CHAPTER 13

Logic, Change and Contradiction

Logic can be used to represent information and to solve problems. But information changes and its representation needs to change accordingly. In this chapter we consider the processes by means of which an information system needs to change in time. The information systems considered include not only programs and databases but also more complex systems of the kind involved in scientific theories and computer-based natural language understanding. We shall consider in detail the role that contradiction plays in guiding the direction of change.

Information systems

Throughout this chapter the terminology _information system_, and sometimes _belief system_, is used to refer to any _collection of assumptions_ (or beliefs) _expressed in logic_ together with a _proof procedure_ and _maintenance procedures_, which manage the way the information system deals with change.

Information systems include both assumptions which are _explicit_ as well as consequences which are _implicit_. In practice whether a sentence is an implicit consequence is a matter of degree. The accessibility of a consequence depends upon the complexity of finding a derivation. The more complex the derivation, the more inaccessible its consequence. If a derivation is too complex, its consequence is as inaccessible as if it were not implied at all. Thus different information systems may entail the same logical consequences but differ significantly in their pragmatic value. Useful consequences may be efficiently accessible in one system but practically inaccessible in another.

Databases can be regarded as simple information systems. A database might change as the result of internal reorganisation or in response to incoming data and queries. The proof procedure is used not only to answer queries but also to assimilate new data into the database. There are four possibilities:

> The new data might already be implied by the database,
> imply existing data,
> be independent from it, or
> inconsistent with it.

It is the last case which is most important. It includes both the case in which new data violates integrity constraints as well as the one in which it is an exception to a general rule.

Programs together with their specifications can also be regarded as information systems. A program which is inconsistent with its specification can be made consistent by modifying either the program or the specification. A program which is consistent with its specification can be changed by replacing an inefficient procedure with a more efficient one. It can also be changed by adapting it to a different purpose.

In text comprehension, the information system consists of the reader's understanding of the text which has been read so far. It needs to change when new information needs to be assimilated. The new information might be the reader's interpretation of the next sentence in the text or it might be an hypothesis needed to explain information previously obtained from the text. In both cases the new information might be one among several alternatives. The new sentence might be ambiguous and admit alternative interpretations or the previous information might be explained by alternative hypotheses. If the new information is inconsistent with the current information system an alternative to the new information or to previously assimilated information may need to be considered.

Scientific theories can be interpreted as information systems which organise past experience and predict future ones. A theory may need to change in the light of new experience or as the result of a new hypothesis. An ambiguous experience can be reported in alternative ways, and alternative hypotheses might explain the same phenomena. The alternatives need to be compared by evaluating their effect on the state of the scientific theory as a whole. If an alternative renders the theory inconsistent then consistency can be restored by restricting or suitably modifying any of the premises which contribute to the contradiction. This includes both the case in which the new sentence is rejected and replaced by an alternative as well as the one in which the new sentence is accepted and an old one is rejected instead.

Dynamics of information system change

Both the situation in which an information system records its interaction with the environment and the situation in which it generates its own hypotheses result in the need to assimilate new information. There are four possible deductive relationships between the new information and the current information system. Each possibility suggests different candidates for the new system.

(1) _The new information can already be derived from the current information system._ The information system successfully anticipates the new information and the new system is the same as the old one. Assumptions which participate in the derivation can be identified and their utility can be evaluated. More generally, assumptions can be evaluated by assessing the extent to which they contribute to the derivation of useful consequences. The evaluation of assumptions according to utility can be used later to help determine which assumptions should be abandoned or modified when a contradiction occurs.

(2) _Part of the information in the current system can be derived from the new information together with the information in the rest of the_

system. The explicit assumptions of the new system consist of the new information together with the explicit assumptions of the old system without the part that can be derived. The new system subsumes the old one. It implies the same consequences and possibly new ones as well. The assessed utility of the assumptions which participate in the derivations can be increased by an amount which takes into consideration the number of derived consequences, the complexity of the derivations and the utility of the derived consequences themselves.

The simplest example is the one in which the new information is an inductive generalisation of existing information. The situation in which it is an abductive assumption [Peirce 1931] is more complicated. Suppose, for example, that the current system already contains the information

(1) A & B & C <— D
(2) A

Then the new information

D

is an abductive hypothesis. Together with (1) it implies (2). Moreover, it also implies B and C. In order to justify its incorporation into the information system, the hypothesis D may need to prove its utility. It can do so, for example, by showing that B or C is already redundantly contained in the existing database or by predicting them when they are introduced as new information later on. Generation of abductive hypotheses is similar to reasoning by means of defaults [Minsky 1975], [Reiter 1978b]. If A is given, then D is assumed by default unless it leads to contradiction or does not lead to sufficiently many useful consequences.

Notice that cases (1) and (2) might both apply. Whether one case is better than the other depends upon the overall utility of the resulting information system.

(3) The new information is consistent with the information system but is independent of it. The new information can neither be derived from the current system nor be used to derive existing information. This is potentially an undesirable situation which may lead the system to seek an explanatory hypothesis, which together with the information in the rest of the system implies the new information. Of course, the hypothesis itself would also be independent and to justify its acceptance would have to imply other useful consequences in addition to the one which motivated its generation. The preceding example illustrates the situation. Suppose the information system contains the assumption

A & B & C <— D

and the new information A is independent. If this leads the system to generate the hypothesis D, then D itself is independent and there is no net gain unless at least one of the additional consequences B or C can be independently confirmed.

It is not always possible to determine in a reasonable time whether one or other of the four deductive relationships apply. In such cases, whether the new information is logically related to the existing information system or not, it will need to be treated as independent and added to it.

(4) <u>The new information is inconsistent with the information system.</u> A contradiction can be derived when the new information is introduced. The assumptions which contribute to the refutation can be identified, and consistency can be restored by rejecting or modifying one or more of the assumptions which lead to the contradiction. The previous record of the utility of assumptions can be used to help determine which assumptions should be changed.

It is this last case, in which a contradiction occurs, which is the most important.

<u>Restoration of consistency</u>

Contradiction and its reconciliation play an important role in philosophy and in theories of problem-solving. It is the driving force behind change (thesis, antithesis and synthesis) in the Hegelian dialectic and the main instrument for advancing knowledge (conjectures and refutations [Popper 1963] and proofs and counter-examples [Lakatos 1973]) in the Popperian philosophy of science and mathematics. In problem-solving, it is an advanced form of intelligent backtracking and an important component of truth maintenance problem-solving systems [Doyle 1978], [Stallman and Sussman 1977].

It is a major feature of Quine's [1953] argument against the distinction between necessary and contingent truths that, when a contradiction arises, consistency can be restored by rejecting or modifying any assumption which contributes to the derivation of contradiction. No belief is immune from possible alteration. Even the laws of mathematics and logic, to the extent that they are included among the assumptions of information systems, are subject to critical assessment and change.

This does not mean that any belief can be altered as easily as any other. Psychological attachment and even computational commitment may vary from one belief to another. Nor is it pragmatically desirable to treat different beliefs the same. Some contribute to the derivation of useful consequences more often than others; and some participate more often in the derivation of contradictions. It benefits the well-functioning of the belief system as a whole, therefore, to abandon, among the beliefs which lead to contradiction, the one which contributes least to the derivation of useful consequences. In the longer term, if contradictions continue and the assessed utility of beliefs changes, it may be necessary to backtrack, reinstate a previously abandoned belief and abandon an alternative instead.

Thus the derivation of inconsistency contributes to the search space of alternative information systems. For each assumption which contributes to the derivation of a contradiction there exists at least one alternative new belief system obtained by abandoning or suitably modifying the assumption. The space can be searched in depth-first fashion, backtracking when a contradiction arises, or several branches can be investigated in parallel. Parallel exploration of alternatives has the advantage that the consequences of abandoning a belief can be explored before a decision is made. Such parallel exploration of several internally consistent, but mutually inconsistent, belief systems may, of course, give an external observer the illusion of a single inconsistent system.

The derivation of inconsistencies plays an important role in the development of computer programs and databases. Generally, when an inconsistency arises between a program and its specification or between data and integrity constraints, it is the program or the data which is rejected. Indeed, by definition, it is a main function of specifications and integrity constraints to rule out incorrect programs and data. None the less there are frequent occasions when it is necessary to abandon or modify the specification or integrity constraint instead. For example, given the conflict which arises between the integrity constraint

> No vehicles are allowed in the park.

and the need for police and other emergency services to have access to the park, it is likely that preference will be given to the police and that the integrity constraint will have to be modified instead:

> No unauthorised vehicles are allowed in the park.

Preference is also given to incoming data when it is treated as an exception to general rules. Early versions of a university department's timetable, for example, might be described by ambitiously general rules:

> All first year lectures are held in room 144.
> All lectures attended by more than
> 80 students are held in room 145.

Subsequent additions to the database

> The first year logic lectures
> are attended by 100 students.

might result in contradiction. Consistency can be restored by treating the new data as an exception to a general rule, replacing the original rule by a more restricted one

> All first year lectures, except
> logic, are held in room 144.

Notice in this last example that the assumption which has been modified is not necessarily the one which has been least useful in the past. What matters in general is not simply the utility of a belief but rather the difference between its utility and that of its replacement. Treating new data as an exception to a general rule when a contradiction arises has the advantage of avoiding the contradiction while preserving most of the useful consequences of the existing information system.

Contradiction also plays an important role in text comprehension. It helps to disambiguate sentences by rejecting interpretations which are inconsistent with the current interpretation of the text-so-far, and it helps to reject inconsistent explanatory hypotheses. If all interpretations of a new sentence lead to contradiction, the system may attempt to restore consistency by altering a previous hypothesis or an interpretation of a previous sentence instead.

Perhaps the classical example in which an information system needs to cope with contradiction is the case in which the report of an empirical observation or experiment contradicts a scientific theory. Whether it is more beneficial to reject the report or a statement of the theory depends on the overall effect on the information system. It is even possible that several alternatives might lead to incomparable, equally viable, but mutually incompatible, theories.

As Lakatos [1974] argues, in a mature theory with a history of useful consequences it is generally more useful to reject an anomalous conflicting report than it is to abandon the theory as a whole.

But it is almost never the case that a whole theory needs to be abandoned anyway. A complex information system is a collection of cooperating individual beliefs, some of which are more useful and more firmly held than others. Propositions which reside in the central core of a theory are more firmly held than those which are located closer to the periphery, where rival hypotheses may coexist as mutually incompatible alternatives. Reports of empirical observations can help to accumulate evidence in favour of one alternative over another.

Even without restoring consistency, an inconsistent system can still organise useful information. Although in theory inconsistent assumptions imply any conclusion, in practice efficient proof procedures derive only relevant conclusions with varying degrees of accessibility. Indeed, it can be argued that practical provability, acheived by efficient resolution-based proof procedures, satisfies all of the criteria necessary for relevant entailment [Anderson and Belnap 1962].

Thus contradiction, far from harming an information system, helps to indicate areas in which it can be improved. It facilitates the development of systems by successive approximation - daring conjectures followed by refutation and reconciliation. It favours bold, easily falsified beliefs, which can be weakened if the need should arise, over safe, timid beliefs, which are difficult to strenthen later on. Better to make mistakes and to correct them than to make no progress at all.

A logic program for natural language

As a test of the theory of information systems outlined in this chapter, a logic program for managing a natural language front-end to a logic database has been designed by the author with Jaqueline Shane and Karen Ritchie. A pilot version is being implemented using a theorem-prover for the standard form of logic written by Krysia Broda.

The top-level of the program

```
Process(x,y,z,x')
```

starting with an initial logic database x, processes a list y of natural language input sentences, producing a correlated list z of output sentences, finishing with a new database x' at the end of the session.

```
              Process (db, nil, nil, db) <-
              Process (db, input.restin, output.restout, newdb) <-
                      Represents (input, logic, control),
                      Assimilate (db, logic, control, output, interdb),
                      Process (interdb, restin, restout, newdb)
```

Here as in the previous chapter, lower case character strings (e.g. "db", "input", "restin") are variables.

Represents(input, logic, control) holds when the natural language input can be interpreted as consisting of a logic statement together with a control component.

Assimilate(db, logic, control, output, interdb) holds when assimilating the logic statement and associated control into the logic database results in an appropriate output and a new intermediate database.

At the simplest level, control simply indicates whether a sentence is a declarative statement or a question. Here clause (1) deals with the case that the input is a question. The result of attempting to answer the question may or may not be a proof. (2) deals with the case that the input is a declarative sentence already implicitly contained in the database. In both cases, (1) and (2), assimilation of the new information does not change the database. In the case A3, the next database consists of the new information together with part (stay) of the existing database. The new database implies all the data in the part (go) of the old database which is no longer explicitly contained in the new database. A4 adds the new information to the database if it cannot be derived or be used to derive existing information. A5 deals with the case in which the new information is inconsistent with the current database. The new database results from analysing the proof of contradiction and restoring consistency.

```
A1  Assimilate(db,logic,control,output,db) <- Question(control),
                              Demonstrate(db,logic,control,result),
                              ExtractOutput(result,output)

A2  Assimilate(db,logic,control,output,db) <- Declarative(control),
                              Demonstrate(db,logic,control,result),
                              Proof(result), IAlreadyKnowThat(output)

A3  Assimilate(db,logic,control,output,nextdb) <- Declarative(control),
              db = stay ∪ go,
              nextdb = stay ∪ {logic},
              ∀data[data ∈ go ->
                   ∃result[Demonstrate(nextdb,data,control,result) &
                         Proof(result)]],
              ThanksForTellingMe(output)

A4  Assimilate(db,logic,control,output,nextdb) <- Declarative(control),
                              Independent(db,logic,control),
                              nextdb = db ∪ {logic},
                              Acknowledge(output)
```

```
A5  Assimilate(db,logic,control,output,nextdb) <— Declarative(control),
                                  incon = db ∪ {logic},
                                  Demonstrate(incon,□,control,result),
                                  Proof(result),
          AnalyseFailureRestoreConsistency(incon,result,output,nextdb)
```

This is only a top-level sketch of part of the natural language program. Important lower level procedures need to be defined and specifications, such as A3, need to be transformed into efficient procedures.

Our intention has been to deal with a restricted subset of natural language suitable for untrained database users. However we do not insist that input sentences be completely unambiguous. Certain ambiguities can be dealt with by allowing Represents to be non-deterministic$_1$; others, such as those resulting from anaphora ("he","she","it",etc.), by adding extra parameters to the Represents relation in order to deal with the context of the previous natural language input.

For users interacting with a database it can be required that all information included in the database be described explicitly. Implicit assumptions, however, cannot be avoided in normal conversation and text comprehension, where hypothesis generation schemes, such as frames [Minsky 1975] and scripts [Schank 1975] are needed to fit sentences into a coherent framework. The natural language program can be extended, in theory at least, to accommodate the abductive generation of assumptions by adding extra procedures. Here, in the case that the input is independent from the existing database, clause A6 generates and adds to the database a new assumption which together with the rest of the database implies the new information. To be worth the effort, the new information must be sufficiently more useful than the incoming information itself.

```
A6  Assimilate(db,logic,control,output,nextdb) <— Declarative(control),
                                 Independent(db,logic,control),
                                 nextdb = db ∪ {newassump},
                                 Demonstrate(nextdb,logic,control,result),
                                 Proof(result),
                                 newassump is more useful in db than logic,
                                 Iassume(newassump,output)
```

Conclusion

The theory of information systems attempts to combine the traditional role which logic plays in epistemology and the philosophy of science with its new role in computing. It attempts to reconcile the use of logic without computational considerations with the use of complex, computer-based computational systems without logical foundations. By exploiting the computational interpretation of logic, it hopes to contribute to a more useful communication of techniques between logic and computing.