Teleology + Bugs = Explanations

Gregg C. Collins

Yale University

In previous work, Schank and Collins (1982) have argued that learning requires the ability to explain the failures of expectations derived from the knowledge structures used in planning and understanding. Building a computational theory based on this idea, however, depends first of all on our ability to understand what kind of explanation of an expectation failure is required for learning, and what a system would need to know in order to construct such an explanation. One approach to this rather formidable task is to analyze actual instances of such explanation performed by people. This paper will look at one such instance in detail, and attempt to sketch the general method which might underlie it.

The example we will analyze involves explaining expectation failures arising from the behavior of a broken vending machine. This problem can be viewed as a kind of "debugging," and in fact prior work on that topic and on the analysis of device models in general (see, e.g., Sussman & Brown, 1974, McDermott, 1976, Stevens & Collins, 1978, Forbus & Stevens, 1981, de Kleer & Brown, 1981) will be relevant to the analysis. However, a big difference in this case is that we begin without a model of how a vending machine operates. Instead, we begin with our everyday knowledge of how to use one when it is functioning properly, plus an understanding of its purpose as seen by its owners and designers. In fact I will argue that we *construct* a model of how the machine works as a necessary part of explaining what is wrong with it. Our ability to reason teleologically about things (i.e. with respect to their "purposes") (deKleer & Brown, 1981, Sussman & Brown, 1974), combined with the information which can be derived from an analysis of the ways in which the device fails to fulfill its purposes, can take us surprisingly far in constructing a model of a machine about which we originally had little specific knowledge. The model must be developed to the point where it can be used to construct an explanation of the cause of the bugs we observed.

The actual episode in which the failures occurred was this: I went up to an ordinary Coke machine, which I had previously used without any problem, and deposited a quarter. The quarter came back in the coin return box. I tried several more times using different quarters, with the same result each time, and with no apparent effect on the machine. I then tried other denominations of coins, which the machine accepted. I tried a quarter again after having deposited some other coins, and it was again rejected. When I had deposited 30 cents in nickels and dimes, the machine lit up, signalling that I could now have a Coke, which it gave me. The usual price for a Coke in that machine was 55 cents. The same behavior occurred subsequently when I tried the same machine again.

The machine was behaving in a way that consistently violated two of the expectations derived from my get-a-Coke *script* (Schank & Abelson, 1977). I expected to be able to use quarters in the machine, and I expected to get the Coke for the advertised price of 55 cents. What was wrong? The explanation that immediately presented itself to me was that the machine consisted of some sort of ramp between the input slot and the coin return box down which coins slid or rolled. Off of this ramp would be chutes for quarters, dimes, and nickels, separately, and at the top of each chute would be some sort of sensing device which would increment a counter whenever a coin went by. What must be happening, I thought, was that a quarter was stuck at the top of the quarter chute, blocking other quarters and forcing them to slide on down to the coin return. This quarter would be at least partially inside the sensing device, and would be "counted" once on each cycle of the machine, so that Cokes effectively would cost 25 cents less than they normally did. I believe this to an interesting example the explanation was not completely obvious, implying that the reasoning required was not trivial, yet there seems to be a fairly unanimous consensus that the explanation is the right one among people to whom I subsequently presented the problem, implying that special knowledge or abilities were not required to produce the explanation, but that it was in fact only a slightly clever instance of whatever it is people normally do in constructing explanations.

Analysis of the example

I will now present an analysis of the processes and prior knowledge which are necessary to produce the above explanation in terms of what a program capable of constructing it would need to know and do. The description will be presented as a series of steps starting with the initial knowledge of the expected behavior of the machine, and leading to a model of the machine which is complete enough to, first, explain how the vending machine *should* accomplish the functions which were perturbed, and second, how the perturbations could arise from plausible malfunctions of those mechanisms.

Step one: Modelling what should have happened

Initially what I knew was a rote plan, or *script*, for using a Coke machine, which told me what to do and what observable behavior to expect from the machine in response. One expectation told me that if I inserted dimes, nickels, or quarters in the machine, they would remain within, as long as the cost of a Coke had not been exceeded. Another told me that it would cost me the advertised price to get the Coke. In both cases, the actual observation differed from the expectation.

What we need in the final explanation of the discrepancy is a model of *how* the things we expected to happen are accomplished. What we have, probably, are just the expectations themselves. What I want to show is that we can bridge the gap by asking ourselves the teleological question: what purposes of the machine were affected by its failure to operate as expected?

We know of two purposes which are relevant to the observed bugs:

1. The vending machine is meant to collect a customer's money.
2. It is meant to give the customer a Coke when a certain amount of money has been collected.

Transforming this teleology into the mechanism required for the explanation is a matter of distributing the teleology among a set of interacting components which are themselves characterized, at first, *only* by their teleology. The interactions among the components can be represented by links which denote various types of causal relations. Two causal link types are needed in this example, corresponding roughly to Conceptual Dependency primitives PTRANS and MTRANS (Schank & Abelson, 1977). These are intended to indicate, respectively, the transfer of some physical object, and the transfer of information, between two components.

In order to distribute the teleology of the mechanism among interacting components, we must first determine suitable components. It may not require inventing new component types; rather, it may suffice to select items from an existing stock of devices described in terms of their teleology. However, our knowledge of the devices includes more than just their teleology, in particular, knowledge of how such devices are commonly implemented. We are therefore not only following deKleer in asserting that teleology is a crucial aspect of our vocabulary for device description, but further claiming that this serves as an organizing principle for memory about devices.

Given an appropriate stock of simple components, our knowledge of the existence of the coin slot from the rote plan enables us to construct the following model accounting for the first teleological fact about the vending machine, namely that it takes the customer's money:

```
                ptrans (coins)
SLOT ----------------------------------> RESERVOIR
```

A "reservoir" is simply a component which collects something. The appropriate causal links between the components is part of our knowledge of the components themselves. Reservoirs, we know, must have a link enabling an inflow of whatever substance they store. The nature of this link depends on what that substance is. In the case of a physical object like a coin, the mechanism by which it is input must of course be physical transfer, i.e., PTRANS.

We can extend this model to account for the second teleological fact, namely, that the vending machine should dispense a Coke when a certain amount of money has been collected:

```
                            mtrans
                 COUNTER -----------> DISPENSER
                    ↑
                    | mtrans
                    |
SLOT ------------ SENSOR ------------> RESERVOIR
              ptrans (coins)
```

A "dispenser" is a component which "gives one something." Inferring the existance of the "counter" and the "sensor" is a bit more complex: we must know that when control of an operation, in this case dispensing, is dependent on detecting a certain amount of something, then a sensor is required for detection, and a counter is required for storing the quantity detected so far. In fact we could abstractly view a counter as a reservoir for a quantity, and analogously with the coin reservoir, determine that since quantity is information, and causally appropriate input link is transfer of information, i.e., MTRANS. The sensor must be inserted in the path between the slot and the reservoir because the coins which are collected to satisfy the first purpose of the vending machine are the same coins which must be counted to satisfy the second purpose.

Step two: Locating the bug

Once we have constructed this simple model of the Coke machine, we must attempt to locate the sources of the observed bugs in the model. At this point we need to invoke domain-independent rules about locating bugs. The first, and most important is the well-known "single-fault assumption," which is the assumption that multiple bugs have a single source.

We can make use of this assumption to locate the bug due to a nice property of our representation, namely that each link and the devices at its endpoints can be seen as a "nearly decomposable" (Simon, 1969) subsystem. Thus, if one can assign bugs to different subsystems in the representation, the actual fault must lie at the intersection of those subsystems. In this case, one bug is clearly in the SLOT-RESERVOIR system, since coins end up in the coin-return instead of the reservoir. The other bug is in the COUNTER-SENSOR system, since the machine seems to be counting wrong. Applying single fault assumption tells us that the source of both bugs should be at the point where the sensor phyically contacts the SLOT-RESERVOIR connection. Thus a major part of the ultimate explanation stems from our ability to apply the single-fault assumption to our model in this way.

Step three: Modelling the bug

At this point, although we have located the source of the bugs, we have not yet explained them. We can look to two sources for the underlying causes which might explain a bug: the properties of the correctly functioning device, and the properties of the fault which is actually producing the bug. The observed symptoms which constitute the bug can generally be viewed as resulting from some relatively straightforward interaction between these two sources. One way of sorting out the interaction is to isolate characteristics of a bug which can be attributed predominantly to either the device alone or to the fault alone.

For example, we might view the entire description of the first bug, "rejects quarters," as such a characteristic. Viewing this bug in isolation from the second bug, we might then attribute it either to the device alone (the machine has a reason for rejecting quarters, e.g., when it is out of change) or to the fault alone (for example,the coin slot is blocked so that nothing as large as a quarter can be inserted). Since neither of these is particularly convincing -- for example, I would have noticed if the coin slot were in fact blocked -- we have reason to assume that "rejects quarters" is not best seen as being directly attributable to either the device alone or the fault alone, but rather is the result of a complex interaction. In such a case, we must attempt to isolate features of the bug at a finer level of detail -- e.g., that different denominations of coins are being discriminated -- which may be attributable to either the normal device alone or to the underlying fault alone.

Depending on which attribution is made, differing constraints will be imposed any subsequent explanation. If it is assumed that the characteristic in question results from the underlying fault, then the constraint is imposed by the single-fault assumption: all buggy behavior explained in this way must be traced to a single underlying cause. If, on the other hand, a property of the correctly functioning device is adduced to account for a characteristic, then the constraint is imposed by another domain-independent debugging rule, the *teleological* assumption. This rule asserts that any hypothesized property of a properly functioning device must have a teleological justification. We always assume that every component of the mechanism must serve some purpose, and if no purpose can be found for some component which is hypothesized to play a role in explaining a bug, then the explanation is probably wrong.

A critical feature of the buggy behavior which must be explained in this example is the fact that different denominations of coins are treated differently by the first bug, so that quarters are rejected while nickels and dimes are not. Applying the procedure above, we must decide whether this distinction is solely a property of the bug, or whether it is owing to a discrimination that the device would make under normal circumstances anyway. Would discriminating among the different denominations of coins serve some purpose in the normal function of the vending machine? In fact we know such a distinction must be present in the SENSOR-COUNTER subsystem, since the physical differences between coins indicate the differences in the amount of money they represent. If we assume that such a distinction is also present in the SLOT-RESERVOIR subsystem (where the bug resides), then the design of the sensor is simplified, since a single complex sensor which must distinguish among different denominations can then be replaced by three simple sensors which merely indicate the presence of a coin. Such a simplification provides evidence that the assumption is plausible. But it should be clear that deducing this consequence requires fairly sophisticated knowledge about the desirability of and means for accomplishing complex sensing by separating the functions of discrimination and detection.

Upon assuming that the machine is in fact designed to make such a discrimination in the SLOT-RESERVOIR subsystem, then we can extend our model, using an additional SORT component capable of routing distinct items into distinct paths:

```
                                         mtrans
                              COUNTER --------> DISPENSER
                                    ↑ mtrans
                   quarters        |
                      +---- SENSOR ---- to RESERVOIR
                     /
       ptrans       / dimes     ↑ to COUNTER
SLOT ---------- SORT ----- SENSOR ---- to RESERVOIR
                     \
                      \ nickels  ↑ to COUNTER
                       +---- SENSOR ---- to RESERVOIR
```

Another aspect of the buggy behavior is the fact that quarters come out of the machine at the coin return slot. Disregarding (for the moment) the *explanation* for this, the mere fact of it implies that there is a deficiency in the model shown above: there is no path for the quarters to travel between the SLOT-RESERVOIR subsystem and the coin return. This requires a PTRANS link from some point in the SLOT-RESERVOIR subsystem to the coin return. This link must connect with the SLOT-RESERVOIR subsystem before the quarter sensor, since otherwise quarters would be counted. Furthermore, using the single fault assumption we can infer that whatever is wrong with the quarter sensor is precluding quarters from taking the normal path through it, forcing them instead to take this alternative path to the coin return. The obvious explanation, of course, is that something is *stuck* in the quarter sensor. Exactly why this is obvious is not clear. We can only conclude that there exists a BLOCKED PATH memory structure which has been indexed by the problem description "a problem ahead in a path is precluding something from continuing along that path," and which indicates what the problem might be depending on the nature of the path. In particular, for physical paths, the BLOCKED PATH structure must indicate that the problem might be that something is stuck in the path.

At this point, we have a complete explanation of why the vending machine is rejecting and ignoring quarters. However, the explanation has not yet been adequately tied to the teleology of the machine, since we do not yet know the purpose of the alternative path the quarters are taking to the coin return. Our explanation of this particular aspect of the buggy behavior will not be complete until we have satisfied the constraint of teleological justification.

In this case, the particular purpose which I inferred was suggested by knowledge about the SORT component of the model. Often when sorting items, one wants an "other" category to take care of objects which have failed the sort. This would be a useful for a vending machine to handle the probability that objects other than the appropriate coins will be deposited; the action of routing such things to the coin return would serve the purposes of removing unwanted items from the machine, and returning them to their owners. Thus our explanation of the first bug meets the teleological constraint.

We turn now to the second bug, which is that the machine is dispensing Cokes for only 30 cents, rather than the usual 55 cents. From the single fault assumption, we already determined that the problem was located at the quarter sensor. Furthermore, the BLOCKED PATH memory structure enabled the additional conclusion that something was stuck in the quarter sensor, in order to explain the first bug. If, again invoking the single fault assumption, we assume that the rest of the SENSOR-COUNTER subsystem functions properly, the problem must be that the quarter sensor is reporting that a quarter has been collected which has in fact not been collected.

The question then is what is causing this to happen? The obvious answer for what might cause a quarter sensor to report a quarter is: a quarter. In this case it would have to be a quarter which was not in fact collected. Since we have already concluded that the problem is something stuck in the quarter sensor, we can now conclude that what is stuck in the sensor is in fact a quarter, and *this* is the quarter which the machine counts once towards the price of each Coke dispensed.

We are now almost finished with respect to the two issues which have driven our construction of the model: each part in our model is teleologically justified, and we have explanations of the bugs which people generally deem adequate. In fact, the mechanisms and explanations match quite closely what we set out to produce.

Step four: Building the machine

Given that we are satisfied with our model's explanatory and teleological characteristics, one question remains: can we build it? That is, can we think of actual physical mechanisms which will do the jobs we have assigned to the various components?

Some of the implementations are straightforward: the PTRANS connections can be viewed as tracks down which the coins roll or slide. The SORT process can be implemented by attaching chutes of various sizes off of a main channel, arranged smallest to largest, such that the coins fall into the chute matching their size. Most of the protocol subjects imagined this kind of sorting device, and most got the idea from remembering children's piggy banks which sort coins in this fashion. Thus the SORT component allows us to index this memory, given that coins are the items to be sorted. Other components, such as the counter, were harder.

The real significance of the attempt to actually "construct" the model, however, lies in the fact that conjectures about the physical implementations of device components not only raise additional demands to elaborate the model (when it seems hard or impossible to implement it as is), they also suggest possible elaborations of the model through fortuitous remindings of additional uses a device component may serve, once it has already been proposed for a different use.

Conclusion

Let us consider again the important points made about the process of explanation in this domain:

1. Teleological knowledge must be transformed into a device model consisting of teleologically-defined components connected by the appropriate causal links.
2. Domain-independent knowledge of debugging must be used to locate the source of the problem in the model.
3. The model must be sufficiently elaborated to explain all observed features of the bugs.
4. All components added to the model in order to explain the buggy behavior must be justified teleologically.
5. The teleological components of the model must, ultimately, be replaced with "real" components -- e.g., relays and wheels -- as a demonstration that the teleological model is implementable.
6. Knowledge about real components in memory must be indexed by the purposes with which they have been associated by experience; conversely, given a real component, one must be able to access knowledge about its possible purposes.

Two striking facts seem to emerge from this analysis. First, it is surprising how elaborate and detailed a model can be constructed in a situation where so little is initially known. When teleology and observed bugs are treated as twin constraints on the process of model building, complex and fairly convincing models can be constructed given only a rudimentary knowledge of device components. Second, given the simple representation constructed in step one, plus the single fault assumption, it seems difficult *not* to conclude that the problem lies in the quarter sensor, which is the cornerstone of the ultimate explanation. However, many people failed to reach this conclusion, despite the fact that they at least considered using the single fault assumption in attacking the problem. This argues that quite a bit of power lies in the simple representation scheme illustrated here.

Acknowledgments: I thank Larry Birnbaum for useful discussions and for comments on an earlier draft of this paper.

References

de Kleer, J. and Brown, J.S. 1981. Mental Models of Physical Systems and their Aquisitition. In J. Anderson (eds), Cognitive Skills and their Aquisition, Erlbaum.

Forbus, K., and Stevens, A. 1981 Using Qualitative Simulation to Generate Explanations. Technical Report 4490, Bolt Beranek and Newman.

McDermott, D.V. 1976. Flexibility and Efficiency in a Computer Program for Designing Circuits. Technical Report TR-402, Massachusetts Institute of Technology.

Schank, R.C. and Abelson, R. 1977. Scripts, Plans, Goals and Understanding. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Schank, R.C. and Collins, G.C. 1982. Looking at Learning. In Proceedings of the ECAI, pages 10-16. ECAI, Paris, France.

Simon, H.A. 1969. The Sciences of the Artificial. The M.I.T. Press.

Stevens, A. and Collins, A. 1978. Multiple Conceptual Models of a Complex System. Technical Report 3923, Bolt Beranek and Newman.

Sussman, G.J. and Brown, A.L. 1974. Localization of Failures in Radio Circuits: A Study in Causal and Teleological Reasoning. Technical Report AIM-319, Massachusetts Institute of Technology.

Step four: Building the machine

Given that we are satisfied with our model's explanatory and teleological characteristics, one question remains: can we build it? That is, can we think of actual physical mechanisms which will do the jobs we have assigned to the various components?

Some of the implementations are straightforward: the PTRANS connections can be viewed as tracks down which the coins roll or slide. The SORT process can be implemented by attaching chutes of various sizes off of a main channel, arranged smallest to largest, such that the coins fall into the chute matching their size. Most of the protocol subjects imagined this kind of sorting device, and most got the idea from remembering children's piggy banks which sort coins in this fashion. Thus the SORT component allows us to index this memory, given that coins are the items to be sorted. Other components, such as the counter, were harder.

The real significance of the attempt to actually "construct" the model, however, lies in the fact that conjectures about the physical implementations of device components not only raise additional demands to elaborate the model (when it seems hard or impossible to implement it as is), they also suggest possible elaborations of the model through fortuitous remindings of additional uses a device component may serve, once it has already been proposed for a different use.

Conclusion

Let us consider again the important points made about the process of explanation in this domain:

1. Teleological knowledge must be transformed into a device model consisting of teleologically-defined components connected by the appropriate causal links.
2. Domain-independent knowledge of debugging must be used to locate the source of the problem in the model.
3. The model must be sufficiently elaborated to explain all observed features of the bugs.
4. All components added to the model in order to explain the buggy behavior must be justified teleologically.
5. The teleological components of the model must, ultimately, be replaced with "real" components -- e.g., relays and wheels -- as a demonstration that the teleological model is implementable.
6. Knowledge about real components in memory must be indexed by the purposes with which they have been associated by experience; conversely, given a real component, one must be able to access knowledge about its possible purposes.

Two striking facts seem to emerge from this analysis. First, it is surprising how elaborate and detailed a model can be constructed in a situation where so little is initially known. When teleology and observed bugs are treated as twin constraints on the process of model building, complex and fairly convincing models can be constructed given only a rudimentary knowledge of device components. Second, given the simple representation constructed in step one, plus the single fault assumption, it seems difficult *not* to conclude that the problem lies in the quarter sensor, which is the cornerstone of the ultimate explanation. However, many people failed to reach this conclusion, despite the fact that they at least considered using the single fault assumption in attacking the problem. This argues that quite a bit of power lies in the simple representation scheme illustrated here.

Acknowledgments: I thank Larry Birnbaum for useful discussions and for comments on an earlier draft of this paper.

References

de Kleer, J. and Brown, J.S. 1981 Mental Models of Physical Systems and their Aquistition. In J. Anderson (eds), Cognitive Skills and their Aquisition, Erlbaum.

Forbus, K., and Stevens, A. 1981. Using Qualitative Simulation to Generate Explanations. Technical Report 4490, Bolt Beranek and Newman.

McDermott, D.V. 1976. Flexibility and Efficiency in a Computer Program for Designing Circuits. Technical Report TR-402, Massachusetts Institute of Technology.

Schank, R.C. and Abelson, R. 1977. Scripts, Plans, Goals and Understanding. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Schank, R.C. and Collins, G.C. 1982. Looking at Learning. In Proceedings of the ECAI, pages 10-16. ECAI, Paris, France.

Simon, H.A. 1969 The Sciences of the Artificial. The M.I.T. Press.

Stevens, A. and Collins, A. 1978. Multiple Conceptual Models of a Complex System. Technical Report 3923, Bolt Beranek and Newman

Sussman, G.J. and Brown, A.L. 1974. Localization of Failures in Radio Circuits: A Study in Causal and Teleological Reasoning. Technical Report AIM-319, Massachusetts Institute of Technology.