Failure-driven reminding for incremental learning

Christopher K. Riesbeck

Computer Science Department
Yale University

ABSTRACT

Much of adult learning is gradual, almost imperceptible. Our model for this knowledge-based, incremental learning is to augment normal story comprehension processing with a failure tracking mechanism. When a comprehension rule fails, the failure and its correction are stored in an exception episode attached to the failing rule. The rule is otherwise unchanged. Subsequent failures of that rule trigger the retrieval of these exception episodes (failure-driven reminding). Rule modification occurs when classes can be found for the known exceptions. The ALFRED program is a preliminary implementation that classifies and remembers failures of "everyday knowledge" in the domain of political economics.

A LEARNING EXAMPLE

One of the members of our learning group read an article in favor of controlling credit cards. The article said that they account for $55 billion of the total credit in the American economy, and this convinced him that credit cards contribute to inflation and probably should be controlled.

But two days later he read an article that ssid that credit cards were insignificant compared to the $1.23 trillion of total credit in the economy. This changed his mind. He realised he had been wrong in thinking $55 billion was a large part of total credit.

A week later, he read an article that said that adding a *lOi* per gallon tax on gas would decresse consumption by 100,000 barrels a day. At first, that effect looked too big, but then he remembered having misjudged the size of $55 billion the week before. Resding further, he found that current consumption was over 6 millions barrels a day, so the expected decrease was actually quite small, in keeping with the small site of the tax.

This work was funded in part by the Office of Naval Research under contract N00014-75-C-1111.

We believe that being reminded of prior failures is part of the following underlying learning process:

1. When new beliefs contradict old beliefs, debugging processes decide which belief to reject and which inference rule to blame for having accepted that belief (a non-trivial problem — see [6] and [12]).

2. An exception episode describing both the failure and the fix is attached to the faulty inference rule.

3. Later, if the same rule is blamed for another failure in some new situation, the previously stored episode is retrieved (this is called iailurc-drmp reminding)-

   When a subset of the exception episodes can be grouped into a class (e.g., episodes within one domain), the failing rule can be modified to treat that class correctly, and the exceptions removed.

We do not have a classification scheme for exception episodes to handle the last step, but we hope thst existing methods (e.g., [7])) will be appropriate. We report here on a special — but common — case of the above process: the failure of "everyday knowledge," auch as that 55 billion is a big number. We use such rules freely and yet we find it very hard to give reasons why we believe them. As we become more expert in some field, we learn to replace these rules with more specific facts, and to use more cautious rules, such as: "don't assume — find out!"

Thus, as we become better at economics, we learn not only the real sices of various economic quantities, but we learn to postpone judging the relative sises of things until we have explicit points for comparison. Outside of economics, of course, we will still think 55 billion is a lot.

Nor do we stop using everyday rules in economics immedistely. The first time one fails, it is neither removed nor changed. It is only tagged with the failure episode. The rule is still used to generate new beliefs, as long as no

further problems trite* If t problem does trite, however, end the rule is considered suspect, its previous failures tre remembered.
The tdvtnttge of thit approach is that rules stay simple and efficient as long ae they work most of the tine. But failures tre noted and chtnges made if a rule fails several tines. The distdvtnttge of this tpprotch is thtt a rule known to have problems may still be tdding plausible, but incorrect, beliefs to the system.

THE ALFRED PROJECT

ALFRED (Automatic Learning using Failure-driven Reminding in an Expert Domain) is a program being developed at Yale to model learning sequences such as the one above. In February and March, 1980, several learning sequences were gtthered by the ALFRED project while retding stories in the Wall Street Journal end the New York Times about politicians and their proposals regarding the economy. These stories were about credit controls, anti-inflation proposals, economic platforms, partisan battles over budget cuts, and to on. As our inititl beliefs tbout inflttion, recession, politicians, and so on, were found wanting, t number of obviout letrning experiences became the basis for our research.

LEARNING AND UNDERSTANDING

Like Solowty 111] tnd Sussmtn [12], we believe thtt letrning does not sttrt from scratch, but occurs in the context of an ongoing application of knowledge that already exists. New things are interpreted as instances of old things, and failures of fit ctute existing knowledge and processing structures to be modified. In our case, we made ALFRED a story understanding program, similar to SAM [4] tnd PAN (13], but with three major differences.

Firtt, ALFRED does not yet ttke natural language input. We give it conceptual repretentttiont equivtlent, tt t crude level, to sentences from selected articles. This is t serious wetkness. Ve feel that skimming and focusing strttegiet tre closely linked with the letrning process. ALFRED needs more than t "natural language front-end." It needs a we11-developed model of language analysis driven by dynamically changing intereats and beliefs.

Second, ALFRED chtnges its knowledge structures on the basis of the stories it understands. It is not enough for ALFRED to understand an argument. It must also decide whether to believe it or not.

•Members of the Yale letrning group htve included Mtrk Burstein, Gregg Collins, Drew McDermott, Shoshana Hardt and Alan Cypher.

Third, ALFRED is designed to detl with expectations thtt ftil rtther then succeed. We deliberately choose stories that contradict what ALFRED already believes.

THE ALFRED PROGRAM

We developed two programs to test the failure-driven reminding aspect of our learning model. One program, written by Mark Burstein, covered the the credit card and gas tax example given at the start of this paper. The program took hand-analyred input, looked for related stored beliefs, and checked for contradictions. If one was found, the belief supported only by an everyday rule was rejected and the everyday rule was marked with an exception episode.

The first input represented "The government has proposed controls on credit cards." ALFRED linked this to a belief that credit causes inflation, and predicted further input supporting the notion that credit card control would reduce inflation.

```
(PROPOSED-ACT
  ACT (GOVT-CONTROL
       OBJECT (CREDIT TYPE (CREDIT-CARD))))
```

Found causal connection

```
(CAUSE
  ANTE (RATE-CH DIR *DVAR OBJ (CREDIT))
  CONSE (RATE-CH DIR *DVAR OBJ (INFLATION)))
```

Inferring PROBLEM is INFLATION

Expecting support for

```
(CAUSE ACTOR (US-COVT)
  ANTE (RATE-CH DIR (NEC)
        OBJ (CREDIT TYPE (CREDIT-CARD)))
  CONSE (RATE-CH DIR (NEG) OBJ (INFLATION)))
```

The next input represented "Credit cards contribute $55 billion in credit." ALFRED used a set of rules called CHECK-SCALES which decided that $55 billion was enough to make credit cards a significant part of total credit and hence a significant factor in cauaing inflation.

```
(FRACTION
  PART (CREDIT
        TYPE (CREDIT-CARD)
        AMOUNT (55 SCALE (BILLION)
                   UNIT ($)))
  OF (CREDIT ACTOR (CONSUMER)))
```

CHECK-SCALES — CREDIT-CARD is a SIGNIFICANT part of total CREDIT

Accepting input as support for

```
(CAUSE
  ANTE (RATE-CH DIR (NEG)
        OBJ (CREDIT TYPE (CREDIT-CARD)))
  CONSE (RATE-CH DIR (NEG) OBJ (CREDIT)))
```

ALFRED now believed credit card control would work. The next input represented "Controls on credit cards will do little to combat inflation," which was contradicted the newly acquired belief. The input was not yet supported however so nothing happened.

```
(CAUSE
   ANTE (GOVT-CONTROL-ECONOMY
            OBJ (CREDIT TYPE (CREDIT-CARD)))
   CON8E (RATE-CH DIR (NEC)
            OBJ (INFLATION) SIZE (SMALL)))
```

Found referent GOVT-CONTROL-ECONOMYO

*** Input is CONTRADICTION to known causal
- expecting support for contradiction statement.

The next input represented "Credit cards are only $55 billion out of $1.23 trillion in total credit." CHECK-SCALES said that this made $55 billion a small fraction of total credit, supporting the new claim. Since it was an everyday rule in CHECK-SCALES, called CS-DEFAULT-WHOLE, that said that $55 billion was big, ALFRED saved the current story as an exception to CS-DEFAULT-WHOLE.

```
(FRACTION OF (CREDIT AMOUNT (1230 UNIT ($)
                       SCALE (BILLION)))
           PART (CREDIT TYPE (CREDIT-CARD)
                        AMOUNT (55 UNIT ($)
                       SCALE (BILLION))))
```

CHECK-SCALES — CREDIT-CARD is a SMALL part of total CREDIT

```
Accepting input as support for negation of
  (CAUSE
     ANTE (RATE-CH DIR (NEC)
              OBJ (CREDIT TYPE (CREDIT-CARD)))
     CONSE (RATE-CH DIR (NEG) OBJ (CREDIT)))
```

***** Processing error — accepted contradictory supports
Searching for errors made in process CHECK-SCALES

```
Found probable source of error in use of
CS-DEFAULT-WHOLE in CHECK-SCALES
 when processing input
   (FRACTION SIZE (LARGE)
           PART (CREDIT TYPE (CREDIT-CARD))
           OF (CREDIT ACTOR (CONSUMER)))
```

Indexing error episode EP1 on mop CHECK-SCALES

Now ALFRED was given, the representation for "The government announced a 10 cent tax on oil to reduce its consumption." This was linked to a belief that prices affect consumption.

```
(PROPOSED-ACT
   ACT (GOVT-CONTROL
          PROBLEM (OIL-CONSUMPTION)
          SOLUTION 4
           (SALES-TAX
              OBJECT (OIL UNIT (GAL))
              AMOUNT (10 UNIT (CENTS)))))
```

```
Found support
  (CAUSE ANTE (CHANGE DIR *DVAR
                  OBJ (PURCHASE-PRICE
                          OBJ *OVAR))
        CONSE (RATE-CH
                  DIR *DINV
                  OBJ ($BUY ACTOR (CONSUMER)
                             OBJ *OVAR)))
for input.
```

The next input represented "This would save 100,000 barrels of oil per day." This was linked to a belief that causal effects are commensurate; hence, a small change in price should lead to a small change in consumption. But CHECK-SCALES, using CS-DEFAULT-WHOLE, said that 100,000 barrels was a large change. An internally generated contradiction was noted, blamed on CS-DEFAULT-WHOLE, and the previous story episode was remembered.

```
(CAUSE
   ANTE (SALES-TAX OBJECT (OIL UNIT (GAL))
                AMOUNT (10 UNIT (CENTS)))
   CONSE (RATE-CH DIR (NEG)
                OBJ ($BUY ACTOR (CONSUMER)
                          OBJ (OIL))
               AMOUNT
                (100 SCALE (THOUSAND)
                    UNIT (BARREL)
                    PER (DAY))))
```

CHECK-SCALES -- (10 UNIT (CENTS)) is a SMALL part of total
(PURCHASE-PRICE OBJ (OIL UNIT (GAL)))

```
CHECK-SCALES — (100 SCALE (THOUSAND)
                 UNIT (BARREL) PER (DAY))
 is a SIGNIFICANT part of total $BUY0
```

Error detected
(SIZE) of consequent - SIGNIFICANT
Does not match expectation given antecedent
(SIZE) - SMALL

Noted error in applying VERIFY-PREDICTION
Found probable source of error in use of
CS-DEFAULT-WHOLE in CHECK-SCALES

** Step CS-DEFAULT-WHOLE cauaed previous error in episode EP1
Reducing certainty of process-step
CS-DEFAULT-WHOLE to 0

There *ate* many problems with the program just presented. It was not a general purpose story understander, and it did not start with a lot of knowledge. But the moat glaring problem to us was that we had no well-defined structure for episodes and no well-defined deacription of the debugging process. Our solution, presented in the rest of this paper, tries to answers both deficiencies with the same data structure.

MOPS

ALFRED'S proceaaing structures are baaed on Schank'e Memory Organisation Packeta (MOPa) (9]. Although MOPa are basically juat frames ([1], 13]), the important thing ia that they organise epiaodic experiences in long-term memory while they simultaneously process those experiencea. Reminding ia basic to understanding, since the proceaa of understanding ia the same aa the proceaa of epiaodic memory search. Furthermore, aa inputs change the aet of categories uaed in memory, the courae of future understanding ia changed.

MOPa in ALFRED have the following parte:

1. a conceptual pattern called the trigger
2. a aet of conceptual patterns, that make up the content of the MOP
3. a aet of indicea to subMOPe or epiaodea
4. a aet of rules for filling in the variablea in the conceptual patterns

In the description below, we shall mostlly ignore the indicea. Each index value labels a link to either a particular epiaode, or a subMOP collecting together a aet of similar epiaodea (aee [7] and [8]).

Here ia an outline of COVT-CONTROL, a MOP organising knowledge about governmental regulation of some activity:

GOVT-CONTROL

Trigger: ?Actor control ?Object

Concepts: TActor authorise
LBGAL-CONSTRAINT(Uee of ?Object)
CAUSE
Rate of TActivity = Decreaae

Goal of TActor
= GOVT-FIX-PROBLEM(TProblem)

Indices: Domain of activity
Kind of regulation
Object regulated

Rulea: TO FILL TActivity:
{Activity <• function of 70bject

TO FILL ?Problem:
Find an undeaired state cauaed by TActivity

Question marks precede the variablee in the conceptual patterna. LEGAL-CONSTRAINT and GOVT-FIX-PROBLEM are other MOPa. LEGAL-CONSTRAINT containa knowledge about how laws work. GOVT-FIX-PROBLEM containa knowledge about reducing unwanted situationa by regulation, de-regulation, taxation, and ao on. "Authorise" ia one of several kinda of prtdicatta and relationships needed in the political domain (aee [10]).

GOVT-CONTROL ia invoked after reading a aentence such as "Carter propoaea controla on credit carda." Thia fills two variablea:

Actor <• Carter/US-Government
Object <- credit carda

Because the normal function of credit carda ia to get credit:

Activity <= Get credit

Because credit causea inflation and inflation ia one of the problems the government wanta to fix:

Problem <■ inflation

In thia way, ALFRED infers that Carter intends to limit credit card uae in order to fight inflation.

The rulea uaed to fill variablea are important in ALFRED because they explicitly repreaent a kind of knowledge that changea during learning. In particular, there ia a class of rulea, called <u>default</u> rulea, that fill in variablea with approximate answera when exact onea can't be found. As ALFRED becomes more expert in political economics, it haa to learn to replace theae default rulea with more apecific, more accurate onea.

To organise rulea, we uae <u>proceaa MOPs</u>. Where a regular MOP organisea evente and other MOPa, a proceaa MOP organises inference rules. A pattern in a proceaa MOP may aay something like "rule TR failed," where the variable R ia filled with a pointer to some rule.

One uee of proceaa MOPa ia to organise a aet of rulea into a atrategy, which can then be uaed aa a rule. For example, CHECK-SCALES ia a aet of rulea for judging the relative aise of a number:

CHECK-SCALES:

Trigger: To find the relative aise TR for TN unite of TX

Rulea: TO FILL TR:
Compare TN againat a known scale for TX

If thia fails, compare TN againat a known scale for a auperclaaa of TX

The aecond rule for filling TR ia a default rule.

Another example of a proceaa MOP ia the EXCEPTION MOP. It recorda what happens when a problem in understanding occurs. Below are the trigger and conceptual parte for the EXCEPTION proceaa MOP (the rulea will be described shortly):

EXCEPTION:

Trigger: Belief 7B1 conflicts with ?B2

Concepts: Belief ?B3 is wrong.
?B3 is supported by rule ?R1.
Use ?R2 instead of ?R1.

This says that when a new belief contradicts an old one, find the incorrect belief, B3, find the rule Rl that led to it, and find a better rule, R2.

The EXCEPTION process NOP provides not only a mechanism for fixing the problem, but a frame for remembering how the problem was fixed. With the EXCEPTION process MOP we have both a mechanism to drive the debugging process, and, at the same time, a knowledge structure to organise the relevant pieces of the episode for long-term memory.

The EXCEPTION process MOP is invoked when a belief conflict is recognized. Sometimes, sn article may explicitly contradict a held belief. More commonly, the conflict arises during the inference process. For example, when the member of our learning group read that an additional tax on gas of 10c* per gallon would cause consumption to decrease by 100,000 barrels per day, he thought that this effect was too big for that small an increase in the price of gas.

In our model, the contradiction arose from inferences triggered by this causal:

Increase price of gas by 10cV a gallon
CAUSE
Decrease use of gas by 100,000 barrels a day

Knowledge about causation includes the following inference rule:

IF A causes B
AND A and B are changes in quantities
THEN the change in B is commensurste with the change in A

In order to use this rule, we find out how big the changes are with the CHECK-SCALES process MOP. CHECK-SCALES compares the 104 gss tax against the cost of a gallon of gas ($1.25) snd concludes that the increase is small. Therefore, the causal rule above predicts that only a small change should result in something else.

But when CHECK-SCALES looks at the decrease of 100,000 barrels per day, it can't find any actual value for gas consumption. Therefore it uses a default value of millions of barrels per year. Millions of barrels per year implies that 100,000 barrels per day is a large change.

The contradiction between the small change predicted by the causal and the large change returned by CHECK-SCALES invokes the EXCEPTION process MOP. Its job is to find out what went wrong and fix it.

The EXCEPTION proceas MOP fills in ita variables by finding the belief at fault, where that belief came from, and what can be done to prevent it from happening again. To do this, the EXCEPTION process MOP has the following rules:

EXCEPTION:

Trigger: Belief ?B1 conflicts with ?B2

Concepts: Belief ?B3 is wrong.
?B3 is supported by rule ?R1.
?R2 should be used instead of ?R1.

Rulea: TO FILL ?B3, ?R1 (the incorrect belief and rule):

If ?B1 is not yet supported then "wait for more input"

If TBI (?B2) ia supported only by a default rule ?R, then ?B1 (?B2) and ?R are at fault

TO FILL ?R2 (the better rule):
If a default rule is at fault, and ?V is the variable that the rule fills, then use the rule: "TO FILL ?V: wait for more input"

The above assumes that ALFRED at least partially remembers how it inferred the faulty belief. Also it only deals with failures by default rules. A more realistic MOP would reconstruct probable sourcea of faulty beliefs and would deal with other kinds of rule failures.

The EXCEPTION process MOP waita until the new input is supported. Then it finds the faulty belief by looking to see which one is supported by s default rule. The belief based on a default rule is replaced by the belief that contradicted it, and the default rule is replaced with the more cautioua "Wait for more input." If the faulty belief is the new one, then the replacement rule can be uaed immediately. In our example, when our learner realised that he might have incorrectly acaled 100,000 barrels in the article he was reading, the "wait for more input" was applied at once. He looked for the real value to use.

"Wait for more input," by delaying variable bindings, can cauae aome complex and difficult problems for a predictive understanding process. An alternative possibility would be to scan the text for the desired information, juat aa the FRUMP program [5] skimmed newspaper articles to fill in its sketchy scripts.

SUMMARY OF THE LEARNING MODEL

Our research hat stretted several idessa regarding the learning procett:

1. Episode-taving — when an inference rule in a MOP fails (or it inadequate) in understanding an episode, an exception link it made from that rule to the episode, tpecifying what the correct rule ahould have been.
2. Failure-driven reminding — when an inference rule in a MOP failt (or it inadequate), itt exception linkt (if any) are followed to tee if a previout epitode providet a better antver.
3. The EXCEPTION procett MOP — thit directt recovery and organizes the memory of the failure for later retrieval.

REFERENCES

[1] Bobrow, D., Winograd, T. and KRL Research Group. "Experience with KRL-0: One cycle of a knowledge repretentation language." In Proc. IJCAI-77 Cambridge. MA, August, 1977, 213 - 222.

[2) Chamiak, E. "Towards a model of children's ttory comprehension." Ph.D. Thesis, Report AI TR-266, Masschusetts Inttitute of Technology, Cambridge, MA, Augutt, 1972.

(3] Chamiak, E. "Me. Malaprop, a language comprehension program." In Proc. IJCAI-77 Cambridge, MA, Augutt, 1977, 1 - 7.

[4] Cullingford, R. E. "Script Application: Computer Understanding of newspaper stories." Ph.D. Thetit, Dept. of Computer Science, Research Report #116, Computer Science Department, Tale Univeraity, New Haven, CT, 1978.

[5] DeJong, G. F. "Skimming Storiet in Real Time: An Experiment in Integrated Underatending." Ph.D. Thesis, Dept. of Computer Science, Retearch Report #158, Computer Science Department, Yale Univertity, New Haven, CT, 1979.

[6] Doyle, J. "A truth maintenance system." Artificial Intelligence. 12:3 (1979), 231 - 272.

[7] Kolodner, J. "Retrieval and organizational ttrategiet in conceptual memory." Ph.D. Thetit, Dept. of Computer Science, Retearch Report #187, Computer Science Department, Yale Univertity, New Haven, CT, 1980.

[8] Lebowitz, M. "Generalization and memory in an integrated underttanding system." Ph.D. Thetit, Dept. of Computer Science, Retearch Report #186, Computer Science Department, Yale Univertity, New Haven, CT, 1980.

[9] Schank,R. "Reminding and Memory Organization: An Introduction of MOPt." Retetrch Report #170, Computer Science Department, Yale Univertity, New Haven, CT, 1979.

[10] Schank, R. C and Carbonell, J. G. "Re: The Gettytburg Address: Representing Social and Political Acts." Retearch Report #127, Computer Science Department, Yale Univertity, New Haven, CT, 1978,

[11] Soloway, E. and Riteman, E. "Levels of pattern detcription in learning." In Prpc. IJCAI-77 Cambridge, MA, Augutt, 1977, 801 - 811.

[12] Suteman, G. J. A Computer Model of Skill Acquisition. American Elsevier, New York, 1975.

[13] Vilentky, R. "Understanding Goal-based Storiet." Ph.D. Thetit, Dept. of Computer Science, Retearch Report #140, Computer Science Department, Yale Univertity, New Haven, CT, 1978.