

Figure 4-4 concluded

the restaurant by 'moving' to it (line 11), by 'going' to it (line 12). Once he was at the restaurant he was 'seated' by the waitress (line 20) and 'given' a menu. Next, after the waiter 'came' to the table (lines 14 and 15), he 'ordered' his food. The waiter 'served' him food (line 23). Before he 'left' the restaurant (line 21), he 'left a tip (lines 19, 18, and 17).

Consider some of the references that TRACE resolves as it constructs a representation of the text. Because John 'ordered' lobster TRACE infers that John ate lobster (line 9). TRACE infers that John ordered his food from the waiter (line 13). The text does not explicitly state that the waiter 'served' John his food, but TRACE infers it (line 23). TRACE infers that John was the he who was 'served' (line 23). TRACE infers that John was the he who 'left' a tip (line 19). John 'leaves' a tip and TRACE infers that the waiter was meant to be the recipient (line 17). TRACE infers that John was the he who 'left' the restaurant (line 22). In each of these cases TRACE resolved references as TRY-PATH tested a path.

4.5 Robbing a Liquor Store

Here we show TRACE's processing of the "Robbing a Liquor Store" story. For the reader's convenience the text of the story has been repeated.

> John wanted money. He got a gun and walked into a liquor store. He told the owner he wanted some money. The owner gave John the money and John left.

Figure 4-5 shows TRACE's output for the story. The central schema is 'robbing'. To 'rob' a liquor store (line 13) John needs to 'travel' to the liquor store (line 14). He 'travels' to the liquor store by 'moving' (line 15), by 'walking' (line 16). John's motive for 'robbing' the liquor store is that he 'wanted' money (line 17). John 'had' a gun (line 21). When he got to the liquor store he 'told' the owner he 'wanted the money' (line 20). Then the owner 'gave' John the money (line 22). John made his 'getaway' (line 23).

'Getting a gun' (line 10) is connected to 'robbing' because if John 'gets' a gun then John 'has' a gun (line 12) which is a precedent of 'robbing' (line 21).

'Leaving the liquor store' is connected to 'robbing' it because a sequel of a 'robbery' is a 'getaway' (lines 23), which is a kind of 'escape', which is a kind of 'leaving'. What is interesting about this connection is that in the process of connecting the 'leaving' to the 'robbery' TRACE finds a more precise term, 'getaway'- thus demonstrating TRACE's potential for word sense disambiguation.

```
*(SPRINT (TRY ROB))
((TRACE
1  ((WANT1 (AGT JOHN) (OBJ MONEY))        ***** INPUT ***** .
2  (GET1 (AGT HE) (OBJ GUN))
3  (WALK1 (AGT HE) (DEST LIQUOR-STORE))
4  (TELL1 (AGT HE) (THM WANT2) (TO OWNER))
5  (WANT2 (AGT HE) (OBJ MONEY))
6  (GIVE (AGT OWNER) (OBJ MONEY) (REC JOHN))
7  (LEAVE (AGT JOHN)))
8 ((((LEAVE (AGT JOHN)                    ***** OUTPUT *****
            (SOURCE LIQUOR-STORE)
9           (SC (ESCAPE830 (AGT JOHN)
                  (SOURCE LIQUOR-STORE)
                  (SC (GETAWAY833 (AGT JOHN)
                        (SOURCE LIQUOR-STORE))))))
10   (GET1 (AGT JOHN)
           (OBJ GUN)
11 ·       (SC (GET2#759 (AGT JOHN)
                 (OBJ GUN)
12               (CONSQ (HAVE2#769 (AGT JOHN) (OBJ GUN))))))
13   (ROB764 (INSTR GUN)
             (OBJ MONEY)
             (AE OWNER)
             (AGT JOHN)
             (LOC LIQUOR-STORE)
14           (ANTE (TRAVEL778 (AGT JOHN)
                      (DEST LIQUOR-STORE)
15                    (SUBSEQ (MOVE1#775 (OBJ JOHN)
                                  (DEST LIQUOR-STORE)
16                              (SC (WALK1 (AGT JOHN)
                                         (DEST LIQUOR-STORE))
                                )))))
17           (PREC (WANT1 (AGT JOHN)
                          (OBJ MONEY)
18                    (CONTINUATION (WANT2 (AGT JOHN)
                                           (OBJ MONEY)))))
```

Figure 4-5: Robbing a Liquor Store
15 event/state concepts

```
19          (SUBSEQ (SAY819 (TO OWNER)
                            (AGT JOHN)
                            (THM WANT2)
20                          (SC (TELL1 (TO OWNER) (AGT JOHN)
                                       (THM WANT2)))))
21          (PREC (HAVE2#769 (AGT JOHN) (OBJ GUN)))
22          (SUBSEQ (GIVE (REC JOHN) (OBJ MONEY) (AGT OWNER)))
23          (SEQ (GETAWAY833 (AGT JOHN)
                       (SOURCE LIQUOR-STORE)))))))))
```

Figure 4-5 concluded

4.6 The Margie Story

In the chapter two we discussed NEXUS' representation of "The Margie Story", here we show TRACE's processing of it. For the reader's convenience I will repeat the text of "The Margie Story".

> Margie was holding tightly to the string of her beautiful new balloon. Suddenly, a gust of wind caught it. The wind carried it into a tree. The balloon hit a branch and burst. Margie cried.

Figure 4-6 shows TRACE's handling of the text. In the "The Tail of the Pig" the concepts 'moving2#' and 'carrying' were needed to represent the fact that the pig carried the laundry to and from the stream. Here the concepts of 'moving2#' and 'carrying' are needed to connect carry1 to hold1. The path from carry1 to hold1 is:

1) A kind of 'taking' is 'catching'. (line 14)

2) If an object is 'taken' then it was previously 'held1'. (line 15)

3) The thing that 'takes' the object 'has' it. (line 16)

4) To 'move' an object the agent of 'moving' must 'have' it. (line 10)

5) A kind of 'moving' is 'carrying1'. (line 11)

The concepts 'moving2#' and 'carrying' are also needed to connect carry1 to hit1.

1) 'Carrying1' is a kind of 'moving2#'. (line 10)

2) A part of the fact that the wind is 'carrying' the balloon is that the balloon is 'moving1#'. (line 12)

3) For the balloon to 'hit1' the branch it had to be (line 21)

```
*(SPRINT (TRY MARGIE))
1((TRACE        ((HOLD1 (AGT MARGIE)   ***** INPUT *****
                        (OBJ BALLOON) (INSTR STRING))
2                (CATCH1 (AGT WIND (MOD GUST)) (OBJ IT))
3                (CARRY1 (AGT WIND) (OBJ IT) (DEST TREE))
4                (HIT1 (OBJ1 BALLOON) (OBJ2 BRANCH))
5                (BURST1 (OBJ IT))
6                (CRY1 (AGT MARGIE)))
7               ((((CRY1 (AGT MARGIE) ***** OUTPUT *****
8                        (PREC (UNHAPPY641 (STATEOF MARGIE))))
9                  (MOVE2#602 (AGT WIND)
                      (OBJ BALLOON)
                      (DEST TREE)
10                    (ANTE (HAVE2#608 (AGT WIND) (OBJ BALLOON)))
11                    (SC (CARRY1 (AGT WIND) (OBJ BALLOON)
                                  (DEST TREE)))
12                    (COORD (MOVE1#616 (DEST TREE) (OBJ BALLOON))))
13                 (TAKE592 (AGT WIND)
                      (OBJ BALLOON)
14                    (SC (CATCH1 (AGT WIND) (OBJ BALLOON)))
15                    (ANTE (HOLD1 (INSTR STRING)
                                   (OBJ BALLOON) (AGT MARGIE))
                            )
16                    (CONSQ (HAVE2#608 (AGT WIND) (OBJ BALLOON))))
17                (HIT1 (OBJ2 BRANCH)
                        (OBJ1 BALLOON)
18                  (SEQ (BREAK638 (OBJ BALLOON)
19                               (SC (BURST1 (OBJ BALLOON)))
20                               (SEQ (UNHAPPY641 (STATEOF MARGIE)))))
21                  (ANTE (MOVE1#616 (DEST TREE)
                                     (OBJ BALLOON)))))))))
```

Figure 4-6: The Margie Story - 12 event/state concepts

'moving1#'.

4.7 The Archer, William Tell

This example comes from the story "The Archer, William Tell". The text of the paragraph is:

> Just then the clatter of horses' hooves was heard. And Gessler, the governor general, galloped into the square. His military retinue followed him. He reined his horse to a stop before the pole.

Figure 4-7 shows TRACE's handling of the text. Here again we encounter the concept 'following'. The governor general was 'moving' (lines 8 and 17) and his military retinue 'followed' (line 7).

What was 'heard' (line 9) was a sound (line 10), the 'clattering' of hooves (lines 10 and 15), which was made as the governor general 'rode' into the square (line 12).

His military retinue 'followed' the governor general as he 'travelled' into the square, as he 'rode' into the square, by 'galloping' (line 15), 'reining' (Line 13), and 'stopping' (line 14).

4.8 The Czar's General and the Clever Peasant

This example (figure 4-8)is the first paragraph from The Czar's General and the Clever Peasant. I have included it for historical reasons. Trying to find the connection between 'digging' and 'hitting' initiated most of the motivations of this research.

```
*(SPRINT (TRY WTELL))
((TRACE
1 ((HEAR1 (THM CLATTER1))                    ***** INPUT *****
2  (CLATTER1 (OBJ HOOVES))
3  (GALLOP1 (AGT GOVERNOR-GENERAL) (DEST SQUARE))
4  (FOLLOW1 (AGT MILITARY-RETINUE) (OBJ HIM))
5  (REIN1 (AGT HE) (INSTR (HORSE (POSSBY HIM))))
6  (STOP1 (AGT HE) (OBJ HORSE) (LOC POLE)))
7 ((((FOLLOW1 (AGT MILITARY-RETINUE)    ***** OUTPUT *****
              (OBJ GOVERNOR-GENERAL)
8             (COORD (MOVE1#445
                      (OBJ GOVERNOR-GENERAL) (DEST SQUARE))))
9    (HEAR1 (THM CLATTER1)
10           (COORD (SOUND437 (THM HOOVES)
                     (SC (CLATTER1 (OBJ HOOVES))))))
11   (TRAVEL474 (AGT GOVERNOR-GENERAL)
                (DEST SQUARE)
                (INSTR HORSE)
12     (SC (RIDE440 (INSTR HORSE)
                    (DEST SQUARE)
                    (AGT GOVERNOR-GENERAL)
13                  (SUBSEQ (REIN1 (LOC SQUARE)
                                   (AGT GOVERNOR-GENERAL)
                                   (INSTR (HORSE (POSSBY HIM)))
14                                 (SEQ (STOP1 (LOC POLE)
                                        (AGT GOVERNOR-GENERAL)
                                        (OBJ HORSE (POSSBY HIM)
                                                   ))))))
15                  (COORD (CLATTER1 (OBJ HOOVES)))
                    (SUBSEQ (GALLOP1 (AGT GOVERNOR-GENERAL)
16                                   (INSTR HORSE)
                                     (DEST SQUARE)))))
17     (SUBSEQ (MOVE1#445 (OBJ GOVERNOR-GENERAL)
                          (DEST SQUARE)))))))))
```

Figure 4-7: The Archer, William Tell - 10 event/state concepts

> A peasant was digging in his garden one day. His spade unexpectedly hit a hard object. (It was not a stone. It was not a root.) He dug around it. soon he uncovered a metal chest.

Placed in parentheses are the sentences that TRACE does not handle. Both of these sentences are descriptions of objects; in Simmons' schema/narrative trees they would fall into the setting portion of the representation. As was stated in chapter two, NEXUS does not concern itself with setting information. Some of this information is useful for match thing concepts, but the match functions lie at the periphery of its concerns.

Figure 4-1 show TRACE's processing of the paragraph. In this story we see a repetition of the 'hitting' concept that we saw in "The Margie Story". TRACE connects 'hitting' (line 5) and 'digging' (line 8) as follows:

1) Part of 'digging' is 'pushing' the spade into the ground. (line 9)

2) If the peasant 'pushes' on the spade then the spade 'moves1#'. (line 11)

3) If the spade and a hard object 'hit' then one them must be 'moving1#'. (line 6)

Again we are seeing how the seven coherence relations of NEXUS capture causal relations; the relationship between the 'digging' and the 'hitting' is causal.

```
*(SPRINT (TRY PEASANT))
((TRACE
1 ((DIG1 (AGT PEASANT)            ***** INPUT *****
         (LOC (GARDEN (POSSBY HIM)))
         (TM (DAY (DET INDEF))))
2  (HIT1 (OBJ1 (SPADE (POSSBY HIM)))
         (MOD UNEXPECTEDLY)
         (OBJ2 (OBJECT (MOD HARD) (DET INDEF))))
3  (DIG2 (AGT HE) (AROUND IT))
4  (UNCOVER1 (AGT HE)
             (OBJ (CHEST (MOD METAL) (DET INDEF)))))
5 ((((HIT1 (MOD UNEXPECTEDLY)     ***** OUTPUT *****
           (OBJ2 (OBJECT (MOD HARD) (DET INDEF)))
           (OBJ1 (SPADE (POSSBY HIM)))
6          (ANTE (MOVE1#230 (OBJ (SPADE (POSSBY HIM)))
                            (INTO EARTH))))
7    (UNCOVER1
      (OBJ (CHEST (MOD METAL) (DET INDEF)))
      (AGT PEASANT)
      (LOC (GARDEN (POSSBY HIM)))
8     (SUBSEQ (DIG1 (TM (DAY (DET INDEF)))
                (INSTR (SPADE (POSSBY HIM)))
                (AE EARTH)
                (AGT PEASANT)
                (LOC (GARDEN (POSSBY HIM)))
9               (SUBSEQ
10                (PUSH234 (AGT PEASANT)
                    (OBJ (SPADE (POSSBY HIM)))
                         (INTO EARTH)
11                       (SEQ (MOVE2#237
                                (AGT PEASANT)
                                (OBJ (SPADE (POSSBY HIM)))
                                (INTO EARTH)
12                              (COORD (MOVE1#230
                                   (OBJ (SPADE (POSSBY HIM)))
                                   (INTO EARTH)))))))
13                (CONTINUATION (DIG2 (AROUND IT)
                                (AGT PEASANT)))))))))))
```

Figure 4-8: The Czar's General and the Clever Peasant
7 event/state concepts

Chapter 5

Question Answering and Summarizing

5.1 Quest

Question answering can be broken out into two subprocesses, question identification and answer retrieval. Answer retrieval is guided by the heuristics associated with each category of question: it is directed to examine a subset of the relationships established among event/state concepts in the text. For example, how questions guide the retrieval process toward the instrumental portion of the understanding, and why questions towards the causal/temporal portion of the understanding. Thus the scope of the heuristics is determined by the semantic relationships supported by the system.

QUEST is concerned with the answer retrieval phase of question answering. The heuristics it uses are coded as a set of relationships between known, mentioned in the question, and unknown, not mentioned in the question, event/state concepts. Associated with each question type are a list of relationship configurations. QUEST proceeds as a pattern matcher, seeking a subschema in the representation produced by TRACE which will match the relationship pattern associated with the question type.

QUEST represents only a part of the answer retrieval process (see figure 5-1). Its role is to produce candidate answers; it uses the organization of text produced by TRACE to suggest possible answers. A later process, which has access to additional semantic information (e.g. causal), either rejects or accepts QUEST's candidate answers. Thus QUEST can be thought of as an early staging operation in the answer retrieval process.

Suppose QUEST is asked a motive question. One strategy that QUEST uses for motivation questions is to look for event/states which are in a precedent relationship to the event described in the question. For a question like,

Why did John rob the liquor store?

QUEST would select instances of precedent concepts, in this case there is are two.

Because he wanted money.
Because he had a gun. (?)

A later process would determine that 'wanting money' is motivational, and therefore a reasonable answer.

The workhorse of QUEST is a procedure called FIND-MATCH. FIND-MATCH tries to establish a relationship between a known, named and instantiated, and unknown, neither named nor instantiated, event/state concepts. It has two arguments; the first is a pattern

ANSWER RETRIEVAL

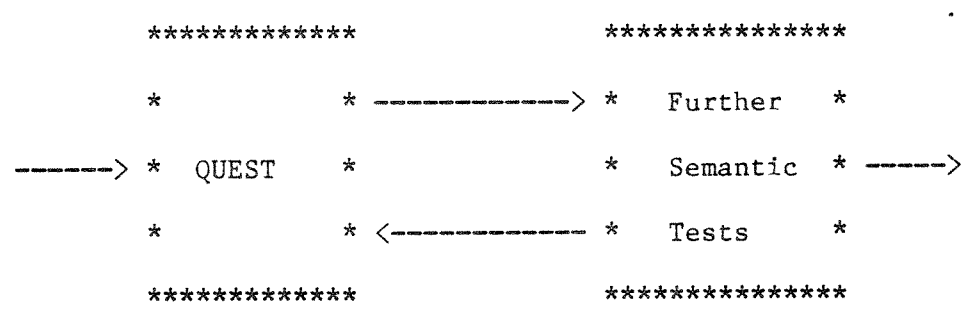

Figure 5-1: Two parts of answer retrieval

that it tries to match, the second a list of concept trees. The concept trees are stored in tuple form. So the relationship between clean and wash is represented:

```
[ (subseq (clean agt pig obj laundry loc stream)
          (wash agt pig obj laundry loc stream) ) . . . ]
```

The pattern argument is a triple:

```
[relation event1 event2]
```

Event1 and Event2 can either be instantiated concepts (i.e. known) or a variable that could match any arbitrary concept (i.e. unknown). The relation field is either one of the seven relations supported by NEXUS or a variable. A typical pattern passed to FIND-MATCH is:

```
(subseq x (wash agt pig obj laundry))
```

FIND-MATCH matches the pattern against each triplet in the schema. If the relationship in the pattern matches the relationship of a triplet in the schema, FIND-MATCH checks to see if corresponding events match. For the pattern shown above the subseq relation

matches the relation of the first triplet in the schema. X is a variable so it matches, and is unified, with clean. The two washing events match because they have the same name and each of the cases of the pattern-wash match an argument of the schema-triplet-wash.

In the remainder of this section we will see that QUEST can in fact answer a range of questions. In particular, because the focus of this research has been on event/state coherence, the kinds of questions that we will be looking at are ones that ask about relationships between events and states. This means that what has frequently been referred to as concept completion (i.e. who, what, and where questions) and verification questions will not be discussed.

5.1.1 Enhancement Questions

Enhancement questions ask the reader to embellish with further details the description of an event that occurred in the text. Answers to enhancement questions either describe subparts or instantiated subclasses of the event in question. Some examples from The Tale of the Pig are listed below. Figures 5-2,5-3, and 5-4 show QUEST's handling of these examples. The figures only show the concepts which QUEST used in answering the question. QUEST's output is the case noted version of the answer.

1. How did the pig clean the clothes? By washing and drying it.

2. How did the pig wash the laundry? By soaking and scouring it.

3. How did the pig dry the laundry? By hanging it in the sun.

Not all how questions are enhancement questions. For example,

How did the pig carry the laundry?
In her hands.

In this case the how question is asking about instruments used in the event and is therefore a concept completion question.

Enhancement questions direct QUEST to look for **subclass**, **coordinate**, and **subsequence** relationships between the question concept and some unknown concept in the text. Calls to FIND-MATCH take three forms.

[FIND-MATCH (subclass (clean agt pig obj laundry) x)]

[FIND-MATCH (subseq (clean agt pig obj laundry) x)]

[FIND-MATCH (coord (clean agt pig obj laundry) x)]

For question 1 FIND-MATCH is asked to find, in the schema list produced by TRACE, an event which is in a subclass relationship with **cleaning**, or a set of events that are either subsequences or coordinates of **cleaning**. For question 1 it finds both **washing** and **drying** are subsequences of **cleaning**.

Question 2 also uses subsequence relationships to answer the question, while question 3 uses the subclass relationship.

```
*(SPRINT (TRY Q1))
*************** QUESTION ******************
((QUEST HOW?
     (CLEAN (AGT PIG) (OBJ CLOTHES))
*************** CONCEPT TREES *************
     (            .
                  .
      (SUBSEQ (CLEAN923 (AGT PIG) (LOC WATER) (OBJ CLOTHES))
              (DRY934 (AGT PIG) (LOC WATER) (OBJ CLOTHES)))
      (SUBSEQ (CLEAN923 (AGT PIG) (LOC WATER) (OBJ CLOTHES))
              (WASH920 (AGT PIG) (OBJ LAUNDRY) (LOC WATER)))
*************** ANSWER ********************
     ((BY ((WASH920 (AGT PIG) (OBJ LAUNDRY) (LOC WATER))
           (DRY934 (AGT PIG) (LOC WATER) (OBJ CLOTHES)))))))
```

Figure 5-2: How did the pig clean the clothes?

```
*(SPRINT (TRY Q2))
************** QUESTION ******************
((QUEST HOW?
     (WASH (AGT PIG) (OBJ LAUNDRY))
************** CONCEPT TREES **************
     (            .
                  .
      (SUBSEQ (WASH920 (AGT PIG) (OBJ LAUNDRY) (LOC WATER))
              (SCOUR1 (AGT PIG) (OBJ LAUNDRY) (LOC WATER)))
      (SUBSEQ (WASH920 (AGT PIG) (OBJ LAUNDRY) (LOC WATER))
              (SOAK1 (AGT PIG) (OBJ LAUNDRY) (LOC WATER)))
************** ANSWER ********************
     ((BY ((SOAK1 (AGT PIG) (OBJ LAUNDRY) (LOC WATER))
           (SCOUR1 (AGT PIG) (OBJ LAUNDRY) (LOC WATER)))))))
```

Figure 5-3: How did the pig wash the laundry?

```
*(SPRINT (TRY Q3))
************** QUESTION ********************
((QUEST HOW?
     (DRY (AGT PIG) (OBJ CLOTHES))
************** CONCEPT TREES **************
     (            .
                  .
      (SC (DRY934 (AGT PIG) (LOC WATER) (OBJ CLOTHES))
          (HANG1 (IN SUN) (LOC WATER) (OBJ CLOTHES)))
******************** ANSWER **************
     ((BY (HANG1 (IN SUN) (LOC WATER) (OBJ CLOTHES))))))
```

Figure 5-4: How did the pig dry the laundry?

5.1.2 Goal Questions

Goal questions specify an action in the text and ask QUEST to specify the purpose of that action. Answers to goal questions are either a state the actor is trying to achieve or that is desired to occur. Some examples from The Wishing Ring are (see figures 5-5 and 5-6:

1. Why did John labor in the field? So he could feed is family and sell food.

2. Why did he want to sell food? So he could eat at the inn.

An example from The Tale of the Pig is (see figure 5-7):

3. Why did the pig trot to the stream?
   So she could clean the clothes.

An example from The Restaurant Story is (see figure 5-8):

4. Why did John order lobster?
   So he could eat lobster.

If the event asked about is a subpart of a second event, then

the second event represents the goal (See question 4). If the event named in the question has a consequent which is an antecedent or precedes some other event then the later event might be the goal (Examples of that are questions 1, 2 and 3). There are four types of patterns that goal questions ask FIND-MATCH to look for:

```
[AND (FIND-MATCH (CONSEQ QUESTION X) )
     (OR (FIND-MATCH (ANTE ANSWER X) )
         (FIND-MATCH (PREC ANSWER X)) ) ]

[FIND-MATCH (ANTE ANSWER QUESTION)]

[FIND-MATCH (SUBSEQ ANSWER QUESTION)]

[FIND-MATCH (COORD ANSWER EVENT)]

[FIND-MATCH (SC ANSWER QUESTION)]
```

5.1.3 What-Caused Questions

What-caused questions ask the reader to identify the cause of an event described in the text. Answers to what-caused questions describe some previous event. Examples from The Czar's General and the Clever Peasant (see figure 5-9) and The Margie Story (see 5-10) are:

1. What caused the spade to hit the hard object?
   The peasant pushed the spade into the earth.

2. What caused the balloon to burst?
   It hit a branch.

3. What caused Margie to be unhappy?
   The balloon broke.

What-caused questions direct QUEST to try three different concept patterns:

```
*(SPRINT (TRY Q10))
**************** QUESTION ****************
((QUEST GOAL?
     (LABOR (AGT JOHN) (LOC FIELD))
**************** CONCEPT TREES ***********
     (              .
                    .
      (ANTE (SELL368 (AGT JOHN)) (HAVE381 (OBJ CROP) (AGT JOHN)))
      (SC (HAVE381 (OBJ CROP) (AGT JOHN))
          (HAVE2#349 (AGT JOHN) (OBJ CROP)))
      (CONSQ (FARM348 (AE CROP) (AGT JOHN) (LOC (FIELD (POSSBY HE))))
             (HAVE350 (AGT JOHN) (OBJ CROP)))
      (SC (HAVE350 (AGT JOHN) (OBJ CROP))
          (HAVE2#349 (AGT JOHN) (OBJ CROP)))
      (ANTE (FEED1 (AGT HE) (AE (FAMILY (POSSBY HE))) (OBJ FOOD))
            (HAVE2#349 (AGT JOHN) (OBJ CROP)))
      (SC (LABOR1 (AGT JOHN) (LOC (FIELD (POSSBY HE))))
          (FARM348 (AE CROP) (AGT JOHN) (LOC (FIELD (POSSBY HE)))))
******************** ANSWER ***********************
     ((SELL368 (AGT JOHN))
      (FEED1 (AGT HE) (AE (FAMILY (POSSBY HE))) (OBJ FOOD)))))
```

Figure 5-5: Why did John labor in the field?

```
*(SPRINT (TRY Q11))
**************** QUESTION *********************
((QUEST GOAL?
     (SELL (AGT JOHN) (OBJ FOOD))
**************** CONCEPT TREES ****************
     (              .
                    .
      (PREC (EAT1 (TM NEVER) (AGT JOHN) (LOC (INN MOD VILLAGE)))
            (HAVE1 (TM EVER) (AGT JOHN) (OBJ (PENNY MOD EXTRA))))
      (CONSQ (SELL368 (AGT JOHN))
             (HAVE1 (TM EVER) (AGT JOHN) (OBJ (PENNY MOD EXTRA))))
**************** ANSWER ************
     ((EAT1 (TM NEVER) (AGT JOHN) (LOC (INN MOD VILLAGE))))))
```

Figure 5-6: Why did John sell food?

```
*(SPRINT (TRY Q13))
********** QUESTION **********
((QUEST GOAL?
     (TROT (AGT PIG) (DEST STREAM))
********** CONCEPT TREES **********
     (
                    .
                    .
      (ANTE (CLEAN923 (AGT PIG) (LOC WATER) (OBJ CLOTHES))
            (MOVE2#928 (AGT PIG) (OBJ LAUNDRY) (DEST WATER)))
      (SC (MOVE2#928 (AGT PIG) (OBJ LAUNDRY) (DEST WATER))
          (CARRY1 (AGT PIG) (OBJ LAUNDRY) (DEST WATER)))
      (SC (TRAVEL857 (TWD STREAM) (AGT PIG) (DEST WATER))
          (COME880 (AGT PIG) (DEST WATER)))
      (SC (MOVE1#860 (DEST WATER) (OBJ PIG) (TWD STREAM))
          (TROT1 (DEST WATER) (AGT PIG) (TWD STREAM)))
      (SUBSEQ (TRAVEL857 (TWD STREAM) (AGT PIG) (DEST WATER))
              (MOVE1#860 (DEST WATER) (OBJ PIG) (TWD STREAM)))
      (COORD (CARRY1 (AGT PIG) (OBJ LAUNDRY) (DEST WATER))
             (TRAVEL857 (TWD STREAM) (AGT PIG) (DEST WATER))))
********** ANSWER **********
     ((CLEAN923 (AGT PIG) (LOC WATER) (OBJ CLOTHES)))))
```

Figure 5-7: Why did the pig trot to the stream?

```
*(SPRINT (TRY Q14))
********** QUESTION **********
((QUEST GOAL?
     (ORDER (AGT JOHN) (OBJ LOBSTER))
********** CONCEPT TREES **********
     (
                    .
                    .
      (SUBSEQ (EAT239 (OBJ LOBSTER) (AGT JOHN) (LOC RESTAURANT))
              (ORDER289 (REC WAITER) (INSTR MENU)
                        (AGT JOHN) (OBJ LOBSTER)))
********** ANSWER **********
     ((EAT239 (OBJ LOBSTER) (AGT JOHN) (LOC RESTAURANT)))))
```

Figure 5-8: Why did John order lobster?

```
[FIND-MATCH (SEQ ANSWER QUESTION)]

[FIND-MATCH (CONSQ ANSWER QUESTION)]

[AND (FIND-MATCH (ANTE QUESTION X))
     (OR (CONSQ ANSWER X)
         (SEQ ANSWER X)) ]
```

Question 1 uses pattern 3. An antecedent of 'hitting' is that one of the object is 'moving1#'. A sequel of 'pushing' is that the object 'moves1#'.

Question 2 uses pattern 1. A kind of 'breaking' is 'bursting'. A sequel of 'hitting' is 'breaking. Question 3 also uses pattern 1. A sequel of a balloon 'breaking' is that its owner could be 'unhappy'.

5.1.4 Motive Questions

Motive questions ask about an actors motive for causing an event. A particular motive is never necessary, so QUEST's heuristic is to retrieve the precedents of the question-concept. Examples from Robbing a Liquor Store and The Margie Story are:

1. Why did John rob the liquor store? He wanted money.
2. Why did Margie cry? She was unhappy.

Figures 5-12 and 5-13 show QUEST's handling of the questions. Notice for the first question QUEST suggests two answers; John robbed the liquor store because either he had a gun (?) or he wanted money.

```
*(SPRINT (TRY Q51))
****************** QUESTION **********************
((QUEST WHATCAUSED?
     (HIT (OBJ1 SPADE) (OBJ2 OBJECT))
****************** CONCEPT TREES ******************
     (                .
                      .
      (SUBSEQ (DIG1 (TM (DAY (DET INDEF)))
                    (INSTR (SPADE (POSSBY HIM)))
                    (AE EARTH)
                    (AGT PEASANT)
                    (LOC (GARDEN (POSSBY HIM))))
              (PUSH42 (AGT PEASANT)
                      (OBJ (SPADE (POSSBY HIM))) (INTO EARTH)))
      (SEQ (PUSH42 (AGT PEASANT)
                   (OBJ (SPADE (POSSBY HIM))) (INTO EARTH))
           (MOVE2#45 (AGT PEASANT)
                     (OBJ (SPADE (POSSBY HIM))) (INTO EARTH)))
      (COORD (MOVE2#45 (AGT PEASANT)
                       (OBJ (SPADE (POSSBY HIM))) (INTO EARTH))
             (MOVE1#38 (OBJ (SPADE (POSSBY HIM)))
                       (INTO EARTH)))
      (ANTE (HIT1 (MOD UNEXPECTEDLY)
                  (OBJ2 (OBJECT (MOD HARD) (DET INDEF)))
                  (OBJ1 (SPADE (POSSBY HIM))))
            (MOVE1#38 (OBJ (SPADE (POSSBY HIM))) (INTO EARTH))))
****************** ANSWER ************************************
     ((PUSH42 (AGT PEASANT) (OBJ (SPADE (POSSBY HIM)))
              (INTO EARTH)))))
```

Figure 5-9: What caused the spade to hit the hard object?

```
*(SPRINT (TRY Q50))
****************** QUESTION ***************************
((QUEST WHATCAUSED?
     (BURST (OBJ BALLOON))
****************** CONCEPT TREES **********************
     (                .
                      .
      (SC (BREAK148 (OBJ BALLOON)) (BURST1 (OBJ BALLOON)))
      (SEQ (HIT1 (OBJ2 BRANCH) (OBJ1 BALLOON))
           (BREAK148 (OBJ BALLOON)))
****************** ANSWER *****************************
     ((HIT1 (OBJ2 BRANCH) (OBJ1 BALLOON)))))
```

Figure 5-10: What caused the balloon to burst?

```
*(SPRINT (TRY Q52))
**************** QUESTION **************
((QA WHATCAUSED?
     (UNHAPPY (STATEOF MARGIE))
**************** CONCEPT TREES *********
     (             .
                   .
     ((PREC (CRY1 (AGT MARGIE)) (UNHAPPY151 (STATEOF MARGIE)))
      (SEQ (BREAK148 (OBJ BALLOON)) (UNHAPPY151 (STATEOF MARGIE)))
**************** ANSWER ****************
     ((BREAK148 (OBJ BALLOON)))))
```

Figure 5-11: What caused Margie to be unhappy?

QUEST makes one type of call to FIND-MATCH to answer motive questions.

[FIND-MATCH (PRECEDENT QUESTION ANSWER)]

5.1.5 What-happened Questions

What-happened questions ask the reader to identify the outcome of an event. Some examples of what-happened questions from The Peasant and the Waterman are (See figures 5-14, 5-15, and 5-16):

1. What happened when the peasant was chopping a tree? He dropped his axe.
2. What happened when the peasant dropped his axe? It fell into the lake?
3. What happened when the peasant sought his axe? He didn't find it.

One heuristic that QUEST uses for retrieving answers to what-happened questions is to look for events which follow the question-concept via either sequel or consequent arcs; a consequent of the peasant 'dropping' the axe is that it 'fell' into the lake, a sequel of the peasant 'seeking' the axe is that he didn't 'find' it. Another heuristic it uses is to find events that the question-concept either precedes or antecedes; a part of the peasant 'chopping' wood is that he 'held' the axe, which is an antecedent to 'dropping' the axe.

There are four calls to FIND-MATCH that QUEST makes:

[FIND-MATCH (CONSQ QUESTION ANSWER)]

```
*(SPRINT (TRY Q20))
********** QUESTION **********
((QUEST MOTIVE?
     (ROB (AGT JOHN) (LOC LIQUOR-STORE))
********** CONCEPT TREES **********
     (                    .
                          .
      (PREC (ROB165 (INSTR GUN)
                    (OBJ MONEY)
                    (AE OWNER)
                    (AGT JOHN)
                    (LOC LIQUOR-STORE))
            (HAVE2#170 (AGT JOHN) (OBJ GUN)))
      (PREC (ROB165 (INSTR GUN)
                    (OBJ MONEY)
                    (AE OWNER)
                    (AGT JOHN)
                    (LOC LIQUOR-STORE))
            (WANT1 (AGT JOHN) (OBJ MONEY))))
********** ANSWER **********
     ((HAVE2#170 (AGT JOHN) (OBJ GUN))
      (WANT1 (AGT JOHN) (OBJ MONEY)))))
```

Figure 5-12: Why did John rob the liquor store?

```
*(SPRINT (TRY Q21))
********** QUESTION **********
((QUEST MOTIVE?
     (CRY (AGT MARGIE))
********** CONCEPT TREES **********
     (                    .
                          .
      (PREC (CRY1 (AGT MARGIE)) (UNHAPPY151 (STATEOF MARGIE)))
********** ANSWER **********
     ((UNHAPPY151 (STATEOF MARGIE)))))
```

Figure 5-13: Why did Margie cry?

[FIND-MATCH (SEQ QUESTION ANSWER)]

[FIND-MATCH (ANTE ANSWER QUESTION)]

[FIND-MATCH (PREC ANSWER QUESTION)]

5.1.6 Enablement Questions

Enablement questions either ask about conditions of events or events which make another event possible; thus an answer to an enablement question either describes a condition or an event which made a condition true. An example from Robbing a Liquor Store is (see figure 5-17):

1. What did John need to rob the liquor store? He needed to have a gun.

An example from The Wishing Ring (see figure 5-18) is:

2. What did John need to eat at the inn?
   He need to have an extra penny.

To answer enablement questions QUESTS looks for concepts either precedent or antecedent to the question concept. So calls to FIND-MATCH take two forms:

[FIND-MATCH (ANTE QUESTION ANSWER)]

[FIND-MATCH (PREC QUESTION ANSWER)]

```
************************ QUESTION ****************
*(SPRINT (TRY Q30))
((QUEST HAPPEN?
     (CHOP (AGT PEASANT) (AE TREE))
************************ CONCEPT TREES ***********
     ((COORD (CHOP1 (AE TREE)
                    (LOC (WOODS (BY LAKE)))
                    (AGT (PEASANT (DET INDEF)))
                    (INSTR (AXE (POSSBY PEASANT))))
             (HOLD454 (AGT (PEASANT (DET INDEF)))
                      (OBJ (AXE (POSSBY PEASANT)))))
      (ANTE (DROP1 (AGT (PEASANT (DET INDEF)))
            (OBJ (AXE (POSSBY PEASANT))))
            (HOLD454 (AGT (PEASANT (DET INDEF)))
            (OBJ (AXE (POSSBY PEASANT))))))
************************ ANSWER ******************
     ((DROP1 (AGT (PEASANT (DET INDEF)))
             (OBJ (AXE (POSSBY PEASANT)))))))
```

Figure 5-14: What happened when the peasant was chopping a tree?

```
*(SPRINT (TRY Q31))
******************** QUESTION ****************
((QUEST HAPPEN?
     (DROP (AGT PEASANT) (OBJ AXE))
******************** CONCEPT TREES ***********
     (                .
                      .
      (CONSQ (DROP1 (AGT (PEASANT (DET INDEF)))
                    (OBJ (AXE (POSSBY PEASANT))))
             (FALL1 (TNS PAST) (OBJ (AXE (POSSBY PEASANT)))
                    (DEST WATER)))
******************** ANSWER ******************
     ((FALL1 (TNS PAST) (OBJ (AXE (POSSBY PEASANT)))
             (DEST WATER)))))
```

Figure 5-15: What happened when the peasant dropped the axe?

```
*(SPRINT (TRY Q32))
******************** QUESTION ****************
((QUEST HAPPEN?
     (SEEK (AGT PEASANT) (OBJ AXE))
******************** CONCEPT TREES ***********
     (                  .
                        .
      (SEQ (SEEK611 (AGT PERSON)
                    (OBJ (AXE (MOD (AND PRECIOUS ONLY)))))
           (FIND1 (AGT PERSON) (TV NOT)
                  (OBJ (AXE (MOD (AND PRECIOUS ONLY))))))
********************* ANSWER *****************
     ((FIND1 (AGT PERSON) (TV NOT)
             (OBJ (AXE (MOD (AND PRECIOUS ONLY))))))))
```

Figure 5-16: What happened when the peasant sought the axe?

```
*(SPRINT (TRY Q40))
***************** QUESTION ***************
((QUEST PREC?
     (ROB (AGT JOHN) (LOC LIQUOR-STORE))
***************** CONCEPT TREES **********
     (              .
                    .
      (ANTE (ROB165 (INSTR GUN)
                    (OBJ MONEY)
                    (AE OWNER)
                    (AGT JOHN)
                    (LOC LIQUOR-STORE))
            (TRAVEL179 (AGT JOHN) (DEST LIQUOR-STORE)))
      (PREC (ROB165 (INSTR GUN)
                    (OBJ MONEY)
                    (AE OWNER)
                    (AGT JOHN)
                    (LOC LIQUOR-STORE))
            (HAVE2#170 (AGT JOHN) (OBJ GUN)))
      (PREC (ROB165 (INSTR GUN)
                    (OBJ MONEY)
                    (AE OWNER)
                    (AGT JOHN)
                    (LOC LIQUOR-STORE))
            (WANT1 (AGT JOHN) (OBJ MONEY))))
************* ANSWER *****************
     ((TRAVEL179 (AGT JOHN) (DEST LIQUOR-STORE))
      (HAVE2#170 (AGT JOHN) (OBJ GUN))
      (WANT1 (AGT JOHN) (OBJ MONEY)))))
```

Figure 5-17: What did John need to rob the liquor store?

```
*(SPRINT (TRY Q41))
************* QUESTION *************
((QUEST PREC?
     (EAT (AGT JOHN) (LOC INN))
************* CONCEPT TREES ********
     (          .
                .
      (PREC (EAT1 (TM NEVER) (AGT JOHN) (LOC (INN MOD VILLAGE)))
            (HAVE1 (TM EVER) (AGT JOHN) (OBJ (PENNY MOD EXTRA))))
************* ANSWER ***************
     ((HAVE1 (TM EVER) (AGT JOHN) (OBJ (PENNY MOD EXTRA))))))
```

Figure 5-18: What did John need to eat at the inn?

5.1.7 Perspective

There is not a wealth of relevant question answering literature. Most of the systems that discuss answer retrieval heuristics either answer questions about a micro-world [Winograd 72, Scragg 75] or are natural language front-ends for a database [Woods, Kaplan, and Nash-Webber 72, Waltz 78, Rich 79, Hendrix 77a, Hendrix 77b, Hendrix, Scerdoti, Saglowicz, and Slocum 78]. The recent work of Lehnert [Lehnert 77] represents the most extensive exploration of text question answering. Her system , QUALM, uses representations produced by SAM (a script applier) and PAM (a plan applier) to answer questions about stories. Lehnert has a classification of thirteen question types. Table 5-1 shows correspondences between Lehnert question types and their QUEST counterparts. QUEST's question types were derived from Lehnert. In a couple cases we have changed the names of the question type (i.e. causal antecedent and causal consequent) because they potentially

QUALM QA TYPE	QUEST QA TYPE
Causal Antecedent	What-Caused & Motive
Goal Orientation	Goal
Enablement	Enablement
Causal Consequent	What-Happened
Instrumental/Procedural	Enhancement

Table 5-1: Lehnert's Question Typology

could be confused with the coherence relations antecedent and consequent. In addition causal antecedent questions have been split into two categories , and instrumental/procedural questions like, "What did John use to eat with?", which ask about the instrument an agent uses in an event are not accounted for. Of the remaining categories, QUEST does not produce answers for them for one of three reasons: either the question category does not concern inter-event relations, or it asks the reader to discuss concepts related to the text but not needed to understand the text, or it is not a relevant question type for text.

Consider the heuristics Lehnert suggests for Goal questions. First QUALM looks for an instantiated script. If it is available QUALM can retrieve an answer by using script-specific retrieval heuristics. The script-specific retrieval heuristics are canned answers which are organized according to question category. QUALM makes no attempt to generalize and take advantage of the structure of

scripts. Since in NEXUS, scripts, like everything else, are represented using the coherence relations, QUEST can base its answer retrieval heuristics on the structure of representation.

If QUALM's script heuristics fail plan-based retrieval heuristics are attempted. So, for example, if an event which precedes the question event is causally related, and is part of a plan, the goal of the plan is the answer to the question. This strategy is analogous to the first of the coherence patterns used by QUEST to answer goal questions.

Finally causal retrieval heuristics are employed by QUALM. During the processing of the text scripts were used to make inferences about the goal structure of the text. These inferences were added to the causal chain representation. Here the kinds of heuristics employed are analogous to QUEST's strategy of ascending subsequence or coordinate arcs.

Another effort in the area of text question answering is Levine's [Levine 80] master thesis. It undertook a preliminary investigation of the utility for answering questions of Simmons'-type schema/narrative trees. The emphasis of her research was on translating english forms to answer verification and concept completion questions. Questions about the relationships between events are largely unexplored. She does discuss some 'why'

questions. Her heuristic is to look for events which precede, via a sequence arc, the question event. So for "The Black and Yellow V2 Rocket" (see Chapter Two) a question like "Why did radar track the rocket?" is answered "It was too high to be seen".

5.2 SUM

Text lies on a continuum, and is in fact, itself, a summarization. The longest summary, which would not be a summary at all, would attempt, and fail in that attempt, to explicitly describe every connection among the events and states it included. The shortest summary names a few events that will implicitly determine the bulk of the rest of it. TRACE produces the no-summary summarizations. SUM uses the representations produced by TRACE to generate the shorter summaries.

In summarizing SUM highlights the most relevant portions of the text; the events it chooses to describe tell the most about the event/state concept it is summarizing. For us the importance of a summarizing capability will be that it demonstrates the utility of the hierarchical organization of the event/state packets produced by TRACE.

We can divide SUM's activities into two types: strategies for summarizing single and multiple concepts. Single concepts are trees and normally can be summarized by selecting the top node in the

concept's tree. For multiple concept representations heuristics can be applied for deleting least significant concepts.

5.2.1 Single Concept Summarization

Because of the hierarchical organization of TRACE's output summarization of single concepts is straightforward. With one exception SUM summarizes single concepts by returning the top node in the concept's tree. Before considering the exception consider an example of the rule. For "The Restaurant Story" TRACE produces a single concept tree. The text of "The Restaurant Story" reads:

> John went to a restaurant. The hostess seated John. The hostess gave John a menu. The waiter came to the table. John ordered a lobster. He was served quickly. He left a large tip. He left the restaurant.

Figure 5-19 shows SUM's handling of the story. The input to SUM is the output produced by TRACE for this story. SUM's output is the case version of a summary of the text. The output would translate in english to: "John ate lobster at a restaurant."

The story does not explicitly mention that John ate at the restaurant, nor that if he did eat he ate lobster, yet the representation that TRACE built is a tree centered around the concept 'eating lobster'. Consequently, when SUM selects the top node in the tree it correctly summarizes the story as an 'eating' event. The algorithm for summarizing single concepts can be formally stated:

```
*(SPRINT (TRY REST-SUM))
((SUM ********** INPUT ************************
      ((EAT651 (OBJ LOBSTER)
          (AGT JOHN)
          (LOC RESTAURANT)
          (ANTE (TRAVEL654 (AGT JOHN)
                    (DEST RESTAURANT)
                    (SUBSEQ (MOVE1#652 (OBJ JOHN)
                              (DEST RESTAURANT)
                              (SC (GO1 (AGT JOHN)
                              (OBJ JOHN)
                              (DEST RESTAURANT))))
                                )))
          (SUBSEQ (ORDER701 (REC WAITER)
                    (INSTR MENU)
                    (AGT JOHN)
                    (OBJ LOBSTER)
                    (PREC (TRAVEL688 (DEST TABLE)
                              (AGT WAITER)
                              (SC (COME1 (AGT WAITER)
                                    (DEST TABLE)))))
                    (CONTINUATION (ORDER1 (AGT JOHN) (OBJ LOBSTER))
                                  )))
          (SEQ (TIP728 (AGT JOHN)
                    (REC WAITER)
                    (SC (GIVE727 (AGT JOHN)
                              (REC WAITER)
                              (OBJ TIP1#)
                              (SC (LEAVE2#1 (AGT JOHN)
                                        (OBJ TIP1#))))
                                ))
          (SUBSEQ (SEAT1 (LOC RESTAURANT)
                    (BEN JOHN)
                    (AGT HOSTESS)
                    (SUBSEQ (GIVE1 (REC JOHN)
                              (OBJ MENU)
                              (AGT HOSTESS)))))
          (SEQ (DEPART755 (AGT JOHN)
                    (SOURCE RESTAURANT)
                    (SC (LEAVE1 (AGT JOHN)
                          (SOURCE RESTAURANT)))))
              (SUBSEQ (SERVE1 (OBJ LOBSTER) (BEN JOHN)
                              (AGT WAITER)))))
************** OUTPUT ************************
      ((EAT651 (OBJ LOBSTER) (AGT JOHN) (LOC RESTAURANT)))))
```

Figure 5-19: The Restaurant Story

Summarization Rule One -
Delete from the top node in a concept tree any subtrees which are related by one of the seven coherence relations or the special relation 'continuation'."

The exception is not much harder to handle. The exception occurs because NEXUS uses property inheritance to represent the net. In a class/subclass relationship the subclass predicts more than the parent class. A subclass inherits all the properties of its parent class. If the parent class is in an antecedent relationship with a 'having', then the subclass concept inherits this relationship. But vice versa is not the case. If a summary mentions an instantiated subclass concept the reader can predict all its relationships as well as the relationships it inherits from its ancestors. If a summary mentions the parent class instead of the subclass then the reader of the summary has lost the relationships which the subclass makes available. So the subclass concept is preferred over the class concept with regards to summarization because it has more predictive power.

For example, in "The Margie Story" to connect the 'carrying' to the 'catching' and 'hitting' TRACE climbed up a subclass arc to 'moving2#'. Figure 5-20 shows the text of the story and the representation TRACE produces of the 'moving2#'.

If SUM were to summarize this concept, given the previously stated summarization rule, SUM would produce the summary; "The wind

Margie was holding tightly to the string of her beautiful new balloon. Suddenly, a gust of wind caught it. The wind carried it into a tree. The balloon hit a branch and burst. Margie cried.

The text of the story.

```
(MOVE2#602 (AGT WIND)
    (OBJ BALLOON)
    (DEST TREE)
    (ANTE (HAVE2#608 (AGT WIND) (OBJ BALLOON)))
    (SC (CARRY1 (AGT WIND) (OBJ BALLOON) (DEST TREE)))
    (COORD (MOVE1#616 (DEST TREE) (OBJ BALLOON))))
```

Figure 5-20: The wind carried the balloon into the tree.

moved the balloon into the tree." But a better summary would be; "The wind carried the balloon into the tree. "

To produce this summary SUM must first promote any concepts which are a subclass to the top node in the concept tree. A subclass of an event is promoted by replacing the name of the top node in the tree by the name of its instantiated subclass. So for the example we have been discussing, before SUM selects the top node of the tree as the summarization of the concept, 'move2#602' gets replaced by 'carry1'. A more formal statement of the promotion rule is:

Summarization Rule Two - (Promotion)
If the top node in a concept tree has an instantiated subclass, promote the instantiated subclass to be the top node in the tree. Apply this rule before rule one.

5.2.2 Multiple Concept Summarization

It is not unusual for TRACE to produce several interconnected trees as a representation of a piece of text. To summarize a multiple tree representation SUM needs heuristics for deleting some of the least significant concept trees. In order to give the reader a idea of the kind of heuristics available, we will discuss two examples.

The first heuristic SUM can apply is to delete identity trees. An identity tree occurs when TRACE has to descend a series of subclass arcs to get a concept which is connected (via something other than a subclass arc) to another concept. An example of this is occurs in "Robbing a Liquor Store." To connect "John left." to the the 'robbing' concept TRACE descends subclass arcs to the concept 'getaway' which is a sequel to a 'robbery'. Figure 5-21 shows the text of the story and the identity tree that TRACE produced as a part of its output.

For summarization purposes identity trees are of little interest. They occur because TRACE needed a particular sense of a high level concept to find a connection to another concept. SUM deals with identity trees by deleting them. SUM recognizes identity trees by checking to see if all the coherence related subtrees are subclasses. If they are SUM can delete that tree from its summary because it is an identity tree. So the tree whose top node is

John wanted money. He got a gun and walked into a liquor store. He told the owner the wanted some money. The owner gave John the money and John left.

The text of the story.

```
(LEAVE1 (AGT JOHN)
   (SOURCE LIQUOR-STORE)
   (SC (ESCAPE830 (AGT JOHN)
          (SOURCE LIQUOR-STORE)
          (SC (GETAWAY833 (AGT JOHN)
                 (SOURCE LIQUOR-STORE))))))
```

Figure 5-21: An identity tree.

'leave1' is an identity tree and can be deleted. Thus our rule for identity trees is:

Summarization Rule Three - (Identity Trees)
Delete from the summarization any concepts which are represented by an identity tree. An identity tree is a tree for which all event/state coherence related subtrees are subclasses. Apply rule three before summarization rules one and two.

A second heuristic that Sum applies to multiple concept representations has to do with planned behavior. Sometimes the text of a story describes an event which enables a later event to occur. The concept trees associated with such events are preparatory and can be deleted from the summary. An example of this occurs in "Robbing a Liquor Store." That "John got a gun." is a preparation for a robbery. Figure 5-22 shows the 'getting' and part of the 'robbing'.

As in the case of identity trees, SUM deals with preparatory trees by deleting them. A concept tree is preparatory if if has a

```
(GET1 (AGT JOHN)
      (OBJ GUN)
      (SC (GET2#759 (AGT JOHN)
            (OBJ GUN)
            (CONSQ (HAVE2#769 (AGT JOHN)
                              (OBJ GUN))))))
(ROB764 (INSTR GUN)
   (OBJ MONEY)
   (AE OWNER)
   (AGT JOHN)
   (LOC LIQUOR-STORE)
   (PREC (HAVE2#769 (AGT JOHN) (OBJ GUN)))
         .
         .
         .
```

Figure 5-22: A preparation of 'robbing' is 'getting a gun'.

consequent (or sequel) which is an antecedent (or precedent) of another event; a consequent of 'getting' is 'having' which is a precedent of 'robbing'. So when SUM summarizes "Robbing a Liquor Store" it deletes the 'getting' concept tree from its summary. The preparatory summary heuristic is:

Summarization Rule Four - (Preparatory Trees)
   If a concept tree is preparatory delete it from the summary.
   A preparatory tree has a consequent (or sequel) which is
   an antecedent (or precedent) of some other concept tree.
   Apply rule four before rules one and two.

5.2.3 Summarization

SUM's algorithm can be summarized as follows:

1) Delete identity trees. (rule 3)
2) Delete preparatory trees. (rule 4)
3) Promote subclass concepts. (rule 2)
4) Delete coherence related subtrees. (rule 1)

Consider how SUM summarizes the story "Robbing a Liquor

Store". Figure 5-23 shows SUM's handling of this story. The input are the three interconnected concept trees produced by TRACE. One of the trees is an identity tree (leaving), and another one a preparatory tree (getting). SUM's output translates in english to "John robbed the owner of a liquor store with a gun." Notice SUM's summary is about a 'robbery' a concept only implicitly in the text.

Figure 5-24 shows how SUM summarizes the text from "The Tale of the Pig". Again SUM deletes a preparatory tree; 'gathering' the laundry' is preparatory for 'carrying' it home. In english SUM's summary of this text translates to: "The prince saw the pig appear at the water. She cleaned her laundry. "

5.2.4 Perspective

A number of researchers have investigated summarization of text. Table 5-2 compares other contributions to the summarization literature to SUM. Column one gives the name of the researcher(s). Column two the base of information they used for summarizing text. Column three indicates if they had any programmed examples, and four their summarization techniques.

Both Rumelhart and Correira & Simmons summarize text by level of tree. Rumelhart works with two trees, one contains the syntactic structure of the story the other its semantic structure. He summarizes the text by simultaneously descending both trees, deleting