Chapter 4 MapTutor: Diagnosis and Teaching

4.1 Introduction

Diagnosis in graphical mapping is more complex than in some other educational situations, because, in the current context, the learner faces two simultaneous and complex tasks, namely reading comprehension and mapping. As Feifer (1989) observes, she can be wrong in either or both of them. Thus, an important component of MapTutor is a mechanism dedicated to identifying causes of errors. Notice, however, that the strategy followed by this mechanism is rather different from that employed by the Sherlock program discussed in **Chapter 2**. For example, once the cause of error has been determined as misunderstanding of concepts (what Feifer, 1989, calls an icon interpretation problem), MapTutor does not attempt to find out what possible interpretation (i.e., underlying misconception) of the concept in question the learner might have. Instead, MapTutor is satisfied with assuming that concept misunderstanding was the cause of error and provides the corrective feedback appropriate to this situation. In other words, Sherlock's main diagnostic approach was about finding out underlying reasons for errors, whereas MapTutor's is about finding nature of errors.

MapTutor's diagnosis is based upon the knowledge representation schema employed to represent and update beliefs, discussed in **Chapter 3**. Also, as seen in the previous chapter, MapTutor's knowledge base includes some information about the text and semantics of links so as to improve its ability to identify sources of potential difficulties for the learner. Most of this chapter concentrates on MapTutor's diagnostic procedures, but it also presents some teaching procedures which are able to deal with each diagnostic value returned by the diagnostic procedures. The pedagogical effectiveness and motivational aspects of these teaching procedures have not been tested yet. Instead, currently, they are only intended to show how each diagnostic value could be accommodated into a specific kind of feedback embodied in a teaching procedure.

4.2 Simplifying Assumptions

Before proceeding, it is important to make clear some assumptions this book makes. First, it is assumed that the set of canonical links provided is sufficient to represent all relationships of interest found in the text at hand. Second, the

program assumes that the target-learner has the background necessary to understand all those concepts whose definitions are not available to MapTutor, as seen in the previous chapter. The learner must also have some degree of linguistic sophistication so that she will have no trouble constructing simple bridging inferences (see McKoon & Ratcliff, 1992; Just & Carpenter, 1987). The latter assumption means that the program is not endowed with capability to deal with students who fall below the entry-level required.

4.3 Modelling the Learner's Performance

Bangert-Drowns, Kulik, Kulik, & Morgan (1991) reviewed a great deal of research into feedback and found that when the learner was sure that her answer was incorrect she was more likely to study the feedback, than when she was unsure about it. In a context similar to the present research, Feifer (1989) reaches an identical conclusion. Feifer's Sherlock was not very successful at determining why a learner's action went wrong, because its ability to identify and explain why a learner's reasoning was faulty was constrained by the use of buggy rules, which required both the designer's intuition and previous protocol analysis[1]. Only when Sherlock could match the learner's reasoning against one of its buggy rules, was it able to explain the reason for a wrong link by using the canned explanation hand coded for the relevant buggy rule. Therefore, most of the time, Sherlock's feedback merely stated that there was a problem, but did not either explain where the problem was or help the learner to build up the necessary understanding. As a consequence, the learners were not willing to change their minds, as Feifer says, 'With very few exceptions subjects did not believe Sherlock's diagnosis, and thus did not change their beliefs as a result of the feedback' (p. 145).

It seems clear then that, in order to be effective, a tutor ought to be able to explain to the learner why she is wrong, when she is. Therefore, what is needed is a diagnostic procedure, which not only determines when, but also, and mostly, why the learner is wrong. Such a diagnostic procedure is essential for good tutoring in the present context and has guided the design of MapTutor.

MapTutor's diagnosis is carried out by a set of procedures shown as boxes in **Figure 4–1**. At a glance, upper-case names represent evaluation constants (i.e., values returned by the diagnostic procedures), and an arrow-up represents the possible return values of the procedure just above it. Arrow-down lines represent procedure calls. The name(s) associated with each of them represent(s) the situation(s) in which the respective call applies; if there are no names, the call applies

[1] Elsom-Cook (1993), presents convincing arguments against the use of buggy rules.

whenever the calling procedure does. These situations in which a procedure can be applied are precisely the triggering conditions introduced earlier in **Chapter 3**. Procedure *DiagnoseWrongLink*, the heart of the diagnosis process, is a virtual decision procedure which determines which basic diagnostic procedures (represented by boxes below it) will be called according to both the situation and the strategy currently used by the program. <*evaluation*> is a pseudo-name representing the returned value of the top-most evaluation procedure. Any, but only one, value returned by its descendant procedures may be assigned to <*evaluation*> at the end of diagnosis. <*suspect concept(s)*>/<*suspect link(s)*> is (are) one or two concept(s)/link(s) the respective procedure believes has(ve) caused the misunderstanding. These values are stored somewhere in order to be used by the teaching procedures when needed. Procedure *AskLearner* is called whenever the calling diagnostic procedure has found a suspect cause of error but its judgement is not conclusive. **Figure 4–1** also shows a hierarchy which holds among MapTutor's diagnostic procedures. That is, the bottom-most procedures are the ones which actually carry out the diagnosis — these are henceforth called *basic diagnostic procedures*. Basic diagnostic procedures are only allowed to return either SUCCESS or FAILURE. On the other hand, the procedures above the basic ones are decision procedures which play two roles: (1) deciding what to do (e.g., calling a basic diagnostic procedure, asking the learner), and (2) establishing which diagnostic returned by the basic diagnostic procedures is the most likely to be correct. This hierarchy makes the program more flexible and adaptable to a number of strategies (see below). In the following sections, this hierarchy will be traversed and discussed in a top-down fashion, but having a look at the bottom procedures in **Section 4.6** before reading on may be helpful.

4.4 Determining the Link between Two Concepts

The top-most diagnostic procedure in **Figure 4–1** determines whether a link made by the learner is correct or not. Given that relationships are represented into MapTutor's knowledge base according to the prototype in **Table 3–9**, finding out whether or not the learner has drawn an expected link is straightforward: simply verify whether the concepts of interest are related to each other, and if so, verify whether one of the links[2] used to represent the relationship corresponds to the one the learner has just drawn.

[2] There may be more than one link which fits a given relationship, and the RELATIONSHIP prototype, presented in **Table 3–9**, could include a rank indicating which link, if any, would be the most appropriate for representing each relationship. However, this does not hinder the current research goals from being fulfilled and this has been left as future work.

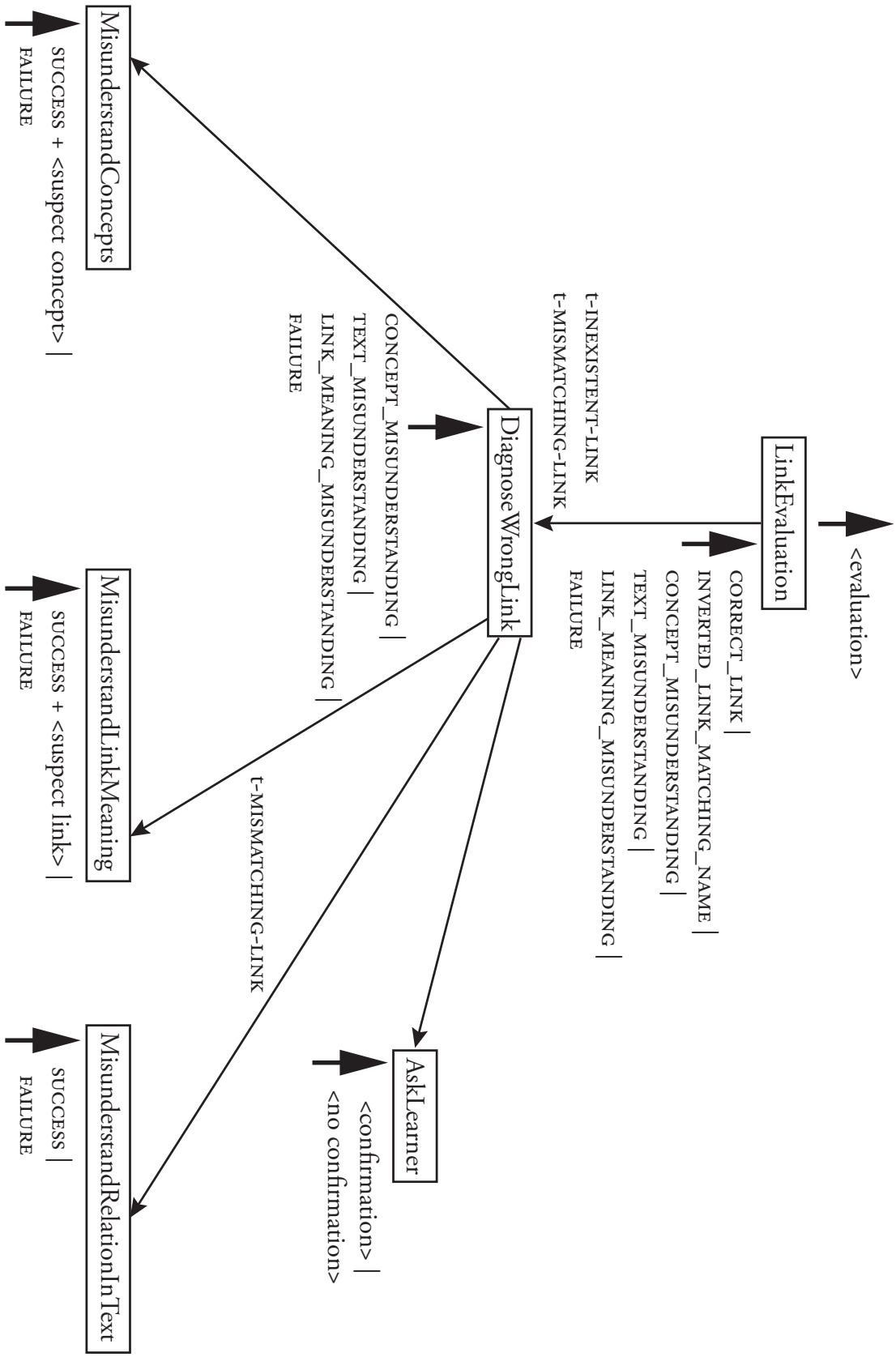

FIGURE 4–1: MAPTUTOR'S DIAGNOSIS

The ideal situation occurs when the learner has drawn the expected link. In this case, MapTutor returns correct_link and stops the diagnostic cycle. On the other hand, the worst situation, from a cognitive point-of-view, is when the learner has drawn a link which does not exist at all; i.e., the program knows nothing about any relationship between the given concepts. In this situation, triggering-condition is set to t-inexistent-link. An intermediary situation occurs when the name of the link used by the learner matches the name of the expected link, but it has been drawn in a direction contrary to that specified by the semantics of the link. In this case, everything is considered fine, except that the link has been inverted due to a mere slip. The link is considered as a wrong one, but the cause is not further investigated. Thus, MapTutor returns inverted_link_matching_name as the result of diagnosis. Finally, the last situation considered is when there is a relationship between the given concepts but there is no match at all between the student's link and any of the expected links. In this case, triggering-condition is set to t-mismatching-link.

When the link is acceptable, no further evaluation is necessary. Otherwise, MapTutor calls the appropriate procedure(s) to proceed the evaluation and try to find out the cause for the wrong link. Procedure *DiagnoseWrongLink*, described next, is responsible for the decisions taken in this subsequent analysis.

4.5 Establishing the Cause of a Wrong Link

Procedure *DiagnoseWrongLink* (see procedural hierarchy in **Figure 4–1**) tries to establish why the link is wrong, once procedure *LinkEvaluation* has determined so. There are two cases to be considered here:

Case 1: The· learner has added a link between two concepts which are not (directly) related to each other (i.e., triggering-condition = t-inexistent-link — indicating that there is no link whatsoever between these concepts in the program's knowledge base). In this case, MapTutor calls *MisunderstandsConcepts* to verify whether the cause was misunderstanding of the concepts. The justification for this decision runs as follows.

To make a link between two concepts, one must (1) find in the text a relationship between the two concepts, and (2) map this relationship onto one of the links provided by the map system. MapTutor assumes that a kind of closed-world assumption (see, e.g., Reiter, 1978) applies to the text, so that all correct, relevant information about any pair of concepts of interest is contained in the text. It follows that any conclusion, which is relevant to understanding the text, about a pair of concepts of interest and the relationship which holds between

them can be drawn from the text (as opposed to, for example, from the student's background knowledge). It is assumed then that the learner will hardly make a non-existent link if she has correctly understood both concepts in the first place[3]. Therefore, MapTutor's best guess in this case is to assume that the learner does not know one or both concepts.

To sum up, when the triggering condition is t-inexistent-link, MapTutor is left with only two options: applying procedure *MisunderstandsConcepts* or procedure *AskLearner* since the other basic diagnostic procedures all assume the existence of a relationship to be mapped onto a link. That is, there is nothing else MapTutor could do here, because all other diagnostic procedures assume that there is at least one correct link, but in the current situation there is none (according to the program's knowledge base). Thus, when *MisunderstandsConcepts* fails the last chance is to ask the learner why she has made the wrong link. If this last resource fails, *DiagnoseWrongLink* returns failure too.

The following alternative strategies only are applicable when the triggering condition is t-mismatching-link.

Case 2: The learner has added a link between two concepts where there is one, but the link is inappropriate (triggering-condition = t-mismatching-link). It seems equally likely that the learner could have misunderstood:

1. the meanings of the appropriate/wrong links — procedure *MisunderstandsLinkMeaning* is responsible for checking this case;
2. the actual relationship between the two concepts in the text — procedure *MisunderstandsRelationship* is responsible for checking this case; or
3. one/both concept(s) of interest — this case is procedure *MisunderstandsConcepts*'s duty.

The diagnostic strategy adopted in this case is to define a criterion of relative confidence which holds among the basic diagnostic procedures, so that MapTutor can compare and then decide which returned diagnostic is the most likely to be correct when two or more procedures return success. By the same token, MapTutor can decide which diagnostic procedure is the most likely to be incorrect when two or more procedures return failure.

Thus, we can define ordering relation[4] $\prec$ on the set of diagnostic procedures with the meaning: is less likely than or as likely as. In other words, *basic procedure* p_1

[3] It might be the case that the learner does not understand the meaning of the link either, but this is not considered as the primary cause of her error because she was not expected to draw any link at all in the current situation.

[4] See, e.g., A Survey of Modern Algebra by Birkhoff & Mac Lane (1965).

$\prec$ *basic procedure* p_2 *if and only if the confidence of procedure* p_1 *is less than or equal to the confidence of procedure* p_2.

There are six possibilities of choosing this confidence precedence[5] and they are all represented in MapTutor's diagnostic decision procedure (i.e., *DiagnoseWrongLink* in **Figure 4–1**). The criterion used to choose the default ordering reflects the basic assumption that the greatest difficulty faced by a novice learner in mapping is to understand the semantics of the graphical representation (i.e., how a given canonical link represents a relationship in the text). Thus, as *MisunderstandsLinkMeaning* is the basic diagnostic procedure closest to this assumption, it should have the highest priority. The next higher priority should be given to *MisunderstandsRelationship*, because if the learner is assumed to know the meanings of the involved links, then it is more likely that she does not understand the relationship between the involved concepts than the concepts themselves (see **Section 4.6**). Therefore, the default total ordering over MapTutor's set of basic procedures has been chosen as:

$$\textit{MisunderstandsConcepts} \prec \textit{MisunderstandsRelationship} \prec \textit{MisunderstandsLinkMeaning}$$

It follows that, when triggering-condition has been determined as t-mismatching-link, MapTutor calls all diagnostic procedure in turn, and then decides based upon the ordering relation defined above what to do with the returned values. There are altogether eight possibilities — called decision sets. These decision sets are explored below. In the discussion which follows, *MCSucceeded*, *MLMSucceeded* and *MRSucceeded* mean the successful results of diagnosis returned by procedures *MisunderstandsConcepts*, *MisunderstandsLinkMeaning*, and *MisunderstandsRelationship*, respectively. The negation ('¬') of any of those symbols means that the corresponding diagnostic procedure failed at arriving at a conclusive result (i.e., returned failure). Also, concept_misunderstanding, link_meaning_misunderstanding, and text_misunderstanding are values returned by decision procedure *DiagnoseWrongLink* when it is confident of the successful outcomes returned by basic diagnostic procedures *MisunderstandsConcepts*, *MisunderstandsLinkMeaning*, and *MisunderstandsRelationship*, respectively. In other words, these values represent the reason the program believes in that led to a wrong link.

[5] The easiest way of seeing this is to consider each possible definition as an ordered list, so that a procedure precedes another when the former comes before the latter in this list. For example, list $[p_2, p_3, p_1]$ means that $[p_2 \prec p_3 \prec p_1]$. Therefore, the number of possible definitions of precedence is the number of permutations in a three-element ordered list, which is 3! = 6.

Decision Set 1: {*MCSucceeded*, ¬*MLMSucceeded*, ¬*MRSucceeded*}. Return CONCEPT_MISUNDERSTANDING as the result of diagnosis.

Decision Set 2: {¬*MCSucceeded*, *MLMSucceeded*, ¬*MRSucceeded*}. Return LINK_MEANING_MISUNDERSTANDING as the result of diagnosis.

Decision Set 3: {¬*M CSucceeded*, ¬*MLMSucceeded*, *MRSucceeded*}. Return TEXT_MISUNDERSTANDING as the result of diagnosis.

Decision Set 4: {*MCSucceeded*, ¬*MLMSucceeded*, ¬*MRSucceeded*}. In this case, both procedures *MisunderstandsLinkMeaning* and *MisunderstandsRelationship* succeeded. Two strategies and variations thereof would be possible here:

(1) apply the confidence ordering over the procedures of interest (i.e., MAPTUTOR is more confident of the result returned by *MisunderstandsLinkMeaning* than of that returned by *MisunderstandsRelationship*);

(2) eliminate one of the possibilities by asking the learner to confirm or not one of them.

Currently, the first strategy has been used by the program. Thus the returned value will be LINK_MEANING_MISUNDERSTANDING in this situation.

Decision Set 5 {*MCSucceeded*, *MLMSucceeded*, ¬*MRSucceeded*}. In this case, both *MisunderstandsConcepts* and *MisunderstandsLinkMeaning* succeeded. The previous comment also applies here, and currently MAPTUTOR uses strategy 1 above, with confidence of *MisunderstandsLinkMeaning* outcome greater than confidence of *MisunderstandsConcepts*'. Therefore, MAPTUTOR returns LINK_MEANING_MISUNDERSTANDING as the result of diagnosis.

Decision Set 6 {*MCSucceeded*, ¬*MLMSucceeded*, *MRSucceeded*}. In this case, both *MisunderstandsConcepts* and *MisunderstandsRelationship* succeeded. Again, the previous comment applies here, and currently, MAPTUTOR use strategy 1, with confidence of *MisunderstandsRelationship* outcome greater than confidence of *MisunderstandsConcepts*'. Therefore, the returned value will be TEXT_MISUNDERSTANDING.

The first three decision sets above represent ideal situations where only one basic diagnostic procedure succeeded. Thus, MAPTUTOR is not left with options to

choose from: it simply returns the diagnostic value corresponding to the procedure which succeeded[6].

Decision Set 7 {¬*MCSucceeded*, ¬*MLMSucceeded*, ¬*MRSucceeded*}. No procedure has succeeded. A number of strategies seem to apply in this situation. For example, ask the learner successively until either she confirms one of the possible diagnostic outcomes, or the program runs out of stem and returns FAILURE. The drawback of this approach is that MAPTUTOR could end up asking tree times in a row (before perhaps realising that it would run out of steam anyway). Another alternative is to ask confirmation only for the procedure MAPTUTOR has more confidence. The advantage of this approach is that the program would be less obtrusive and annoying. The disadvantage is that it cuts the chances of succeeding, but nonetheless, the latter alternative has been chosen anyway. Thus, in this case, MAPTUTOR's last chance is to ask the learner whether she understands the meanings of the correct and the wrong links so as to try to establish link meaning misunderstanding as the nature of her wrong link. If after asking the learner it still cannot determine the cause, it will return FAILURE. Notice that MAPTUTOR has not run out of steam, actually, as it had two alternative questions to ask the learner. Instead, this is based upon a pedagogical decision: MAPTUTOR does not ask twice in a row so as not to annoy the learner.

Decision Set 8 {*MCSucceeded*, *MLMSucceeded*, *MRSucceeded*}. All procedures have succeeded. The strategy used here is similar to that in **Decision Set 7**: MAPTUTOR asks the learner for confirmation of the diagnosis it has more confidence (i.e., LINK_MEANING_MISUNDERSTANDING). If the learner does not confirms this diagnostic, it will return the next diagnostic according to the default confidence ordering (i.e., TEXT_MISUNDERSTANDING).

From this point on, MAPTUTOR can use a number of strategies to test a number of hypotheses. What follows is the strategy currently used by the program. The important point to bear in mind is that, despite the fact that the default ordering

[6] Of course, we always have the option of asking the learner for confirmation, and indeed this option is also implemented in MAPTUTOR. Nonetheless, it has not been used, because doing so would be terribly annoying for the learner.

relation above is somewhat arbitrary, the general diagnostic strategy allows for easy modification during an interactive session with the program if the current strategy proves to be ineffective with a particular learner. This shift can be done either by the program itself (e.g., by using some machine learning mechanism), or by human intervention (e.g., by simply choosing another strategy in a pull-down menu — as shown in **Chapter 5**).

4.6 Basic Diagnostic Procedures

In general, MapTutor's basic diagnostic procedures attempt to answer the following questions:

- Does the learner understand the meaning of both concepts of interest? Procedure *MisunderstandsConcepts* (see **Section 4.6.1**) is responsible for checking this.
- Does the learner understand the (implicit/explicit) information in the text, especially and the relationship which holds between both concepts of interest? Procedure *MisunderstandsRelationship* (see **Section 4.6.2**) is responsible for checking this.
- Does the learner understand the semantics of both the expected and the wrong links? Procedure *MisunderstandsLinkMeaning* (see **Section 4.6.3**) is responsible for checking this.

At a first sight, it may appear that the first two cases above are inconsistent, or at best, the second case is contained in the first. After all, it is generally accepted (see, e.g., Howard, 1987) that no concept stands for itself; i.e., the meaning of a concept stems from its relationship to other concepts. Nevertheless, we do not need to know all relationships that could possibly exist between a concept and other concepts in order to be able to categorise that concept. For example, we do not necessarily need to know that cat is a member of family Felidae in order to know the meaning of concept cat. Knowing that a cat is a domestic animal which has a short muzzle, large eyes, whiskers, sharp claws, etc., is enough to know the meaning of cat in most practical situations. Suppose we have a hypothetical text which discusses the relationship between cat and Felidae, but does not contain this basic definition of cat. If this simple example were the case, procedure *MisunderstandsConcepts* would be in charge of checking whether the learner knows that basic definition of cat (i.e., that cat has a short muzzle, whiskers, etc.), whereas if the relationship being investigated were that between

cat and Felidae, procedure *MisunderstandsRelationship* would be called upon to try to verify whether she understands this relationship in the hypothetical text.

4.6.1 Basic Diagnostic I: Concept Misunderstanding

Procedure *MisunderstandsConcepts* checks whether the learner knows one or both concepts she has just linked according to the learnability criterion adopted by MapTutor, which is that of membership of the known-concepts list described earlier in **Chapter 3**. MapTutor keeps a list, called culprit-concepts, where it temporarily stores concepts it suspects the learner of misunderstanding. Whenever a concept being investigated by procedure *MisunderstandsConcepts* does not satisfy the learnability criterion, it is added to this list. Since, only two concept are examined each time this procedure is called, this list has in fact at most two elements. It follows that if one of the concepts under consideration in not in list known-concepts, it is considered a culprit and consequently put in list culprit-concepts. If neither of the concepts is in known-concepts, both are considered culprits, and both are added to list culprit-concepts.

Remember from **Chapter 3** that a concept is a member of list known-concepts if it satisfies one of the following criteria:

1. It is part of the student's prior knowledge; i.e., the program assumes beforehand that the learner knows this concept. This requirement prevents the program of providing silly feedback such as the definition of concepts like snail, trees, etc.
2. The learner is assumed to have mastered it by having it satisfy the concept-learnability criterion defined in **Section 3.5**.

Table 4–1 sums up the algorithm followed by procedure *MisunderstandsConcepts*.

algorithm
1. Reset suspect-concepts to the empty list. 2. Check if both concepts of interest are in list known-concepts. 3. If any of the concepts is not in list known-concepts, add it to list suspect-concepts. 4. If any concept has been added to list suspect-concepts, return success; otherwise, return failure.

Table 4–1: Procedure MisunderstandsConcepts

4.6.2 Basic Diagnostic II: Misunderstanding the Relationship

Procedure *MisunderstandsRelationship* tries to determine whether the learner understands the piece of text containing the relationship which holds between both concepts whose link is being investigated. This procedure operates by quantifying how difficult it is to grasp the given relationship before mapping it onto the appropriate link. As seen in **Section 3.7.3**, information about the text is included into the program which allows it to predict that certain pieces of text may cause trouble for the learner. For example information that a given piece of text describing the relationship between two concepts is ambiguous or requires some inference to uncover the relationship itself strengthens the belief that the learner might not have understood that piece of text well. **Table 4–2** summarises the algorithm executed by Procedure *MisunderstandsRelationship*.

ALGORITHM

1. Reset evidence counter to zero.
2. Verify whether the relationship is explicit in the text. If it is not explicit and require some inferences even to uncover it, increase evidence by one.
3. Check whether the relationship is ambiguous. If it is ambiguous and may have more than one interpretation, increase evidence by one.
4. Verify how much reasoning is needed to go from the text representation to a map representation. If some reasoning is required, increase evidence by one. If it may be tricky, increase evidence by two.
5. Return SUCCESS if the confidence in the diagnosis (i.e., evidence counter/maximum score that could be accumulated) is at least equal to a pre-set threshold; otherwise, return FAILURE. This pre-set threshold corresponds to 75%.

TABLE 4–2: PROCEDURE MISUNDERSTANDSRELATIONSHIP

In **Table 4–2**, there are three tests (**Steps 2–4**) to be performed. The first two tests (**Steps 2** and **3**) may increase the evidence counter by at most one, whereas the third test (**Step 4**) may increase it by at most two. Thus, the maximum value which the evidence counter can accumulate is four. The confidence in the diagnosis is defined as the proportion of the actual value of the evidence counter at the end of tests to the maximum value it could accumulate. The diagnosis

confidence threshold referred to in **Step 5** corresponds to the fact that this procedure will have succeeded when the evidence counter accumulates three points (three out of four equals 75%), which seems fair enough.

4.6.3 Basic Diagnostic III: Misunderstanding Semantics of Links

Procedure *MisunderstandsLinkMeaning* tries to establish whether the learner understands the meanings of both the expected and the wrong types of canonical links. Specifically, it verifies whether either of the involved link types have been correctly used before. MapTutor maintains a list of suspect-links where it stores canonical links considered suspects by this procedure. A canonical link type is considered a suspect if either it has been misused most of the time (i.e., the number of wrong uses of the link is greater than the number of correct ones), or it has never been used before. A link type can also be considered a suspect if it is inherently ambiguous, i.e., confusing for the learner. As seen in **Section 3.7.2**, MapTutor keeps this information for each canonical link type represented in its knowledge base. **Table 4–3** sums up the algorithm followed by procedure *MisunderstandsLinkMeaning*. The two involved canonical links referred to in this table correspond to the correct (i.e., the expected) link name and the wrong name the learner used in drawing her link.

ALGORITHM

1. Reset suspect-links to the empty list.
2. For each of the two involved canonical links do:
   - 2.1 If the link type has never been used before, add it to suspect-links.
   - 2.2 Get a list of the link's usage. If the link has been misused most of the time, include it into suspect-links.
   - 2.3 Check whether this link itself is inherently ambiguous. If so, add it to suspect-links.
3. If at least one link has been added to list suspect-links, return success; otherwise, return failure.

Table 4–3: Procedure MisunderstandsLinkMeaning

4.7 Asking the Learner

The procedure described here is called whenever MapTutor decides to ask the learner to confirm or not the suspicion of a given diagnostic. This procedure has

a parameter which specifies which diagnostic the confirmation has been requested for. Accordingly, it may be called to ask whether the learner is sure about her understanding of the concepts of interest, the meaning of canonical link types, or a relationship which occurs in the given text. Moreover, to avoid asking many times whether the user know something (e.g., the meaning of a given relationship), the program keeps a list of already-asked questions along with the respective answers to these questions.

This procedure uses question templates corresponding to each kind of question. The question templates currently in use are as follows[7]:

- **Case 1**: **the suspicion is about misunderstanding of concepts**. 'Are you sure you understand the meaning of concept <*concept*>?'
- **Case 2**: **the suspicion is about misunderstanding of semantics of links**. 'Are you sure you understand the meaning of link <*link*>?'
- **Case 3**: **the suspicion is about misunderstanding of a relationship in the text**. 'Are you sure you understand the relationship which holds between concepts <c_1> and <c_2> as presented in the selected piece of text?' In this case the program also highlights the corresponding piece of text it is asking about.

Table 4–4 summarises the algorithm followed by procedure *AskLearner*.

ALGORITHM

1. In the case where misunderstanding of concepts is being investigated, do the following:
   - 1.1 If there are two suspect concepts, ask for confirmation of both; if there is only one, ask for confirmation of it. In any case, keep the suspect concept (if any) the learner confirms not knowing, and retract from the list the one (if any) the learner is sure to know.
   - 1.2 If there are no suspects yet, try to find one by asking the learner about the concepts she has just linked together.
   - 1.3 Return SUCCESS if the learner confirms not knowing at least one concept. Otherwise, return FAILURE.

TABLE 4–4: PROCEDURE ASKLEARNER (CONTINUES)

[7] Names enclosed by angle brackets represent slots to be instantiated with the corresponding object (concept or canonical link name) being inquired.

2. In the case where misunderstanding of semantics of canonical links is being investigated, do the following:
   2.1 If there are two suspect links, ask for confirmation of both; if there is only one, ask for confirmation of it. In any case, keep the suspect link (if any) the learner confirms not knowing, and retract from the list the one (if any) the learner is sure to know.
   2.2 If there are no suspects yet, try to find one by asking the learner about both the link she has just used and the most correct, expected link.
   2.3 Return SUCCESS if the learner confirms not knowing at least one link. Otherwise, return FAILURE.
3. In the case where misunderstanding of a relationship is being investigated, ask the learner whether she understands the piece of text which represents the proposition she intended to depict when drew her last link. If she confirms that she does not understand this piece of text, return SUCCESS; otherwise, return FAILURE.

TABLE 4–4 (CONTINUED): PROCEDURE ASKLEARNER

As can be apprehended, in the cases where the program asks about a suspicion of either a concept or a link, it keeps the suspect in the respective list (i.e., SUSPECT-CONCEPTS or SUSPECT-LINKS) when the learner confirms the suspicion, or otherwise, retracts the suspect from the respective list. Notice, however, that this procedure acts slightly different in the case where it asks about misunderstanding of relationships, because in this case the program keeps no list of suspects. Thus, in this situation, MAPTUTOR asks simply whether the learner understands the relationship of interest by using data from the relationship representation (see **Section 3.7.3**).

4.8 Updating the Performance Model

MAPTUTOR's performance model attempts to represent the learner's understanding of the mapping session — mostly by means of keeping lists BELIEVED-CONCEPTS and KNOWN-CONCEPTS updated. Given that the learner has just linked concepts c_i and, c_j, updating the performance model consists in calculating their BDs (see **Section 3.5**) and putting them in lists BELIEVED-CONCEPTS or KNOWN-CONCEPTS, accordingly. Procedure *UpdatePerformanceModel*, described in **Table 4–5**, does this job.

ALGORITHM

1. If c_i is in list BELIEVED-CONCEPTS, calculate the BD of c_i by using the following formulae:

$$new\ BD(c_i) = \frac{old\ BD(c_i) \times LV(c_i) + ass(c_i, c_j) \times rank(c_j)}{LV(c_i)}$$

   otherwise, calculate the BD of c_i as:

$$BD(c_i) = \frac{ass(c_i, c_j) \times rank(c_j)}{LV(c_i)}$$

2. Update list BELIEVED-CONCEPTS by storing c_i along with its respective BD.
3. If $BD(c_i) \geq KT$, add c_i to list KNOWN-CONCEPTS; if c_i is already there, do nothing.
4. If $BD(c_i) < KT$ and c_i is in list KNOWN-CONCEPTS, take it out of there; if c_i is not there, do nothing.
5. Repeat **Steps 1** to **4** for concept c_j.
6. Finally, check whether the learner already know all major concepts. Is so, call the appropriate method to close the session.

TABLE 4–5: PROCEDURE UPDATEPERFORMANCEMODEL

When the learner deletes a link, MAPTUTOR calls a procedure similar to the one just described to update both lists BELIEVED-CONCEPTS and KNOWN-CONCEPTS in order to take this fact into consideration. Note that a concept formerly linked by a deleted link may also be included into KNOWN-CONCEPTS, in the case where the learner has deleted a wrong link so that the concept's BD increases.

MAPTUTOR provides two short-cuts in the case where the learner wants to modify a link drawn on her map: link renaming and link inversion (see **Chapter 5**). However, as far as updating is concerned, when the learner inverts or renames a link, MAPTUTOR proceeds as if she had deleted the old link and drawn a new one in its place in the opposite direction or with a new name, respectively.

4.9 Providing Feedback

MAPTUTOR teaching procedures are schematised in **Figure 4–2**. These procedures are divided into two groups:

1. **Basic teaching procedures** — which are in charge of providing either corrective feed back, in case the program has diagnosed some misunderstanding, or informative feedback, which informs the learner about her performance. These procedures are the ones connected in form of tree in **Figure 4–2**.
2. **Auxiliary teaching procedures** — which are responsible for providing the learner with both additional information and suggestions so as to keep the interaction going smoothly. Auxiliary procedures currently implemented in MapTutor are the unattached ones at the bottom of **Figure 4–2**.

In **Figure 4–2**, *Teach* is a decision procedure; attached to it are the basic teaching procedures; unattached procedures at the bottom are auxiliary teaching ones.

4.9.1 Basic Teaching Procedures

MapTutor basic teaching procedures provide one kind of feedback for each diagnostic output, and at most one variation thereof. MapTutor's general approaches to feedback consist in:

- providing immediate feedback;
- telling the learner whether the answer is right or wrong;
- supplying the learner with the correct answer if her last task resulted in a wrong link; and
- providing more elaborate explanation of why her link was wrong, when this is the case.

It follows that, when MapTutor believes the learner is wrong, it will tell her, in the first place, that she is wrong and then provide instructional corrective feedback according to the reason pointed out by the diagnostic procedures described above. Accordingly, the program patterns itself on the following principles:

- If the learner does not understand one or both concepts, clarify concepts and facts about the domain;
- If the learner does not know how to use the appropriate link, teach about domain relationships and semantics of links. That is, explain the meaning of the right link and why it is appropriate in that situation.
- If the learner cannot grasp a piece of text containing the relationship between two given concepts, teach her the kind of reasoning she could employ in order to under stand the given relationship using the text at hand.

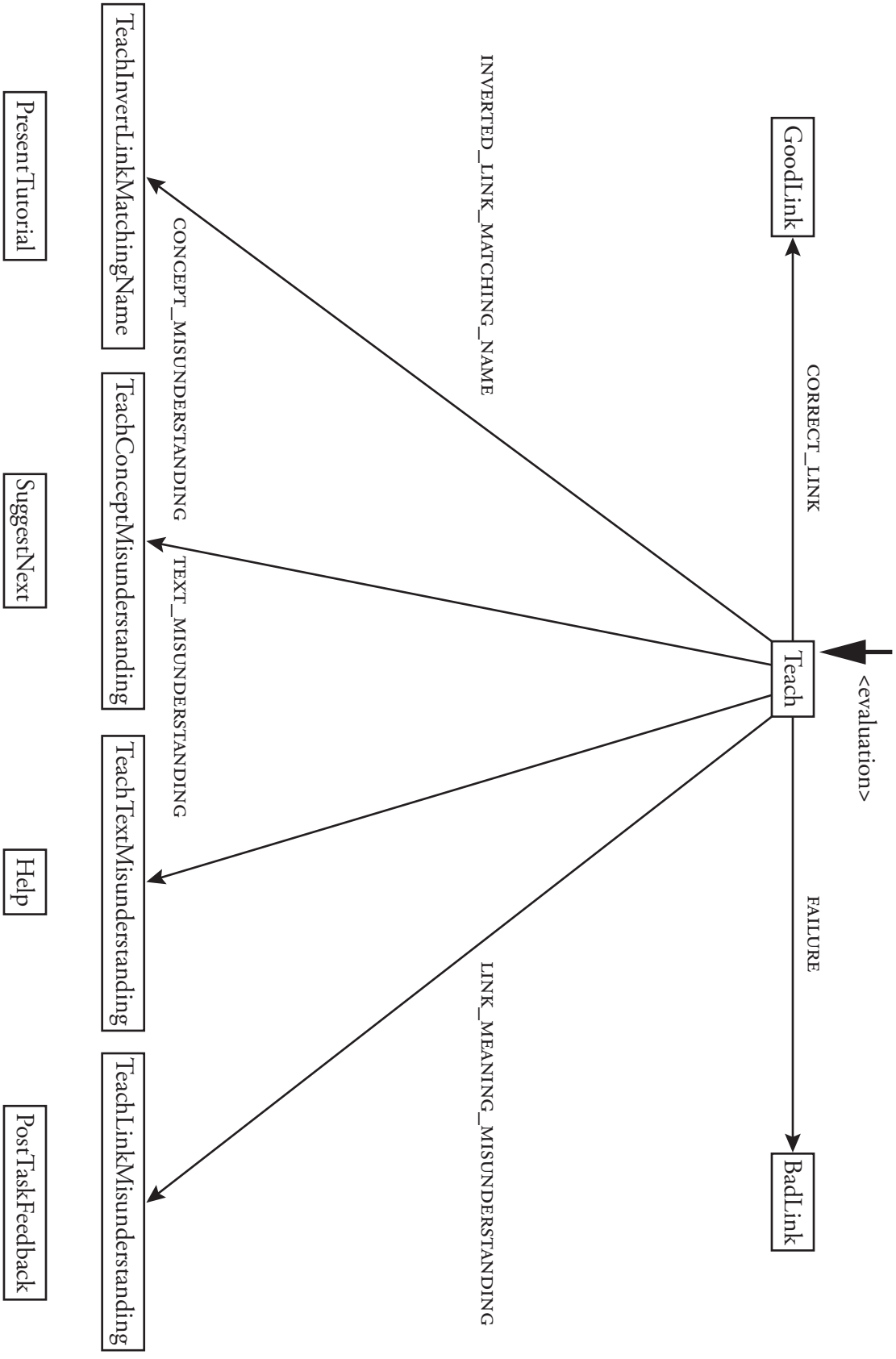

FIGURE 4–2: MAPTUTOR'S TEACHING PROCEDURES

Whatever the outcome of diagnosis is, MapTutor calls meta-level procedure *Teach*, which in turn, is in charge of calling the appropriate basic teaching procedure to deal with the case accordingly. **Table 4–6** shows how procedure *Teach* operates. The basic procedures called by *Teach* are the ones which actually carry out the duty of providing corrective or informative feedback. These procedure are described next.

When diagnostic is...	Call procedure...
CORRECT_LINK	*GoodLink*
INVERTED_LINK_MATCHING_NAME	*TeachInvertLinkMatchingName*
CONCEPT_MISUNDERSTANDING	*TeachConceptMisunderstanding*
TEXT_MISUNDERSTANDING	*TeachTextMisunderstanding*
LINK_MEANING_MISUNDERSTANDING	*TeachLinkMisunderstanding*
FAILURE	*BadLink*

Table 4–6: Decision Procedure Teach

Procedure GoodLink. This basic informative procedure is called upon whenever the learner has done well — i.e., when she has drawn an expected, correct link. MapTutor just says 'Good link!', so that she can follow her performance. It also highlights for a while the actual relationship in the text so as to make sure the learner will become aware of the connection between the relationship in the text and the link she has just drawn.

Procedure TeachInvertedLinkMatchingName. This corrective procedure applies whenever INVERTED_LINK_MATCHING_NAME is returned by the diagnostic procedures. This is the simplest of all corrective teaching actions implemented by MapTutor, because the program understands that, in this situation, the only mistake made by the learner was to invert the orientation of the link. That is, it is assumed that the learner has not done so bad, since she knows that there is a relationship between the two involved concepts and has chosen the right link to apply, although in the wrong direction.

The form of feedback presented by this procedure is shown in **Table 4–7**[8].

[8] In this table and in others to follow, the expressions enclosed by angle brackets are to-be-filled slots, and their meanings should be clear. Furthermore, the feedback messages are not presented as a single frame as it may appear. Instead, they are distributed over a number of balloons (see **Seção 5.7**).

Concepts <c_1> and <c_2> are in fact related to each other and the appropriate link is really <*link name*>. But you applied the link in the wrong direction. You should invert it.

TABLE 4–7: FORM OF FEEDBACK PROVIDED BY TEACHINVERTEDLINKMATCHINGNAME

Procedure TeachConceptMisunderstanding. This is another corrective procedure which applies when CONCEPT_MISUNDERSTANDING is returned as the value of evaluation. It is based on the following two premises:

1. If the learner does not understand one (or both) concept(s), clarify the concept(s) by presenting definitions more elaborate than those found in the text so as to clarify the misunderstanding.
2. If there are two concepts c_1 and c_2 to teach, verify whether knowing concept c_1 is necessary before concept c_2 can be learned. If so, teach concept c_1 first. MAPTUTOR uses slot **Prerequisite** of the CONCEPT **Prototype**, defined in **Section 3.7.1**, for this purpose.

The form of feedback presented by this procedure is shown in **Table 4–8**.

I believe you don't know the real meaning of <*concept*>. This concept can be defined as <*concept definition*>.

TABLE 4–8: FORM OF FEEDBACK PROVIDED BY TEACHCONCEPTMISUNDERSTANDING

Procedure *TeachLinkMeaningMisunderstanding*. This corrective procedure applies whenever LINK_MEANING_MISUNDERSTANDING is returned by the diagnostic procedures. It attempts to show the learner that using the wrong link, as she is supposed to have done, will lead to a map configuration which could be interpreted in a way that is incompatible with what is stated in the text. In addition, if the learner does not know how to use the appropriate link, MAPTUTOR teaches her about the meaning (semantics) of the right link and why it is appropriate in the current situation. Hence, the feedback takes the general form shown in **Table 4–9**. This form of feedback message is divided into four pieces (identified by [1], [2], [3] and [4] in **Table 4–9**). Piece [2] is only used when the program has determined that the learner does not understand the meaning of the wrong link, whereas piece [3] is only used when the program believes she does not understand the meaning of the correct one. Pieces [1] and [4] are common to both situations.

[1] I'm afraid you've just made a mistake, because linking concept <c_1> to <c_2> by using link <*wrong link name*> leads to a proposition like: <*proposition*> which according to the text is clearly false. [2] Linking a concept to another using <*wrong link name*> means that <*wrong link meaning*>. [3] On the other hand, linking a concept to another using <*correct link name*> means that <*correct link meaning*>, so that <*wrong link name*> is not the appropriate link to use in this situation. [4] You should use <*correct link name*> instead.

TABLE 4–9: FORM OF FEEDBACK PROVIDED BY TEACHLINKMEANINGMISUNDERSTANDING

The feedback template in **Table 4–9** may sound a bit weird, so that an instantiation of the template above is called upon. Suppose, for example, that the learner has just made a link from concept MICROHABITAT to concept HABITAT, using canonical link IS A, but MAPTUTOR believes that link PART OF would be more appropriate. Then, assuming that the program has determined that both links must be taught, it would deliver the following message presented in **Table 4–10**[9].

I'm afraid you've just made a mistake, because linking concept MICROHABITAT to HABITAT by using link IS A leads to a proposition like: MICROHABITAT IS A HABITAT which according to the text is clearly false. Linking a concept to another using IS A means that the concept represented by the origin node IS A MEMBER, SUBSET or EXAMPLE of the concept represented by the terminal node. On the other hand, linking a concept to another using PART OF means that the concept represented by the origin node IS PART OF the concept represented by the terminal node, so that IS A is not the appropriate link to use in this situation. You should use PART OF instead.

TABLE 4–10: EXAMPLE OF FEEDBACK PROVIDED BY TEACHLINKMEANINGMISUNDERSTANDING

Procedure TeachTextMisunderstanding. This corrective procedure applies whenever TEXT_MISUNDERSTANDING is returned by the diagnostic procedures.

[9] Lest you still feel this example to be a bit weird, remember that this message would not be delivered as a single frame. In this particular case, the message would be distributed over four balloons.

Each correct link represented in the program's knowledge base has an associated justification based on the following general rules of justification — RJs (see **Section 3.7.3**):

RJl: the relationship between concepts <c_i> and <c_j> is *R*, because the text says so. Very simple paraphrasing (rewording) is also included in this category.

RJ2: the relationship between concepts <c_i> and <c_j> is *R*, because it can be inferred from information in the text. In this case, the information necessary to infer *R* is also included into the representation.

RJ3: the correct link between concepts <c_i> and <c_j> is *L*, because the relationship between concepts <c_i> and <c_j> is *R* (from either RJl or RJ2) and *R* belongs to, or is very close to a member of, the set of keywords of link *L*, and therefore indicates the use of *L*.

RJ4: the correct link between concepts <c_i> and <c_j> is *L*, because the relationship between concepts <c_i> and <c_j> is *R* (from either RJl or RJ2) and as *L* is the link, among the canonical set provided, which has the meaning closest to *R*, it is the indicated link. In other words, *R* can be mapped onto a keyword which indicates the use of *L* and there is no better choice. (Since canonical link systems sometimes do not cover all relationships a given text contains, it may be the case that there is no link among those provided so close to the actual relationship.)

Summing up, each correct link has a two-part justification:

1. The first part is either of type RJl or RJ2 above. This rule represents the justification of the actual relationship.
2. The second part is a rule of type RJ3 or RJ4, and corresponds to the justification of the mapping of the actual relationship onto a canonical link.

Hence, the feedback in the current situation is presented according to the algorithm in **Table 4–11**.

Procedure BadLink. This basic informative procedure is called whenever MapTutor fails at arriving at a reliable diagnostic of a wrong link. In this case, the program says simply: 'I'm afraid your last link was wrong.' There is nothing else the program could do in this situation, because, as it failed at determining the cause of the wrong link, it does not know what kind of corrective feedback it should provide.

ALGORITHM

1. Present the reasoning involved in uncovering the actual relationship:
   - 1.1 If the information in the text is explicit (RJl), show her, by highlighting the relevant piece of text, where she can find it. Form of feedback: 'You can easily verify based on the highlighted piece of text that <*justification*>'.
   - 1.2 If the information is implicit (RJ2), tell her what the actual relationship is and present her with a chain of reasoning that is capable of uncovering the relevant relationship. Form of feedback: 'Based on the highlighted piece of text you should be able to infer that <*justification*>'. (A program intended to teach how to draw inferences from text should, at the very least, tutor about heuristic reasoning or some form of metacognitive awareness which would help the learner to learn how to make inferences. The primary purpose of MAPTUTOR, however, is not to teach heuristic inferences from text.)
2. Present the reasoning involved in mapping the actual relationship onto the appropriate link:
   - 2.1 If RJ3 applies, present the keyword which indicates the use of link *L* in the current situation. Form of feedback: 'The fact that the relationship between <*concept* c_1> and <*concept* c_2> can be expressed using keyword <*keyword*> indicates the use of link <*link*>.'
   - 2.2 If RJ4 applies, present the (sometimes rough) chain of reasoning necessary to map the actual relationship onto the most indicated link L. Form of feedback: 'You can map the actual relationship between <*concept* c_1> and <*concept* c_2> using link <*link*> based on the fact that <*justification*>.'

TABLE 4–11: ALGORITHM FOLLOWED BY TEACHTEXTMISUNDERSTANDING

4.9.2 Auxiliary Teaching Procedures

Auxiliary teaching procedures — represented by the boxes at bottom of **Figure 4–2** — are not called as a result of evaluation, as basic teaching procedures are. Instead, they are intended to complement the teaching environment by providing help at request, hints, and suggestions, among others. These procedures are described next.

Pre-Teaching Tutorial — Procedure PresentTutorial. This procedure simply launches the tutorial program which in turn presents a welcome message and a guided tutorial through the program. This is also the opportunity the program has to present its syntax. Part of the tutorial program is also responsible for implementing the pre-teaching stage of the general approach proposed in **Chapter 3**. The tutorial program is in fact a separate program which works collaboratively with MapTutor (see **Chapter 5**).

Providing More Feedback — Procedure Help. Auxiliary procedure *Help* simply launches the help (or assistant) program — a separate program that helps the learner to use the main program. This program will be discussed in **Chapter 5**.

Suggesting Actions — Procedure SuggestNext. Auxiliary procedure *SuggestNext* presents two concepts for the learner to consider linking together. It is called when the program believes the learner has got stuck. This procedure suggests at least one major concept for the learner to consider next, and it takes as a model the algorithm presented in **Table 4–12**.

Technically speaking, this procedure uses a direct-path, depth-first search to determine whether there is a path (not necessarily a direct one) between the two given concepts. Also, it uses the following suggestion formats:

- When both concepts are drawn: 'Why do not you consider linking <*first concept*> to <*second concept*>?'
- When one concept is drawn but the other is not: 'Look for the relationship between <*the drawn concept*> and other concepts in the text.'
- When none of the concepts is yet drawn: 'Look for the relationship between <*first concept*> and <*second concept*> in the text.' It does not make much sense to suggest two not-yet-considered concepts when they are not related to each other.
- When there is only one concept to suggest and this concept has already been drawn on the map pane: 'Why don't you consider linking <*concept*> to another concept?'
- When there is only one concept to suggest but this concept has not been selected yet: 'Why don't you examine concept <*concept*> in the text?'

Presenting Final Suggestions — Procedure PresentPostTaskFeedback. Auxiliary procedure *PresentPostTaskFeedback* is responsible for presenting the learner with

post-session suggestions about how to improve her future maps. This procedure matches the post-teaching phase of the general approach proposed in **Chapter 3**.

ALGORITHM

1. Start with the last used concept and see if there is a path between this concept and any major concept.
2. If a path has been found, ask the learner to examine the relationship between its last used concept and the first concept in this path.
3. If either there is no last used concept (perhaps the session has just begun) or there is no unknown path between such a last concept (if any) and an unknown major concept, do not suggest the last drawn concept (if any) because it will not lead to mastery of any major concept. The choice here is arbitrary: simply take the first concept in the MAJOR-CONCEPTS list. The preference, however, is for those concepts which are connected (i.e., that has already been linked to another concept on the map). The justification in this case is that the learner is probably closer to mastery of a major concept she has already linked to other concept(s) than to mastery of a concept she has never considered to link before.
4. MAPTUTOR has not any better offer when the attempts above fail to find out a suggestion: it simply suggests the first major concept in the MAJOR-CONCEPTS list (be it drawn or not).

TABLE 4–12: ALGORITHM FOLLOWED BY SUGGESTNEXT

Procedure *PresentPostTaskFeedback* constructs and executes an instructional plan based on the list of major concepts which are still unknown at the end of the session. **Table 4–13** sums up the algorithm followed by this procedure. Some steps in **Table 4–13** are worthy commenting. The sorting carried out in **Step 2.3** is in order to accelerate the process of mastery of major concept MJ. Also, teaching a link between two concepts (**Step 2.5.2**) involves: (1) showing the learner where she can find the relationship between these concepts in the text; (2) telling her how to map this relationship onto an appropriate canonical link; and (3) drawing the appropriate link in the map. Updating the performance model (**Step 2.5.3**) guarantees that **Step 2.5** will be satisfied at some point, because, as only correct links are drawn, BD(MJ) will increase at each cycle.

1. Remind the learner about the wrong links she made during the session:
   - 1.1 For each link the learner has made between two concepts the program does not know of the existence of any relationship between them, tell her that that link may be wrong.
   - 1.2 While there are any wrong links (excluding those in 1.1) left in the map, do:
     - 1.2.1 Correct their wrong names or directions.
     - 1.2.2 Update the performance model as if the learner herself had made the correction.
2. Get the list of major concepts which are unknown to the learner.
3. While there are concepts left in the list of unknown major concepts do:
   - 3.1 Pick a concept in the unknown major concepts list. Let MJ identify this concept.
   - 3.2 Get the list of all concepts related to MJ.
   - 3.3 Sort this latter list according to the ranks of its concept members, so that the concept with highest rank becomes the first in the list, and so on.
   - 3.4 If MJ has not yet been drawn, tell the learner she should have considered selecting this concept in the text, and then draw the concept in the map; otherwise, do nothing.
   - 3.5 While BD(MJ) < KT, do:
     - 3.5.1 If the first concept in the sorted list of concepts related to MJ has not yet been drawn, tell the learner she should have considered selecting it in the text, and then draw it in the map.
     - 3.5.2 Teach the link between this concept and MJ.
     - 3.5.3 Update the performance model as if the learner herself had made this link.
     - 3.5.4 Remove the first concept from the sorted list of concepts related to MJ.
   - 3.6 Remove MJ from the unknown major concepts list.

TABLE 4–13: ALGORITHM FOLLOWED BY PRESENTPOSTTASKFEEDBACK

4.10 Conclusion

MapTutor's diagnostic process is based upon the following analytical reasoning:

- To make a link between two concepts in a graphical map, which represents a proposition in the text at hand, one must:
  1. understand the proposition itself, including the concepts under consideration;
  2. understand the meanings of the canonical links provided so that the textual relationship can be translated appropriately onto the graphical representation.

This chapter has identified three forms of misunderstanding which may hamper this process:

1. Misunderstanding of the piece of text in question;
2. Misunderstanding of the concepts under consideration; and
3. Misunderstanding of meanings of canonical links.

This chapter has also presented procedures which are to some extent able to diagnose each of these forms of misunderstanding. Moreover, one must also have a high-level procedure which supervises the action of these basic diagnostic procedures. The usefulness of this high-level procedure is twofold: (1) to make decisions both about when a given basic diagnostic procedure should be called and about the validity of the returned diagnostic values; and (2) to allow for real-time modification of the current diagnostic strategy.

Furthermore, this chapter has presented three basic types of corrective feedback provided by MapTutor: (1) feedback about the text being read, (2) feedback about the map being constructed, and (3) feedback about the meanings of the canonical links provided. MapTutor's basic feedback procedures are driven by the diagnostic values returned by its diagnostic process, and are intended to show how those values can reconcile themselves with the teaching process.

In summary, his chapter has dealt with two of those issues which may hinder the effective application of a learning strategy pointed out in **Chapter 1**, namely (1) misunderstanding of the text — which has been subdivided into misunderstanding of concepts and misunderstanding of relationships; and (2) misunderstanding of the semantics (i.e., meanings of canonical links) of the learning strategy. From a conceptual point-of-view, dealing with the yet missing factor pointed out in **Chapter 1** — namely, selection of the learning objectives — seems to be

trivial: simply tell the learner she has not selected a relevant portion of the text, in case she has actually done so. Technically speaking, we could simply construct an interface containing the text of interest which would raise a flag whenever the learner has selected a piece of text which is not relevant for the learning objectives. Sadly, despite its low conceptual import, the interface of a computer program like MapTutor is one of the most time-demanding of the whole project. The following chapter discusses the design and implementation of MapTutor interface.