# 1

# Fundamental Logic and the Definition of Engineering Tasks



***What there is to learn:*** *For many important engineering tasks computers should provide choices, not answers. The type of engineering task will determine important aspects of computer support. The 'open-world' nature of engineering is a difficult challenge.*

The success of computer software in engineering rarely depends upon the performance of the hardware and the system-support software. Computer applications have to match engineering needs. Although the principal objective of this book is to introduce fundamental concepts of computing that are relevant to engineering tasks, *it is essential to introduce first the fundamental concepts of engineering that are relevant to computing*. This is the objective of this first chapter, and key concepts are described in the following sections.

## 1.1 Three Types of Inference

From the perspective of fundamental logic, engineering reasoning is categorized into the following three types of inference: deduction, abduction and induction. The best way to understand inference types is through an example. Consider a bar of length $L$ and area $A$ that is fixed at one end and loaded at the other with a load $Q$ (Figure 1.1).

From fundamental solid mechanics and assuming elastic behaviour, when a load $Q$ is applied to a prismatic bar having a cross-sectional area $A$, this bar deforms axially (elongates) by an amount $\delta = QL/AE$ where $E$ is the Young's modulus of the material. Thus, for a given bar, we have the following items of information:

- two facts: $Q$ and $\delta$
- one causal rule: if $Q$ then $\delta$ (a causal rule has the form, if *cause* then *effect*)

The difference between deduction, abduction and induction originates from what is given and what is inferred. *Deduction* begins with one fact $Q$ and the causal rule. With these two items as given information, we infer that a deformation $\delta$ occurs. *Abduction* begins with the effect $\delta$ and the causal rule. Starting with these two items, we infer that a load $Q$

*Engineering Informatics: Fundamentals of Computer-Aided Engineering*, Second Edition.
Benny Raphael and Ian F. C. Smith.

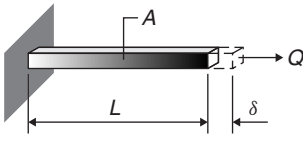


**Figure 1.1** An axially loaded bar

**Table 1.1** Three types of inference

| Type of inference | Given information | Inferred information |
|---|---|---|
| Deduction | Cause and rule | Effect |
| Abduction | Effect and rule | Cause |
| Induction | Cause and effect | Rule |

is present. *Induction* begins with the two facts, $Q$ and $\delta$, and then infers the causal rule. Table 1.1 summarizes the important elements of these three types of inference.

Logically, deduction is the only legal inference type in an open world. The use of abduction and induction requires a closed-world hypothesis. A closed world hypothesis involves the assumption that all facts and all rules are known for a given task. Such hypotheses are typically found at the beginning of mathematical proofs using statements such as 'for all x ...' and 'y is an element of ...'.

For illustration, refer to the example of the loaded bar. We can state with some confidence that if there is a load $Q$ and when the causal rule, *if* $Q$ *then* $\delta$, is known, we can infer that a deformation $\delta$ is expected (deduction). However, when we know the rule and only that a deformation $\delta$ is observed, we cannot necessarily say in an open world that this deformation was caused by a load $Q$ (abduction). For example, the deformation may have been caused by an increase in temperature. Similarly, just because we observe that there is a load $Q$ and a deformation $\delta$, we cannot necessarily formulate the rule, *if* $Q$ *then* $\delta$ (induction). Other facts and contextual information may justify the formulation of another rule. Abduction and induction become valid inferences for the example on hand only when we make the closed-world hypothesis that the only relevant items of information are the two facts and one rule.

Scientists and mathematicians frequently restrict themselves to closed worlds in order to identify in a formal manner kernel ideas related to particular phenomena. Also, the closed-world idea is not limited to research. Job definitions in factories, offices and many other places often only require decision-making based on closed, well-defined worlds. However, engineers (as well as other professionals such as doctors and lawyers) are not allowed to operate only in closed worlds. Society expects engineers to identify creative solutions in open worlds. For example, engineers consider social, economic and political aspects of most of their key tasks before reaching decisions. Such aspects are difficult to define formally. Moreover, attempts to specify them explicitly, even partially, are hindered by environments where conditions change frequently.

Therefore, engineers must proceed carefully when they perform abductive and inductive tasks. Since a closed-world hypothesis is not possible for most of these tasks, any supporting software needs to account for this. For example, engineering software that supports abductive tasks has to allow for frequent user interaction in order to accommodate open-world information. Also, providing good choices rather than single answers is more attractive to engineers. Differences in 'world hypotheses' provide the basic justification for variations between requirements of traditional computer software, such as those applied to book-keeping, and those applied to certain tasks in engineering. The next section will expand on this theme by providing a classification of engineering tasks according to their type of inference.

## 1.2 Engineering Tasks

Important engineering tasks are simulation, diagnosis, synthesis and interpretation of information. All of these tasks transform one type of information to another through inference. Some more terminology is needed. In this discussion the word *structure* refers to information that is commonly represented in drawings, such as dimensions, location, topography and other spatial information, as well as properties of materials and environmental influences such as temperature, humidity and salinity of the air. This category also includes the magnitude, direction and position of all loading that is expected to be applied to the object. The word *behaviour* is used to include parameters that describe how the structure reacts, such as deformations, stresses, buckling, shrinkage, creep and corrosion.

In *simulation*, causes are applied to particular structures in order to observe effects (or more precisely, behaviour). For example, a factory process is simulated to observe output and other aspects, such as sensitivity to breakdown of a machine. *Analysis* is a special case of simulation. Analysis is performed when behavioural parameters are required for a given physical configuration in a particular environment (the word structure is used loosely to define this information). For example, bridges are analysed for various loading (such as wind, truck and earthquake) in order to determine behaviour that is expressed in terms of stresses and deflections. These transformations are summarized in Figure 1.2.

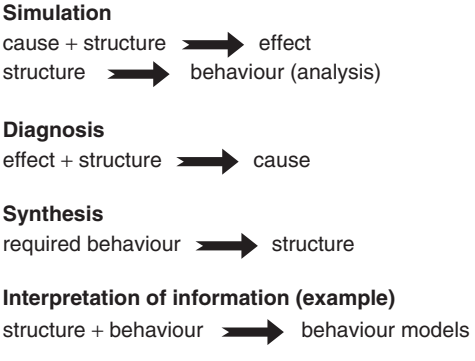


**Figure 1.2** Four common engineering tasks. The arrows indicate transformation of information through inference

*Diagnosis* can be thought of as the reverse of simulation. For this task, observed effects are considered with the physical configuration and the environment to infer causes. For example, maintenance engineers observe the vibrations of motors in order to identify the most likely causes. Similarly, from an information viewpoint, *synthesis* is the reverse of analysis, where target behaviour is used to infer a physical configuration within an environment. For example, engineers design building structures to resist many types of loading in such a way that stresses and deflections do not exceed certain values.

*Interpretation of information* includes a wide range of engineering activities. For example, interpretation of information refers to tasks that infer knowledge from activities such as load tests, commissioning procedures and laboratory experiments. Test results in the form of values of behavioural parameters are combined with known structure to infer models of behaviour. The form of the model may already be known; in such cases, testing is used for model calibration. For example, engineers test aircraft components in laboratories to improve knowledge of strength against fatigue cracking. These results are often used to improve models that predict behaviour of other components. Although such models are usually expressed in terms of formulae, they are transformable into causal rules, as in the example in Figure 1.1.

Each arrow in Figure 1.2 refers to an inference process where the left-hand side is the input, and the right-hand side is the output. Reviewing the definitions for deduction, abduction and induction that are given in Section 1.1, a classification of engineering tasks in terms of inference types becomes possible. Since deduction begins with known causes and causal rules to infer effects, analysis and simulation are examples of deduction. Abduction begins with known effects and causal rules to infer causes; therefore, diagnosis and synthesis are abductive inferences. Interpretation of information begins with known causes and known effects to infer causal rules (models of behaviour), therefore this task is an example of induction. Results are summarized in Table 1.2.

Therefore, important engineering tasks of diagnosis, synthesis and interpretation of information involve logical inferences that are not reliable in open worlds (abduction and induction). This result has a very important impact on requirements for software that is intended to support such tasks. This classification also provides clues to an explanation for the early success of analysis and simulation software. Nevertheless, it can be argued that engineering tasks that provide the greatest added value to society are mostly abductive and inductive. Therefore, the potential gain with appropriate computer-aided engineering support is correspondingly greater as well. It is 'just' harder to do things correctly.

**Table 1.2** Engineering tasks classified into inference types

| Engineering task | Inference type |
|---|---|
| Analysis | Deduction |
| Simulation | Deduction |
| Diagnosis | Abduction |
| Synthesis | Abduction |
| Interpretation of information | Induction |

Indeed, failure to recognize logical differences between engineering tasks has been a central reason for failures of software such as early artificial intelligence applications (expert systems) proposed during the last quarter of the twentieth century (see Chapter 7 for further discussion). More generally, software applications that support abduction and induction in open worlds are much more sensitive to changing knowledge and contexts. Therefore, maintaining such software over long periods is more difficult than software that supports deduction. The open-world nature of engineering presents important challenges to those who wish to use computers to support abductive and inductive tasks.

## 1.3 A Model of Information and Tasks

In Section 1.2 we explained that inference processes transform information from one type, such as structural information, to another type, such as behavioural information. How many types of information are there? What is the best way to organize engineering information so that we can obtain the most support from computer-aided engineering (CAE)?

These questions have been the subject of much discussion. A framework that has been thoroughly discussed by CAE researchers and has stood the test of time is the function–behaviour–structure schema. In this schema, information is divided into the three categories that form its name. Information in the *function* category includes functional requirements such as design criteria, specifications, desires of society and purpose. The categories of *behaviour* and *structure* have been defined in the previous section.

The transformations associated with these categories of information have been collected into a schema that was initially proposed by Gero (1990). For the purpose of describing this schema systematically, a design example is used in the following discussion. Subsequent discussion demonstrates the importance of this schema for a range of engineering activities.

At the beginning of a design activity, engineers identify functional requirements for future objects through contacts with clients and meetings with other people that could influence design decisions. Engineers then transform these requirements into required behaviour. For example, the functional requirement of a building to provide safe shelter for its occupants leads to behaviour requirements related to strength and stability. Behaviour requirements are different from functional requirements because behavioural parameters are explicitly mentioned. Other requirements for occupant comfort lead to serviceability requirements such as maximum beam deflections. This transformation is a task called formulation and it is shown schematically in Figure 1.3.

In the second stage of design, parameters associated with the required behaviour, along with environmental parameters such as loading, are used to generate spatial information within the structure category. For example, the foundations and load-carrying elements of the building are chosen and sized during this stage. As introduced in the previous

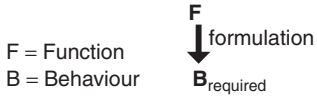


**Figure 1.3** Formulation

section, this transformation is a task called synthesis and this is described schematically in Figure 1.4.

Once the as-designed structural information has been completed, the structure is then analysed in order to obtain predicted behaviour. In the building example, the structural configuration would be analysed to determine lateral deflections.[1] Such predicted behaviour is then compared during an evaluation with required behaviour. Analysis and comparison tasks are included in Figure 1.5.

Results of the evaluation may lead to a new formulation or to new synthesis tasks and new analyses. These additional tasks are particularly likely if the evaluation reveals that predicted behaviour is unsatisfactory. For example, if predicted building deflections exceed the maximum deflections fixed during formulation, designers may reformulate the required behaviour, redo the synthesis so that the building is stiffened, or rework the analysis to provide a more accurate prediction. These three actions are often not equivalent in their

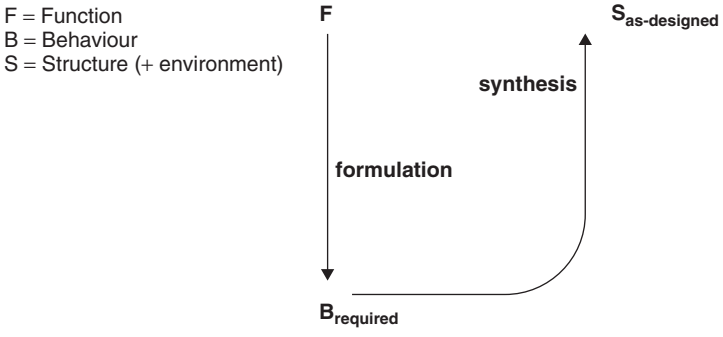


**Figure 1.4** Synthesis added to Figure 1.3

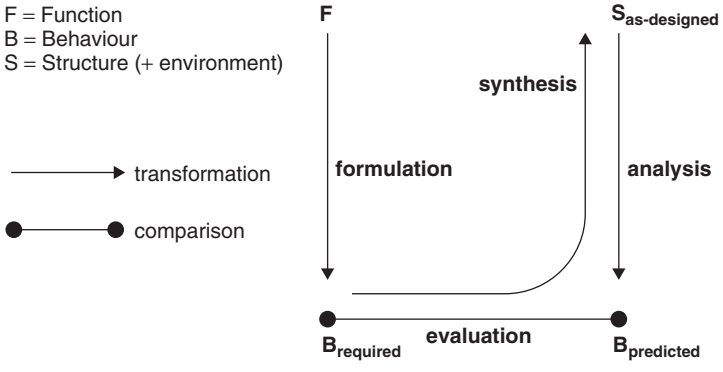


**Figure 1.5** Analysis and evaluation

[1] There is often more than one synthesis–analysis cycle. For example, once the element layout is determined and section properties are estimated for a building in a synthesis task, this may be analysed to determine bending moments which are then used to fix the cross-sectional dimensions of elements. This information is then used to re-analyse the structure to obtain values for additional behavioural parameters such as deflections.

ease of application and their effectiveness. For example, if the formulated requirements are parts of legal documents (such as building codes), reformulating required behaviour is not possible. Also, reworking the analysis may not provide the necessary correction to the calculations. The best course of action is usually to redo the synthesis step. If the designer is satisfied with the results of all the evaluations, the construction task is begun. Adding this to the schema results in Figure 1.6.

At this point in the process, engineers are often motivated to question the applicability of two general assumptions. The first assumption is that the as-built structure is the same as the as-designed structure. Since structural information in this schema includes information related to the environment, including loading, such an assumption is notoriously risky. The second assumption is that the results of the analysis task are sufficiently accurate for the comparison with required behaviour. Accuracy implies other, more specific assumptions related to material properties (for example, homogeneity, isotropy, linear elastic behaviour, and so on), deflections, and boundary conditions such as perfect pins and absolute fixity.

Often these two general assumptions are not acceptable. For example, engineering products are often not built exactly as they are designed; real loading may be very different from design loading; material may be non-homogeneous; deflections could lead to geometric non-linearities; finally, engineers know that all connections between elements are neither perfectly rigid nor perfectly free. Indeed, the evaluation task in Figure 1.6 may seem dangerously far away from the as-built structural information in many practical situations.

When too much doubt surrounding these assumptions exists and when engineers determine that such uncertainties may have an important influence on subsequent decisions, the structure might be monitored. The monitoring task provides a third type of behaviour, measured behaviour, and this is compared with predictions from analysis to improve models as well as the accuracy of values of parameters that quantify environmental effects. Finally, more accurate structural information may be used to improve predictions of structural behaviour. These tasks are described in Figure 1.7.

Opportunities for increasing the details related to required behaviour, such as more specific constraints related to functional requirements, arise when predicted values of behaviour are more reliable. Such activities are often referred to as performance-based engineering.

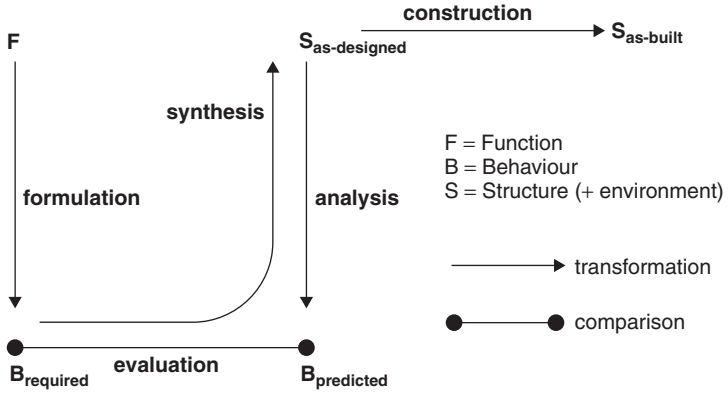


**Figure 1.6** Adding the construction task

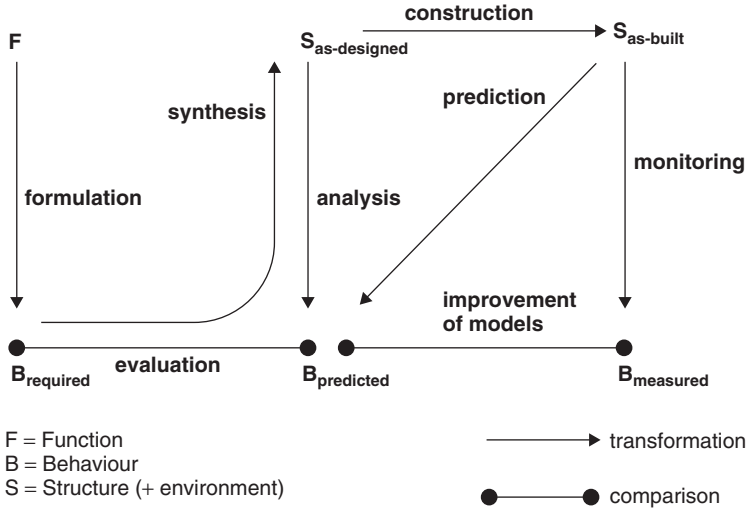


**Figure 1.7** Addition of monitoring and prediction transformation tasks as well as an additional comparison task to improve models

## 1.4 Another Task Definition

In 1972, Newell and Simon introduced another classification of tasks that accounts for how well tasks were defined.[2] Well-defined tasks are:

- carried out in closed worlds
- defined by goals explicitly and unambiguously expressed using, for example, a mathematical formula (this may be used as an objective function, see Chapter 8)
- described by algorithms that lead to identification of solutions.

Many engineering tasks are decomposed into well-defined tasks in order to simplify matters. This strategy is a common engineering tactic. When applied judiciously, reasonable solutions to complex tasks are possible. However, when simplification is not possible, engineering tasks are usually poorly defined. Poorly-defined tasks occur when at least one of the following aspects is important to the nature of the solution:

- open-world conditions prevail
- goals are defined only partially
- definitions of solutions and their forms are incomplete
- procedures for obtaining solutions are not known completely.

In reality, most engineering tasks are poorly defined. Accommodating such conditions while being able to perform useful tasks is the job of engineers. Requirements for software

[2] Newell used the terms 'well structured' and 'ill structured' to classify tasks. For the purposes of avoiding ambiguity in terminology, the word 'structured' is replaced by the word 'defined' in this text. Also, the word 'ill' is replaced by the word 'poorly' for clarity.

that support engineering tasks are subsequently more challenging for poorly-defined tasks when compared with well-defined tasks.

## 1.5 The Five Orders of Ignorance

When tasks begin, engineers are often not able to appreciate every subtlety associated with aspects such as requirements, interdependencies, goals and contextual information. Armour (2000) believed that the core value-added activities are those of acquiring knowledge and reducing ignorance. While much discussion has centred on acquiring knowledge, little attention has been given to its corollary, reducing ignorance. Armour proposed a classification of ignorance into the following five orders:

- $0^{th}$- order ignorance (0OI) – *lack of ignorance*. This means that answers are available and skills for completing tasks exist. For example, most engineers know how to find the roots of a polynomial equation in one variable.
- $1^{st}$- order ignorance (1OI) – *lack of knowledge*. This shortfall is exactly quantifiable, tasks required to overcome it are known, and skills exist to complete the tasks. For example, engineers who have learned one programming language can easily learn another and thereby satisfy their 1OI related to the new language.
- $2^{nd}$- order ignorance (2OI) – *lack of awareness*. This is when it is not known what is not known. For example, having 1OI and not knowing it is 2OI. Some engineers, thinking that computers are just big calculators, suffer from 2OI regarding the topics discussed in this book. They are not aware that computing is a science and not a skill.
- $3^{rd}$- order ignorance (3OI) – *lack of process*. This order occurs when it is not known how to go about finding out what is not known. For example, there are people who do not realize that going to university is the best way to find out what is not known (2OI) about important aspects of engineering knowledge and methodologies.
- $4^{th}$- order ignorance (4OI) – *meta ignorance*. 4OI occurs when one is unaware of the other four orders of ignorance. If you have read and understood this far, then you no longer suffer from 4OI.

The first four orders have a generality that exceeds the boundaries of computer-aided engineering. Nevertheless, they are particularly pertinent for developing and using computer-aided engineering software. Engineers need mechanisms to identify and organize knowledge that is important for the task. Now that 4OI is no longer present, the goal of this book is to provide readers with capabilities to overcome 3OI, 2OI and 1OI when confronted with tasks that involve computers in engineering.

## 1.6 Summary

- Inference is the basis of engineering reasoning.
- There are three types of inference: deduction, abduction and induction.
- Use of conclusions from abduction and induction implicitly requires a closed-world assumption.
- Engineers work in open worlds.

- In open worlds, software that provides choices and not just a single answer is often the best for abductive and inductive tasks.
- Engineers deal with three categories of information: function, behaviour and structure.
- Many important engineering tasks involve the transformation of this information from one category to another.
- Although engineers are required to carry out both well-defined and poorly-defined tasks, the added value to society is usually greater when poorly-defined tasks are successfully completed than when well-defined tasks are successfully completed.
- Development of engineering software is a process of information design and construction. This process involves setting up mechanisms where developers become able to discover the existence of information.

## Exercises

1.1 Find an example of (a) deduction, (b) abduction and (c) induction in engineering.

1.2 For the example in Exercise 1.1b, formulate another causal rule that would make abduction ambiguous.

1.3 Identify two engineering tasks that would be hard to carry out with an open-world hypothesis.

1.4 Should a computer give only one answer when supporting the engineering tasks identified in Exercise 1.3?

1.5 In Figure 1.7, which tasks are deductive, abductive and inductive?

1.6 Name three tasks that are well defined.

1.7 Name three tasks that are poorly defined.

1.8 Which tasks add the most value to society when performed successfully: well-defined or poorly-defined tasks?

## References

Armour, P.G. 2000. The five orders of ignorance. *Communications of the ACM*, **43**, 10, 17–20.
Gero, J.S. 1990. Design prototypes: a knowledge representation schema for design. *AI Magazine*, **11**, 4, 26–48.
Newell, A. and Simon, H. 1972. *Human Problem Solving*. Englewood Cliffs, NJ: Prentice Hall.