Chapter 1
Model Building

Val Lowndes, Stuart Berry, Marcello Trovati and Amanda Whitbrook

Section 1.1 introduces the use of system dynamics in modelling and then uses this approach to construct models to describe real applications.
Section 1.2 introduces the concepts needed to construct models using available data, modelling using Big Data.
Section 1.3 introduces modelling using blackboard architecture; this provides a flexible, symbolic artificial intelligence (AI) method for the cooperative modelling and then solution of complex problems.

1.1 Introduction to System Modelling

The purpose of system dynamics modelling is to develop understanding and then the improvement of systems. The first stage in this process is the construction of a logical model (influence diagram) to describe a system.

V. Lowndes (Retired)
University of Derby, Kedleston Road, DE22 1GB Derby, UK
e-mail: v.p.lowndes@derby.ac.uk

S. Berry (✉) · A. Whitbrook
College of Engineering and Technology, University of Derby,
Kedleston Road, DE22 1GB Derby, UK
e-mail: s.berry@derby.ac.uk

A. Whitbrook
e-mail: a.whitbrook@derby.ac.uk

M. Trovati
Computer Science, Edge Hill University, St Helens Road, Ormskirk,
L39 4QP Lancashire, UK
e-mail: marcello.trovati@edgehill.ac.uk

S. Berry et al. (eds.), *Guide to Computational Modelling for Decision Processes*,
Simulation Foundations, Methods and Applications,
DOI 10.1007/978-3-319-55417-4_1

This model can then lead to sets of equations describing the operation of the system. These can be used to simulate the system to gain understanding of its dynamic behaviour and to be able to evaluate alternative policies, leading to improvements within the system.

A series of small examples are used to introduce this modelling process. Where information is available, the behaviour predicted by these models is compared with reality, i.e. what has happened in reality.

1.1.1 Introducing Influence Diagrams

Modelling using influence diagrams is introduced through the following illustrative examples:

- Stock control model: used to illustrate the basic modelling notation.
- Spending/saving model: used to illustrate the construction of an influence diagram and to introduce the concept of "positive" and "negative" feedback loops
- House building, financial models and population modelling: so that the predictions from the models (positive or negative loops) can be compared with reality.
- Transport modelling: extending the work to demonstrate the effect of government policy on transport provision (the "Beeching" cuts for example).

Example A: Stock Control Policies

A company holds stocks of finished goods to be able to satisfy demand; when stocks are low, more newly manufactured items are added to the finished goods stock; in this example, the available stock (for use) is "influenced" by production and demand (see Fig. 1.1).

The direction of the arrow from [despatched] to [production] indicating the production levels is influenced by the quantity of items dispatched, and the arc label (D) indicates the delays between each event.

Where

Demand	Influences	Number dispatched
Number dispatched	Influences	Production
Production	Influences	Available stock
Available stock	Influences	Number dispatched

The model constructed from this diagram will have the form:

Dispatched is described by the function f(Demand, Stock)
$Dispatched_i = Minimum(Demand_{i-3}, Stock_i)$
Assuming a delay of 3 between receipt of order and despatch.

Production is defined by f(Dispatched) or following through
f(Dispatched, Demand, Stock) or by implication
f(Dispatched, Demand, Stock, production))
$Production_i = Maximum(Dispatched_{i-2}, Demand_{i-5}, MinProduction)$
Assuming a delay of 2, dispatched and production request.

Stock is described by the function $f(stock_{i-1}, Production_{i-1}, Dispatched)$
$Stock_i = Stock_{i-1} + Production_{i-1} - Dispatched$
Assuming a delay of 1 between production and stock ready for use.

The next stage gives examples to introduce approaches to the production of "influence diagrams" and the notation used to analyse the resultant model.

1.1.1.1 Categorising Dependencies (Links) in a Model

An initial (causal) analysis is used to categorise an influence diagram and hence the underlying model, essentially the causal analysis asks: (Fig. 1.2).

If the input value increases, what is the effect on the output value? leading to the categorisation of the links as either positive (+) or negative (−) links.

Connecting between inflation rate and prices, as inflation rises, then so too do prices giving:

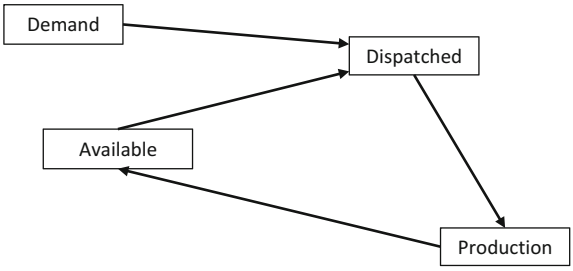


Fig. 1.1 Production

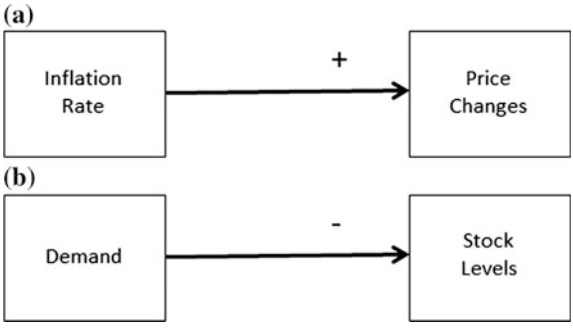


Fig. 1.2 **a** Positive arc: if "inflation rate" increases then "prices" will rise. **b** Negative arc: if "demand" increases then "stock levels" will fail

Connecting between demand and stock levels, as demand rises, it follows that stock levels will fall:

In carrying out this analysis always start with…"if the input rises…" starting with the opposite "…if the input falls…" can lead to double negative statements and some confusion in the following analysis.

1.1.1.2 Categorising a Model

Having categorised all the links, a loop in a model can fall into one of the two states: positive or negative feedback loops. In general, a negative loop indicates a "goal-seeking" model here there will be convergence, whereas a positive feedback loop indicates unrestricted growth or decay (Fig. 1.3).

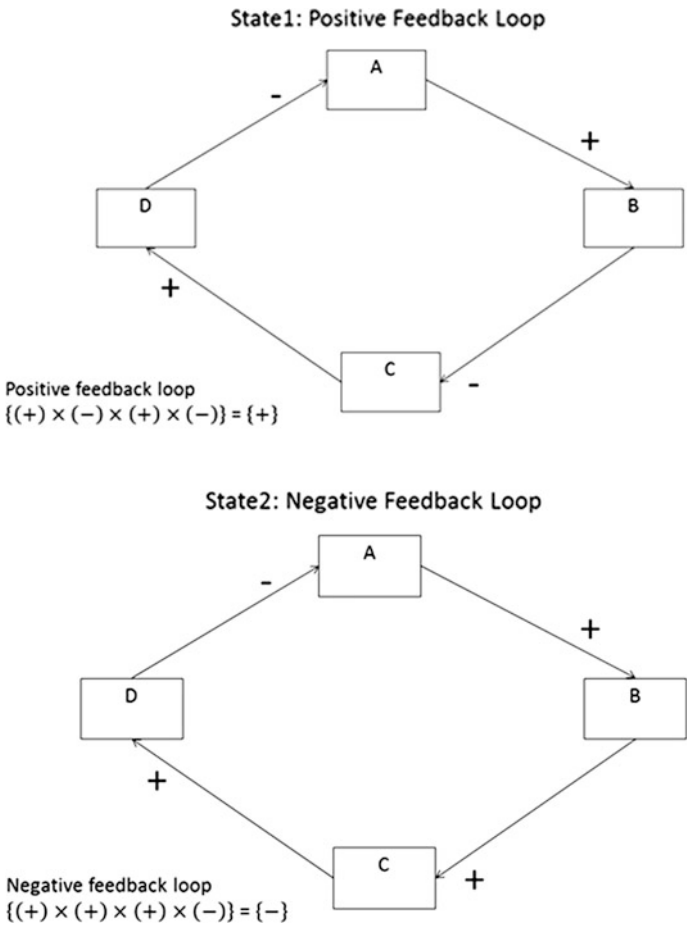


Fig. 1.3 Categorising feedback loops

- State 1, a positive feedback loop would lead to "unconstrained" growth or decline, while
- State 2, a negative feedback loop would lead to a steady-state solution (goal-seeking).

1.1.2 Model Evaluation/Validation, Comparing the Model with Historic Data

Here, a model is constructed and evaluated showing that its behaviour replicates the real situation.

Example B: House-Building Model

An initial model links house buying with house prices and housing stock giving a model with a negative feedback loop, ignoring the overall demand for housing (Fig. 1.4).

Analysing Influences in the diagram leads to the consequent effects:

House prices	Up	Then	House building	Down	Negative effect
House building	Up	Then	Housing stock	Down	Negative effect
Housing stock	Up	Then	House prices	Down	Negative effect

Thus giving the model cycle from and returning to a given starting position

House building	Up	Gives	Housing stock	Down
Housing stock	Down	Gives	House prices	Up
House prices	Up	Gives	House building	Down

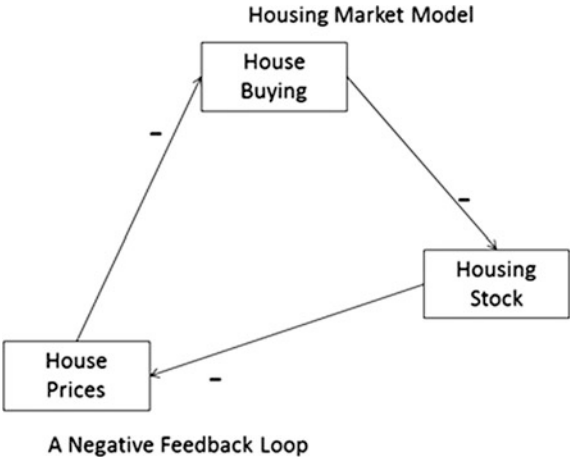


Fig. 1.4 First house price model

Continuing to complete the cycle of effects

House building	Down	Gives	Housing stock	Up
Housing stock	Up	Gives	House prices	Down
House prices	Down	Gives	House building	Up

Completing the cycle shows the goal-seeking behaviour of this model.

Extending the model through the addition of house construction gives a second loop (Fig. 1.5).

The additional loop adds the influences

House construction	Up	Then	Housing stock	Up
Housing stock	Up	Then	House prices	Down
House prices	Up	Then	House buying	Down
House buying	Down	Then	House construction	Down

Thus, adding to the existing model cycle starting from the same starting position

House buying	Up	Gives	House construction	Up
House construction	Up	Gives	Housing stock	Up
Housing stock	Up	Gives	House prices	Down
House prices	Down	Gives	House buying	Up

Finally, adding financial considerations into the influence diagram gives the model: (Fig. 1.6).

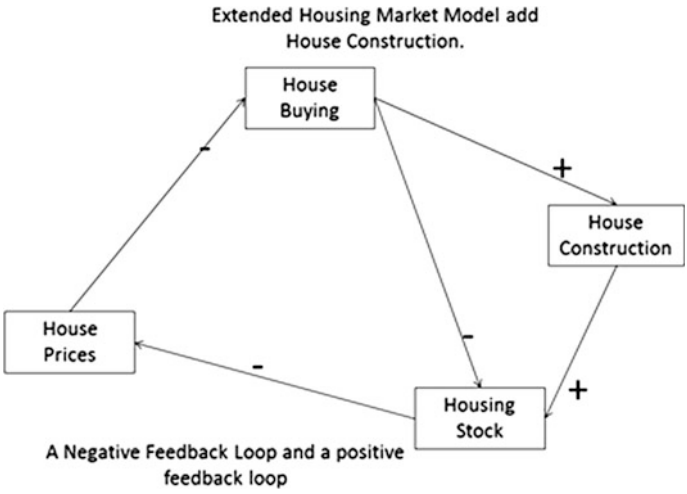


Fig. 1.5 Second house price model

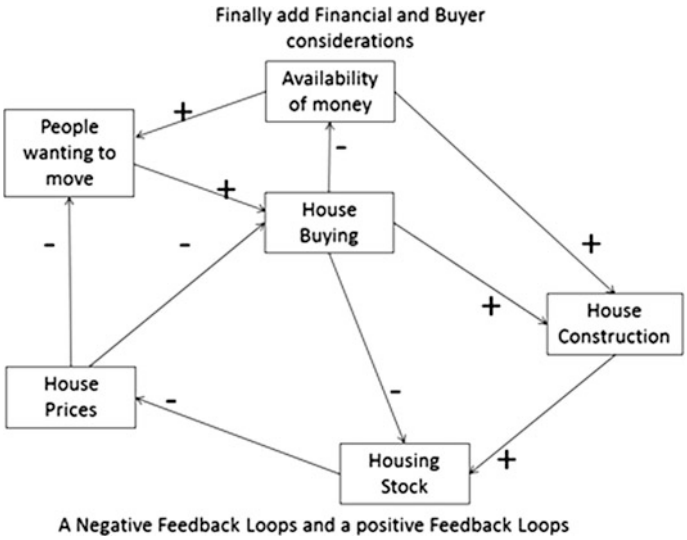


Fig. 1.6 House pricing model

All housing models contain a negative loop; however, a fuller model would include "people needing housing" and "construction of social housing" (see appendix for an extended model).

1.1.3 Example C: Developing Financial Models

These models investigate the movement of stock market share prices. The first model investigating the effect of investor confidence on share price movements and the second (extended) model adding the effect of short selling on the first model. Notice that these models demonstrate the effect of positive feedback loops and explain the movement of stock market prices (Fig. 1.7).

1.1.3.1 Effect of Investor Confidence on Financial Markets

This model can be constructed in stages.

Stage 1: share price and its effect on investor confidence, as the price increases, then confidence will also increase (Fig. 1.8)
Stage 2: now add the effect of investor confidence on the sale of shares, assuming that when confidence is rising, the investors will tend to hold onto the shares to give
Stage 3: as the sale of shares increase, then the share price will fall, giving the completed model (Fig. 1.9).

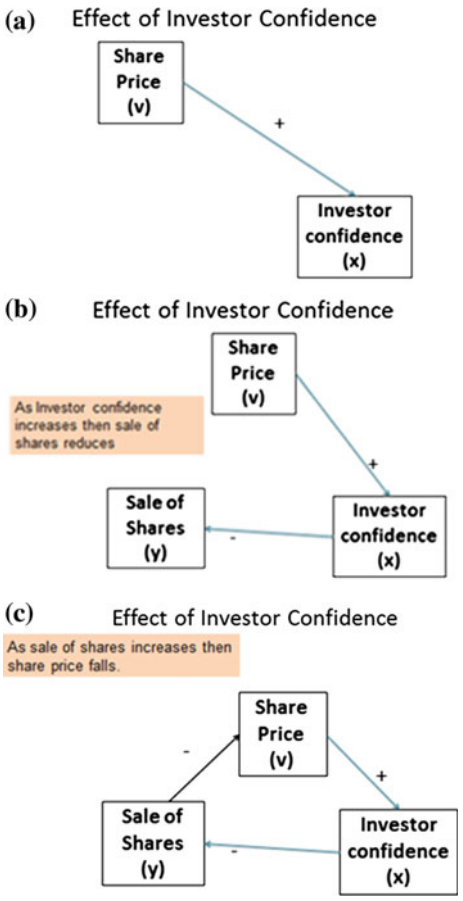


Fig. 1.7 **a** Start financial model **b** start financial model **c** first financial model

This model exhibits a positive feedback loop {+, −, −}, and hence, either an uncontrolled increase or decrease in share values will result from this model:

Increase in share prices gives				
Share price	Up	Leads to	Investor confidence	Up
Investor confidence	Up	Leads to	Sale of shares	Down
Sale of shares	Down	Leads to	Share price	Up
Decrease in share prices gives				
Share price	Down	Leads to	Investor confidence	Down
Investor confidence	Down	Leads to	Sale of shares	Up
Sale of shares	Up	Leads to	Share price	Down

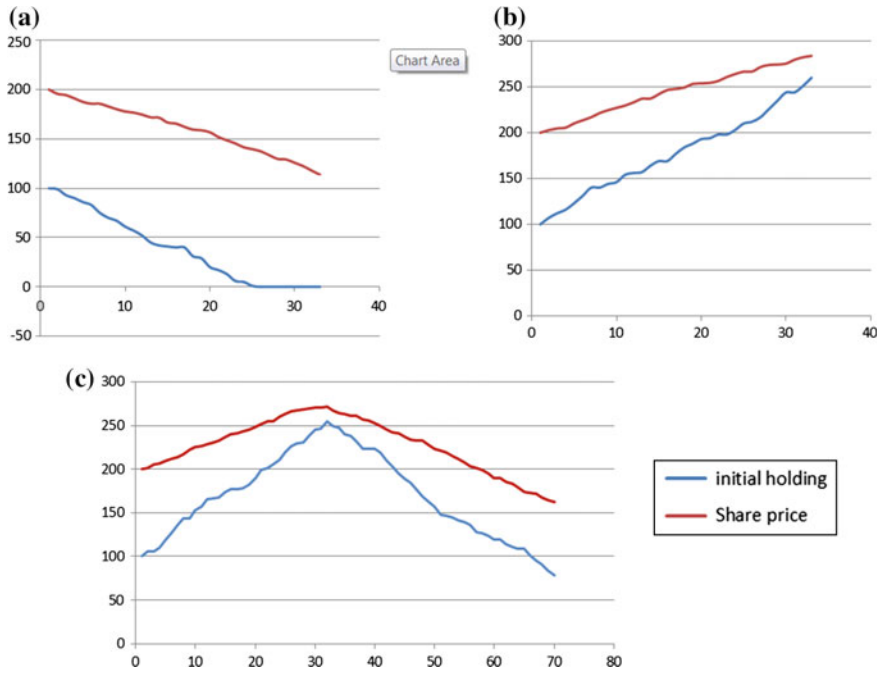
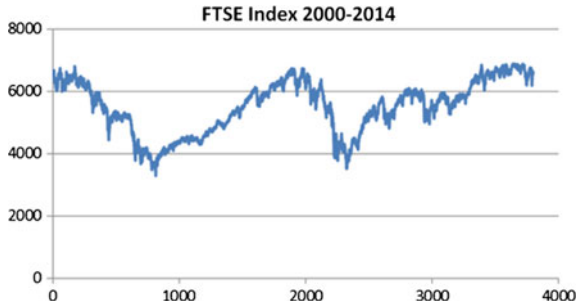


Fig. 1.8 **a** , **b** and **c**: Simulating stock holdings

Fig. 1.9 FTSE index



The direction (of movement of share prices) is changed only through the effect of external factors.

Using this model, simulation models were constructed first starting with low investor confidence; the results from this simulation produced the plots of "share price" and "share holding" showing how they both fall with time.

Conversely, if the investors start with high confidence, the plots show how both share price and share holding grow with time.

Finally, if the investor state starts in one mode (high confidence), then switching the next plot shows the effect on share prices and investor holding.

Validating the Model

This effect is demonstrated in both FTSE and Dow Jones historic data log plots; both the FTSE and the Dow Jones data can be shown to have several changes of direction moving from a period of continual growth to a period of continual decline; in addition, the Dow Jones plot also demonstrates that there can be periods where the plot is stationary (Fig. 1.10).

Turning points, changes from growth to fall or from fall to growth in the FTSE Index, occur at (about) 2003; 2007; 2009; 2010.

Similarly, plotting the Dow Jones, log(price) and movements for the period 1902–2012 gave (Fig. 1.11).

With Dow Jones turning points, marked above in Fig. 1.11, at (about) 1902; 1929; 1936; 1946; 1964; 1984; 2002;

Replotting the Dow Jones Index (log price) over a shorter time period, up to 1935, shows the dramatic effect of this model around the time of the Wall Street Crash (1929–1933); here, attempts to halt the fall failed until external events caused the changed direction.

Here, turning points seem to occur at 1924, 1929, 1932 and 1934.

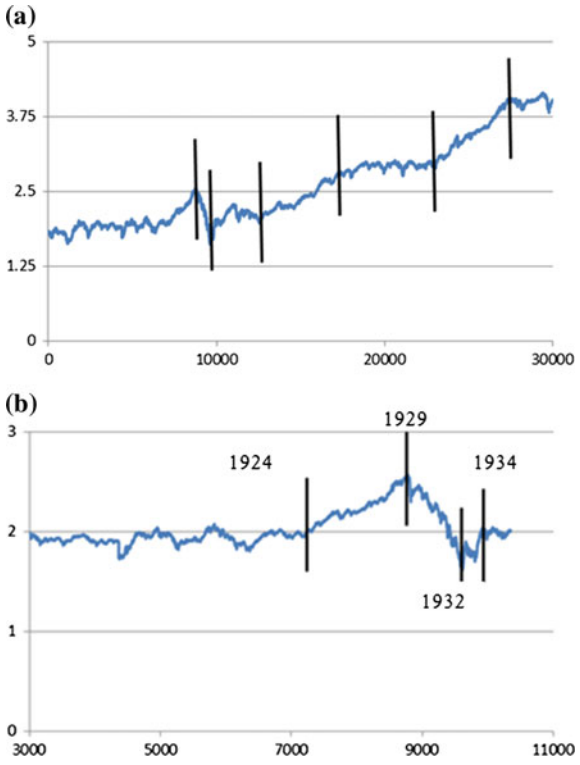


Fig. 1.10 **a** Dow Jones index **b** growth periods in Dow Jones index

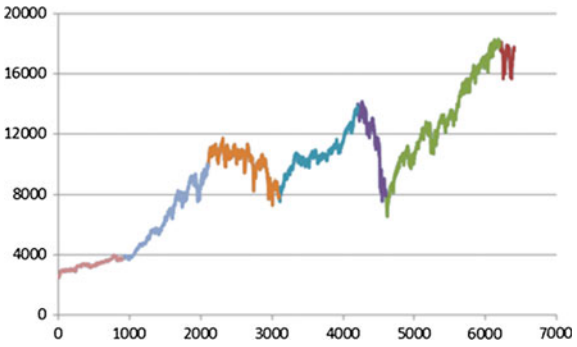


Fig. 1.11 Growth periods in Dow Jones index

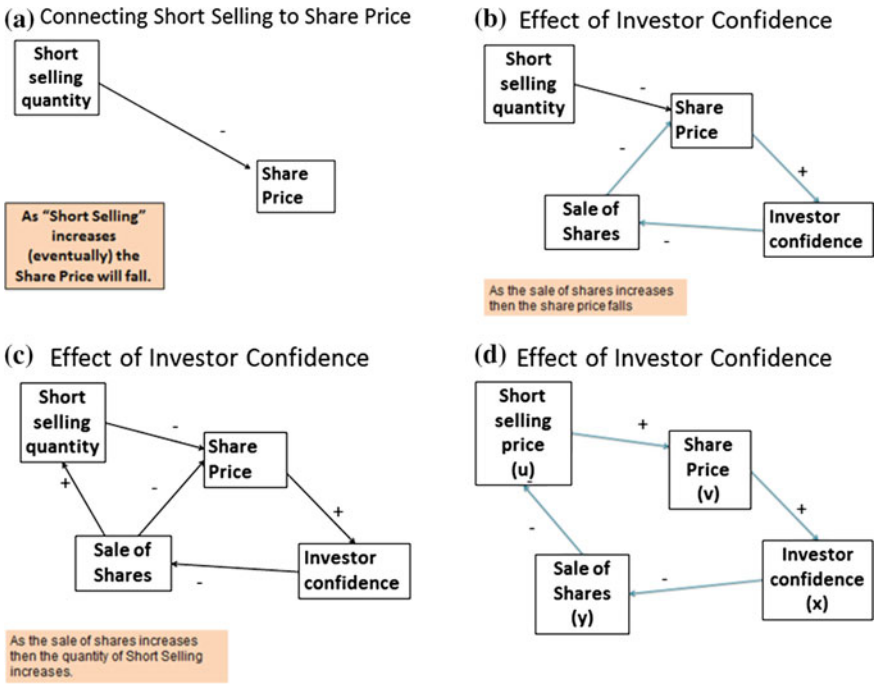


Fig. 1.12 **a** Modelling investor confidence and shares, **b** modelling investor confidence and shares and **c** modelling investor confidence and shares **d** alternative model for investor confidence and shares

Finally, plotting the Dow Jones Index over the period from 1991 not only shows the same effects during this period but also that the changes in direction tend to occur more frequently (Fig. 1.12).

Here turning points at 1991, 1994, 1999, 2003, 2007, 2009 and 2015 on average every 4 years.

1.1.3.2 Example D2: Effect of Investor Confidence and Short Selling

Definition: Short sale is the sale of securities or commodities not owned by the seller (who hopes to buy them later at lower price); hence, short-selling cause falls in asset prices.

Short selling and the results from short selling have been known for a long time:

- "He who sells what isn't his'n, must buy it back or go to pris'n."—Daniel Drew, 1797–1879, American financier.
- 1609—The Dutch East India Co protests to the Amsterdam Exchange after short sellers make enormous profits on its stock. Leading to the first ever regulations on shorting in the following year
- 1733—Britain bans naked short selling. "Investopedia" ref South Sea Bubble.
- 1932—US President Herbert Hoover condemns short selling for speculative profit on the New York Stock Exchange. www.reuters.com/article/us-sec-shortselling-history

Developing the influence model, from the base investor confidence model (Fig. 1.13).

Then, adding in investor confidence events gives:

Finally, as the sale of shares increases, then the short-selling quantity will increase.

This model exhibits only positive feedback loop {+, −, −} and {+, −, +, −}; hence, short selling will have the following effect on share prices:

Increase in short selling gives				
Short selling	Up	Leads to	Share price	Down
Share price	Down	Leads to	Investor confidence	Down
Investor confidence	Down	Leads to	Sale of shares	Up
Sale of shares	Up	Leads to	Short selling	Up

Notice that a similar model exists with respect to the short-selling price:

An analysis of this model gives:

Fall in short selling price gives				
Short selling price	Down	Leads to	Share price	Down
Share price	Down	Leads to	Investor confidence	Down
Investor confidence	Down	Leads to	Sale of shares	Up
Sale of shares	Up	Leads to	Short selling price	Down

Comparing this model with the stock market when "falls" occurred in the 1990s and 2000s and 2010s.

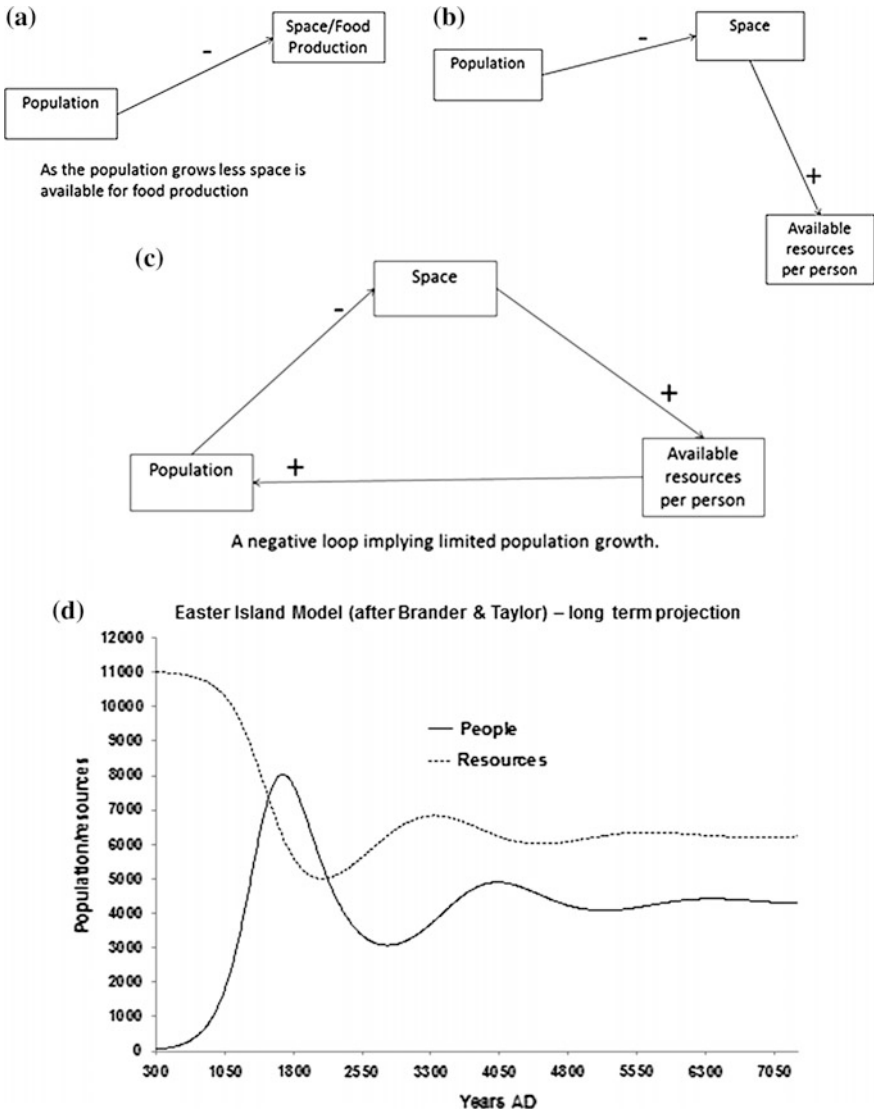


Fig. 1.13 **a**–**c** Developing the model, **d** results from a simulation based upon this model

1.1.4 Population Modelling

Again, two models are presented investigating population changes in a closed society, for example Easter Island an isolated Pacific Ocean island, in the first model no technological developments and in the second there are technological developments.

Basic "Easter Island" model

Stage 1: as the population increases, then the space available for food production falls.
Stage 2: as the space for food production increases, then the resources available per person also increases.
Stage 3: linking to give the final model, a negative goal-seeking loop (Fig. 1.14).

Notice that									
Population	Up	Leads to	Available space	Down					
Available space	Down	Leads to	Resources per person	Down					
Resources per person	Down	Leads to	Population	Down	Population	Down	Leads to	Available space	Up
Available space	Up	Leads to	Resources per person	Up					
Resources per person	Up	Leads to	Population		Up				

Applying this model using Easter Island, data gave the population plot

Where population and resources are tending towards a steady state, as expected with periods of population growth and population decline.

Note Easter Island was "discovered" at a time when the population was in the first "state of decline" shown in the population plot.

Allowing for technological development produces the model (Fig. 1.15):

Leading to an increasing population, compare this with the Earths continual development of food sources and the total population growth.

1.1.5 Transport Modelling

Travellers choose the most appropriate mode of transport to reach their chosen destination. They could travel by rail, or car, or bicycle or by other forms of public transport or even walk.

This model investigates the interaction between the use road (car) and rail transport and the subsequent expansions of the road and rail systems.

Stage 1: only road travel is considered giving (Fig. 1.16):

The model suggesting that road capacity is continually increasing, generally true, in the absence of any alternative mode of transport.

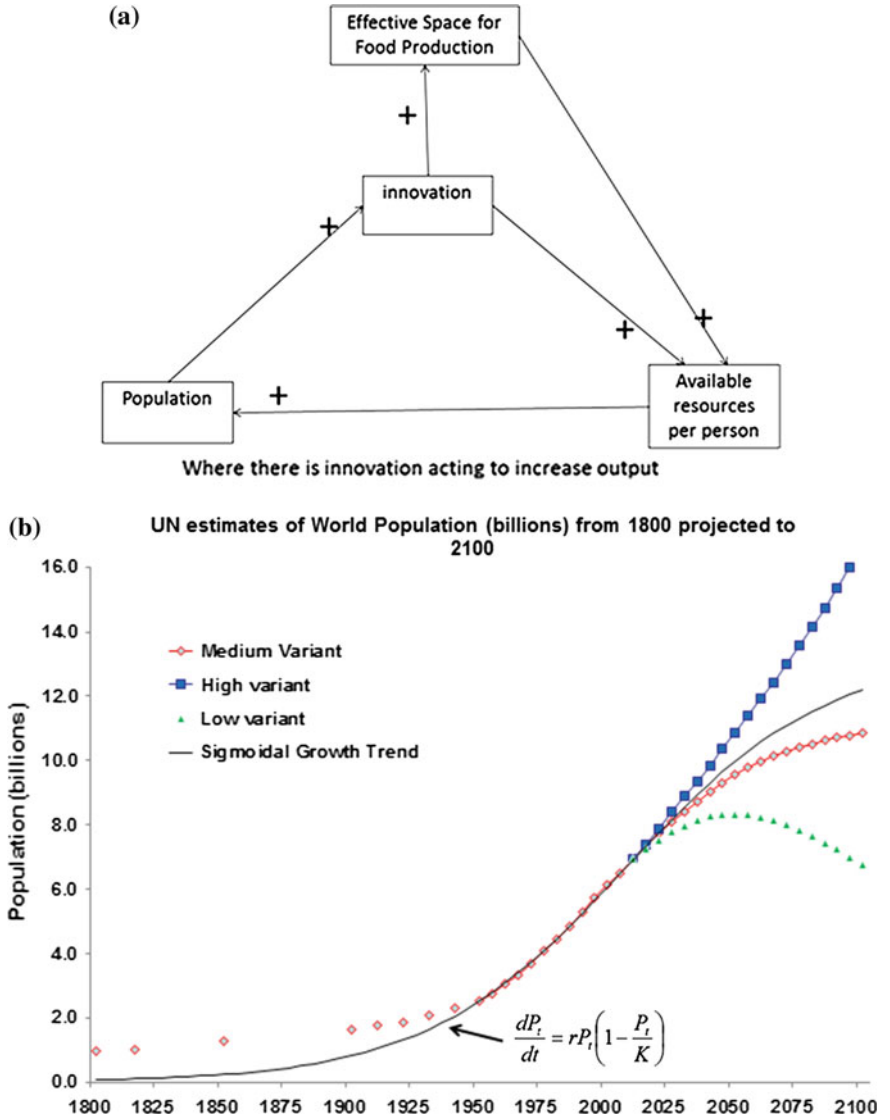


Fig. 1.14 **a** Second population model **b** world population estimates

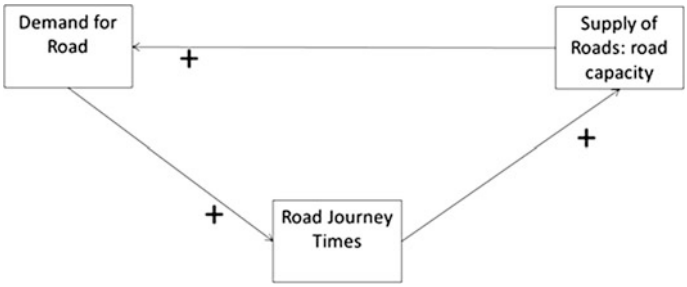


Fig. 1.15 Modelling road building

Journey times	Up	Leads to	Road Building	Up					
Road building	Up	Leads to	Demand for Roads	Up	Demand for Roads	Up	Leads to	Journey times	Up
Journey times	Up	Leads to…							

Similarly, extending the model by incorporating rail changes could produce another model containing only positive feedback loops, thus implying, as in the 1960s, that road provision is (continually) increasing and rail provision declining.

However, when increases in road capacities or the development of the road network are not, or no longer, a viable option, this model becomes:

Here there exists a negative feedback loop leading to an equilibrium state:

- {Demand for Road Travel; Road journey times; Demand for railways; Supply of Railways; Demand for road travel}
- { +, +, +, −}

The alternative modelling approach, using travel cost, also gives a model with a negative feedback loop

- {Supply of railways; Rail Journey costs; Demand for Railways}
- {+, −, +}

Here, paradoxically, the increased demand for rail travel can act to cause developments in the rail system and thus act to reduce demand!

Further development leads to the more comprehensive model given by:

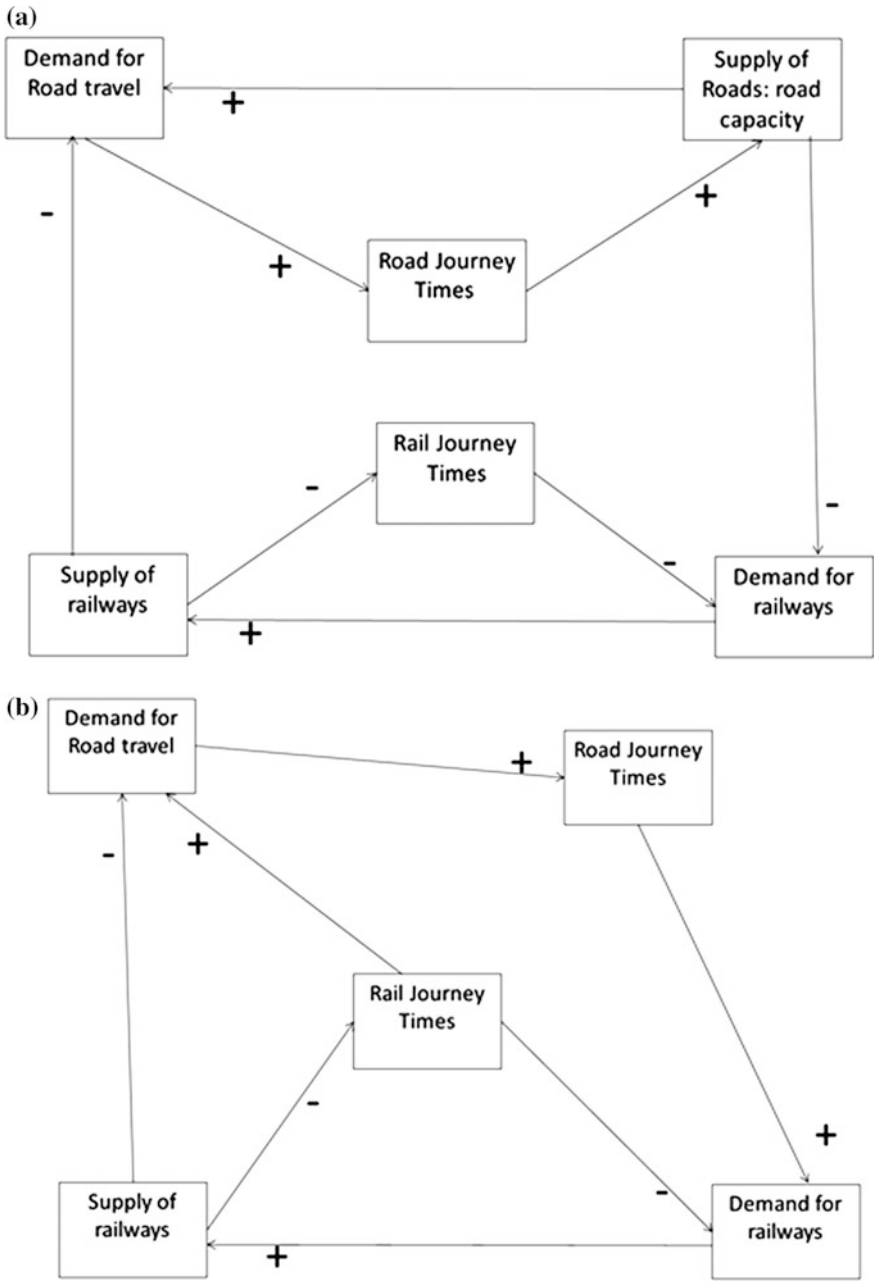


Fig. 1.16 **a** Modelling the effects of roads on rail demand **b–d** modelling the effects of roads on rail

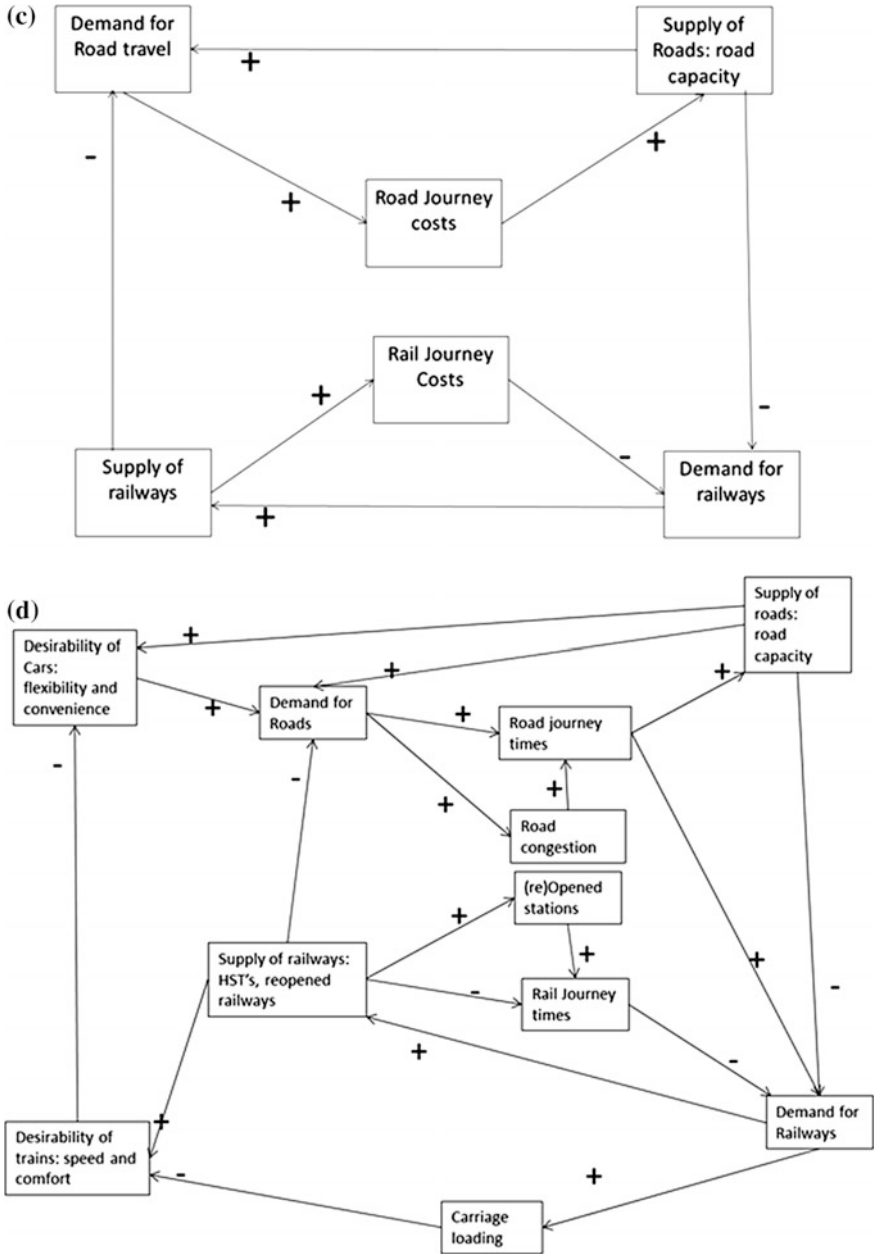


Fig. 1.16 (continued)

A negative feedback loop with respect to rail travel gives an overall goal-seeking model for both road and rail travel.

	{Desirability of cars;	
	Demand for roads;	
	Road journey time;	
	Supply of roads;	
	Demand for rail;	
	Carriage loading;	
	Desirability of trains;	
	Desirability of cars}	
Changes	{+, +, +, −, +, −, −}	A negative loop

1.2 Constructing Models from "Big Data"

1.2.1 Introduction

Data science is a relatively new research field consisting of well-established methods and approaches to address the need to identify actionable knowledge from the continuous creation of data. In particular, the use of the existing techniques has led the discovery of new research directions, with groundbreaking results.

There are a variety of mathematical and statistical tools to identify and discover knowledge, which can be used to facilitate the decision-making process. In particular, these include Bayesian and dependency networks, which provide modelling tools to determine how mutual relationships between concepts influence the knowledge captured by such networks. In the rest of this section, we will discuss their main theoretical properties, which be fully exploited in the two case studies discussed in part 2.

Bayesian and Dependency Networks

Bayesian networks (BNs) are graphical models, which capture independence relationships among random variables, based on a basic law of probability called *Bayes' rule* [1]. They offer an efficient modelling framework in risk and decision analysis with a variety of applications, such as safety assessment of nuclear power plants, risk evaluation of a supply chain and medical decision support tools [2]. More specifically, BNs are defined by nodes, representing objects based on a level of uncertainty, also called *random variables*, which are connected by edges indicating a dependence relationship between them. Furthermore, Bayesian networks also contain quantitative information, which represents a factorisation of the joint probability distribution of all the variables in the network.

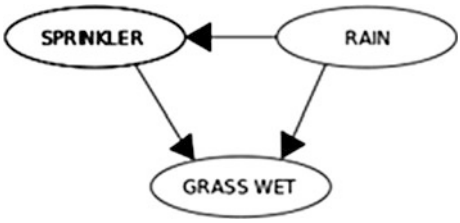


Fig. 1.17 Representing a Baysian network

Suppose, for example, we want to explore the chance of finding wet grass on any given day. In particular, assume the following

1. A cloudy sky is associated with a higher chance of rain,
2. A cloudy sky affects whether the sprinkler system is triggered and
3. Both the sprinkler system and rain have an effect on the chance of finding wet grass.

In this particular example, no probabilistic information is given. The resulting BN is depicted in the following Fig. 1.17.

It is clear that such graphical representation provides an intuitive way to depict the dependence relations between variables.

In the definition of BNs, the most complex statements do not refer to dependencies, but rather about independences (i.e. absence of edges in the graph), as it is always possible to determine dependence through the conditional probability tables when an edge is present, even though the reverse is not true.

The definition of a BN can be carried out either through explicit data analysis, or via literature review and expert elicitation. These are typically manually intensive tasks depending on the size and complexity of the data sets analysed, especially when they exhibit unstructured components. There is extensive research on the extraction of BNs from text corpora. For example, in [3], the authors suggest a domain-independent method for acquiring text causal knowledge to generate Bayesian networks. Their approach is based on a classification of lexico-syntactic patterns which refer to causation, where an automatic detection of causal patterns and semi-validation of their ambiguity is carried out. Similarly, in Kuipers [4], a supervised method for the detection and extraction of causal relations from open domain texts is presented. The authors provide an in-depth analysis of verbs, cue phrases that encode causality and, to a lesser extent, influence.

Dependency networks (DNs) have also been attracting increasing attention within several research fields. These types of networks are similar to BNs, which, however, allow cycles (that is a path starting and ending at the same node) enabling a more computationally efficient approach and making them more applicable especially when large and unstructured data sets are considered [5].

Text Mining

Text mining (TM) consists of computational techniques to achieve human language understanding via linguistic and semantic analysis. Such methods have been shown to be crucially important in the way we can represent knowledge described by the interactions between computers and human (natural) languages [6].

Language is based on grammatical and syntactic rules which can be captured by *patterns* or, in other words, templates that sentences with similar structure follow. Such language formats enable the construction of complex sentences, as well as framing of the complexity of language.

In order to understand the properties of human language, a variety of methods have been developed to address the complexity and challenges posed by it. These, broadly speaking, fall into three categories: *symbolic, statistical* and *connectionist*.

In the symbolic approach, linguistic properties are mapped onto precise and well-understood knowledge representation [7]. Once the linguistic rules have been defined, the hierarchical structure of the semantic concepts within the corresponding textual fragments is identified. Subsequently, the properties associated with the different textual components are investigated to provide an insight into their structure. Symbolic methods have been widely exploited in a variety of research contexts such as information extraction, text categorisation, ambiguity resolution, explanation-based learning, decision trees and conceptual clustering.

On the other hand, the statistical properties from observable data and the investigation of large documents can be used to develop generalised models based on smaller knowledge data sets and significant linguistic or world knowledge [8]. They have many applications such as parsing rule analysis, statistical grammar learning and statistical machine translation, to name but a few.

The connectionist approach integrates statistical learning with representation techniques to allow an integration of statistical tools with logic-based rule manipulation, generating a network of interconnected simple processing units (often associated with concepts) with edge weights representing knowledge. This typically creates a rich system with an interesting dynamical global behaviour induced by the semantic propagation rules. In Troussov et al. [9], a connectionist distributed model is investigated pointing towards a dynamical generalisation of syntactic parsing, limited domain translation tasks and associative retrieval.

General Architecture and Various Components of Text Mining

A grammar is a set of well-defined rules which govern how words and sentences are combined according to a specific syntax. A grammar does not describe the meaning of a set of words or sentences, as it only addresses the construction of sentences according to the syntactic structure of words. Semantics, on the other hand, refers to the meaning of a sentence [8]. In computational linguistics, semantic analysis is a much more complex task since its aim is the full understanding of the meaning of text.

Any text mining process consists of a number of steps to identify and classify sentences according to specific patterns, in order to analyse a textual source. Broadly speaking, in order to achieve this, we need to follow these general steps:

1. Textual data sources are divided into small components, usually words, which can be subsequently syntactically analysed.
2. These, in turn, create *tokenised* text fragments, which are analysed according to the rules of a formal grammar. The output is a parsing tree—in other words, an ordered tree representing the hierarchical syntactic structure of a sentence.
3. Once we have isolated the syntactic structure of a text fragment, we are in the position of extracting relevant information, such as specific relationships and sentiment analysis.

More specifically, the main components of text mining are as follows:

Lexical Analysis

Lexical analysis is the process which analyses the basic components of texts and groups into *tokens* [10]. In other words, lexical analysis techniques identify the syntactic role of individual words which are assigned to a single *part-of-speech* tag.

Lexical analysis may require a *lexicon* which is usually determined by the particular approach used in a suitably defined TM system, as well as the nature and extent of information inherent to the lexicon. Mainly, lexicons may vary in terms of their complexity as they can contain information on the semantic information related to a word. More research is currently being carried out to provide better tools in analysing words in semantic contexts [see 11 for an overview].

Part-of-Speech Tagging.

Part-of speech tagging (POS) allows to attach a specific syntactic definition (noun, verb, adjective, etc.) to the words which are part of a sentence. This task tends to be relatively accurate, as it relies on a set of rules which are usually unambiguously defined. Often, POS tasks are carried out via the statistical properties of the different syntactic roles of tokens [8]. Consider the word *book*. Depending on the sentence it belongs to, it might be a verb or a noun. Consider "a book on chair" and "I will book a table at the restaurant". The presence of specific keywords, such as "a" in the former, and "I will" in the latter, provides important clues as to the syntactic role that *book* has in the two sentences. One of the main reasons for the overall accuracy of POS tagging is that a semantic analysis is often not required, as it is based on the position of the corresponding token.

Parsing

Once the POS tagging of a sentence has identified the syntactic roles of each token, each sentence can be considered in its entirety. The main difference with POS tagging is the fact that parsing enables the identification of the hierarchical syntactic structure of a sentence. Consider, for example, Fig. 1.18b depicts the parsing tree structure of the sentence "This is a parsing tree". Note that each word is associated with a POS symbol which corresponds to its syntactic role [8].

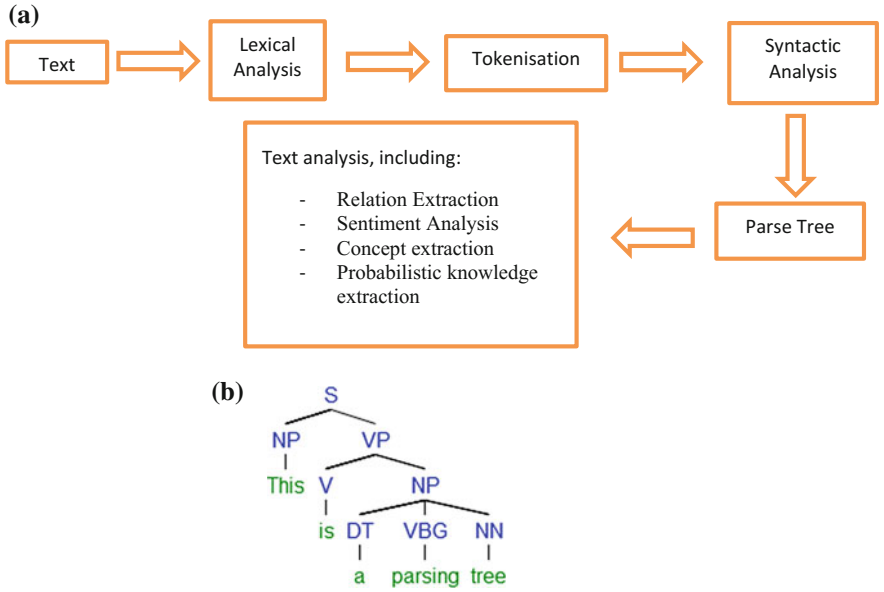


Fig. 1.18 **a** Text missing, **b** the parsing tree of the sentence "This is parsing tree"

Named Entity Recognition

An important aspect of text analysis is the ability to determine the type of the entities, which refer to words, or collections of them. For example, determining whether a noun refers to a person, an organisation or geographical location (to name but a few) substantially contributes to the extraction of accurate information and provides the tools for a deeper understanding. For example, the analysis of "dogs and cats are the most popular pets in the UK" would identify that dogs and cats are animals and the UK is a country. Clearly, there are many instances where this depends on the context. Think of "India lives in Manchester". Anyone reading such sentence would interpret, and rightly so, India as the name of a specific person. However, a computer might not be able to do so and determine that it is a country. We know that a country would not be able to "live" in a city. It is just common sense. Unfortunately, computers do not have the ability to discern what common sense is. They might be able to guess according to the structure of a sentence, or the presence of specific keywords. This is a very effective example of semantic understanding, which comes natural to humans, but a very complex task to computers.

Coreference Resolution

Coreference resolution is the process of determining which text components refer to the same objects. For example, *relation resolution* attempts to identify which individual entities or objects a relation refers to. Consider the following sentence,

"We are looking for a fault in the system". Here, we are not looking for *any* fault in the system, rather for a specific instance.

Relation Extraction

The identification of relations between different entities within a text provides useful information that can be used to determine quantitative and qualitative information linking such entities. For example, consider the sentence "smoking potentially causes lung cancer". Here, the act of smoking is linked to lung cancer by a causal relationship. This is clearly a crucial step in building BNs, even though such analysis requires a deep understanding of the associated textual information.

Concept Extraction

A crucial task in information extraction from textual sources is concept identification, which is typically defined as a one or more keywords, or textual definitions. The two main approaches in this task are supervised and unsupervised concept identification, depending on the level of human intervention in the system.

In particular, formal concept analysis (FCA) provides a tool to facilitate the identification of key concepts relevant to a specific topical area [12]. Broadly speaking, unstructured textual data sets are analysed to isolate clusters of terms and definitions referring to the same concepts, which can be grouped together. One of the main properties of FCA allows user interaction, so that user(s) can actively operate the system to determine the most appropriate parameters and starting points of such classification.

Sentiment Analysis

This instance of information extraction from text focuses on the identification of trends of moods or opinions associated with textual sources.

Broadly speaking, its aim is to determine the *polarity* of a given text which identifies whether the opinion expressed is positive, negative or neutral. This also includes emotional states, such as anger, sadness and happiness, as well as intent, such as planning and researching. Sentiment analysis can be an important tool is obtaining an insight into relationships among concepts in BNs, since it can support the process of relation discovery. In fact, not all the information contained in text is unambiguously described. Consider, for example, the sentence "I am very surprised by your irrational fear that spiders can kill you". Here, rather than drawing a definite conclusion that spiders are linked with death, a sceptical assessment of such relationship is noted.

Topic Recognition

This procedure attempts to identify the general topic of a text by grouping a set of keywords which appear frequently in the documents. These are then associated with one of more concepts to determine the general concept trend.

Semantic Analysis

Semantic analysis determines the possible meanings of a sentence by investigating the interactions among word-level meanings in the sentence. This approach can also incorporate the *semantic disambiguation of words* with multiple senses. Semantic disambiguation allows the selection of the sense of ambiguous words, so that they can be included in the appropriate semantic representation of the sentence [13]. This is particularly relevant in any information retrieval and processing system based on ambiguous and partially known knowledge. Disambiguation techniques usually require specific information on the frequency with which each sense occurs in a particular document, as well as on the analysis of the local context, and the use of pragmatic knowledge of the domain of the document. An interesting aspect of this research field is concerned with the purposeful use of language where the utilisation of a context within the text is exploited to explain how extra meaning is part of some documents without actually being constructed in them. Clearly, this is still being developed as it requires an incredibly wide knowledge dealing with intentions, plans and objectives [8]. Extremely useful applications in TM can be seen in inference techniques where extra information derived from a wider context successfully addresses statistical properties [4].

1.2.2 The Automatic Extraction of Bayesian Networks from Text

The mathematical constraints posed by Bayes' rule and general probability theory create a significant challenge. As a consequence, the identification of suitable Bayesian networks is often carried out manually usually by a modeller. However, this can be extremely time-consuming and based on only specific, often limited, sources depending on the modeller's expertise. As a consequence, the ability to automatically extract the relevant data would potentially add enormous value in terms of increased efficiency and scalability to the process of defining and populating BNs. However, extracting both explicit and implicit information, and making sense of partial or contradictory data, can be a complex challenge.

Dependence Relation Extraction from Text

As discussed above, nodes in BNs, which are connected by edges, indicate that the corresponding random variables are dependent. Such dependence relations must be therefore extracted from textual information, when present. The conditional dependencies in a Bayesian network are often based on known statistical and computational techniques, which are based on a combination of methods from graph theory, probability theory, computer science and statistics. Linguistically speaking, a dependence relation contains specific keywords which describe that two concepts are related to a certain degree. Consider the sentence "lung cancer is more common among smokers". There is little doubt that we would interpret this as clear

relation linking lung cancer with smoking. However, there is not a precise linguistic definition to determine a relationship between two concepts from text, due to its content dependence. When a full automation of the process of textual information extraction is carried out, a clear and unambiguous set of rules ensures a reasonably good level of accuracy. As a consequence, it is usually advisable to consider *causal relationships,* which are a subgroup of dependence relationships [1]. In fact, they are likely to convey a much stronger statement, and they are more easily identified due to a more limited set of linguistic rules that characterise them. Going back to the above example, saying that smoking *causes* lung cancer assumes a direct link between them. We cannot arguably say the contrary, but there are other cases where there is a less marked cut-off. If we are only looking for causal relationships when populating a BN, we might miss out several dependence relations. However, accuracy is much more preferable. The integration of an automatic BN extraction with human intervention usually addresses such issue.

Variables' Identification

Mapping a representative to a specific variable is closely linked to the task of relation extraction. However, this is partially a modelling choice by the user based on the set of relevant concepts. Consider again the sentence "smoking causes lung cancer". If this was rephrased as "smokers are more likely to develop lung cancer", we would need to ensure that "smoking" and "smokers" are identified as a single variable associated with the act of smoking. In a variety of cases, this can be addressed by considering synonymy. However, such as in our example, it might also happen that they refer to the same concept, rather than being the *same* concept. Formal concept analysis (FCA) is one of the computational techniques that can be successfully applied in this particular context [8].

Probability Information Extraction.

An essential part in the extraction and subsequent creation of BNs involves the processing of the textual sources to determine probability of variables.

Sentences may capture some probabilistic relationships between concepts even though very few of them might provide conclusive and unambiguous information, which can be utilised to reason. In fact, the combination of qualitative and quantitative data creates a variety of challenges, which need to be addressed to produce relevant and accurate information.

The identification, assessment and ranking of specific keywords (and their combinations), when describing probability, can provide a useful insight into the structure of the corresponding relationships. However, Big Data research focuses on the interrelations of diverse and multidisciplinary topics, resulting in the intrinsic difficulty in finding a common ground in terms of the linguistic features that specific probabilistic description should have.

In [2], an automated method to assess the influence among concepts in unstructured sets is introduced. Despite not being directly related to BNs, it shows potential in the extraction of the mutual co-occurrence properties between nodes.

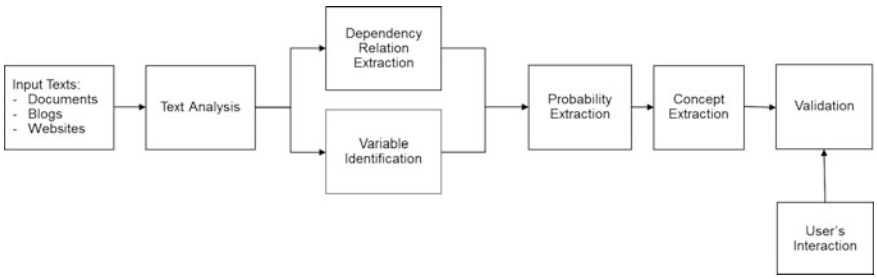


Fig. 1.19 Architecture of Bayesian network extraction from textual information

Aggregation of Structural and Probabilistic Data.

This step integrates the steps discussed above, to construct fragments of BNs via user's interaction.

Figure 1.19 depicts this process in a sequential set of steps. However, the repeated implementation of such steps in a series of loops might be required to obtain meaningful BN fragments.

General Architecture

The general architecture of the extraction of fragments of BNs from text corpora consists of the following components:

1. Existing and predefined information on specific topics would be incorporated into a database, or *Knowledge Database* (KDB) consisting of
   (a) Historical data from structured DBs,
   (b) Bayesian networks built on existing data and
   (c) Data entered by modeller(s) and manually validated.

The KDB is an important component since it is based on information which is considered "reliable". In a variety of cases, the KDB is maintained by modelling experts to ensure that the data are regularly updated to prevent any inconsistency and ambiguity.

2. The user would interact with the system by specifying further textual sources and structured data sets.
3. The extraction and data aggregation stage consists of the identification of the appropriate textual data associated with such unstructured data sets, as well as the removal of any data duplication. An essential part of this process is to address any qualitative and quantitative inconsistency. As discussed above, BNs have strict mathematical constraints which make any fully unsupervised automatic extraction prone to inaccuracies and inconsistencies. As a consequence, human intervention is often advisable to minimise any such error.

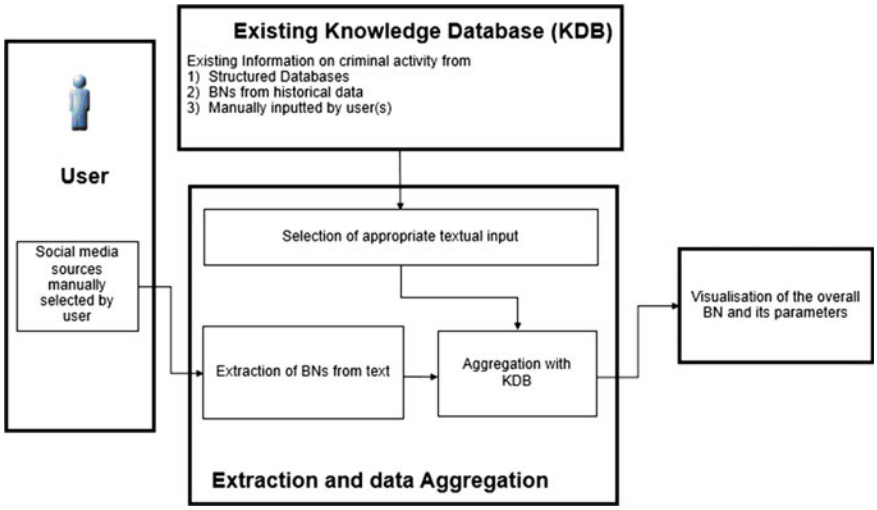


Fig. 1.20 General architecture of Bayesian networks for crime detection as discussed in Trovati [5]

4. Finally, the BN is visualised, providing
   (a) Relevant information on the structure of the BN,
   (b) Description of the different parameters and
   (c) Any required action in order to address any inconsistency which could not be resolved automatically. This is typically an interactive step, where the result can be updated by the user as well as focused on a specific part of the BN (Fig. 1.20).

1.3 The Blackboard Architecture

1.3.1 Introduction

The blackboard architecture represents a flexible, symbolic artificial intelligence (AI) method for the cooperative solution of complex problems. Systems that use this architecture have been in existence since the 1970s. In the beginning, they were used mainly for solving signal-processing problems, for example speech recognition with Hearsay-II (see [14]) and interpretation of sonar with HASP (see [15]). Following Hearsay-II and HASP, the blackboard architecture became very popular and was associated with many diverse application areas including mission control systems for satellites, military object tracking and detection, printed text recognition for scanners, fault diagnosis, assembly arrangement, and planning and scheduling, to name just a few. In the early nineties, the architecture endured a period of relative

obscurity, which has mostly been attributed to a lack of formalism, but since the turn of the century it has enjoyed something of a renaissance, in some cases as a hybrid system coupled with a Bayesian Belief Network, with applications such as robotic mapping (see [16]) and logistics planning for the US military (see [17]). The blackboard architecture has also been adopted by the computer game AI community for solving decision-making and agent coordination problems (see [18]), although there is some controversy about whether game AI blackboard systems are actually "true" blackboard systems.

Blackboard systems (BBSs) have many properties that make them suitable for solving complex problems that require progression through different stages (with many paths to those stages) to reach the final solution [19]. These include problems that can be viewed as a search for the "best" solution given a set of constraints [20]. BBSs provide a flexible method for incremental reasoning about a problem and its solution, building up the solution in step-by-step manner by opportunistically examining different paths at different levels of abstraction. This means that the system's exploratory properties are high compared with systems or algorithms that used fixed, predetermined methods such as forward or backward chaining rule-based systems. BBSs can also deal with large quantities of diverse, uncertain, incomplete or inaccurate data [21] and can integrate different kinds of knowledge in order to solve a given problem. This makes them highly useful for solving problems with limited and imperfect input data, problems that require the combination of many separate diagnostic components, dynamic decision-making problems and problems involving systems of systems.

1.3.2 Architecture

The original blackboard architecture was developed in the early 1970s with the Hearsay-II (HSII) speech recognition project [14], but has evolved to some extent since then (literature review for further details). The aim of the HSII project was to build a system that could manage complex and ill-defined problems without the requirement for a formal model, so that enhancements could easily be made [19]. The deliberate flexibility of the design has led to a number of different interpretations of BBS terminology and definitions. This section aims to clarify its core definition and design and also to dispel some of the myths surrounding BBSs. First, the analogy to a set of experts gathered around a blackboard is explained, and then, the key components of the BBS are listed and described. The essential properties of the BBS are then reported, either as advantages or disadvantages of the architecture. This section also includes a review of the main problem types that are suited to solution with BBSs, provides some definitions and briefly discusses the history and availability of toolkits for frameworks for BBSs.

1.3.2.1 Analogy

The BBS architecture is based on the concept of a group of experts using a physical blackboard as a shared workspace for constructing a solution to a problem. The first step is to write a description of the problem and the initial data on the blackboard. Each expert waits until an opportunity arises to apply their particular knowledge to the problem, at which point they may contribute by writing information on the blackboard; this information may take many forms including a partial solution, an informed suggestion about which solution paths or avenues of exploration to select next, an alternative solution, a candidate complete solution, an agreement to a candidate solution or partial solution, or a disagreement about a candidate solution or partial solution. The aim of each contribution is to provide insights that will enable the experts to progress closer to a complete solution that is optimal in some way. Thus, the contributions continue until an agreed solution is reached.

An important consideration is managing the flow of the problem-solving, i.e. deciding who takes the chalk next in the event that more than one expert identifies an opportunity to contribute; they cannot interact with the blackboard simultaneously as this would cause confusion. The intuitive way to impose control is to nominate an independent arbiter, who decides which expert should contribute next. However, a consistent basis for making that decision is still needed. The arbiter needs to be able to assess the benefits of the potential contributions in some way. One method of proceeding is to ask all of the candidate experts to estimate the value of their contribution, selecting the one who makes the highest estimate.

The scenario described above has some important properties. First, the experts act independently and are self-contained. Their approach to solving the problem and the knowledge they possess about it (in terms of their prior experience, the level they have reached and the focus of their expertise) can differ vastly, which means that very diverse contributions can be made. Second, the problem is solved in an opportunistic fashion, i.e. the flow of expert contributions is not predetermined or prescriptive. They respond when they have something worthwhile to contribute based on their perception of previous contributions. This is also an event-based method of problem-solving. The event that triggers an expert to ask to contribute is the writing of a partial solution (or other information) that allows that expert to apply his or her knowledge to the problem.

1.3.2.2 Components

The blackboard AI architecture is analogous to the scenario described above. It consists of three main components:

- **The Knowledge Sources** (analogous to the experts in the human system)
- **The Control Component** (analogous to the arbiter in the human system)
- **The Blackboard** (analogous to the physical blackboard in the human system)

Each of these components is now described in detail.

Knowledge Sources
In the AI system, the contributions to the problem solution are made by a set of "artificial experts", i.e. a set of diverse problem-solving algorithms known as knowledge sources (KSs). As in the analogy, these programs are self-contained, operate completely independently from one another and may have different strengths and weaknesses to bring to the problem. Thus, within the system of systems, each KS may be viewed as a black box with its implementation details hidden from the others. A given KS may reason by any means, for example by top-down, bottom-up, goal-driven, data-driven or opportunistic methods [22]. However, each KS needs to be able to understand the information that is contributed by the others and also the current state of the problem-solving. Thus, the BBS requires a common information representation that is understood by all KSs. For example, a particular blackboard application may have one KS based on an artificial neural network (ANN) design, one that uses a forward-chaining production system, one that uses fuzzy logic reasoning and one that implements Dempster–Shafer models. The production system KS does not need to understand how the ANN works, but it needs to be able to interpret the contributions that are output by the ANN as the KSs are required to work together to solve the problem, as in the human analogy. If the KSs contribute different types of information, then each KS must be able to understand each of these different types.

The KSs contribute information when they are able (they have something to contribute) and when activated (they are selected for contribution). Each KS has a precondition attached to it to determine whether it should be activated and an action that describes what it will do when activated.

Control Component
In the AI system, the control component is analogous to the arbiter. It is independent of the KSs and is tasked with selecting an estimate of the "best" candidate KS to make a contribution to the problem solution. The mechanism for control in BBSs has evolved considerably over time but was fairly simple in early systems; each candidate KS was asked to determine an estimate of its cost (to system resources) and value (expected usefulness of the contribution). The control component merely combined these two metrics in some way to establish which KS should contribute next.

Blackboard
The component in the AI system that is analogous to the physical blackboard in the human system is also called a blackboard. However, in the AI system, this is a shared memory location that acts as a repository for all of the information about the problem including the input data, the problem statement and all of the KS-contributed information (solution path suggestions, partial solutions, alternative solutions, candidate complete solutions, final complete solutions, agreements and disagreements). The information deposited on the blackboard is often referred to as

a *hypothesis*. The blackboard component can be thought of as a global database for KS hypotheses.

The blackboard serves as the main, in-direct communication mechanism between the KSs; their outputs are recorded on it, and these are read from it and understood by the other KSs. In addition, contributions made to the blackboard (events) may trigger a response from one or more KSs.

Some BBSs have blackboards that are subdivided into some way to categorise the hypotheses that they hold. The division may take the form of levels, sections, areas, a hierarchy of levels (as in HSII) or some combination of these, and the information in each part of the blackboard may also be sorted in some way, for example alphabetically or by the time it was contributed. In addition, some systems use a number of blackboards for categorisation. The reason for subdivision is to enable information to be located efficiently. Early BBSs tend to retain all of the information placed on them for the entire duration of the program life cycle. Information that appears to be irrelevant is kept because its use often becomes apparent later on during the program execution. However, some later systems permit deletion of information that is regarded as "outdated".

1.3.2.3 Similarities to Other Systems

There are some similarities between BBSs and symbolic rule-based decision-making systems, also known as production systems (see [23]), for example expert systems. Rule-based systems consist of a semantic reasoning subsystem and a set of rules with preconditions and actions. The semantic reasoning subsystem takes an action from the rule set based on its relationship to the input data, i.e. a rule is fired when conditions in the input data match its precondition. The rules in these systems are like the KSs in BBSs, which are triggered under certain conditions, but rule-based systems differ in that the rules are interdependent, which reduces the flexibility of these systems. As stated above, KSs operate independently of one another and are thus highly flexible. BBSs were, in fact, first proposed as a generalisation of rule-based systems with "rules" that could take any format and trigger [18].

1.3.2.4 Algorithm

As in the analogy, the BBS algorithm proceeds in an event-based manner. Any change to the blackboard is an event, and KSs can also cause events when they write their hypothesis to the blackboard. Any event can result in a match with a KS precondition. When its precondition is matched, a KS becomes a candidate for contributing to the blackboard, i.e. executing its action, and it competes with other KS candidates to execute that action on the blackboard.

The control component is responsible for selecting one of the candidate KSs, based upon the current state of the blackboard. In some systems, the control component is also charged with managing the events that trigger the KSs, but its chief function is to rank the candidates and select the most highly ranked. The methods for rating and ranking the KSs differ between BBSs. Some of the different approaches are discussed in the literature review.

1.3.2.5 Properties

This section is subdivided into the advantages and disadvantages of BBSs. The main contenders in each category are summarised as follows. Advantages: BBSs are flexible and general and solve problems incrementally in a step-by-step manner. They are opportunistic in the way that they tackle the search space leading to more sophisticated searches. They can also handle both a wide variety of data types and imperfect data. Disadvantages: the methodology suffers from a lack of formalism. In addition, the BBS architecture does not scale down to more simple problems and scales poorly for large search spaces. Further details about each of these properties are provided in the following subsections.

Advantages

BBSs have an important advantage over more traditional problem solvers that use rules, for example production systems. Production systems are prescriptive; there is a set of order for firing the rules, i.e. all reasoning is either forward or backward chaining. In BBSs, the problem-solving is opportunistic in that the most appropriate KS is selected, for example in speech recognition words that are understood well can be used to limit the search for interpretations of poorly sensed words. In addition, the solution is built incrementally in a step-by-step manner in response to events on the blackboard; there is no predetermined prescription for the triggering of KSs. KSs respond to events as and when they happen, and it is the interaction between the KSs (via the blackboard) that determines the solution paths explored and thus the final solution. The exploratory properties of the BBS search are therefore high in comparison with rule-based systems, which results in a much more sophisticated search of the solution space. In fact, BBSs show emergent properties in that their behaviour is governed by the interactions between a number of independent, interacting agents (the KSs) that follow relatively simple rules.

The BBS architecture is modular and is thus highly general and flexible when it comes to solving knowledge-based problems. As the KSs are independent, they can be adapted, added and removed from the system to improve its performance or to adapt it to solve other problems. This is in contrast to rule-based systems, where, for example, removal of a rule might affect the logic of the system leading to poor performance.

The control component and the structure of the blackboard can be designed in many different ways, permitting much freedom for the BBS application developer. BBS applications can also be built incrementally because of the system flexibility.

When designing, the application a developer may thus postpone decisions about, for example, which KSs to include or what control strategy to employ, until the system has reached an appropriate level of experimentation and testing. This is especially important for novice developers, who are learning to build BBS applications.

Another major advantage is the ability of BBSs to handle quite severe limitations in input data and also to handle a wide variety of data types. They are capable of dealing with ambiguity, incomplete data, uncertain and imprecise data and large quantities of data. They can also work with and can integrate different data types, which makes them particularly suited to the solution of problems that require the combination of many separate diagnostic components, dynamic decision-making problems, data fusion problems and problems involving systems of systems. They can also work with data that arrives asynchronously and sporadically [17].

BBSs are also capable of following multiple lines of reasoning concurrently. Once the relationships between hypotheses are established, each can be linked to others that justify or clarify it.

Disadvantages

Unfortunately, the chief advantage of BBSs, flexibility, is directly related to their main disadvantage in that it has led to a lack of formalism in defining them. This drawback is often cited as the major reason for their abandonment following the "AI Winter" (around 1990), where more formal frameworks were generally adopted for dynamic decision-making. The dismissal of less formal methods has had an impact on the perception of BBSs and their place in AI. Notably, BBSs receive only three sentences in Russel and Norvig [24], which is widely regarded as the leading textbook on AI, and in the latest edition [25], there is no mention of them. BBSs were recast as Bayesian Blackboard Systems (BBBSs) in response to this drawback (see [26]).

BBSs have traditionally been used for solving problems that are complex and ill-defined. Erman has suggested that the advantages of BBSs do not scale down to simpler problems [19]. Thus, unless a problem is sufficiently complex to warrant use of a BBS, then it is best to use a simpler architecture. A simpler architecture is also favourable when dynamic decisions do not need to be made or if an application contains only one system. Another well-documented drawback is the difficulty in estimating the value of potential KS actions. This has led to the design of more sophisticated control components for BBSs and also, later, their integration with Bayesian Belief Networks. These developments are discussed further in Sects. 3.3 and 3.4.1.

Although the early BBSs demonstrated that they worked well with complex, dynamic decision problems, they ran into difficulties when the search space became too large [20]. In fact, this was another reason for their decline in the 1990s, as the processors of the time were not powerful enough to deal with the extra computation burden.

1.3.2.6 Definitions

The flexibility that is a key feature of the BBS design has led to some confusion about its definition and some of the terminology surrounding it. This section aims to clarify some of the misunderstandings.

First, the game AI community and some other sources tend to define BBSs as merely a means for sharing common data among subsystems, i.e. the view is that they can be thought of simply as a global database or communication medium. In the seventies, eighties and nineties, this view was very much frowned upon; for example, Corkill states that a system with a global database also requires a set of KSs and a control component to be classed as a BBS [19]. However, as BBSs are now widely used in game development, this definition has become somewhat acceptable. Tuple spaces, a form of distributed shared memory, are also often confused with BBSs (see [27]). They provide a repository for tuples that can be accessed concurrently by, for example, different processors, but there is no requirement for a set of KSs to work together to solve a problem. Tuple spaces are said to use the blackboard metaphor (referred to in this chapter as the blackboard analogy), but the metaphor is, in fact, only partially implemented.

Second, there has been a lack of agreement between authors regarding some of the terminology. To counteract this, Engelmore has listed a set of definitions to distinguish between blackboard systems, models, frameworks, applications, architectures and shells [28]. This chapter adopts most of the definitions in Engelmore with some slight modifications. The terminology used in this chapter is summarised as follows: the term *architecture* is used to refer to the general core design and set of components of the blackboard AI method. The term *method* refers to the procedural details of the algorithm, which effectively represent a computational interpretation of the analogy [29]. The term *system* refers to a particular design implementation; for example, HSII is a blackboard system (BBS) with a specific design for the control component. *Application* is used to refer to the use of a given BBS to solve a particular problem; for example, HSII is an application when it is used to solve the speech recognition problem. *Framework* is used to refer to commercial or academic tools that enable a developer to build a BBS based on, for example, supplied program methods and subroutines. *Shells* are synonymous with *systems* in some sources (e.g. [30]), but the term is not used in this chapter.

1.3.2.7 Problem Types and Applications

This section examines the types of problems that are particularly suited to solution using a BBS. The discussion in the section "Analogy" highlighted the advantages of BBSs and linked the flexibility of allowed input data (e.g. incomplete and uncertain data), the modular design and the incremental solution-building approach with capacity for solving problems that require the combination of many separate diagnostic components. Thus, case-based reasoning and dynamic decision-making problems such as speech recognition, signal understanding, symbolic learning,

robot vision, image understanding, structure identification, vehicle monitoring and tracking, robot map creation, fault and disease diagnosis, planning and scheduling, and information fusion problems are all suited to solution using a BBS. For example, a robot building a map of an area may rely upon infrared and sonar sensor data to attempt to trace an image of its environment. Both of these data types are unreliable, and the environment may be changing dynamically. However, a BBS is able to fuse the data from the two input types as it arrives, make sense of the resulting information and reason about it.

In general, any problem with a number of different, interacting components that requires the integration of diverse expertise is a candidate BBS problem. This includes complex problems with very large search spaces and problems that suffer from combinatorial explosion, as well as ill-defined and poorly structured problems, for example problems where the goals are not clear or where the solution path from the initial state to the goal state is highly irregular [21]. Cooperating agent systems and distributed and parallel AI problems are also good candidates for solution with a BBS. It is the integration of several different KSs that makes BBSs applicable to the solution of a wider variety of problem types. Most other symbolic AI methods, for example production systems, can only deal with knowledge about a single problem domain. This tends to limit them to the solution of diagnosis-type problems, which are much simpler than interpretation-type problems such as speech recognition. The reasons for this are discussed further in the section "Combination with Bayesian Belief Networks". Furthermore, many production systems are only capable of working offline; they cannot handle the arrival of new information on the fly, whereas BBSs have this capability.

It is also possible to solve problems with hard, real-time constraints (i.e. problems with deadlines for the solution) using the BBS model (see [31]), although more sophisticated and predictable control strategies are needed to allow accurate estimation of action durations, as the computation time and use of other resources need to be considered. The system also needs a mechanism for assessing the trade-offs between cost and effectiveness of actions. This requires it to be able to predict resource usage [20].

The literature review provides examples of BBSs solving some of the problem types listed in this subsection. It is presented in chronological order so that it also acts as a guide to the "History and Evolution" of BBSs as well.

1.3.2.8 BBS Frameworks and Toolkits

There was very little software to support the development of BBSs for a long time after their conception. Thus, initially, many researchers and application developers had to build them from scratch [19]. Hearsay-III (see [32]) was developed in response to the demand for a domain-independent framework upon which to build a BBS. The framework, which sat on top of a relational database called AP3, was used to develop many other systems including a BBS that was applied to a crisis management task (see Hayes-Roth et al. [33]). Another early development toolkit

was Corkill's GBB framework (see [34], which was used to build a satellite control system for the Canadian Space Agency. It was also used for logistics planning in the US Army and for design engineering projects at Ford. Many toolkits are now available commercially, for example BEST (Blackboard-based Expert System Toolkit), designed by the Mihailo Pupin Institute. BEST, which is designed to run in the Windows environment, is implemented using C++ and Arity Prolog and uses the MEKON inference engine. Many open-source resources are also available to assist developers; for example, the source code, a toolkit and a software development kit for the computer game "No One Lives Forever 2" (NOLF2, developed by Monolith Productions), which implemented a BBS architecture, are available (see [35]).

1.3.3 Literature Review

BBSs began with the Hearsay-II (HSII) speech recognition system built at Carnegie Mellon University in the early 1970s, although the notion of an "artificial blackboard" being used to share ideas to solve a problem was first conceived by Allen Newell in 1962 (see [36]). The success of the Hearsay-II project inspired many other researches to adopt the blackboard architecture for applications such as signal interpretation, for example in the HASP/SIAP systems [15], and planning [37]. The model became very popular throughout the 1980s for solving complex AI problems, both with academic researchers and also in industry where it was employed by, for example, Cambridge Consultants Ltd. as the BLOBS system to implement reasoning about time-dependent data [38] and as the MUSE system to solve real-time problems [39]. Fujitsu in Japan also developed a BBS called ESHELL for problem diagnosis in cranes, as well other applications (see [28]).

There has been considerable development of the architecture since the early systems. Many of the initial enhancements concentrated upon modifying the control component, and there was a general move away from data-based control towards more goal-oriented control [20]. Other researchers focused on extending the architecture to distributed applications using, for example, networks of BBSs [40]. Parallel BBSs were also investigated in [41] and by Bell Laboratories (see [42]). BBSs became very scarce in the academic literature following the early 1990s. This has largely been attributed to their lack of formalism, but may also be related to the move of many of the early BBS proponents and researches from academia to industry. BBSs made a reappearance in the literature from around the year 2000 when several researchers attempted to merge the architecture with Bayesian Belief Networks (see, e.g. [16]) in an attempt to give it more rigour. Although academic papers about BBSs remain fairly scarce, the architecture continues to be very popular today in commerce and industry, both in the Bayesian hybrid format and in formats that are very close to the original architecture. For example, they are widely used by the computer gaming industry to control non-player character

(NPC) behaviour (see [35]), are used by Adobe to recognise text and have been used to manage satellite operation in Canada.

1.3.3.1 Prehistory

Allen Newell was the first person to use the term "blackboard" in AI literature [36]. In his 1962 paper about organisational problems in programs such as chess-playing programs, he refers to a set of workers looking at the same blackboard, each capable of reading what is on it, writing to it and judging when they have something worthwhile to contribute to it. He likens the situation to Selfridge's Pandemonium [43], where a set of demons shriek with varying levels of loudness in response to what they see. He goes on to discuss ways to organise the synthesis of complex processes using hierarchically organised subroutines [44]. These early ideas eventually led to the development of production systems with preconditions and actions. Interestingly, the term "demons" is still used to describe the set of rules with satisfied preconditions in production systems.

The term "blackboard" was mentioned again by Herbert Simon in 1966 in an article that was later published in 1977 [45]. In this article, the information generated about problem-solving that was fixed in permanent memory was referred to as the "blackboard". The article also talks about the creation of subgoals and the use of a hierarchy of goals and subgoals to achieve the original overarching goal. Simon suggested his ideas about blackboards to Raj Reddy and Lee Erman, when they were preparing for the Hearsay project, although many of the ideas that eventually emerged from Hearsay were centred around the needs of the application, i.e. speech understanding [44]. Thus, most of the core properties and components of blackboard systems, such as opportunistic problem-solving, different levels of abstraction and KS collaboration, were derived directly from those needs.

1.3.3.2 The First BBS—Hearsay-II

This section looks in detail at the system that is widely regarded as the first BBS, Hearsay-II (HSII). HSII was created for speech recognition, evolving directly from Hearsay-I (HSI) (see [46]). HSI was a prototype blackboard system, but is probably not cited as the first BBS because there was no dynamic control component and also because it did not work well. In HSI, information sharing among the KSs was carried out only at the word level, so it was difficult to add non-word KSs and determine the value of their hypotheses [47]. KSs were also activated using a hypothesise-and-test paradigm, rather than by a control component that dynamically selected the most appropriate KS.

The most popular reference for HSII is [14], which describes the final developed BBS. Earlier incarnations of HSII implemented a data-directed control approach, where KSs were activated in response to events on the blackboard if their preconditions were satisfied, and all KSs were checked for this. The data-driven

approach tries to answer the question "what should the system do given the available data?"

The final HSII system (described in [14]) has an agenda-based control mechanism referred to as the *scheduler*. Rather than checking the preconditions of all the KSs, which can be time-consuming, this incarnation of HSII categorises events into types and the KSs provide lists of the event types they are interested in. The scheduler then needs only to check the preconditions of KSs with event types that match the given blackboard event. KSs with matching types and satisfied preconditions are instantiated, are referred to as Knowledge Source Instantiations (KSIs) and are placed on the agenda. Note that some sources refer to Knowledge Source Activation Records (KSARs) rather than KSIs, but the terms are interchangeable. Once instantiated, the scheduler has access to information about the action each KSI would execute and the likely changes this would make to the blackboard. The agenda thus consists of all the actions that the system could possibly take next. The scheduler chooses the best KSI action based upon ratings of its contribution and cost, i.e. there is an estimate of how much progress it is likely to make towards solving the problem and another estimate of computational cost; the winning KS is then removed from the agenda. The cycle repeats until a different event takes place or there are no more KSIs on the agenda. The scheduler's rating calculation is a weighted linear function of several variables and is known as the expected value of the KSI; the result of the calculation tells the system which particular line of reasoning it is best to pursue; i.e., the hypothesis with the maximum expected value is the one considered worthy of further investigation.

The blackboard in HSII consists of a hierarchy of levels; for example, there is phrase level, a word level and a syllable level. There are also classes for each hypothesis associated with each level, and the levels contain a set of dimensions so that information can easily be retrieved when needed.

The KSs consist of acoustic, lexical, syntactical and semantic reasoning systems [17]. These KSs examine the blackboard for information that they can work with, for example hypotheses about adjacent words. When activated, a KSI posts a new hypothesis onto the blackboard. For example, a KSI may use the rules of grammar to generate words likely to appear next in a phrase or sentence, and another may actually detect words directly from the source. When a KS posts a hypothesis, which could be a partial solution, it then attempts to verify it in a test stage, where it may be refined. The entire process represents a search for a hypothesis that explains the data at each level of abstraction [17].

ARPA funded the speech recognition umbrella research program of which Hearsay was a part, and at the end of the project, the various systems that emerged were analysed and compared [29]. HARPY (see [48]), which employed a Markov algorithm to perform its analysis, was deemed the best in terms of performance, although it was not as flexible as HSII. Control was generally considered a problem in HSII because the decisions were based upon the scheduler's rating of the local and immediate effects of KSI actions. Using such a limited expected value, function can lead to poor performance because actions are not independent of one another; indeed, they can have very complex interrelationships. A more accurate expected

value would depend upon when the actions were executed and the next actions in the sequence. Thus, in HSII, the global effects of actions and the long-term state of the blackboard had to be abstracted by using models of the intermediate state to predict them. This was a way around the problem, but what was really needed was a method for linking potential actions and goals. HSII struggled because it was forced to deal with uncertainty about whether a given hypothesis was part of a solution and also with uncertainty about the expected values of actions. This led the HSII team and other researchers to enhance the control mechanism in HSII; effective control is especially important when uncertainty of the input data and problem-solving knowledge is high [20].

1.3.3.3 Development of Control Mechanisms

The BBSs that followed HSII tended to adapt and improve the control component in some way because of the flaws in the original design. In addition, HSII was only capable of handling a single input phrase and could not deal with strict scheduling deadlines. It thus became necessary to create more elaborate control mechanisms to address these problems since multiple inputs could potentially overload the blackboard and the agenda, and predictable control structures were needed for hard time constraints to allow accurate estimation of action durations.

In general, there was a move away from implicit representation of goals, as in HSII, towards more explicit representation of goals and their relationships to the overarching goal, so that more efficient and sophisticated goal-oriented control mechanisms could be implemented [20]. Thus, goal-directed reasoning, which can be defined as reducing a problem from a set of abstract high-level goals into more detailed low-level subgoals and planning, was introduced to BBSs. In contrast to the data-driven approach, the goal-directed approach tries to answer the question "what should the system do to solve the problem?" This is essentially achieved by identifying sequences of actions capable of satisfying goals and subgoals. The system terminates when all the goals have been achieved, and the resulting solution is deemed acceptable in some way. There may be other solutions that meet the constraints of the input data, but the system must be capable of selecting the "best" one. Thus, many different search strategies can be employed, for example a focused depth-first search that pursues a particular solution if it is preferred, or a more exhaustive breadth-first search that pursues all potential solutions until a particular one is favoured.

The control mechanisms that evolved from HSII began to use methods for making more accurate predictions about the long-term effects and global value of an action by representing goals in a more detailed and explicit way. They also began to limit the rerating of KSIs by restricting the number on the agenda to those more likely to be executed, increasing the efficiency of the algorithm. This subsection chronicles the development of the control component from its agenda-based approach in HSII through to the event-based approach seen in HASP/SIAP [15], the hierarchical approach used in CRYSALIS [49, 50] and the goal-directed

architecture of DVMT [40], which also included incremental planning. The BB1 [22] and RESUN [51] and Carver and Lesser [52] control mechanisms are also discussed as further extensions to the goal-directed approach.

Event-based control in HASP/SIAP

The HASP/SIAP BBS (see [15, 53]) was built to interpret sonar signals and identify the ships and submarines that produced them. The control architecture extended that of HSII by specifying the KS preconditions as predefined event types, so that, in effect, preconditions and event types were merged. Thus, as soon as an event occurred, the activated KSs were immediately known making the control algorithm much more efficient. The system used blackboard events, clock events (a time and a set of events expected to occur at that time), expectation events (events expected to occur in the future) and problem events (e.g. missing information). HASP also used a limited hierarchical control structure to distinguish domain knowledge from knowledge about its application; this was a precursor to the hierarchical control structure used in CRYSALIS.

The main disadvantage of the modified control mechanism in HASP was that opportunism in the system became more limited because the predefined event types governed the KS sequence. In HSII, the KSs were triggered in an ad hoc fashion by general events.

Hierarchical control in CRYSALIS

CRYSALIS (see Engelmore and Terry 1979; [49]) was used for protein crystallography. Its control mechanism was built on a hierarchy of control knowledge sources (CKSs) that were tasked with selecting the domain knowledge source (DKS) to be executed. There were two levels of CKS, a single strategy CKS and a set of task CKSs corresponding to the system subgoals. The strategy CKS selected a sequence of task CKSs for execution, which in turn selected a sequence of DKSs for execution. This technique eliminated the need for KS preconditions. The strategy CKS looked at the current blackboard hypotheses and made a decision about where to focus the problem-solving, i.e. determined which subgoals had the maximum expected value and should be pursued next; it thus provided coarse focus for the problem-solving. Its selection led to the sequential execution of a set of task CKSs, which provided the fine focus. Each task CKS in the sequence examined the conditions on the event list and selected appropriate DKSs for sequential execution. As in the core BBS architecture, the DKSs examined the blackboard in order to add or change hypotheses.

The opportunism was reduced in CRYSALIS as in HASP because there was no method to change focus once a path forward was decided. In HSII, switching between paths was simple. Moreover, HSII could pursue multiple lines of reasoning concurrently without a need for backtracking to a previous one. In contrast, CRYSALIS could follow only one line of reasoning at a time and thus had to backtrack when required. However, this was not a major problem for CRYSALIS because it was not designed to solve real-time problems. When researchers began to require the solution of such problems using BBSs, a different approach to control was necessary.

Goal-directed control in DVMT

A goal-directed approach to control was first used for the application of distributed vehicle monitoring in the DVMT BBS (see [40]). This system inserted a goal blackboard and goal processor (for creating goals on the goal blackboard) into the core BBS architecture. The goal processor employed three mapping functions: hypothesis-to-goal, goal-to-subgoal and goal-to-KS. Data-directed goals were created on the goal blackboard following the addition or amendment of a hypothesis on the domain blackboard and use of the hypothesis-to-goal mapping. Goal-directed subgoals were created on the goal blackboard after the creation of others via the goal-to-subgoal mapping. KSs were selected to have their preconditions checked in response to the insertion of a goal on the goal blackboard and use of the goal-to-KS mapping. The resulting KSI ratings incorporated both a data-directed and goal-directed element by using information about the super-goal, the hypothesis that gave rise to the goal and the level of the blackboard. DVMT succeeded in representing explicit goals and the global effects of actions by linking local effects with higher-level goals.

Control in BB1

In 1986, Hayes-Roth published work on the application of a BBS to solving arrangement-assembly problems [22]. The task was to arrange a set of given objects such that a given set of constraints was satisfied. The BBS with domain-independent knowledge about arrangement assembly was named ACCORD, and several other BBSs were also created based on the same design but with focus in a particular domain; for example, the PROTEAN system was tailored to protein-structure analysis and the SIGHTPLAN system was used for designing construction site layouts. Both of these systems made use of ACCORD as a KS. Collectively, the set of assembly-arrangement BBSs were known BB1 systems.

BB1 modified the original HSII control architecture so that additional KSs were used to build the control plan for the system's behaviour. In BB1's control architecture, the domain problem and the control problem were both solved using the blackboard model. The architecture used CKSs as in CRYSALIS but also inserted a control blackboard. It implemented an agenda-based approach to solving the control problem as in HSII, but introduced a more complex control planning method where the CKSs incrementally developed control plans on the control blackboard. The overall architecture can thus be thought of as "blackboard within a blackboard". BB1 worked at three levels of abstraction: the strategy (long-term plans), the focus (goal) and the heuristic (rating function). The long-term plans were reduced to sequences of substrategies, which in turn were reduced to foci, a set of goals. Each focus had an associated set of heuristics that were used to rate potential KSI actions that matched the focus. These could be changed dynamically to suit different problem-solving stages. KSIs from both the DKSs and CKSs were placed on the same agenda and rated using the same function.

In addition to a more sophisticated control system, BB1 also introduced learning KSs to modify facts in the KS knowledge bases and provided an additional capability for explaining its actions. On each solution cycle, and in response to user

requests, it was capable of providing information about how the selected actions matched with the control plan.

The control architecture of BB1 did not compromise the opportunism of the system as in CRYSALIS and HASP, i.e. actions and plans were both implemented opportunistically. However, although BB1 succeeded in implementing the determination of high-level, long-term goals, it did not carry out goal decomposition.

Control in RESUN

Planning represents a search for the best solution and a search for the best way to find it. The RESUN BBS (see [51, 52]) extended the HSII BBS by replacing the agenda strategy and engineered complex rating functions with an incremental control planner that simply carried out a number of less complex searches to make the best decisions. It was also able to preserve the core BBS opportunistic properties as it incorporated an additional refocusing mechanism that permitted a posteriori changes to the planner focusing decisions. This enabled postponement of decisions when there was insufficient information about a plan, or when two plans could not be rated against each other without sufficient refinement. Refocusing was allowed at both a data/event level and at a planning/hypothesis level so that the system could switch focus in response to developing plans, new hypotheses and input data.

RESUN enabled better resolution of uncertainty in the hypotheses. It worked with detailed information about the uncertainty of hypotheses and their alternatives and used explicit statements about the sources of the uncertainty called source of uncertainty statements (SOUs). The SOUs were attached to hypotheses so that goals could be generated to eliminate the uncertainty.

RESUN was the most goal-directed control approach of all the BBSs discussed in these subsections. Most of the others simply added goal-directed methods to the agenda method. RESUN started with the goal-directed approach and used focusing to allow data-directed control when necessary for opportunism.

1.3.3.4 More Recent Developments

As mentioned at the start of this section, following the "AI Winter" around 1990, documentation of BBSs began to decline in the academic literature, largely due to a lack of formal underpinnings for belief in and decisions about actions. This meant that the systems could only be assessed empirically. Although they remained popular in industry, academic researchers were beginning to prefer non-symbolic (modern AI) approaches to problem-solving or more rigorous statistics-based approaches. In addition, there was a perception that BBSs were large and unwieldy requiring vast quantities of code to manage them and their complex data structures [18]. Another problem was that the initial successes documented in the literature did not scale well to larger dimensioned problems because the computers in use at the time did not have the storage or processing power required. Most of the problems

that had been solved with BBSs thus far were NP hard, which meant that solutions were not tractable in the large limit at the time. For example, in the speech recognition field, many researchers began to work with simple Hidden Markov models as they could outperform BBSs like HSII. BBSs were thus recast as Bayesian Blackboard Systems (BBBSs) in an attempt to define them in modern AI concepts [26].

During the "AI Winter" some researchers, no doubt dismayed by the sudden demise of BBSs, undertook work to begin to formalise BBSs. In 1991, Craig provided a relatively complete mathematical specification for a sample interpreter BBSs using the Z specification language [see 54]. Velthuijsen used a similar approach in 1992 but applied the specification to a number of concurrent BBSs using the CCS language [55]. Craig then extended his 1991 work to provide a formal account of BBSs that showed that control information could be derived and represented in temporal logic [56]. Following this, the interpretation problem was formalised by Whitehair (see [57]) and his paper also analysed BBS systems when applied to the problem. The paper demonstrated that it was possible to develop models for opportunistic AI architectures like BBSs.

Combination with Bayesian Belief Networks

A BBBS closely resembles a traditional BBS, with KSs, a blackboard and a control component. However, in a BBBS, the role of the KSs is to modify Bayesian Belief Networks (BBNs) on the blackboard rather than direct hypotheses (see [17]). The Bayesian component provides the hybrid architecture with a foundation in probability theory, validating the underlying reasoning, i.e. it allows probabilistic models (that can reason about uncertainty) to be built rather than symbolic ones. In addition, the integration not only formalises the BBS but extends traditional belief-based systems by allowing them to be built incrementally. However, Carver states that some of the flexibility and opportunism is lost in interpretation BBBSs that work in time slices as they have to compute exact probabilities for all interpretations of new data with no ability to search selectively [26].

A belief network is a directed graph where the set of nodes represents propositions with associated probability distributions (PDs), some of which may be conditional (dependent on the PDs of the nodes pointed to). Nodes with unconditional probabilities are termed evidence nodes. The conditional PDs are stored in tables known as conditional probability tables (CPTs).

One of the first attempts at integrating BBSs with BBNs was the AIID system (Architecture for the Interpretation of Intelligence Data), which was developed for information fusion in military scenarios (see [17]). The problem was to infer an enemy unit's strategy given military intelligence, and it required the system to be able to cope with heterogeneous data (of varying precision and reliability) arriving sporadically from many different sources. Thus, it was necessary to be able to understand the meaning of data when it arrived. This is not possible using a simple data-driven approach as the search space is too large. A method for reasoning about uncertainty was also needed. Traditional BBSs such as HSII dealt with uncertainty

implicitly and heuristically via the scheduler rating function. However, in AIID, each hypothesis was directly associated with a probabilistic uncertainty.

In AIID, the data corresponds to evidence with conditional probability information, and the belief network is dynamically created and grown as data are received and processed. The network consists of nodes that have a type and set of arguments that identify it, with nodes being associated with hypotheses, previous observations or background knowledge. In addition to preconditions and actions, the KSs have a confidence property that embodies their usefulness. When KSIs are activated, they alter the network in some way; for example, they may post new nodes to the blackboard, remove existing nodes, add edges or change conditional probabilities. The KSs themselves can also exist as belief networks that represent small fragments of knowledge. When a node in the fragment matches a node on the blackboard, i.e. their types and arguments match, a KS becomes a candidate KS and may post its knowledge fragment on the blackboard. After being posted, a knowledge fragment's two conditional PDs (the one from the KS knowledge fragment and the one on the blackboard) are combined. The evidence combination methods vary between BBBSs. As in traditional BBSs, only one KS is run at a time to maintain tractability, but the difference is that control is maintained by computing the expected utility of an action, given the available evidence.

BBBSs arose from a need to improve existing methods for solving complex interpretation problems (such as vehicle tracking, robot map making and speech understanding) as well as a desire to formalise BBSs. These problems are inherently much harder to solve than diagnosis problems as an interpretation requires that instances of input data types are explained in terms of hypotheses about events that might have given rise to them [26]. Moreover, there are multiple possible interpretations for a given set of input data, as there are many different instances of each type and many uncertainties because of data noise. The solution space can become exponentially large for high-dimensioned problems. A system that is able to reason about the validity of evidence for alternative hypotheses can thus make better judgements regarding which solution path to follow. Diagnosis problems do not share the same complexity as there is a single, fixed set of hypotheses, for example possible faults in engine fault diagnosis. Furthermore, the relations between the data and the hypotheses are known, and complete, static probability models are easily constructed. Interpretation problems, on the other hand, lack a model for connecting data instances with hypotheses. This phenomenon is known as the data association problem (DAP). BBBNs that attempt to solve interpretation problems can thus only work with estimates of conditional probabilities, and when performing evidence propagation, the solutions can only approximate the optimal [26].

Other recent work

Culliton describes the hypothetical use of a BBS to coordinate intelligent units in a combat computer game [58], referring to BBSs as "the perfect system to use", although he states that, initially, the complexity of the architecture prohibited its use in game design because of limited time, resources and budgets. However, there is documentation of the use of a BBS for coordinated behaviour in the game NOLF2,

developed by Monolith Productions in 2002 (see [35]). He states that a BBS was used to handle coordination with respect to the timing of behaviours, pathfinding and tactics. For example, the architecture solved problems such as preventing duplication and repetition of behaviour in agents. Contrary to the belief that BBSs are "code-heavy", Orkin stresses that use of a BBS allows a reduction in code volume, is simple to implement and is flexible and maintainable. Furthermore, he asserts that it simplifies the agent architecture, permits reuse and sharing and allows complex reasoning to take place. This conflict of opinion may have arisen because Orkin implemented a more simplified version of the classic BBS, whereas Culliton was discussing the original, more complex architecture. Indeed, Dill reports that the term "blackboard architecture" is used differently in game AI than in the academic literature [59], often being used merely to describe shared memory space that AI components can use to store knowledge. He cites line-of-site (LOS) checks, path-planning checks and the coordination of AI components as common uses for BBSs in game AI, emphasising that the latter may require the more complex, classic use of BBSs, i.e. independent KSs posting partial solutions to a problem on the blackboard rather than merely using it as shared memory space.

Millington and Funge on the subject of AI for games, reporting that this method has been used extensively by game programmers as a mechanism for coordinating the actions of several independent decision-makers [18]. Their description of BBSs implies that many systems developed for games follow a more simplified form of the more classical architecture as seen is HSII, rather than the modernised Bayesian hybrid architecture. This makes sense as there is generally a less rigorous requirement in game AI; the purpose behind it is to create believable behaviours rather than to solve complex problems with large search spaces. Millington and Funge cite a typical example use of BBSs in a computer game, ballistics planning, where three different AI systems work with a blackboard to fire at enemy tanks. There is a route planning subsystem, a target selection subsystem and a ballistics calculation subsystem. Running each system sequentially is not efficient as the game environment changes rapidly, and this approach does not allow information to flow back in the opposite direction. There is a need for all of the AI subsystems to communicate freely without having to set up individual communication channels. The solution to this problem is to use the blackboard architecture. Suggested actions are written to the blackboard by the subsystems where they are stored along with agreement flags. Actions are only executed if there is full agreement between relevant subsystems. For example, "fire at tank X" would have an agreement slot for the ballistics subsystem. The ballistics subsystem could agree, disagree or even remove the suggested action from the blackboard. It could also post a new suggested action, for example "move into firing position for tank X", leaving the original proposition still on the blackboard, but deferring agreement until the correct position had been reached. This example strongly suggests the game AI definition of BBSs, i.e. the shared data paradigm. Champandard also discusses the use of BBSs for computer game behaviour coordination [60], citing their main advantage in game AI as their modularity; as the various systems that need coordinating are independent, they only interact by exchanging information on the

blackboard. Champandard assets that this reduces the coupling between them, making the coding structure much more straightforward. Another article that provides insight into the use of BBSs for game AI is David Mark's account of an interview with Damian Isla, a developer at the MIT Media Laboratory who worked on games such as Halo 2, released in 2004 (see [61]). Mark reports that Isla views BBSs as architectural constructs for decoupling information gathering and storage from the decision-making process; multiple decision-making subsystems can work with the same information simply by looking at the blackboard; for example, the threat level of an enemy character can be calculated once and saved for all the subsystems to work with. Isla also likened the blackboard to a unified interface for all game data, storing the data in a contextually significant way so that relevant subsystems know what it means. This is an allusion to the definition of BBSs in the game AI sense of a shared data source for different decision-making components.

Aside from game AI, there have been other recent uses of BBSs. For example, Khosravi and Kabir used a classic BBS integrated with offline ANN training for optical character recognition (OCR) of typed text in the Farsi language [62]. In this BBS, some of the KSs that were used were generated offline a priori as a training exercise. These took the form of multilayer perceptrons (MLPs), a type of ANN. Other KSs, mostly classifiers boosted by the MLP training, ran online. In addition, some KSs were static and some were changed dynamically during run-time. The control component used confidence values that were usually taken directly from the classifier outputs to rate the value of potential actions on a scale 0 through 100. As an example, one KS was a segmentation and recognition module with the task of breaking words down into individual characters and then recognising them. Another KS was a vocabulary "expert" containing 55,000 words. Its task was to match recognised words with words in its database. If a recognised word did not exist (e.g. because of misclassification of characters), then it would find the word most similar to the recognised one. The system also used tools to help with the text recognition, for example a spellchecker and a line detector. The system was capable of recognising 10 popular Farsi fonts and was tested on 20 real-life documents producing a recognition rate of about 97% at the word level and about 99% at the character level.

Fox provided an interesting extension to BBBSs in 2012 by using a hierarchical Bayesian Blackboard integrated with a Metropolis–Hastings algorithm and a BBN for map building with a whiskered robot known as CrunchBot [16]. The system was required to process very large quantities of sparse sensory information in order that CrunchBot could recognise table-like objects (such as tables, chairs, desks) in its environment, using only four whisker-like tactile sensors. The solution architecture was composed of a hierarchical BBS that implemented hypothesis priming and pruning heuristics integrated with a Metropolis–Hastings algorithm and a Monte Carlo Markov chain (MCMC) sampling Bayesian network. The observations for the MCMC were the position and orientation data (with respect to the contact surfaces) from the whisker sensors. This information was fused with information about the pose of the robot and hierarchical models relating to furniture objects (e.g. the recognition of a table leg object infers the presence of a complete table object)

in order to make inferences about the objects encountered and thus build the map. Each time step was treated as independent inference problem.

The problem tackled was difficult as many incorrect hypotheses can arise when sparse data are used, and there was often not enough information about the furniture objects to resolve ambiguities. The authors previously tried to solve the same mapping problem using particle filtering techniques, but this did not produce the same level of success as the BBBN system.

1.3.4 Summary

BBSs have been used extensively for solving a wide variety of complex, uncertain, real-time, dynamic and ill-defined problems, and over the many years since their conception, they have produced some excellent results. Their problem-solving power lies in their flexibility, modularity, incremental solution-building approach, and their ability to handle imperfect data, large quantities of data and data arriving sporadically and/or asynchronously. They have proved to be a very valuable tool in domains that require complex, multidimensional searches, for example complex scheduling and interpretation problems. Moreover, the optimal solution of such problems remains an open research question, which means that there is ample scope and opportunity for further research into BBSs, with many different potential design choices for developers. Although the blackboard architecture is not as popular with the academic community as it was in its early days, it remains an excellent vehicle for interesting research projects with a lot to offer for academics, industry and the game AI community. Symbolic methods may have declined, but the BBS, even in its core incarnation, is a hybrid architecture capable of incorporating elements of both modern and classic AI approaches. A recurrent theme throughout the BBS literature is that the control of the system is at least as important as the domain knowledge if the system is to be useful, effective and efficient.

References

1. Pearl J (1998) Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers Inc., San Francisco
2. Trovati M, Bessis N (2015) An influence assessment method based on co-occurrence for topologically reduced sets. Soft Comput 20(5):2021–2030
3. Sanchez-Graillet O, Poesio M (2004) Acquiring from text. LREC
4. Kuipers BJ (1984) Causal reasoning in medicine: analysis of a protocol. Cogn Sci 8:363–385
5. Trovati M (2016) An overview of some theoretical topological aspects of big data, Big-Data analytics and cloud computing, theory, algorithms and applications, computer communications and networks, Springer
6. Liddy ED (2001) A robust risk minimization based named entity recognition system. In: Encyclopedia of library and information science. Marcel Decker, Inc., New York

7. Laporte E (2005) Symbolic natural language processing. In: Lothaire (ed) Applied combinatorics on words, pp 164–209
8. Manning CD, Schutze H (1999) Foundations of statistical natural language processing. MIT Press, Cambridge
9. Troussov A, Levner E, Bogdan C, Judge J, Botvich D (2010) Spreading activation methods. In: Dynamic and advanced data mining for progressing technological development: innovations and systemic approaches, pp 136–167
10. Dale R, Moisl H, Somers HL (2000) Handbook of natural language processing. Marcel Dekker, Inc., New York
11. Korhonen AYK (2006) A large subcategorisation lexicon for natural language processing applications. In: Proceedings of LREC
12. Stumme G (1998) Efficient data mining based on formal concept analysis. Lecture Notes in Computer. Springer, New York
13. Wilks Y, Stevenson M (1998) The grammar of sense: using part-of-speech tags as a first step in semantic disambiguation. Nat Lang Eng 4:135–143
14. Erman LD, Hayes-Roth F, Lesser VR, Reddy DR (1980) The Hearsay-II speech-understanding system: integrating knowledge to resolve uncertainty. ACM Comput Surv 12(2):213–253
15. Nii HP, Feigenbaum EA, Anton JJ, Rockmore AJ (1982) Signal-to-symbol transformation: HASP/SIAP case study. AI Mag 3:23–35
16. Fox CW, Evans MH, Pearson MJ, Prescott TJ (2012) Towards hierarchical blackboard mapping on a whiskered robot. Robot Auton Syst 60(11):1356–1366
17. Sutton C, Morrison C, Cohen PR, Moody J, Adibi J (2004) A Bayesian blackboard for information fusion. In: Svensson P, Schubert J (eds) Proceedings of the seventh international conference on information fusion, pp 1111–1116
18. Millington I, Funge J (2009) Artificial intelligence for games, 2nd edn. CRC Press, Boca Raton, pp 459–466
19. Corkill DD (1991) Blackboard systems. AI Expert 6(9):40–47
20. Carver N, Lesser V (1994) Evolution of blackboard control architectures. Expert Syst Appl 7:1–30
21. Pang GK-H (2009) Blackboard architecture for intelligent control. In: Unbehauen H (ed) Control systems, robotics and automation: and intelligent control systems, vol 17. EOLSS, Oxford, pp 303–316
22. Hayes-Roth B, Johnson V, Garvey A, Hewett M (1986) Application of the BB1 blackboard control architecture to arrangement-assembly tasks. Artif Intell 1(2):85–94
23. Newell A, Simon HA (1972) Human problem solving. Prentice-Hall, Englewood Cliffs
24. Russell S, Norvig P (1995) Artificial intelligence: a modern approach. Prentice-Hall, New Jersey
25. Russell S, Norvig P (2016) Artificial intelligence: a modern approach, 3rd edn. Pearson, Harlow
26. Carver N (1997) A revisionist view of blackboard systems, In: Proceedings of the 1997 midwest artificial intelligence and cognitive science society conference. The AAAI Press, Dayton, Ohio
27. Gelenter D (1983) Generative communication in Linda. ACM Trans Program Lang Syst 7(1): 80–112
28. Engelmore RS, Morgan AJ, Nii HP (1988a) Introduction. In: Engelmore R, Morgan T (eds) Blackboard systems. Addison-Wesley, Boston, pp 1–22
29. Craig ID (1995) Blackboard systems. Ablex Publishing Corporation, Norwood, NJ
30. Jones J, Millington M, Ross P (1986) A blackboard shell in PROLOG. In: Proceedings ECAI-86, pp 428–436
31. Dodhiawala RT, Sridharam N, Pickering C (1989) A real-time blackboard architecture. In: Jagannathan V, Dodhiawala R, Baum L (eds) Blackboard architectures and applications. Academic Press Inc., San Diego

32. Balzar R, Erman L, London P, Williams C (1980) HEARSAY-III: a domain-independent framework for expert systems. In: Proceedings of the first annual conference on artificial intelligence, pp 108–110
33. Hayes-Roth F, Waterman DA, Lenat DB (1983) Building expert systems. Addison-Wesley, Reading
34. Corkill DD, Gallagher KQ, Murray KE (1986) GBB: a generic blackboard development system. In: Proceedings of AAAI-86, pp 1008–1014
35. Orkin J (2003) Applying blackboard systems to first person shooters. [online] slidepayer.com. Available at: web.media.mit.edu/~jorkin/utgameAI03-Orkin.ppt and http://slideplayer.com/slide/6102412/. Accessed 16 Apr 2016
36. Newell A (1962) Some problems of the basic organization in problem-solving programs. In: Yovits M, Jacobi G, Goldstein G (eds) Proceedings of the second conference of self-organising systems, pp 393–423
37. Hayes-Roth B, Hayes-Roth F, Rosenschein S, Cammarata S (1979) Modelling planning as an incremental, oppotunistic process. In: Proceedings IJCAI-79, pp 375–383
38. Zanconato (1988) BLOBS—an object-oriented blackboard system framework for reasoning in time. In: Engelmore R, Morgan T (eds) Blackboard systems. Addison-Wesley. Reading, pp 335–345
39. Reynolds D (1988) MUSE: A toolkit for embedded, real-time AI. In: Engelmore R, Morgan T (eds) Blackboard systems. Addison-Wesley, Reading, pp 519–532
40. Lesser VR, Corkill DD (1983) The distributed vehicle monitoring testbed: a tool for investigating distributed problem solving networks. AI Mag 4(3):15–33
41. Nii HP (1986b) CAGE and POLIGON: two frameworks for blackboard-based concurrent problem solving. Technical Report KSL-86-41. Stanford University, Stanford
42. Ensor JR, Gabbe JD (1986) Transactional blackboards. Int J Artif Intell Eng 1(2):80–84
43. Selfridge O (1959) Pandemonium: a paradigm for learning. In: Proceedings of symposium on the mechanisation of thought processes, pp 511–529
44. Nii HP (1986) Blackboard systems: the blackboard model of problem solving and the evolution of blackboard architectures. AI Mag 7(2):38
45. Simon HA (1977) Scientific discovery and the psychology of problem solving. In: Models of discovery, Reidel, Boston
46. Reddy DR, Erman LD, Neely RB (1973) A model and a system for machine recognition of speech. IEEE Trans Audio Electro Acoust AU-21:229–238
47. Engelmore RS, Morgan AJ, Nii HP (1988) Hearsay-II. In: Engelmore R, Morgan T (eds) Blackboard systems. Addison-Wesley, Reading, pp 25–29
48. Lowerre BT, Reddy R (1980) The HARPY speech understanding system. In: Lea W (ed) Trends in speech recognition. Prentice-Hall, Englewood Cliffs
49. Terry A (1988) Using explicit strategic knowledge to control expert systems. In: Engelmore R, Morgan T (eds) Blackboard systems. Addison-Wesley, Reading, pp 159–188
50. Englemore RS, Terry A (1979) Structure and function of the CRYSALIS system. In: Proceedings of IJCAI-79, pp 250–256
51. Carver N (1990) Sophisticated control for interpretation: planning to resolve sources of uncertainty. Ph.D. Thesis. University of Massachusetts, Computer and Information Science Department, Amherst
52. Carver N, Lesser V (1990) Control for interpretation: planning to resolve sources of uncertainty. Technical Report No. 90-53. University of Massachusetts, Computer and Information Science Department, Amherst, MA
53. Feigenbaum EA, Nii HP (1978) Rule-based understanding of signals. In: Waterman D, Hayes-Roth F (eds) Pattern-directed inference systems. Academic Press, New York
54. Craig ID (1991) Formal specification of advanced AI architectures. Ellis Horwood, Chichester
55. Velthuijsen H (1992) The nature and applicability of the blackboard architecture. Ph. D. Thesis. Faculty of General Science, Limburg University, Maastricht
56. Craig ID (1993) Formal techniques in the development of blackboard systems. Int J Pattern Recogn Artif Intell 7(2):197–219

57. Whitehair R (1996) A framework for the analysis of sophisticated control. Ph. D. Thesis. University of Massachusetts. Computer Science Department
58. Culliton P (2003) Implementing a blackboard-like system for squad-level combat AI Part I. [online] GameDev.net. Available at: http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/implementing-a-blackboard-like-system-for-squad-r1931. Accessed 15 Sept 2016
59. Dill K (2014) Structural architecture—common tricks of the trade. In: Rabin S (ed) Game AI PRO: collected wisdom of game AI professionals. CRC Press, Boca Raton
60. Champandard AJ (2007) Using a static blackboard to store world knowledge. [online] aigamedev.com. Available at: http://aigamedev.com/open/article/static-blackboard/. Accessed 16 Apr 2016
61. Mark D (2010) Damián Isla Interview on Blackboard Arch. [online] intrinsicalgorithm.com. Available at: http://intrinsicalgorithm.com/IAonAI/2010/02/damian-isla-interview-on-blackboard-arch/. Accessed 19 Sept 2016
62. Khosravi H, Kabir E (2009) A blackboard approach towards an integrated Farsi OCR system. IJDAR 12(1):21–32