Knowledge Structures, Strategies and Learning Processes in Fault Finding
by
Anders Tunevi
Knowledge Structures, Strategies and Learning Processes in Fault Finding

Anders Tunevi
Infologics AB
Box 91
191 22 Sollentuna
SWEDEN
Tel: +46 8 92 20 00
April 6, 1989

Abstract

A learning system for fault finding has been constructed. This system contains many different types of knowledge, three ways to find faults and four ways to learn fault finding. The constructed learning system works for a class of fault finding problems. This class has been described in the paper. The developed system could be viewed as an architecture of a general learning system for fault finding. The system could also be used as a testbench of learning mechanisms.

The experiences from this project indicates that it is possible to build a learning system when the structure of the knowledge is known.

In this paper the following ideas will be discussed: How can the fault finding and learning techniques be integrated? How can the knowledge structure, fault finding mechanisms and learning mechanisms emerge with the help of a simulator and general mechanisms?
1 The purpose and the plan ................................................................. 1

2 The background ............................................................................. 2
   2.1 Sesam .................................................................................. 2
   2.2 The Sesam Fault Finder project .......................................... 3
   2.3 The ideas and their origin ................................................... 3
   2.4 How this project is different ............................................... 4

3 The system: an overview ............................................................. 6
   3.1 Types of knowledge ........................................................... 6
   3.2 Main components .............................................................. 6
   3.3 Ways to find a fault ............................................................. 6
   3.4 Ways to learn .................................................................. 6

4 Knowledge structures in fault finding ................................ ......... 7
   4.1 An overview of knowledge associated with fault finding of a Sesam
       terminal ........................................................................... 7
   4.2 The knowledge base .......................................................... 8
   4.3 Two main groups of knowledge ......................................... 10
   4.4 Background knowledge ..................................................... 10
   4.5 Experiences ................................................................ 11

5 Strategies in fault finding ............................................................. 13
   5.1 Examples of how to find faults in a Sesam terminal ............ 13
      5.1.1 Fault finding depending on competence ..................... 13
      5.1.2 Fault finding depending on different assumptions ...... 14
   5.2 Fault finding using shallow knowledge - a model ............. 14
      5.2.1 An example of a simulated repair .............................. 14
      5.2.2 Background of the mechanism ................................. 16
      5.2.3 Algorithm .............................................................. 16
   5.3 A combined mechanism .................................................... 16
   5.4 Fault finding using deep knowledge ................................. 17
      5.4.1 An example ............................................................ 17
      5.4.2 Algorithm .............................................................. 17
   5.5 Experiences ................................................................ 18

6 Learning processes in fault finding ............................................. 19
   6.1 How to learn fault finding ............................................... 19
      6.1.1 Examples of how a novice could learn fault finding . . 19
      6.1.2 Two phases in the learning process ......................... 19
   6.2 Models of learning mechanisms ......................................... 21
      6.2.1 Learning by questioning ......................................... 21
      6.2.2 Learning from deep knowledge ............................... 23
      6.2.3 Learning by "watching" the expert working ............... 24
      6.2.4 Learning by trial and error .................................... 25
   6.3 Experiences ................................................................ 26

7 Using a learning system for fault finding on real cases ............. 28
   7.1 Purpose and plan ............................................................ 28
   7.2 Adjusting the knowledge representation and mechanisms .... 28
   7.3 Expert interface ............................................................... 28
   7.4 Integrating the techniques ................................................. 28

8 A system that learns how to learn ............................................. 29
8.1 The purpose and plan ......................................................... 29
8.2 The background of this project idea ..................................... 29
8.3 How new types of knowledge could be invented .................... 30
8.4 An investigation of how the different mechanisms for learning and fault finding could have been invented .......................... 30
8.5 Learning from a small set of knowledge ............................... 31
  8.5.1 The simulator ............................................................. 31
  8.5.2 What the simulated novice knows .................................. 31
  8.5.3 A plausible scenario of how the simulated novice could learn ................................................................. 32
  8.5.4 Mechanisms .............................................................. 33

9 Results ................................................................................. 34
  9.1 What has been done ........................................................ 34
  9.2 Experiences and conclusions ............................................. 34
  9.3 Future work .................................................................... 35

10 References ........................................................................... 36

A1 Appendix 1 - Knowledge associated with the fault finding of a Sesam terminal
  A1.1 A Sesam terminal ............................................................ 37
    A1.1.1 Physical structure ..................................................... 37
    A1.1.2 Functional structure and the relation between the functional and the physical structure ........................................... 37
    A1.1.3 Primitive actions and possible results ......................... 38
    A1.1.4 Faults ................................................................... 38
    A1.1.5 Tests ..................................................................... 39
    A1.1.6 Actionlists .............................................................. 41
    A1.1.7 An abstract description of the knowledge and the problems ................................................................. 42
  A1.2 A generalized description of knowledge ............................ 42
    A1.2.1 Physical structure ..................................................... 42
    A1.2.2 Relations between the physical and the functional structure ................................................................. 42
    A1.2.3 Faults ................................................................... 42
    A1.2.4 Functional structure .................................................. 42
    A1.2.5 The primitive actions and the results ......................... 42
    A1.2.6 Tests and actions ..................................................... 42
    A1.2.7 Fault finding ............................................................ 43
    A1.2.8 Case knowledge ....................................................... 43
    A1.2.9 Commonsense knowledge ......................................... 43
    A1.2.10 Relations between the physical knowledge and other types of knowledge .................................................. 43
    A1.2.11 Relations between functional knowledge and other types of knowledge .................................................. 43
    A1.2.12 Relations between test knowledge and other types of knowledge .................................................. 44
    A1.2.13 More examples of knowledge .................................... 44

A2 Appendix 2 - Vocabulary for the characterization of knowledge ......................... 45

A3 Appendix 3 - Actionlists a background ................................................. 47
1 The purpose and the plan

The main purpose with this project is to explore how "learning" can be used in fault finding.

The plan was to select one object (a Sesam terminal) as an example and carry through the following activities:

Find knowledge associated with the troubleshooting of a Sesam terminal. Study different types of knowledge as: heuristic rules, the functional and the physical description of a Sesam terminal, how the different types of knowledge are related to each other. Discuss criteria for deciding what types of knowledge should be saved in the knowledgebase and what should be derivable. Build a knowledgebase containing different types of knowledge.

Find different ways to find faults in a Sesam terminal using different types of knowledge. Make models of some of the different ways of finding faults.

Find different ways to learn how to troubleshoot a Sesam terminal. Make models of the different ways of learning.

While studying and constructing the models, the following questions were discussed: What would happen if the complete Sesam system instead of just one component (a Sesam terminal) was selected? How general was the model? Could it be extended to a general mechanism? What were the problems? How could the different techniques be integrated?
2 The background

The background of this project consists of experiences from the construction of a fault finding system for a computer called Sesam, studies of different learning techniques and studies of different systems for fault finding.

![Diagram](image)

Figure 1 Sesam

2.1 Sesam

Sesam is a system that helps telephone operators in about 150 companies in Sweden. It is an appendage to a company's switchboard consisting of a control unit and one terminal per telephone operator (figure 1). Sesam is called a referral computer because it will refer a call to an extension whose subscriber can't presently be reached via telephone to the operator. A referred call is displayed as a message on the telephone operator's terminal.

The Sesam terminal was selected as the object that we made a detailed model of. In this model was also the control unit used but it was just represented as one box. The control unit is also used in the discussion about the generality of the models of fault finding and learning.
2.2 The Sesam Fault Finder project

The Sesam Fault Finder instructs a service technician how to find and repair a fault in a Sesam computer. The system runs on a portable PC.

The construction of the Sesam fault finder (Sesam FF) was initiated by the Swedish Telecom. The project team consisted of one knowledge engineer and the constructors of the Sesam computer.

Many of the ideas of knowledge representation, fault finding and learning mechanisms that will be described in this paper have emerged from the Sesam FF project.

2.3 The ideas and their origin

Combining learning and fault finding

Building an expert system for fault finding was a time-consuming process. Developing a learning system for fault finding could be one way to change this.

The idea of using learning in fault finding emerged from the paper "Learning in second generation expert systems" [8]. This paper describes a system that uses an ordinary car as a source for a concrete example of a learning system for fault finding. The system learns heuristic rules from the deep model consisting of a causal network. Another example of a learning system for fault finding called CRIB is described in the book "Expert Systems Technology" [2].

The architecture

To be able to test and experiment with different learning techniques a simulated "world", consisting of a Sesam terminal, an expert and a novice was constructed. I choose a simple object because the problem of making a learning system is difficult enough without complicating it with a complicated example.

Using experiences from the Sesam FF project to build a learning system

In the paper "Acquisition of Proof Skills in Geometry" [1] a model of how students learn to solve geometry problems is presented. This paper gave me the idea of using my own experiences of how I learned how to troubleshoot a Sesam terminal to develop a learning system.

Deciding what types of knowledge to put into the knowledgebase and what methods for fault finding to use

To build a learning system for fault finding I had to know what should be learned and what fault finding mechanisms should be used. In the Sesam FF project I had studied how technicians found faults in Sesam and also learned how to repair Sesam myself. I also studied different papers about fault finding as "Expert Systems Technology" [2] and "A theory of diagnosis from first principles" [6].
Building a list of knowledge associated with the fault finding of a
Sesam terminal

Learning can be viewed as a process where pieces of knowledge continuously are
changing. Acquiring a list of these pieces of changing knowledge generate the steps of
the learning process.

I started to build a list of different types of knowledge associated with the fault finding
of a Sesam terminal. Some types of knowledge from this list were selected and put into
a knowledge base. Fault finding mechanisms were constructed.

Deciding what learning techniques to use

I learned fault finding of Sesam in different ways. I was given instructions how to do, I
experimented with the terminal and so on. The ideas of what learning techniques to use
emerged from these experiences.

There exist other projects with the idea of using many learning techniques. Disciple [7]
is an example of a system that integrates different learning techniques.

2.4 How this project is different

The ideas of this project are similar to the ideas in the papers presented above. But I have
not found any project based on the following ideas:

- trying to make a list of all knowledge in one domain
- using an architecture consisting of a simulated expert, a simulated novice and a
  simulated object
- using learning by questioning in the field of fault finding
Figure 2 A learning system
3 The system: an overview

In this chapter an overview of the system developed is going to be presented. The background of this system is going to be presented in chapter 4, 5 and 6. A picture of the system is showed in figure 2.

3.1 Types of knowledge

The system uses two main types of knowledge: shallow and deep knowledge (Appendix 2). There also exists knowledge produced temporary. A trace is an example of a type of knowledge produced temporary.

3.2 Main components

The system consists of three main components: A simulated Sesam terminal, a simulated expert troubleshooter of a Sesam terminal and a simulated novice.

The Sesam terminal is simulated by a forward chaining mechanism interpreting the deep knowledge. The simulated expert troubleshooter produces a trace while troubleshooting the Sesam terminal and is able to answer questions from the simulated novice. The simulated novice is supposed to learn how to troubleshoot the Sesam terminal in different ways.

3.3 Ways to find a fault

Two methods for the simulated expert to find a fault in a Sesam terminal have been implemented. The two methods use the following different types of knowledge: shallow knowledge and deep knowledge.

One method for the simulated novice to find a fault in a Sesam terminal has been implemented. This method is a combination of two methods one working from rules and the other working from knowledge about the physical structure.

3.4 Ways to learn

Four ways for the simulated novice to learn how to find a fault in the Sesam terminal have been implemented.

Learning by questioning works in an interactive way. The knowledge used: The answers of the expert. Knowledge produced: Actionlists, knowledge about the physical structure tests and knowledge about actions (eg how to repair parts).

Learning from deep knowledge uses the following types of knowledge: Knowledge about the physical structure, the functional structure and knowledge about faults. Knowledge produced: Actionlists

Learning by "watching the expert working" uses the following type of knowledge: Traces of the expert. Knowledge produced: Actionlists

Learning by "trial and error" integrates problem solving into the learning process and uses the following types of knowledge: Knowledge about tests, the physical structure. Knowledge produced: Traces giving examples, examples giving rules.
4 Knowledge structures in fault finding

![Diagram of a Sesam terminal](image)

**Figure 3** The physical description of a Sesam terminal

The first activity in the plan (chapter 1) was to find knowledge associated with the troubleshooting of a Sesam terminal. An overview of this knowledge is going to be presented in this chapter. A knowledge base consisting of parts of this knowledge is also going to be presented. General knowledge that forms the background of the fault finding and learning mechanisms is going to be described.

A vocabulary used to discuss about knowledge has been developed (Appendix 2).

4.1 An overview of knowledge associated with fault finding of a Sesam terminal

A more detailed description of the knowledge associated with the fault finding of a Sesam terminal is presented in Appendix 1. Here is an overview of the knowledge:

- physical structure (figure 3)
- knowledge about faults
- functional structure (figure 4)
- primitive actions and results
- knowledge about tests
- troubleshooting knowledge
- common sense knowledge
- relations between different types of knowledge

Case knowledge is created during a consultation.

4.2 The knowledge base

A knowledgebase was constructed from parts of the knowledge found in the investigation of knowledge described in 4.1. The following types of knowledge from the knowledgebase are going to be presented:

- actionlists
- rules
- knowledge about tests
- knowledge about the physical structure
- knowledge about the functional structure

Actionlists

An actionlist consists of: a test with a result indicating a fault and an ordered list of actions and subtests. Here is an example together with an explanation:

actionlist([quicktest,screen_black], [[reset_screen_test,screen_black], [repair,control-unit]]).

Explanation: If quicktest gives as a result that the screen is black then try to make quicktest correct by one of the following actions:

- making the reset_screen_test correct
- repairing the control unit

The actionlists are used by the routine that finds fault from shallow knowledge. They are produced by the following routines: learning by questioning, learning by watching and learning from deep knowledge.

The background of the actionlists is described in Appendix 3. It is explained how the actionlists have been constructed from rules in order to make the knowledge more compact. It is also showed how the actionlists could be transformed to rules and thus be viewed as containing the same knowledge as a set of rules.
*1 The reset function connects the keyboard function and the screen function

Figure 4 The functional description of a Sesam terminal

Rules

Rules are used and produced by the routine "learning by trial and error". A rule consists of some tests and a faulty part as conclusion. The following is an example of a rule:

rule[[[quicktest, seekformula_bad]], [repair, cable]].
Knowledge about tests

The knowledge about a test consists of:

- a description of the test
- possible results of the test
- a list of connections used by the test

A list of connections used by the test is needed because different tests need different configurations.

Physical structure

The knowledge about the physical structure (figure 3) consists of:

- a list of the different parts
- knowledge about how the parts are connected

Functional structure

The functional structure (figure 4) is represented in form of rules. These rules are interpreted by a forward chaining mechanism to simulate the Sesam terminal. The functional knowledge is also used by the routine that finds fault from deep knowledge.

4.3 Two main groups of knowledge

The knowledgebase consists of the following two main groups of knowledge: deep knowledge and shallow knowledge. (description Appendix 2).

It is possible to partly derive the shallow knowledge from the deep knowledge. It is only possible to derive the shallow knowledge partly, because the deep knowledge does not contain any knowledge about how difficult/easy it is to make the different actions or how often a certain fault occurs.

4.4 Background knowledge

Here is a list consisting of suitable assumptions in the domain of fault finding a Sesam terminal. The learning and fault finding mechanisms that will be presented later are appropriate for the class of fault finding problems that this list characterizes.

1. The object consists of parts.
2. Some of the parts are connected.
3. Every part is linked to an action: repair
4. Repairing a faulty part makes the part function properly.
5. Every connection is linked to three actions: disconnect, connect and check
6. Checking or connecting a connection makes the connection correct.
7. The whole object is functioning properly if all connections and all parts belonging to the object are functioning properly.
8. A fault can not cause another fault. Faults does not emerge while fault finding.
9. There exist tests and every test has one result indicating that the subsystem is correct and one or more results indicating a fault in the subsystem.
10. Every test is linked to a question.
11. Some of the tests need different configurations.
12. One of the tests is the main test. The main test is used to test whether the object is functioning properly.
13. Every test with a value indicating a fault has a set of possible causes. The relation between, two sets of causes of results indicating faults, belonging to two different tests is one of the following:
   A. The sets are disjoint.
   B. One of the sets is a subset of the other set.
14. The tests can be ordered according to simplicity. For every test with a result indicating a fault, there exist a list. This list consists of actions and subtests. The subtests are all simpler than the test. If all subtests in the list are ok and all actions are performed then the test will no longer give a result indicating a fault.

Observe that point 14 could be derived from the other points in the list.

4.5 Experiences

The relations between different types of knowledge

The knowledge found are related to each other in different ways. They could be partly or completely derivable from each other. An example of the relation between two types of knowledge is illustrated in figure 5.

Deciding what to save in a knowledge base

The first thing that was noticed was that it existed a very lot of different types of knowledge that could be associated with the fault finding of a Sesam terminal. It would not be suitable to save all in the knowledgebase. How to decide what types of knowledge to put into a knowledgebase? An example of a question that has risen is: Should one allow redundancies in the knowledgebase? If two representations are derivable from each other should one save both or just one? How should one do with concepts that are similar but not exactly the same? Eg: connected to versus connectable. A framework to decide the right combination of pieces of knowledge to put into the knowledgebase is probably needed.

Scaling up

When scaling up to the whole of Sesam may there exist different configurations. There may for example be one or two disk drives. It is important to be able to use standard models of Sesam and only describe the exceptions from the standard model. New types of knowledge will also be needed. There is for example a need to describe preconditions of an action.
Figure 5  Relations between the physical and the functional structure in a Sesam terminal

*1 The reset function connects the keyboard function and the screen function
5 Strategies in fault finding

5.1 Examples of how to find faults in a Sesam terminal

The way to find a fault in a Sesam terminal could depend of the following factors:

- the competence of the technician
- different assumptions

5.1.1 Fault finding depending on competence

Here are some of the ways that a technician could use, when troubleshooting a Sesam terminal, depending on his competence:

Fault finding from shallow knowledge

A technician experienced with a Sesam terminal could work from specialized rules. Behind these rules is both the insight of how a Sesam terminal functions and knowledge about what is a suitable ordering of the actions/tests. An example of such a rule could be:

If the reset screen test gives as a result that the screen is black then try with the following actions:

- check if there is electricity
- check the connections between the printing unit and the cable
- check the connections between the display unit and the cable
- exchange the display unit
- exchange the printing unit
- exchange the cable

Fault finding from deep knowledge

A technician with knowledge of how a Sesam terminal functions uses his knowledge about the relations between the tests to find the fault. The following is an example of how he might reason:

If the quicktest gives as a result that the screen is black he knows that the quicktest uses a screen function and that this function could be controlled by doing a reset. He resets and if this test gives a black screen he knows that reset needs electricity and uses the printing unit, the cable, the display unit and the connections. He starts by checking the electricity and the connections because they are easy to check. If he does not manage to make the reset screen test ok this way he continues by successively exchanging the cable, the display unit and the printing unit until the reset is ok. He does not know which parts are most likely to be faulty and the exchanges are made in a random order.

Fault finding using trial and error

A technician with no knowledge of how a Sesam terminal works could try to make the terminal correct by checking the connections and successively exchanging all the parts until the main test indicates that the terminal works..
5.1.2 Fault finding depending on different assumptions

When one troubleshoots an object one could use different assumptions such as:

- there exists just one fault
- one fault cannot cause another fault
- the things one uses to exchange the different parts are correct

If one does not manage to find the cause of some symptom one drops some of the assumptions. Different assumptions are more or less suitable for different kind of objects. When the domain is expanded to the complete Sesam the following assumption is less suitable: one fault cannot cause another fault

Under different assumptions could it be suitable to use different types of knowledge for fault finding. The following is an example: If one has the assumption that there exists just one fault then one could make a trace of the experts actions for every possible fault. All these traces could be combined into a decision tree. The resulting tree would for a Sesam terminal consist of a reasonable number of possible paths. If one leaves the one fault assumption but uses the same type of knowledge there will be a combinatorial explosion.

5.2 Fault finding using shallow knowledge - a model

A model of how a technician could find a fault in a Sesam terminal using shallow knowledge has been developed (Chapter 3). This model (example Chapter 5.2.1) is implemented in the form of production-rules with a forward-chaining mechanism. The inference mechanism together with a small set of rules forms a mechanism for fault finding. This mechanism works on actionlists and knowledge about tests.

The mechanism above works by successively either finding new tests to locate the faults or by repairing/exchanging parts. There may be more faults found while working in this manner. While the system solves the problem it also produces a trace of its activities. The traces are used by the routine "learn by watching".

5.2.1 An example of a simulated repair

TEST: What symptom?
RES: terminal_doesnt_work

TEST: Push the buttons "name off name" What happens on the screen?
RES: screen_black

TEST: Push the reset-button. What happens on the screen?
RES: frikab_text_ok

TEST: Push the buttons "name off name" What happens on the screen?
RES: screen_black

TEST: Push the reset-button. When the FRIKAB-text is shown on the screen then push some buttons on the keyboard. What happens on the screen?
RES: nothing_happens

ACT: Check the connection between the printing-unit and the keyboard

TEST: Push the reset-button. When the
FRIKAB-text is shown on the screen then push some buttons on the keyboard. What happens on the screen?
RES: nothing happens

ACT: Exchange the keyboard

TEST: Push the reset-button. When the
FRIKAB-text is shown on the screen then push some buttons on the keyboard. What happens on the screen?
RES: nothing happens

ACT: Exchange the printing-unit

TEST: Push the reset-button. When the
FRIKAB-text is shown on the screen then push some buttons on the keyboard. What happens on the screen?
RES: letters shown

TEST: Push the buttons "name off name". What happens on the screen?
RES: seekformula_ok

TEST: What symptom?
RES: ok

generated trace:

[[symptom, terminal_doesnt_work], [quicktest, screen_black], [reset_screen_test, frikab_text_ok], [quicktest, screen_black], [reset_keyb_test, nothing_happens], [[checkconn, [printing_unit, keyboard]], yes], [reset_keyb_test, nothing_happens], [[repair, keyboard], yes], [reset_keyb_test, nothing_happens], [[repair, printing_unit], yes], [reset_keyb_test, lettersShown], [quicktest, seekformula_ok], [symptom, ok]]

The development of the problemstack

1. [symptom, terminal_doesnt_work]
2. [quicktest, screen_black], [symptom, terminal_doesnt_work]
3. [reset_screen_test, frikab_text_ok], [quicktest, screen_black], [symptom, terminal_doesnt_work]
4. [reset_keyb_test, nothing_happens], [quicktest, screen_black], [symptom, terminal_doesnt_work]
5. [[checkconn, [printing_unit, keyboard]], yes], [reset_keyb_test, nothing_happens], [quicktest, screen_black], [symptom, terminal_doesnt_work]
6. [[repair, keyboard], yes], [reset_keyb_test, nothing_happens], [quicktest, screen_black], [symptom, terminal_doesnt_work]
7. [repair, printing_unit], yes], [reset_keyb_test, nothing_happens], [quicktest, screen_black], [symptom, terminal_doesnt_work]
8. [reset_keyb_test, nothing_happens], [quicktest, screen_black], [symptom, terminal_doesnt_work]
9. [quicktest, screen_black], [symptom, terminal_doesnt_work]
10. [symptom, terminal_doesnt_work]
11. []
5.2.2 Background of the mechanism

The background of the mechanism consists of the following points from the background knowledge described in Chapter 4.3.

12. One of the tests is the main test. The main test is used to test whether the object is functioning properly.

14. The tests can be ordered according to simplicity. For every test with a result indicating a fault there exist a list. This list consists of actions and subtests. The subtests are all simpler than the test. If all subtests in the list are ok and all actions are performed then the test will no longer give a result indicating a fault.

The following is an abstract description of the problem: Help a technician to solve a problem in as few and easy steps as possible. (a step = an actions or a test)

5.2.3 Algorithm

An algorithmic description of the mechanism (example Chapter 5.2.1) follows:

Put a symptom on top of the problemstack and continue with the following activities until the problemstack is empty:

1. If the element on the top of the problemstack is a test then ask of the result of the test and proceed as follows:

   A. If the result of the test indicates correctness then take away the test from the problemstack.

   B. If the result of the test indicates a fault then use the actionlist for the test to find a subtest or an action that has not been tried. Put this action/subtest on top of the problemstack.

2. If the first element in the problemstack is an action then make this action and take away this action from the problemstack.

5.3 A combined mechanism

Two fault finding mechanisms are combined in the following way: Start with the first mechanism. If the first mechanism finds the fault then finish. Otherwise try with the second mechanism.

The first mechanism uses rules to find the faults. The second mechanism checks all connections and repair all parts until the main test is correct. A trace is produced.

The background to this combined mechanism consists of the general background knowledge described in chapter 4.3 with the exception that there are no conditions on the tests (point 13 and 14 in the description in chapter 4.3).

This combined mechanism is used in the learning by "trial and error" routine.

16
5.4 Fault finding using deep knowledge

A model of how faults could be found using the theory of Reymond Reiter [6] has been constructed (chapter 3). This method (example Chapter 5.4.1) uses a description of the system and some observations. The algorithm gives a list of possible faults as a result. This list consists of lists that are minimal subset of faulty components that explain the observations.

This method is very general but maybe time consuming when tried on examples that are more complicated than the one presented in Chapter 5.4.1.

5.4.1 An example

The following facts are given:

- quicktest gives the result screen black
- reset-screen-test gives the result ok
- reset-keyb-test gives the result ok

The system gives the following possible faults:

- the control unit function, the receive function or the send function

5.4.2 Algorithm

1. Make a node that consists of:

- assumed faulty components = empty
- rest-components = consists of all components
- type = expandable

2. Select nodes with the type "expandable" until there are no more such nodes. For every such node proceed as follows:

   A. Check if the assumed faulty components are consistent with the observation. Use the functional description to make this check.

   B. If the assumed faulty components are consistent with the observations then mark this node with ok. Take away surplus nodes which means: Take away nodes that are related to this node in the following way: Their assumed faulty components have the assumed faulty components of this node (= the node marked ok) as a subset.

   C. If the assumed faulty components are not consistent with the observations expand this node by doing the following: Move one component from rest-components to assumed faulty components in every possible way. Make a new node in every such case if the generated node is not covered by any existing node. A node is covered by another node:

      - if the assumed faulty components of the generated node are the same as the assumed faulty components of an existing node
      - if the assumed faulty components of the generated node have the assumed faulty components of a node that is marked with ok, as a subset
3. Collect all nodes that have been marked with ok and present their assumed faulty components as the result.

5.5 Experiences

The way a technician find faults could depend on his competence and the actual assumptions. This gives as a result that different types of fault finding problems seem to need different fault finding mechanisms working from different types of knowledge.

Methods using deep knowledge have the following advantages:

- it gives the possibility to find complicated faults without specifying how to find the fault
- it is flexible (it is easier to change the functional and physical description than to change specific knowledge)

Methods using deep knowledge could however be limited. One example of this is that it is difficult to give the knowledge behind the choice of a certain order of actions. It is easier to give the order directly.

One way to use the advantages of both deep and shallow knowledge could be to build a mechanism that uses both types of knowledge. The shallow knowledge could for example be used as a model that is transformed with the help of deep knowledge.
6 Learning processes in fault finding

6.1 How to learn fault finding

6.1.1 Examples of how a novice could learn fault finding

How one learns fault finding depends on the knowledge source type that exists. Here are some different sources of knowledge:

- an expert answering questions
- an expert solving problems
- description of a Sesam terminal
- a Sesam terminal

An expert answering questions - learning by questioning

The novice could interview an expert. He could ask questions like: What tests are applicable? What should one do if some test do not work? In this way he could directly acquire the knowledge needed for fault finding.

An expert solving problems - learning by watching

The novice could also watch when the expert working and carefully record how the expert finds faults in a Sesam terminal.

Descriptions - learning from deep knowledge

The novice could use a description of how a Sesam terminal works and is constructed to acquire knowledge about the physical and functional structure.

A Sesam terminal - learning by trial and error

He could try different tests before he repairs the terminal. The next time will he be able to recognize the same fault by it's symptoms. He collects cases consisting of different tests and a fault. The fault finding knowledge could be learned from this knowledge.

Combination

He could also combine the methods above in different ways.

6.1.2 Two phases in the learning process

The following two phases in the learning process were found:

- the collection of some initial knowledge
- the transformation of knowledge into other types of knowledge
The result from the work with collecting knowledge could be:

- rules that helps him to find faults
- traces of how an expert finds a fault
- a functional and physical description
- examples consisting of tests and the faulty parts

If the novice acquires rules that helps him to find faults the process is finished. Otherwise he could transform his knowledge to the fault finding knowledge.

6.2 Models of learning mechanisms

Four models of learning will be presented in this Chapter (see figure 6). The first model "learning by questioning" illustrates how to directly acquire the knowledge needed. Two of the models "learning from deep knowledge" and "learning by watching" use deduction to transform the input knowledge into actionlists. One of the models "learning by trial and error" uses induction to transform the input knowledge into rules.

6.2.1 Learning by questioning

A model of how to learn using questioning will be presented. It works in an interactive way. The knowledge used: The answers of the expert. Knowledge produced: Physical structure, actionlist, knowledge about tests and knowledge about actions (e.g., how to repair parts).

The method implemented is an effective method when the structure of the knowledge is known and there exist an expert that is able to answer the questions.

An example from the developed system (chapter 3)

Here is a conversation between the simulated expert and the simulated novice.

Nov: What is the name of the object
Exp: SESAM
Nov: What are the parts of the object
Exp: [printing_unit, power_supply, cable, test plug, display unit, keyboard, control_unit]
Nov: What are the parts that normally are connected to: printing_unit
Exp: [cable, power_supply, control_unit, keyboard]
Nov: What are the connections between: printing_unit cable
Exp: [conn1, conn1]
Nov: What are the connections between: printing_unit power_supply
......

<Tests>

Nov: What are the tests.
Exp: [symptom, quicktest, reset_screen_test, reset_keyb_test, plug_test]
Test: symptom

Nov: What is the question to: symptom
Exp: What symptom?
Nov: What are the possible results of: symptom
Exp: [ok, terminal_doesnt_work]
Nov: What are the connections that are used by the test: symptom
Exp: [[printing_unit, cable], [printing_unit, power_supply], [printing_unit, control_unit], [printing_unit, keyboard], [cable, display unit]]

<Test: quicktest>

Nov: What is the question to: quicktest
Exp: Push the buttons "name off name". What happens on the screen?
Nov: What are the possible results of: quicktest
   Exp: [seekformula_ok, seekformula_bad, screen_black]

Nov: What are the connections that are used by the test: quicktest
   Exp: [printing_unit, cable], [printing_unit, power_supply], [printing_unit, control_unit], [printing_unit, keyboard], [cable, display unit]

......

Nov: What are the actions/tests that are used to correct: [symptom, terminal doesn't work]
   Exp: [quicktest, screen_black]

Nov: What are the actions/tests that are used to correct: [quicktest, seekformula_bad]
   Exp: [reset_screen_test, frikab_text_bad]

Nov: What are the actions/tests that are used to correct: [quicktest, screen_black]
   Exp: [reset_screen_test, screen_black], [reset_keyb_test, nothing_happens], [plug_test, nothing_happens], [repair, control_unit]

Nov: What are the actions/tests that are used to correct: [reset_screen_test, frikab_text_bad]
   Exp: [checkconn, [printing_unit, cable]], [checkconn, [cable, display unit]], [repair, display unit], [repair, cable]

......

Nov: What is the question to: [repair, printing_unit]
   Exp: Exchange the printing-unit

Nov: What is the question to: [repair, power_supply]
   Exp: Repair power_supply.

......

Knowledge behind the mechanism

The following knowledge from the background knowledge described in 4.3 constitutes the background knowledge of the mechanism:

1. The object consists of parts.
2. Some of the parts are connected.
3. Every part is linked to an action: repair
4. Every connection is linked to three actions: disconnect, connect, and check
5. There exist tests and every test has one result indicating that the subsystem is correct and one or more results indicating a fault in the subsystem.
6. Every test is linked to a question.
7. The tests can be ordered according to simplicity. For every test with a result indicating a fault, there exist a list. This list consists of actions and subtests. The subtests are all simpler than the test. If all subtests in the list are ok and all actions are performed then the test will no longer give a result indicating a fault.

The mechanism

1. The system asks about the physical structure:
   - What parts the object consist of.
   - How these parts are connected.

2. The system asks about tests:
   - What tests exist.
3. For every test the system asks about the following:
   - A description of how the test is made.
   - A list of possible results of the test.
   - A list of connections that the test uses. (the configuration may vary depending on different tests eg: plug test)

4. For each pair consisting of a test and a result of the test that shows a fault in the object the system asks for a list of subtests/actions.

5. The system asks for a description of how to repair every part in the object.

6. The system asks for a description of how to connect disconnect and check every connection in the object.

The order of the questions is important. The knowledge acquired by one question is used to form the next question(s).

6.2.2 Learning from deep knowledge

A model of how to learn using deep knowledge will now be presented. In this routine actionlists are produced from knowledge about the physical structure, the faults and the functional structure.

An example from the developed system (chapter 3)

The simulated novice simulates every fault and tries every test

```
Fault [power_supply,OBJ,broken]
   quicktest screen_black
   reset_screen_test screen_black
   reset_keyb_test nothing_happens
   plug_test nothing_happens
Fault [printing_unit,keyb_func,broken]
   quicktest screen_black
   reset_screen_test ok
   reset_keyb_test nothing_happens
   plug_test nothing_happens
.....
```

Possible causes of a failed test are collected

```
TEST [quicktest,screen_black]
FAULTS [[conn,[cable,display
unit]],[conn,[printing_unit,keyboard]],[conn,[printing_unit,control_unit]],[conn,[printing_unit,cable]],
[control_unit,control_unit_func,broken],[keyboard,keyb_func,broken],[display_unit,screen_func,broken],
[cable,screen_func,broken],[printing_unit,screen_func,broken],[printing_unit,receive_func,broken],[printing
_unit,send_func,broken],[printing_unit,keyb_func,broken],[power_supply,-0,broken]]

TEST [quicktest,seekformula_bad]
FAULTS [[display_unit,screen_func,weak],[cable,screen_func,weak],[printing_unit,screen_func,weak]]

TEST [plug_test,nothing_happens]
FAULTS [[conn,[cable,display_unit]], [conn,[printing_unit,keyboard]], [conn,[printing_unit,cable]],
[keyboard,keyb_func,broken],[display_unit,screen_func,broken],[cable,screen_func,broken],
```
Action lists are produced from the knowledge above

**Mechanism**

This module uses a description of how a Sesam terminal works.

1. Every possible fault is simulated. For every possible fault every test is tried. For every pair consisting of a test and faulty value all possible causes (= faults) are collected into a list.

2. Put the main test in a que of tests and proceed as follows until the que of tests is empty: Chose a test from the que and for every pair consisting of a test and a result indicating a fault make the following:

   A. Use the lists produced in 1 and make the following: Divide the test and the results into subtests as far as possible. Put subtests in the que.
   B. Make actions of the causes of the test that are not covered by any subtest.
   C. Make an action list consisting of the test and the result as first argument and a list consisting of the subtests produced in A and the actions produced in B as second argument.

**Comments**

The deep knowledge in the developed system does not contain all knowledge that is used in fault finding. For example there is not any knowledge about the probability of certain faults. The order of the action/tests in the generated action lists is therefore not always optimal.

When we compared the knowledge produced from this module with the knowledge produced from the routine "learning by questioning" differences were found. When the differences were investigated there were faults found in both the functional description and in the action lists. This shows that the use of techniques using different input knowledge to acquire the same output knowledge could be a way of cross-checking the knowledge.

**6.2.3 Learning by "watching" the expert working**

A model of how to learn watching the expert working will now be presented. Traces from the simulated expert are converted to action lists. The system could be viewed as the inverse of the fault finding mechanism (Chapter 5.2).

The background knowledge of this mechanism consists of: the general background knowledge described in Chapter 4.3 and the assumption that the simulated expert finds fault using the fault finding mechanism described in Chapter 5.2.

Traces is a very general form of knowledge and may be used in different ways. The implemented mechanism could be viewed as one way to use this knowledge.

**An example from the developed system (Chapter 3)**

Traces from the simulated expert are used as input
Action lists are produced

\[
\text{actionlist([[\text{quicktest,screen blackout}}, [\text{reset screen test, flex}], [\text{reset keyb test, nothing happens}]]), actionlist([\text{reset keyb test, nothing happens}], [[\text{check conn, printing unit, keyboard}]], [\text{repair, keyboard}], [\text{repair, printing unit}])}. 
\]

Algorithm

Proceed as follows until the trace is empty:

1. If the first element of the trace is a test with a result indicating a fault then:
   
   A. Make an action list for this test using the trace.
   
   B. Take away this test from the trace and make action lists of the rest of the trace.

2. If the first element of the trace is not a test with a result indicating a fault then take away the first element of the trace and make action lists of the rest of the trace.

The action list produced from the trace is combined with the action lists produced from other traces.

6.2.4 Learning by trial and error

A model of how to learn using "trial and error" will now be presented. The following knowledge is used: possible faults, knowledge about the physical structure, list of tests and descriptions of tests. The following knowledge is produced using induction: traces gives examples and examples gives rules. Learning from examples (trial and error) could be used when it is not possible to learn fault finding in another way.

The general background knowledge described in chapter 4.3 with the exception that there are no conditions on the tests (point 13 and 14 in the description in chapter 4.3) constitutes the background of the mechanism.

An example from the developed system (chapter 3)

<From the beginning there exists only one rule>

rule [[\text{quicktest, seek formula ok}}], [\text{all, ok}]]

Fault [\text{printing unit, keyb_func, broken}]

rule [[\text{quicktest, screen black}]], [\text{repair, printing unit}]
rule [[\text{quicktest, seek formula ok}}], [\text{all, ok}]]

Fault [\text{printing unit, keyb_func, broken}] <- The system manages this fault with the existing rules
Fault [\text{cable, screen_func, weak}]

25
Mechanisms

Two different mechanisms for fault finding were used:

- fault finding with help of rules
- fault finding with help of physical knowledge and knowledge about tests

How they are combined is described in Chapter 5 and in the algorithm below (2-4).

Algorithm

The system works in the following cycle:

1. Generate a fault.
2. If the system solves the problem with the existing rules go back to 1.
3. Try all tests.
4. Try all possible actions until the main-test is ok.
5. Generate an example from the trace and add this example to the collected examples.
6. Induce new rules from the collected examples.
7. Go back to 1.

Algorithm for the production of rules from examples

AQ15 [5] is used and works in the following way:

As long as there exist an example not covered by the rules proceed as follows.

1. Chose an example that is not covered by any rule.
2. Decide the most general rules that cover this example and none of the negative examples.
3. Chose one of these rules and add this rule to the collection of rules.

6.3 Experiences

All different ways of learning have situations when they are appropriate.

Two phases have been identified in the learning process: The collecting of knowledge and the transforming of knowledge to other types of knowledge.

The "learning by watching" and "learning by questioning" routines use the simulated expert using shallow knowledge. The "learning from deep knowledge" and "learning by trial and error" routines use the deep knowledge and the simulator. This gives as a result a difference in what is learned. The methods that uses the deep knowledge directly or indirect are not able to learn the correct order between the different actions.

Cross-checking different sources of knowledge using different ways of learning the "same" knowledge has proved to be a way of finding faults in the knowledge.
More learning mechanisms could have been produced by combining the input knowledge of one mechanism with the method of another mechanism. One example of this could be to use the traces of the expert and the inductive method from "learning from examples".

The experiences from this project indicates that it is possible to build a learning system when the structure of the knowledge is known.

The learning mechanisms works for a class of fault finding problems that have been described in Chapter 4.3.
7 Using a learning system for fault finding on real cases

7.1 Purpose and plan

The purpose with this project idea is to investigate the possibility of extending the developed prototype for use on real cases. The plan is:

1. Select a more complicated case (e.g., the complete Sesam) representing a new class of fault finding problems. Adjust the knowledge representation, the fault finding and learning mechanisms to this new class of fault finding problems.
2. Build an expert interface to make it easy to add knowledge.
3. Integrate the developed learning techniques in such a way that it will be possible to learn from different sources of knowledge in parallel. The knowledge may be inconsistent and incomplete.
4. Test the developed system on a new case belonging to the same class of fault finding problems as the case that were selected in point 1.

7.2 Adjusting the knowledge representation and mechanisms

When adjusting the knowledge representation to the complete Sesam new types of knowledge will be needed. Examples of new types of knowledge that will be needed are:

- preconditions of actions
- the outside inside relation

There will also be a need of different levels of abstraction. When specifying the physical and functional organization it should be possible to refer to some general structure specifying only the exceptions (e.g., using default hierarchies). The fault finding and learning mechanisms should be adjusted to this new representation.

7.3 Expert interface

An expert interface is needed. It should be possible to update and change different types of knowledge directly to be able to test the system on different cases.

7.4 Integrating the techniques

To be able to integrate the learning techniques the following has to be done: The developed mechanism must be able to handle incomplete knowledge. There will also be a need of a mechanism that checks the knowledge derived from different sources against each other and generates question when inconsistencies or incompleteness are found.

To be able to test the integration of the learning techniques the following has to be done: The simulated expert should give his information in a more realistic way. The information from him could be a mix of different types of knowledge and the knowledge he gives could be incomplete and faulty.
8 A system that learns how to learn

8.1 The purpose and plan

The purpose with this project idea is to investigate if it is possible to build a learning system that is able to invent new types of knowledge representations, learning and fault finding mechanisms for different classes of fault finding problems. This project could be viewed as a theoretical continuation of the work described in Chapter 1 - 6. The plan is:

1. Investigate how different types of knowledge associated with the fault finding of a Sesam terminal could be invented. Use the list of different types of knowledge and investigate how they are related. Does there exist a path from a small set of knowledge to more and more complicated knowledge?

2. Investigate how the mechanisms in the developed system (Chapter 3) could be generated (e.g. from the background knowledge).

3. Let the simulated novice experiment with the simulator of a Sesam terminal working only from a small set of knowledge. Investigate if it is possible for him to learn the knowledge types, background knowledge, fault finding and learning mechanisms. Only general mechanisms should be used. Test the mechanisms developed on another example.

8.2 The background of this project idea

The idea of searching for the mechanisms behind the developed system emerges from the AM and EURISCO project [3] and [4]: The following ideas are presented in the paper [4]:

1. New domains of knowledge can be developed using heuristics.
2. As new domains of knowledge emerge and evolve, new heuristics are needed.
3. New heuristics can be developed by using heuristics.
4. As new domain of knowledge emerge and evolve, new representations are needed.
5. New representations can be developed by using heuristics.

One question could be: Does there exist a set of heuristics from which all heuristics could emerge? Transforming this question to this project: Does it exist a set of general mechanisms from which knowledge associated with the fault finding of a Sesam terminal could be generated?

What could be used from work already done

The following could be used from the project described in Chapter 1-6:

- the list of knowledge
- background knowledge
- the mechanisms
- parts of the architecture
8.3 How new types of knowledge could be invented

The problem of finding new types of knowledge to use could be divided into the following questions:

How to generate new types of knowledge?
How to select types of knowledge to be saved?
How to represent the knowledge?

One way to find the answers of the first question above could be to use the list of different types of knowledge and investigate how they are related. Does there exist a path from a small set of knowledge to more and more complicated knowledge? An idea of how one type of knowledge could emerge is described in Appendix 3.

Here are some examples of concepts that could be used to develop criteria that could be used in the selection of new types of knowledge. They are taken from the vocabulary Appendix 2.

- narrow versus broad
- usefulness
- operationality for a certain problem

The words above could be used to formulate rules that could be used to decide which types of knowledge to put into the knowledgebase. One problem that has to be solved is: How to quantify the factors above?

8.4 An investigation of how the different mechanisms for learning and fault finding could have been invented

Investigate in what way the fault finding and learning mechanisms could be generated. One could for example investigate the following: In what way could the fault finding and learning mechanisms be generated from the background knowledge? One example of how one mechanism could have been generated from the background knowledge will be illustrated:

**Background knowledge (from Chapter 4.3)**

4. Repairing a faulty part makes the part function properly.
6. Checking or connecting a connection makes the connection correct.
7. The whole object is functioning properly if all connections and all parts belonging to the object are functioning properly.
12. One of the tests is the main test. The main test is used to test whether the object is functioning properly.

By using the background knowledge above one could prove that the second mechanism in the combined mechanism (Chapter 5.3) works. This mechanism checks all connections and repairs all parts until the main test is correct.
8.5 Learning from a small set of knowledge

How could the simulated novice learn the knowledge types, background knowledge, fault finding and learning mechanisms experimenting with the simulator of the Sesam terminal? Only a small set of knowledge and general mechanisms should be used. Some ideas of how to solve this problem is going to be presented in this Chapter.

8.5.1 The simulator

The simulator is used by the simulated novice and here are the actions the simulator recognizes:

- quicktest
- reset
- wait
- push some buttons
- conn OBJ1 OBJ2
- disc OBJ1 OBJ2
- measure OBJ1 POINT
- repair OBJ

The objects the simulator knows:

- printing unit
- display unit
- cable
- keyboard
- plug

8.5.2 What the simulated novice knows

The novice has the following capabilities:

- has a vocabulary
- can perform actions on his environment
- makes observations about his environment

The concepts which the simulated novice recognizes

- what objects are connected
- what happens on the screen
- the values of different measurements

Words that the simulated novice knows

- printing unit
- display unit
- cable
- keyboard
- plug
- quicktest
- reset
- wait
- push some buttons
- con WORD1 WORD2
- disc WORD1
- measure WORD1 WORD2
- repair WORD

8.5.3 A plausible scenario of how the simulated novice could learn

The first phase

In the first phase the simulated novice randomly makes actions like quicktest, wait, disc. Here is an example of a trace consisting of actions made in a random way:

ACTION: Quicktest
WATCH: Screen seekformula
ACTION: Disconnect cable and display unit
WATCH: The connection between the cable and the display unit disappears
ACTION: Connect the cable and the display unit
WATCH: The cable and the display unit are connected
ACTION: Push some buttons
WATCH: Nothing changes
ACTION: Reset
WATCH: Screen frikab text
ACTION: Push some buttons
WATCH: Screen letters shown
INTERNAL: A fault occurs
ACTION: Quicktest
WATCH: Screen black

From the trace above the simulated novice learns rules like:

- if quicktest is done the seekformula could be showed
- if the cable and the display unit are disconnected the connection between the cable and the display unit disappears

The second phase

The rules are compacted and concepts like parts, tests, faults are found. The simulated novice starts to work with the goal of making the quicktest work. The following is an example of how the concept "parts" could be found:

From the first phase a lot of rules consisting of a list of objects have been generated:

It is only possible to repair: printing_unit ...
It is only possible to watch: printing_unit ...

When the same list of words are found in many rules these rules could be compacted by exchanging the list of words with one word. Instead we have the following rules:

It is only possible to repair parts.
It is only possible to watch parts.
The third phase

Rules are transformed to mechanisms. In this phase a backward chaining mechanism is applied on rules as the following.

Repairing a part could make seekformula ok
Repairing the same part could not make seekformula ok

From the backward search traces are produced. From traces mechanisms are constructed.

8.5.4 Mechanisms

Some ideas of mechanisms that could be used to develop the system described are going to be presented.

There could be one mechanism that discovers rules. One general problem solver could be used to solve problems (e.g., making the terminal to work) with help of the rules produced by the mechanism that discovers rules. There could be one mechanism that constructs mechanisms from traces produced by the general problem solver.

To keep the memory clean from garbage there could be two mechanisms: One mechanism could be used to clean memory from unused concepts that could be explained from other types of knowledge. One mechanism could be used to search for patterns in the rules and make the knowledge base more compact by inventing new concepts that use the patterns.

One mechanism could evaluate the results of the different mechanism and distribute resources in accordance with their results.
9 Results

9.1 What has been done

Investigating knowledge structures in fault finding

A list of different types of knowledge associated with faults in a Sesam terminal have been produced. This list could be used in the construction of fault finding systems. It could also be used in an investigation of the possibilities to build a learning system that learns fault finding with help of general mechanisms.

A vocabulary used to discuss about knowledge was developed. A part of this vocabulary has been taken from Worden [9].

A list consisting of suitable assumptions, in the domain of fault finding a Sesam terminal, is produced. This list is a background of the learning and fault finding mechanisms.

Constructing a learning system

A learning system for fault finding has been constructed. This system contains many different types of knowledge, three ways to find faults and four ways to learn fault finding. The constructed learning system works for a class of fault finding problems that is described in Chapter 4.3.

The developed system could be viewed as an architecture of a general learning system for fault finding. The system could also be used as a testbench of learning mechanisms.

9.2 Experiences and conclusions

Knowledge

The knowledge found are related to each other in different ways. They could be partly or completely derivable from each other.

It existed a very lot of different types of knowledge that could be associated with the fault finding of a Sesam terminal. It would not be suitable to save all. How to decide what types of knowledge to put into a knowledgebase? An example of a question that has risen is: Should one allow redundancies in the knowledgebase? If two representations are derivable from each other should one save both or just one? How should one do with concepts that are similar but not exactly the same? Eg: connected to versus connectable.

A framework to decide the right combination of pieces of knowledge to put into the knowledgebase is probably needed. To build this framework a theory that could explain why new types of knowledge emerge is needed.

Fault finding

The way a technician find faults could depend on his competence and the actual assumptions. Different types of fault finding problems seem to need different fault finding mechanisms working from different types of knowledge.
Learning

The learning mechanisms are appropriate in different situations. More learning mechanisms could have been produced by combining the input knowledge of one mechanism with the method of another mechanism.

Two phases have been identified in the learning process: The collecting of knowledge and the transforming of knowledge to other types of knowledge.

The learning routines "learning from deep knowledge" and "learning by trial and error", that use the deep knowledge and the simulator, are not able to learn the correct order between the different actions.

Cross-checking different sources of knowledge using different ways of learning the "same" knowledge has proved to be a way of finding faults in the knowledge.

The experiences from this project indicates that it is possible to build a learning system when the structure of the knowledge is known.

9.3 Future work

A project idea is described where the purpose of the project is to investigate the possibility of extending the prototype to a system for use on a real case. The plan is: Adjust the knowledge representation and the mechanisms to new more complexed case. Build an expert interface. Integrate the learning mechanisms. Test the developed system on another case.

A project idea is described where the work in this project is used to build a learning system that invents the knowledge types, fault finding and learning mechanisms. Ideas have been presented of:

- how new types of knowledge could be invented
- how the learning and fault finding mechanisms could emerge from the background knowledge
- how it could be possible to generate the background knowledge, new knowledge types, fault finding and learning mechanisms with the help of a simulator of a Sesam terminal a small set of knowledge and general mechanisms
10 References

    An Artificial Intelligence Approach 1984


    Pearl J.: Search and Heuristics, 1983

    An Artificial Intelligence Approach 1984

    System: An Overview and Experiments July 1986

    Intelligence Volume 32 Number 1 April 1987

    Expert Knowledge.

    This article appeared in J.S. Kowalik 1985

A1 Appendix 1 - Knowledge associated with the fault finding of a Sesam terminal a listing

A1.1 A Sesam terminal

A1.1.1 Physical structure

A Sesam terminal consists of the following parts:

- power
- printing unit
- a cable
- a display unit
- a keyboard
- a control unit
- a test plug

The printing unit is connected to: power supply, cable, keyboard, control unit. The display unit is connected to the cable. The plug could be connected to the printing unit.

A1.1.2 Functional structure and the relation between the functional and the physical structure

A Sesam terminal (see figure 2) could also be divided into the following functions:

- power
- reset (input)
- keyboard (input)
- send
- plug
- control-unit
- receive
- screen (output)

The power function delivers electricity to the reset function and the keyboard function.

The reset function is initiated with a push on the rest button. It needs electricity and delivers an internal picture called "Friakab text". This picture is sent to the screen function. The reset function also connects the keyboard function to the screen function. The reset function is located inside the printing unit.

The keyboard function is initiated with a push on the keyboard. The keyboard sends signals to the send function. The keyboard function is located inside the keyboard and the connection between the keyboard and the printing-unit.

The send function is initiated with a signal from the keyboard function. The send function sends these signals to the control unit function or the plug function depending on which function it is connected to. The send function is located inside the printing unit.

The control unit function is initiated with a signal from the send function. The control unit function transforms a special signal called "name, off, name" to a seek formula. The
seek formula is sent to the receive function. The control unit function is located inside the control unit and inside the connection between the control unit and the printing unit.

The plug function is initiated with a signal from the send function. The plug function sends this signal to the receive function. The plug function is located inside the plug and inside the connection between the plug and the printing unit. The plug function is assumed to be ok.

The receive function is initiated with information from either the plug function or the control unit function. The receive function sends this information to the screen function. The receive function is located inside the printing unit.

The screen function is initiated with information from either the keyboard function or the receive function. The screen function presents the result on the screen. There may be two kind of faults in this function. The function may be weak or broken. This function is located inside the printing unit, inside the cable and inside the display unit.

### A1.1.3 Primitive actions and possible results

The following primitive actions exist:

- repair of a part (exchange)
- connect a connection
- disconnect a connection
- checking a connection
- watch screen
- push some buttons on the keyboard
- push "name off name"
- push the reset button
- wait until pictures shown
- wait

The results of watch screen could be:

- screen black (nothing happens)
- seek formula ok
- seek formula bad
- frikabtext ok
- frikabtext bad
- letters shown

### A1.1.4 Faults

The following faults may exist:

**power function**

- the power function is broken inside the power supply
- the connection between the power supply and the printing unit is broken.

**reset function**

- the reset function is broken inside the printing unit
keyboard function

- the keyboard function is broken inside the keyboard
- the connection between the keyboard and the printing unit is broken.

send function

- the send function is broken inside the printing unit.

control unit function

- the connection between the printing unit and the control unit is broken.
- the control unit function is broken inside the control unit.

receive function

- the receive function is broken inside the printing unit

screen function

- the screen function is weak inside the printing unit.
- the screen function is broken inside the printing unit.
- the connection between the cable and the printing unit is weak.
- the connection between the cable and the printing unit is broken.
- the screen function is weak inside the cable.
- the screen function is broken inside the cable.
- the connection between the cable and the screen is weak.
- the connection between the cable and the screen is broken.
- the screen function is weak inside the display unit.
- the screen function is broken inside the display unit.

A1.1.5 Tests

There exist different tests to test different parts of the terminal.

quicktest

description: To test if the system is ok the telephonist pushes the buttons "name off name". If a seek formula is shown on the screen everything is ok. If not something is wrong.

Consists of actions:

- push "name off name"
- watch screen

Tests the functions:

- keyboard
- send
- control unit
- receive
- screen

Expected values:

- seek formula ok
- seek formula bad
- screen black

reset screen test

Description: Push a button on the back of the printing unit. If everything is ok FRIKAB will be displayed on the screen.

Primitive actions:

- push the reset button.
- watch the screen

Test function:

- screen function

Expected values:

- frikab text ok
- frikab text bad
- screen black

reset keyboard test

Description: Push a button on the back of the printing-unit. After the text FRIKAB is written type some letters on the keyboard. If everything is ok the letters typed on the keyboard should be shown on the screen.

Consists of actions:

- push the reset button
- wait for text on the screen
- push some buttons
- watch screen

Test functions

- keyboard function
- screen function

Expected values

- letters shown
- screen black (letters not shown)
plug test

Put a test plug instead of the connection between the printing unit and the control unit. Type some letters on the keyboard. If the terminal is ok the letters typed on the keyboard should be shown on the screen.

Consists of the actions

- disc printing unit and the control unit
- connect the test plug and the control unit
- push some buttons
- watch screen

Test of functions

- keyboard
- send
- receive
- screen

Expected values

- letters shown
- screen black

A1.1.6 Actionlists

fattest screen black
- make reset screen test frikab text ok
- reset screen test screen black
- reset keyboard test nothing happens
- plug test nothing happens
- repair control unit

quicktest seek formula bad
- reset screen test frikab text bad

reset screen test screen black
- repair power supply
- connect the printing unit and the cable
- connect the cable and the display unit
- repair the display unit
- repair the printing unit
- exchange the cable

reset screen test frikab text bad
- connect the printing unit and the cable
- connect the cable and the display unit
- repair the display unit
- repair the printing unit
- repair the cable
reset keyboard test nothing happens
- reset screen test screen black
- connect the printing unit and the keyboard
- repair the keyboard

plug test nothing happens
- reset keyboard test nothing happens
- repair printing unit

A1.1.7 An abstract description of the knowledge and the problems

An abstract description of the domain of a Sesam terminal is described in Chapter 4.3.

A1.2 A generalized description of knowledge

A1.2.1 Physical structure

- what parts does the main object consist of
- how are these parts connected to each other
- what connections does a certain part consist of
- what connections are connectable

A1.2.2 Relations between the physical and the functional structure

- what functions does an object consist of
- in which parts a function is located

A1.2.3 Faults

- the possible faults
- how often certain faults occur
- the caused by relation
- a hierarchy of fault classes

A1.2.4 Functional structure

- a hierarchy of functions
- which functions are connected
- rules that contains the precondition of a function and the result of a function (special rules when a fault occur)
- the normal behavior of the main object

A1.2.5 The primitive actions and the results

- which are the primitive actions
- which are the results of an action
- the cost of a primitive action
- a description of a primitive action
- a description of the possible results of an action

A1.2.6 Tests and actions

- which are the tests
- which primitive actions does a test consist of
- a description of how the test is made
- possible results of a test
- which parts and connections does a test use
- which functions does a test use

A1.2.7 Fault finding

- a relation between a result of a test indicating a fault and a list of actions that could cure this fault
- rules that given a test indicating a fault gives an action that could cure this
- rules that given a certain action tell a test whose value this action could have changed
- rules that given the value of some tests tell the faults
- relations between tests and faults that is: given the value of one test which could be the faults
- given a fault which are the consequences
- decision tree that tells exactly how to repair the terminal.
- decision tree that gives faults

A1.2.8 Case knowledge

- the values of tests
- if an action has been made or not
- the actual physical configuration
- a list of problems that the technician is working with
- a trace of actions
- examples that contains the faulty tests and an action
- modes

A1.2.9 Commonsense knowledge

- If you repair a part then it is not faulty.
- if A is connected to B then B is connected to A

A1.2.10 Relations between the physical knowledge and other types of knowledge

- every part is linked to an action (repair).
- every part has one or more faults linked to it.
- every connection is linked to the actions connect, check connection and disconnect
- every part participates in one or more functions
- every part should take part in some test

A1.2.11 Relations between functional knowledge and other types of knowledge

- every function is located in some part
- two functions that are connected have one part in common or are physically connected
- every function could be faulty
- every function should be a part of some test

43
A1.2.12 Relations between test knowledge and other types of knowledge

- every test uses some functions
- every test uses some parts
- different tests could test the same parts
- if a test is not correct then some function is faulty
- if a test is not correct then some part belonging to a faulty function is faulty

A1.2.13 More examples of knowledge

- abstract descriptions
- knowledge about how to find the knowledge
- knowledge about how to use the above knowledge to find a fault.
- knowledge about how to use the knowledge to learn.
- detailed knowledge
A2 Appendix 2 - Vocabulary for the characterization of knowledge

In order to be able to discuss about knowledge a vocabulary to discuss knowledge is needed. Worden gives in his paper [9] the following words that could be used to characterize different types of knowledge:

- deep/shallow
- procedural/declarative
- meta level
- abstract versus concrete
- symbolic versus analogue
- narrow versus broad

Shallow to Deep

The shallow knowledge consists of heuristics (actionlists and rules), knowledge about tests and knowledge about the physical structure (figure 3). The deep knowledge consists of knowledge about the functional and the physical structure (figure 3 and 4).

"There is an implication that shallow correlates with empirical and heuristic - with knowledge close to direct observation of the world, that deep is a result of theorizing. The aim of theorizing is to describe more data with less knowledge, so deep knowledge is more economical." It is more economical memorywise but it could be more costly timewise.

Procedural and Declarative

"Knowledge is either about the way the world is (declarative) or about sequences of steps to do things (procedural)"

Meta-level

"Knowledge about knowledge is meta-knowledge."

Abstract versus Concrete

"Abstraction means throwing away details" One important thing is to discuss is: How should this description of knowledge relate to "Deep/Shallow" and "General/Specific" (see below).

Narrow versus broad

"This is one of several pairs of terms which approach the important questions of amount of knowledge." Another question could be: How should the amount of knowledge be measured?

Other types of words that could be used in a discussion about knowledge:

- general versus specific
- usefulness
- operationality for a certain problem
- how easy is it to acquire
- safety
- stability
- types of knowledge
- the relations between different sources

**General versus specific**

How general is the concept? What kind of knowledge is it supposed to contain. A rule is an example of a very general concept.

**Usefulness**

Which types of problems can be solved with the help of this piece of knowledge? Is it possible to use this piece of knowledge for a lot of purposes or is it very specialized. May it be used to produce other types of knowledge that are used often?

**Operationality for a certain problem**

If this type of knowledge gives the correct solution in a direct and fast way then the knowledge is operational for the problem.

**How easy is it to acquire**

Is it easy to acquire this knowledge or is the knowledge hidden? Is it possible to ask an expert directly about this knowledge?

**Safety**

Is it easy to verify the knowledge by inspecting it? Could it be verified in another way?

**Stability**

How stable is the knowledge? How often is it modified?

**Types of knowledge**

Knowledge about how to find a fault is one type of knowledge. Knowledge about the physical structure is a different type of knowledge.

**The relations between different sources knowledge**

In what way are the sources of knowledge related to each other? Are they partly or totally derivable from each other.
A3 Appendix 3 - Actionlists a background

In this chapter is the background of the actionlists going to be illustrated.

To illustrate this an example of rules (example 1) and an example consisting of actionlists (example 2) will be used. The following will be shown:

- the regularities in the rules that are used to transform them to the actionlists
- that the actionlists could be viewed as containing the same knowledge as the rules. This is shown by showing how the actionlists could be transformed to the rules.

Example 1:

rule: 1

If the last action was check electricity and
reset screen test before this action was made has given the result screen black
then set reset screen test as unknown

rule: 2

If the last action was check electricity and
quicktest before this action was made has given the result screen black
then set quicktest as unknown

rule: 3

If the last action was exchange the display unit and
reset screen test before this action was made has given the result screen black
then set reset screen test as unknown

rule: 4

If the last action was exchange the display unit and
quicktest before this action was made has given the result screen black
then set quicktest as unknown

rule: 5

If quicktest gives the result screen black and
reset screen test gives the result screen black and
the electricity is not checked
then check the electricity

rule: 6

If quicktest gives the result screen black and
reset screen test gives the result screen black and
the display unit is not exchanged
then exchange the display unit
Example 2:

actionlist([(quicktest,screen_black), [reset_screen_test,screen_black])].
actionlist([reset_screen_test,screen_black],[[check,electricity],[exchange,display unit]]).

Regularities inside the rules in example 1:

rule 1 to 4

If last action was ACTIONX and
TESTY before this action was made has given the result VALY
then set TESTY as unknown

rule 5 to 6

If TEST1 gives the result VAL1 and
TEST2 gives the result VAL2
then set ACTIONX as unknown

Transformation of the knowledge in example 2 to the knowledge of example 1

The following transformation of the actionlists to the rules shows that the actionlists could be viewed as containing at least the same knowledge as the rules..

Build actionlist 2 from the two actionlists in example 2:

actionlist2([(quicktest,screen_black), [reset_screen_test,screen_black]],
            [[check,electricity],[exchange,display unit]])..

The rules 1 to 4 in example 1 could be produced from actionlist 2 by combining each element ([(TEST,VAL)]) in the first list inside actionlist 2 with each element in the second list inside actionlist 2 ([(ACTION)]) in the possible four ways. Each of these four combinations should be put into the following rule.

If the last action was ACTION and
the test TEST has given the result VAL
then set TEST as unknown

The rules 5 and 6 in example 1 could be produced from actionlist 2 by combining the first list ([TEST1,VAL1],[TEST2,VAL2]) with each element ACTION in the second list in the possible two ways. Each of these two combinations should be put into the following rule.

If TEST1 gives the result VAL1 and
the TEST2 gives the result VAL 2
and ACTION has not been made
then make ACTION
Regularity were found in the rules in example 1. The regularities were used to transform the rules in example 1 to the actionlists in example 2. This resulted in a compact description but less operational and less general.

There could be problems with the more compact structure if for example a rule is found that does not follow the regularity above. The following rule is an example of a rule that does not follow the patterns above and thus is an irregularity:

If reset screen test has not been made and reset screen test gives as a result that frikab text is ok then set quicktest as unknown
Research Reports


R 86002  Hausman, B., A Simulator of the OR-Parallel Token Machine, 1986


R 86005C  Hallnäs, L., Partial Inductive Definitions, 1988 (Revised version of R 86005 "On the Interpretation of Inductive Definitions" and R 86005B)


R 86009B  Appleby, K., M. Carlsson, S. Haridi and D. Sahlin, Garbage Collection for Prolog Based on WAM, 1988 (Revised version of R 86009) (Also published in Communications of the ACM, June 1988, pp. 719-740)

R 86010  Hallnäs, L., Generalized Horn Clauses, 1986 (No longer distributed; replaced by R 88005)


R 86012  Carlsson, M., An implementation of "dif" and "freeze" in the WAM, 1986 (Out-of-print, see R86011 above)


Holmgren, F. and A. Wärm, *A Scheme for Compiling GHC to Prolog Using Freeze*, 1987


Sjödin, P., *Optimizing Protocol Implementations for Performance - A Case Study*, 1987 (Author's Licentiate thesis at Uppsala University, Sweden)

Franzén, T., *Algorithmic Aspects of Intuitionistic Propositional Logic*, 1988 (Revised version of R 87010)


Khayri A. M. Ali, *OR Parallel Execution of Horn Clause Programs Based on WAM and Shared Control Information*, 1986


**Technical Reports**


T 89008  Tunevi, A., A Learning System for Fault Finding, 1989

T 89009  Tunevi, A., Knowledge Structures, Strategies and Learning Processes in Fault Finding, 1989