Chapter 2

Knowledge Representation Using Formal Logic

Now we begin to dig into the nitty-gritty of our subject matter. Before discussing querying and analysis of complex, heterogeneous spatiotemporal and contextual knowledge, we must discuss representation of temporal knowledge (as well as, to a certain extent, spatial knowledge) . . . and before that, we must address knowledge representation in general.

In the course of our investigation we must work through a number of difficult questions regarding knowledge representation, including:

- Which of the many species of uncertain logic to use as the basis for our knowledge representation
- How specifically to represent temporal knowledge?
- How specifically to represent spatial knowledge?
- What is the best low-level (e.g. graph) representation of logical knowledge for efficient storage and processing?

"Logic" itself is not a monolithic entity; it comes in many different flavors. At the highest level, there is the dichotomy between predicate logic and term logic (and there are also systems that hybridize the two, e.g. (Goertzel *et al.*, 2008; Wang, 2006a)). There are also many types of logical system within each of these broad categories, some of which will be reviewed later on.

The material in this chapter becomes somewhat formal and technical, for which we apologize to the reader who lacks the relevant taste or experience; but which unfortunately seems unavoidable if we are to give a serious treatment of our topic. The reader lacking appropriate expertise may either consult relevant background material (Copi & Cohen, 1998), or less ideally, skim this material and proceed to the later chapters, some of which will be quite clearly comprehensible without grasp of these preliminaries, some less so.

2.1 Basic Concepts of Term and Predicate Logic

Term logic, or *traditional logic*, was founded by Aristotle and was the dominating logical framework until the late nineteen century. Term logic uses subject-predicate statements of the form "S is P" (for instance, "Socrates is a man"). There are singular and universal terms (the former correspond to unique subjects). There are just four forms of propositions in term logic:

- Universal and affirmative (e.g. "All men are mortal")
- Particular and affirmative (e.g. "Some men are philosophers")
- Universal and negative (e.g. "No philosophers are rich")
- Particular and negative (e.g. "Some men are not philosophers").

New conclusions are derived from premises by *syllogisms*. Aristotle introduced fourteen syllogisms, of which we will give just two here for illustrative purposes:

- (*Barbara*) If every M is L, and if every S is M, then every S is L. (for instance, "if every man is mortal, and if every philosopher is a man, then every philosopher is mortal")
- *(Celarent)* If no M is L, and if every S is M, then no S is L. (for instance, "if no philosopher is rich and if every poet is a philosopher, then no poet is rich").

Syllogisms provide a method for deduction – deriving new facts from already proved facts.

In addition there are rules for induction and abduction:

- *(Induction)* If every M is L, and if every M is S, then every S is L. (for instance, "if every poet is mortal, and if every poet is a philosopher, then every philosopher is mortal")
- *(Abduction)* If every L is M, and if every S is M, then every S is L. (for instance, "if every poet is mortal, and if every philosopher is mortal, then every philosopher is poet")

Notice that the induction and abduction rules do not neccesarily derive true statements. Nevertheless these are important forms of inference in the face of insufficient evidence, in modern AI reasoning systems as well as in classical Aristotelian term logic (Dimopoulos & Kakas, 1996). Induction and abduction are omnipresent in human commonsense inference.

Put simply, induction aims at *generalization*. In the above example ("if every poet is mortal, and if every poet is a philosopher, then every philosopher is mortal"), the first premise yields that all philosophers that are also poets are mortal, but then it is generalized to conclude that all philosophers are mortal. Yet, it is possible that there are some philoso-

phers that are not poets, so potentially not mortal, so the above generalization rule does not neccesarily lead to true conclusions.

Similarly, abduction aims at *explanation*. In the above example, the explanation for the fact that "every philosopher is mortal" may be that it is because "every philosopher is a poet".

In the late nineteenth century, classical term logic was the subject of criticism, for its weak expressive power and the limited forms of reasoning it permitted. For example, in classical term logic from "every car is a vehicle" one cannot infer "every owner of a car is an owner of a vehicle." In that period, predicate logic was designed, and it still serves as a basis for most mathematical and philosophical formal reasoning. However, modern theorists have extended classical term logic in various ways (Englebretsen & Sommers, 2000; Wang, 2006b), so that there are now term logics which equal predicate logic in expressive power. There are also systems that hybridize term and predicate logic, such as our own Probabilistic Logic Networks framework (Goertzel *et al.*, 2008), which will be discussed below. Advocates of term logic often argue that it more closely matches the patterns of human commonsense reasoning.

In standard *predicate*, or *first-order logic*, statements have arbitrary propositional form (involving conjunctions, disjunctions, negations, ...) and arbitrary use of quantifiers (for instance, "for every man, there is a woman, such that for every man, ..."). Modern variants of term logic provide this same expressive flexibility.

Pure predicate logic is a framework in which one can describe other theories. This framework is defined by the set of axioms and the set of inference rules (such as "if P and if P yields Q, then Q"). The proofs are sequences of derivation steps based on these axioms and rules. For example, one can represent in predicate logic and derive "not every man is a philosopher if and only if there is a man such that it is not a philosopher". For first-order logic there are also inductive and abductive rules, not used in mathematical theorem-proving, but for uncertain reasoning, most often in AI. First order logic is also used as a basis for many specific logics, including modal, deontic, temporal and spatial logics as will be discussed below.

2.2 Review of Propositional Logic

In order to explain predicate logic in more depth, we must begin with a simpler variant called "propositional logic." Propositional logic can express simple facts while first-order

logic (or predicate logic) also involves quantification and more complex statements. In this sense, first-order logic subsumes propositional logic. Both propositional and first-order logic have many practical applications beyond the ones considered here, in describing different processes and concepts. Most important perhaps are applications in computer science, ranging from chip design (Aehlig & Beckmann, 2007) to natural language processing (Meulen, 2001).

Both propositional logic and first-order logic have three important aspects:

- syntax – describing well-formed formulae and their basic properties;
- semantics – describing *meaning* of well-formed formulae;
- deduction – describing systems for syntactically deriving new formulas from other formulas (with no respect to their meaning).

We now give a brief mathematical exposition of these three aspects.

First, syntax. Let P be an infinite, but countable set, whose elements will be called *propositional letters* or *atomic propositions.* The set of *propositional formulas* is defined by the following rules:

- all elements of P are propositional formulas;
- $\bot$ and $\top$ are propositional formulas (which as we will see below, are normally taken to semantically represent False and True)
- if A is a propositional formula, then so is $\neg$A (which is normally taken to represent the negation of A)
- if A and B are propositional formulas, then so are A$\wedge$B, A$\vee$B, A$\Rightarrow$ *B*, A$\Leftrightarrow$B (which are normally taken to represent And, Or, and implication and equivalence)
- each propositional formulas is obtained by a finite number of applications of the above rules.

Next, semantics. A valuation v is defined as a mapping from P to the set $\{0,1\}$. That is, a valuation assigns either 0 or 1 to any propositional letter. An interpretation I_v is an extension of a valuation v, mapping propositional formulas to the set $\{0,1\}$, defined in the following way:

- $I_v(\bot) = 0$;
- $I_v(\top) = 1$;
- I_v (p)$=$v(p), if p belongs to P;
- I_v ($\neg$A)$= 1-I_v$ (A);

- $I_v\ (A \wedge B) = \min(I_v\ (A), I_v\ (B))$;
- $I_v\ (A \vee B) = \max(I_v\ (A), I_v\ (B))$;
- $I_v\ (A \Rightarrow B) = \max(1\text{-}I_v\ (A), I_v\ (B))$;
- $I_v\ (A \Leftrightarrow B)=1$, If $I_v\ (A)= I_v(B)$ and $I_v\ (A \Leftrightarrow B)=0$ otherwise.

The above semantics is referred as to Tarski's semantics (that he introduced in (Tarski, 1994)). Put simply, this allows the interpretation of propositional formulas like

$$A \wedge (\neg B \vee (C \Rightarrow A))$$

as having truth values drawn from the set $\{0,1\}$ (given the truth values for A, B and C), with 0 usually interpreted as meaning False and 1 as meaning True.

A formula A is satisfiable if there is a valuation v such that $I_v\ (A) = 1$ (otherwise, it is *unsatisfiable* or inconsistent, aka self-contradictory). A formula A is a tautology if for an arbitrary valuation v $I_v\ (A) = 1$. If a formula A is a tautology, then we denote that by $\models A$.

For example, it holds that $I_v(p \vee q) = 1$ in a valuation v such that $v(p) = 1$, $v(q) = 0$ – so the formula $p \vee q$ is satisfiable. On the other hand, the formula $p \vee \neg p$ is tautology.

The problem of checking the satisfiability of a formula made of conjunctions of disjunctions of literals (variables or negations of variables) is decidable (there is an effective algorithm for solving it) and is called SAT. SAT is on of the most important NP-complete problems. Programs for testing satisfiability are called SAT solvers. There are different methods for checking satisfiability of a formula: the simplest is based on truth-tables (i.e. tabular enumeration of all possible combinations of values for the formula's variables, and evaluation of the truth value of the formula for each combination). Other include Davis-Putnam-Logmann-Loveland (DPLL) procedure, and modern solvers – DPLL-based, resolution-based solvers, tableaux-based solvers etc. Modern SAT solvers can decide propositional formulas with thousands variables and clauses (Lynce & Marques-Silva, 2002).

2.2.1 *Deduction in Propositional Logic*

There is a number of inference systems for propositional logic. Most of them are actually restrictions of inference systems for first-order logic. Hilbert-style inference system consists of the following axiom schemes (Mendelson, 1997):

(A1) $A \Rightarrow (B \Rightarrow A)$
(A2) $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$
(A3) $(\neg B \Rightarrow \neg A) \Rightarrow ((\neg B \Rightarrow A) \Rightarrow B)$

and the inference rule *modus ponens*: $A, A \Rightarrow B \vdash B$.

A *proof* or a *derivation* in a Hilbert system is a finite sequence of formulas such that each element is either an axiom or follows from earlier formulas by the rule of inference. A proof of a derivation from a set S of formulas is a finite sequence of formulas such that each term is either an axiom, or is a member of S, or follows from earlier formulas by the rule of inference.

If there is a proof for A, then A is a *theorem* and we denote that by $\vdash A$. For example, it can be proved that $A \Rightarrow A$ is a theorem, as follows:

1. $(A \Rightarrow ((A \Rightarrow A) \Rightarrow A)) \Rightarrow ((A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A))$ (instance of A2)
2. $A \Rightarrow ((A \Rightarrow A) \Rightarrow A)$ (instance of A1)
3. $(A \Rightarrow (A \Rightarrow A)) \Rightarrow (A \Rightarrow A)$ (from 1 and 2, by MP)
4. $A \Rightarrow (A \Rightarrow A)$ (instance of A1)
5. $A \Rightarrow A$ (from 3 and 4, by MP)

This may seem like a lot of work to prove that "A implies A", but that's the nature of formal logic systems! Derivations are broken down into extremely small steps that are rigorously mathematically justified. In human commonsense inference we tend to proceed in large leaps instead, at least on the conscious level – but unconsciously, our brains are carrying out multitudes of small steps, though the analogy between these small steps and the small steps in logical proofs is a subject of debate in the AI and cognitive science community.

There is a link between the semantics of propositional logic and the above Hilbert-style system stating that the system is *sound* and *complete:* every theorem is tautology, and every tautology is theorem, i.e., $\models A$ if and only if $\vdash A$.

While the above is a standard and workable, approach, there are also other inference systems for propositional logic, including the one obtained as restrictions of Gentzen natural deduction and sequent calculus (see the section on first-order logic). There are also variants for classical and for intuitionistic propositional logic: in the former $A \vee \neg A$ is a theorem, and in the latter it is not (the above Hilbert-style system is classical).

2.3 Review of Predicate Logic

Standard, first-order predicate logic builds on propositional logic as defined above. We will review it using the same categories of syntax, semantics and deduction.

Firstly, syntax. Let Σ be a finite or a countable set, its elements will be called function symbols. Let Π be a finite or a countable set, its elements will be called predicate symbols. Let *arity* be a function that maps elements of Σ and Π to natural numbers. The triple

$(\Sigma, \Pi, arity)$ is called a *signature*. The set of *terms* over a signature $(\Sigma, \Pi, arity)$ and a countable set of variables V is defined in the following way:

- all elements of V are terms;
- if f is a function symbol and arity(f) = 0, then f is a term;
- if f is a function symbol and arity(f) = n, and if t_1, ..., t_n are terms, then $f(t_1, \ldots, t_n)$ is a term.
- each term is obtained by a finite number of applications of the above rules.

The set of atomic formulas over a signature $(\Sigma, \Pi, arity)$ and a countable set of variables V is defined in the following way:

- $\bot$ and $\top$ are atomic formulas;
- if p is a predicate symbol and arity(p) = n, and if t_1, ..., t_n are terms, then $p(t_1, \ldots, t_n)$ is an atomic formula.
- each atomic formula is obtained by a finite number of applications of the above rules.

The set of formulas over a signature $(\Sigma, \Pi, arity)$ and a countable set of variables V is defined in the following way:

- each atomic formulas is a formula;
- if A is a formula, then so is $\neg$ A;
- if A and B are formulas, then so are $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$;
- if A is a formula and v is a variable, then $(\forall x)A$ and $(\exists x)A$ are formulas;
- each formula is obtained by a finite number of applications of the above rules.

Next, semantics. The *meaning* of a formula is defined with respect to a pair (D,I), where D is a non-empty set, called the *domain*, and I is a mapping such that:

- To each function symbol of arity 0, I associates an element c from D;
- To each function symbol of arity $n > 0$, I associates a total function f^I from D^n to D;
- To each predicate symbol of arity $n > 0$, I associates a total function p^I from D^n to $\{0, 1\}$.

A valuation v, in this context, is defined as a mapping from the set of variable V to the set D. An *interpretation* I_v of terms, with respect to a pair (D,I) and a valuation v is defined in the following way:

- $I_v(t) = v(t)$, if t is an element V;
- $I_v(t) = I(t)$, if t is a function symbol and $arity(t) = 0$;

- $I_v(f(t_1,\ldots,t_n)) = f^I(I_v(t_1),\ldots,I_v(t_n)))$ where $f^I = I(f)$.

An interpretation I_v of formulas, with respect to a pair (D,I) and a valuation v is defined in the following way:

- $I_v(\bot)=0$ and $I_v(\top)=1$
- $I_v(p(t_1, \ldots, t_n))= p^I (I(t_1), \ldots, I(t_n)))$ where $p^I=I(p)$.
- $I_v (\neg A)=1- I_v (A)$;
- $I_v (A\wedge B)=\min(I_v (A), I_v (B))$;
- $I_v (A\vee B)=\max(I_v (A), I_v (B))$;
- $I_v (A\Rightarrow B)=\max(1-I_v (A), I_v (B))$;
- $I_v (A\Leftrightarrow B)=1$, If $I_v (A)= I_v(B)$ and $I_v (A\Leftrightarrow B)=0$ otherwise.
- $I_v((\forall x)A)=1$, if for every valuation w that is identical to v, with a possible exception of x, it holds $I_w(A)=1$. Otherwise, $I_v((\forall x)A)=0$;
- $I_v((\exists x)A)=1$, if there is a valuation w that is identical to v, with a possible exception of x, such that it holds $I_w(A)=1$. Otherwise, $I_v((\exists x)A)=0$.

If there is a pair (D,I) and a valuation v, such that $I_v(A)=1$, then A is *satisfiable*, otherwise it is unsatisfiable, or inconsistent. If for a fixed pair (D,I) it holds that $I_v(A)=1$ for arbitrary valuation v, then A is *valid with respect to* (D,I). If it holds that $I_v(A)=1$ for arbitrary pair (D,I) and for arbitrary valuation v, then A is *valid* and we denote that by $\models A$.

For instance, if the domain D is the set of natural numbers and if I maps p to the relation $\leqslant$, then $I_v((\forall x)p(x,x))=1$ in every valuation v, hence the formula $(\forall x)p(x,x)$ is valid with respect to (D,I).

2.3.1 *Deduction in First-Order Logic*

Deduction in first-order logic is similar conceptually to its analogue in propositional logic, but more complex in detail due to the presence of quantified variables. There are several different deductive systems available; one of the first was developed by Hilbert in the early 20th century and we will describe it now. In Hilbert's systems, formulas are built using only the connectives $\Rightarrow$ and $\neg$, and the quantifiers $\forall$ ("for all") and $\exists$ ("there exists"). The system consists of the following axiom schemes:

(A1) $A \Rightarrow (B \Rightarrow A)$
(A2) $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$
(A3) $(\neg B \Rightarrow \neg A) \Rightarrow ((\neg B \Rightarrow A) \Rightarrow B)$

(A4) $(\forall x)A \Rightarrow A[x \rightarrow t]$, while the term t is free for x in A
(A5) $(\forall x)(A \Rightarrow B) \Rightarrow (A \Rightarrow (\forall x)B)$, while A does not involve free occurrences of x

and the following inference rules:

Modus ponens: $A, A \Rightarrow B \vdash B$
Gen: $A \vdash (\forall x)A$

A proof or a derivation in a Hilbert system is a finite sequence of formulas such that each element is either an axiom or follows from earlier formulas by one of the rules of inference. A proof of a derivation from a set S of formulas is a finite sequence of formulas such that each formula is either an axiom, or is a member of S, or follows from earlier formulas by one of the rules of inference. If there is a proof for A, then A is a theorem and we denote that by $\vdash A$.

There is a link between the semantics of first order logic and the above Hilbert-style system stating that the system is *sound* and *complete:* every theorem is valid formula, and every valid formula is theorem, i.e., $\models A$ if and only if $\vdash A$.

In Hilbert-style systems, even for trivial statements, proofs can be non-trivial and rather unintuitive. However, although this kind of system is very demanding for practical use, it is very suitable for formal analyses of formal logic (since it has just a few axioms and inference rules). On the other hand, Gentzen constructed (in the 1930's) a "natural deduction" system, that better reflects usual mathematical reasoning. The price for this increased naturalness is a larger set of inference rules – one for eliminating and one for introducing each logical connective ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$) and quantifier ($\forall$, $\exists$), and one for eliminating the logical constant $\perp$ (13 rules altogether). On the other hand, there is just one axiom scheme ($A \vee \neg A$), which is not even needed if one adopts the intuitionistic version of Gentzen's logic.

Some of the rules in Gentzen's deductive system are:

$$\frac{A}{A \vee B} \text{ (introducing } \vee\text{)}$$

$$\frac{A \quad \neg A}{\perp} \text{ (eliminating } \neg\text{)}$$

$$\frac{A \quad A \Rightarrow B}{B} \text{ (eliminating } \Rightarrow\text{)}$$

and so forth.

Proofs in Gentzen's natural deduction are usually represented as trees with the statement to be proved in the root (at the bottom), and with axioms or assumptions in leaves

(all these assumptions have to be *eliminated* along the proof). Somewhat different in nature from usual mathematical proofs, is the Gentzen's sequent calculus, suitable for formal analyses and for automation. The elementary object in this system is a *sequent*, a construct of the form $A_1, A_2, \ldots, A_n \vdash B_1, B_2, \ldots, B_m$. The calculus itself consists of the inference rules for introducing logical connectives and quantifiers on both sides of sequents, for instance:

$$\frac{\Gamma \vdash \Theta, A}{\neg A, \Gamma \vdash \Theta} \text{ (introducing } \neg \text{ left)}$$

There are also structural rules for dealing with formulas within one side of a sequent, for instance:

$$\frac{\Gamma \vdash \Theta}{D, \Gamma \vdash \Theta} \text{ (weakening)}$$

There are many variations of the above inference systems, both for first order logic and for other theories. The above systems can also be used as inference systems for propositional logic – it suffices to omit all axioms and rules involving quantifiers.

Finally, it is worth noting that first-order logic is semi-decidable, which means that there can be an algorithm that can always confirm a valid formula is indeed valid (i.e, is a theorem), but cannot always detect that a non-valid formula is not valid. Things are simpler in propositional logic which is decidable, meaning that there is an algorithm which can always decide whether a given propositional formula is valid or not.

2.3.2 *First-order Theories*

Next, one can extend basic predicate logic by defining "theories" within it, which extend the basic axioms by adding other specialized axioms. We will be doing a lot of this in the present book, in the context of specialized theories about time and space.

Mathematically, we say that each such theory has a certain "signature," consisting of specific sets Σ and Π of function and predicate symbol (with certain arities) extending the basic ones used in predicate logic. Beside the new symbols, to create a theory one has to provide a list of axioms (in the described language). These axioms are then used within a selected deductive framework (e.g., Hilbert's system, Gentzen's natural deduction, etc.) as additional axioms of the system.

If a theory T is defined by a signature (Σ, Π,*arity*), and within a deductive system, then we often write $T \vdash F$ or $\vdash_T F$ to denote that the formula F can be derived in the theory T (i.e., F is a theorem of T).

As an example within mathematics, the branch of math called "group theory" can be constructed easily as an extension of "pure" predicate logic. In this formalization, the signature of theory of group consists of:

- Functional symbol 0 (of arity 0);
- Functional symbol $+$ (of arity 2);
- Functional symbol $-$ (of arity 1);
- Predicate symbol $=$ (of arity 2),

And, the axioms of the theory of groups are:

$$(\forall x)(x = x)$$
$$(\forall x)(\forall y)(x = y \Rightarrow -x = -y)$$
$$(\forall x_1)(\forall x_2)(\forall y_1)(\forall y_2)(x_1 = y_1 \wedge x_2 = y_2 \Rightarrow x_1 + x_2 = y_1 + y_2)$$
$$(\forall x_1)(\forall x_2)(\forall y_1)(\forall y_2)(x_1 = y_1 \wedge x_2 = y_2 \Rightarrow (x_1 = x_2 \Rightarrow y_1 = y_2))$$
$$(\forall x)(\forall y)(\forall z)(x + (y + z) = (x + y) + z)$$
$$(\forall x)(x + 0 = 0 + x = x)$$
$$(\forall x)(x + (-x) = (-x) + x = 0)$$

These axioms, added on to the regular axioms of predicate logic, tell you how to use the entities $\{0, +, -, =\}$ in the manner required in group theory. For instance, it can be proved that the following formula is theorem of the theory of groups:

$$(\forall x)(\forall y)(\forall z)(x + z = y + z \Rightarrow x = y)$$

What we'll see later on in this book are similar theories that involve, not $\{0, +, -, =\}$ but rather entities such as time intervals, spatial regions, and relationships between them.

Finally, although we won't make use of this here, it's worth noting that, apart from the axiomatic approach to defining theories, theories can be also defined semantically. For a given signature L and a corresponding pair (D,I), a theory of a structure is the set of all sentences over L that are true in (D,I).

2.3.3 *Forward and Backward Chaining*

There are two basic search strategies for using inference rules in theorem-proving, and other related AI areas:

- Forward chaining

- Backward chaining

Both strategies are applied to tasks of the same sort: given a set of facts (axioms) and a set of inference rules, the task is to check whether some given fact (formula) F can be derived. The difference between the two strategies is the direction of their search. Namely, forward chaining (also known as *data-driven search*) starts with the available facts and derive all facts that can be derived, until the given fact F is reached. On the other hand, backward chaining (also known as *goal-driven search*) starts with the given goal F and apply inference rules in opposite direction, producing new subgoals and trying to reach the facts already available.

Let us consider the following example. Let there be given facts:

1) Derek will go out for lunch.
2) If Alison goes out for lunch, then Bob will go out for lunch as well.
3) If Bob goes out for lunch, then Clark will go out for lunch as well.
4) If Derek goes out for lunch, then Ellen will go out for lunch as well.
5) If Ellen goes out for lunch, then Clark will go out for lunch as well.

and assume the (only) inference rule is *modus ponens*: if X and $X \Rightarrow Y$, then Y. The goal to be proved is "Clark will go out for lunch".

Forward chaining will try to match any two facts with X and $X \Rightarrow Y$ in order to apply modus ponens and to derive new facts. One (and the only at this step) option is to use the facts 1 and 4 and derive the fact "Ellen will go out for lunch". Then, in next step, this new fact and the fact 5 yield "Clark will go out for lunch", which was required.

Backward chaining starts from the goal – from the fact "Clark will go out for lunch". If it is matched with Y in modus ponens, then the new subgoals will be X and $X \Rightarrow Y$, where X can be *anything*. The subgoal $X \Rightarrow Y$ matches the fact 3, so X is matched with "Bob will go out for lunch". When proving this subgoal, the fact 2 is used and a new subgoal "Alison will go for lunch" should be proved. However, this leads to failure. Another option is the following: the subgoal $X \Rightarrow Y$ matches also the fact 5, so X is matched with "Ellen will go out for lunch". When proving this subgoal, the fact 4 is used and a new subgoal "Derek will go out for lunch" should be proved. This is trivially true, by the fact 1. So, it was proved that "Clark will go out for lunch", as required.

The above description of forward and backward chaining is simplified. Typical applications of forward and backward chaining involve different methods of directing the search. Namely, the main problem that both strategies face is search expense, due to typically huge

numbers of combinatorial options that have to be considered. Neither of these approaches is superior to the other one, and there are domains in which one if more appropriate than another. There are also hybrid approaches that combine forward and backward search.

2.3.4 *Decidability and Decision Procedures*

In this section we briefly introduce a distinction that may be important in practical large-scale logic-based systems: the distinction between proof procedures and decision procedures. Put simply:

- a proof procedure finds a specific series of steps for getting to a certain conclusion from the axioms of a logical theory
- a decision procedure checks whether a certain conclusion can be obtained from the axioms of a logical theory, without necessarily directly supplying the proof (the series of steps)

In practical cases, given the outcome of a decision procedure, plus knowledge of the algorithm used to carry out the decision procedure, it is in principle possible to construct a proof. But this may be quite laborious. In some situations, if one just needs to know whether something is true or not in a certain formal theory, it may be easier to use a decision procedure than to find a specific proof.

In the formal lingo of logic, one says a theory T is *decidable* if there is an algorithm that for any sentence F of T it can decide whether F is theorem of T or not. Such an algorithm is called a *decision procedure*. An example of a decision procedure for propositional logic is the DPLL (Davis-Putnam-Logemann-Loveland) procedure, which lies at the core of all modern SAT solvers. A SAT solver is a program that checks if a large propositional-logic formula can possibly be satisfied by any assignment of values to variables or not; and doing this via a decision procedure rather than a proof procedure is vastly more computationally efficient. SAT solvers are used in a huge variety of practical applications nowadays, including circuit analysis, natural language parsing, and all manner of large-scale discrete optimization problems. If one had to approach these problems using direct theorem proving in propositional logic, then one would run into terrible combinatorial explosions and quite possibly formal logic would need to be set aside in favor of some different analytical approach.

There are many widely applied decision procedures for first-order theories, including decision procedures for linear arithmetic over reals (involving only addition, not multipli-

cation), multiplicative arithmetic over reals (involving only multiplication, not addition), theory of lists, theory of arrays, etc. (Barrett *et al.*, 2009). There is a family of modern decision procedures for first-order theories, heavily using propositional reasoning and SMT (satisfiability modulo theory) techniques. Decision procedures and SMT solvers are widely applied in software and hardware verification.

Mathematically speaking, there are many important theories that are not decidable, such as arithmetic over natural numbers, the theory of groups, etc. Hence, for these theories there can be no decision procedures, only heuristics that can prove/disprove certain classes of formulas. However, undecidability always involves infinity in some form; if one restricts attention to a finite set of data items (as is always the case in practical applications), then it is not an issue.

In practical applications, the use of decision procedures is sometimes unavoidable. However, developing an efficient decision procedure is not an easy task – it requires specific knowledge about the theory. There is no generic (decidable) method for constructing efficient decision procedures. In the case of temporal and spatial logics such as we will discuss here, scientists have not yet created appropriate specialized decision procedures, but this seems a highly worthy area of investigation.

2.4 Simple Examples of Formal Logical Inference

We've been discussing logic in an extremely abstract and mathematical way – but our main goal here is *real-world* reasoning, not mathematical reasoning. So before progressing further in the direction of elaborating abstract logic systems, we will digress a bit to clarify the relationship between logic and commonsense inference. This is after all where logic started: formal logic originated, not as a branch of math, but from the motivation of formalizing everyday human thinking. Many real world problems can be formulated in terms of propositional or first order logic. It is only during the last century and a half that logic has become a sophisticated branch of mathematics.

2.4.1 *Example of Kings and Queens*

Consider the following example, from (Hayes, 1997). The king and his family want to make a party for some ambassadors in their kingdom. The king, the queen, and the prince give their orders to the head of protocol and he has to make a final list of guests. The king said that either the ambassador of Peru, or the ambassador of Qatar, or the ambassador of

Romania should be invited. The queen said that if the ambassadors of Qatar and Romania are invited, then the ambassador of Peru should be invited too. The prince said that if the ambassador of Romania is invited, then the ambassadors of Peru or the ambassador of Qatar should be invited. The question is whether the head of protocol can obey all orders. The problem can be formulated in terms of propositional logic in the following way. Let us denote by p, q, r the fact that the ambassador of Peru, Qatar, Romania will be invited.

Then the orders can be formulated as follows:

- King's order: $p \vee q \vee r$
- Queen's order: $(q \wedge r) \Rightarrow p$
- Prince's order: $r \Rightarrow p \vee q$

Or, equivalently:

- King's order: $p \vee q \vee r$
- Queen's order: $\neg q \vee \neg r \vee p$
- Prince's order: $\neg r \vee p \vee q$

To solve a problem, the head of protocol has to check whether the formula

$$(p \vee q \vee r) \wedge (\neg q \vee \neg r \vee p) \wedge (\neg r \vee p \vee q)$$

is satisfiable. He can use a SAT solver for that (and he can find that he can meet all orders by inviting only the ambassador of Peru). Or, as this is a simple case, using any reasonable propositional logic theorem prover would also work fine.

2.4.2 *Example of Minesweeper*

Or consider, as an another example, the popular computer game of Minesweeper, as depicted in Figure 2.1:
There is a board of $n \times m$ places. When a player selects one position on the board, the game is over if it contains a mine. Otherwise, the player gets the number of mines in the adjacent positions. Let us denote by p, q, r, s, t, u six positions in the left upper corner of the board: Let us suppose that we open the position p, and there is no mine. Let us also suppose that we got the number 1 for the position p. If we open a position q and we get the answer 1 again, we can safely open the positions r and u. The explanation is as follows. We associate a propositional variable to each position – the variable is true if there is a mine on the position, and the variable is false otherwise. Since there is no mine on the position p, it

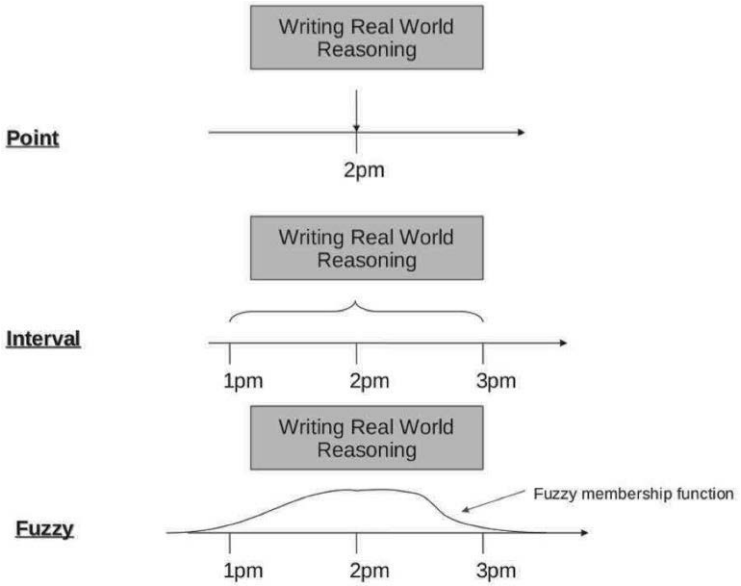

Fig. 2.1 A screenshot of a Minesweeper GUI

follows that:

$$\neg p$$

Since the position p has the associated value 1, there is one mine on either q, s, or t. We can encode this as follows:

$$q \vee s \vee t$$

$$q \Rightarrow \neg s \wedge \neg t$$

$$s \Rightarrow \neg q \wedge \neg t$$

$$t \Rightarrow \neg q \wedge \neg s$$

Since there is no mine on the position q, it follows that:

$$\neg q$$

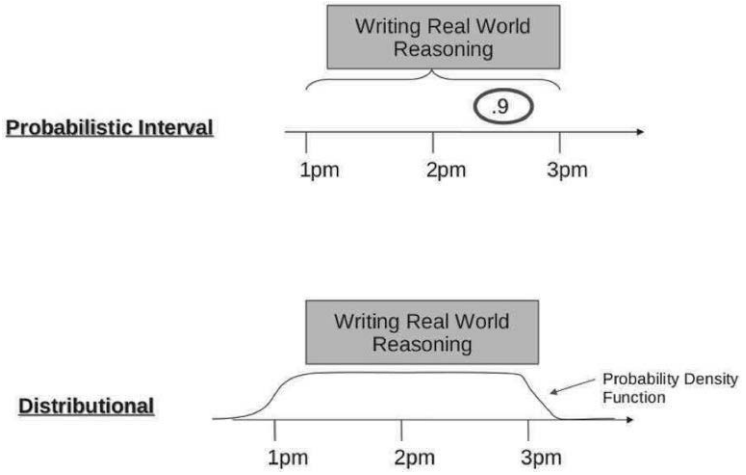

Fig. 2.2

Since the position q has the value 1, it means that there is one mine on either p, s, t, r, or u. We can encode this as follows:

$$p \vee s \vee t \vee r \vee u$$

$$p \Rightarrow \neg s \wedge \neg t \wedge \neg r \wedge \neg u$$

$$s \Rightarrow \neg p \wedge \neg t \wedge \neg r \wedge \neg u$$

$$t \Rightarrow \neg p \wedge \neg s \wedge \neg r \wedge \neg u$$

$$r \Rightarrow \neg p \wedge \neg s \wedge \neg t \wedge \neg u$$

$$u \Rightarrow \neg p \wedge \neg s \wedge \neg t \wedge \neg r$$

The question is whether one can safely open the position r, i.e., whether r is consistent with the above formulas (i.e., is it possible that there is a mine on r). A SAT solver (applied to the above formulas) would easily find that such a set of formulas is not consistent, so there cannot be a mine on the position r (and, similarly, there cannot be a mine on the position u).

2.4.3 *Example of Socrates*

The above examples all involve propositional rather than predicate logic. Concerning first-order logic, consider the following classical example. One is given the following two facts:

- Socrates is a man.

- All men are mortal.

We can encode this in first-order logic as follows:

- man(Socrates)
- $(\forall x)(\text{man}(x) \Rightarrow \text{mortal}(x))$

Applying deduction allows us to conclude that, since Socrates is a man, he must be mortal. For instance, a very simple proof procedure can verify that mortal(Socrates) is a consequence of the above facts by verifying that $\neg$mortal(Socrates) is inconsistent with the above formulas.

2.5 Modal logic

Propositional logic and first-order logic are sometimes not expressive enough for some sorts of common-sense reasoning. For instance, one may wish to extend them using so-called modal statements including qualifiers like necessary, possibly, future, past, etc. For such cases, specific logics are used, such as modal and temporal logics. These logics are not contradictory to first-order logic, and in fact could be viewed as specialized theories constructed within first-order logic; but this is not always the most useful way to look at them. It is often more helpful to think about them as alternative formalizations of logic.

Modal logic describes logical relations of modal notions. These notions may include metaphysical modalities (necessities, possibilities, etc.), epistemic modalities (knowledge, belief, etc.), temporal modalities (future, past, etc.), and deontic modalities (obligation, permission, etc.). Modal logics are widely used in philosophy, artificial intelligence, database theory, and game theory. Modern work on modal logics started in 1918. with the monograph *A Survey of Symbolic Logic* by C. I. Lewis.

The main metaphysical modal notions are *necessity* and *possibility*; and the modal logic that describe logical relations over them is called *alethic modal logic*. In alethic modal logic, one can express a statement like "It is possible that Bob quit his job". The modailities are represented by modal operators. Formulas are built using propositional connectives and these modal operators. The basic unary modal operators are usually written $\Box$ (or L) for necessarily and $\Diamond$ (or M) for possibly. These two operators are linked in the following way:

$$\Diamond \mathrm{p} \Leftrightarrow \neg \Box \neg \mathrm{p}$$
$$\Box \mathrm{p} \Leftrightarrow \neg \Diamond \neg \mathrm{p}.$$

There is a number of modal extensions of some underlying logics, including the extension of first-order logic, called *modal predicate logic*.

The standard semantics of modal logics is Kripke semantics. The concept of the Kripke semantics of propositional modal logic includes:

- A non-empty set W – a set of possible worlds;
- A two-place relation R on elements from W – the *accessibility relation* between worlds, which represents the possible worlds that are considered in a given world, i.e., if we consider a world w_0, every world v such that it is in relation R with w_0 represents a possibility that is considered at a world w_0;
- A *frame* – a tuple (W, R);

Given a frame (W, R), a *model* M is a tuple (W, R, V) where V is a map that assigns to a world a valuation on propositional variables, i.e. for a given world w, $V(w)$ is a function from the set of propositional variables to $\{0, 1\}$. Interpretation of a formula F with respect to M *is* defined in the following way (the fact that F is true at a world w in a model M) is denoted by $M, w \models F$:

$M, w \models p$ iff $V(w)(p) = 1$ (where p is a propositional variable)

$M, w \models F \wedge F'$ iff $M, w \models F$ and $M, w \models F'$.

$M, w \models \neg F$ iff *not* $M, w \models F$

$M, w \models \Box F$ iff, for every world w' such that it is in relation R with w it holds that,

$M, w' \models F$

The semantics of other propositional connectives and the operator $\Diamond$ are implied by the above definition. A formula is then defined to be *valid* in a model M if it is true at every possible world in M. A formula is *valid* if it is valid in all frames (or every model).

For the given semantics, there is a sound and complete inference system for propositional modal logic – the system **K**. In addition to underlying propositional axioms and inference rules, there is the axiom

$$\Box(\mathrm{F} \Rightarrow \mathrm{F}') \Rightarrow (\Box \mathrm{F} \Rightarrow \Box \mathrm{F}')$$

and the rule

$$\text{if } \vdash \mathrm{F}, \text{ then } \vdash \Box \mathrm{F}$$

There are various inference systems for propositonal modal logic obtained by adding extra axioms to **K**.

There are also fuzzy versions of modal logics (Thiele, 1993; Ying, 1988).

2.6 Deontic logic

An interesting specialization of modal logic, which is potentially useful for doing logical inference about human behaviors and motivations, is deontic logic – which is concerned with the ideal and actual behavior of social agents, and involves notions like permissible, impermissible, obligatory, gratuitous, optional, etc. Deontic logic has many analogies with alethic modal logic. In addition to theoretical interest, deontic logics are used for formalizing different real-world concepts and problems such as morality, normative law, legal analysis, social and business organizations and security systems, computer security, electronic commerce, or legal expert systems.

A survey of applications of deontic logic in computer science can be found in (Wieringa & Meyer, 1993), which supplies the following systematization of applications of deontic logic in computer science:

1. Fault-tolerant computer systems.
2. Normative user behavior.
3. Normative behavior in or of the organization.
   (a) Policy specification.
   (b) Normative organization behavior (e.g. contracting).
4. Normative behavior of the object system.
   (a) The specification of law.
   (b) The specification of legal thinking.
   (c) The specification of normative rules as deontic integrity constraints.
   (d) Other applications, not discussed above.

The first formalization of deontic logic was given by E. Mally in 1926. More details on deontic logic can be found in (McNamara & Prakken, 1999).

In the "Traditional scheme" for deontic logic, there are five normative statuses considered:

- it is obligatory that (OB)
- it is permissible that (PE)
- it is impermissible that (IM)
- it is gratuitous that (GR)
- it is optional that (OP)

The first one of the above can be used as a basis, while the remaining ones can be defined in the following way:

$$\mathbf{PE}\mathrm{p} \Leftrightarrow \neg\mathbf{OB}\neg\mathrm{p}$$

$$\mathbf{IM}\mathrm{p} \Leftrightarrow \mathbf{OB}\neg\mathrm{p}$$

$$\mathbf{GR}\mathrm{p} \Leftrightarrow \neg\mathbf{OB}\mathrm{p}$$

$$\mathbf{OP}\mathrm{p} \Leftrightarrow (\neg\mathbf{OB}\mathrm{p} \wedge \neg\mathbf{OB}\neg\mathrm{p})$$

Standard Deontic Logic (SDL) is the most studied deontic logic. It extends propositional logic by the (one-place) **OB** deontic operator. Formulas are built in the standard modal-logic way. The semantics of SDL is usually given in Kripke-style. The inference system for SDL consists of axioms for (classical) propositional calculus and inference rules, the additional inference rule "if $\vdash p$ then $\vdash \mathrm{OB}p$" *and the following axioms:*

$$\mathbf{OB}(\mathrm{p} \Rightarrow \mathrm{q}) \Rightarrow (\mathbf{OB}\mathrm{p} \Rightarrow \mathbf{OB}\mathrm{q})$$

$$\mathbf{OB}\mathrm{p} \Rightarrow \neg\mathbf{OB}\neg\mathrm{p}$$

Consider the following simple example. Let us assume that the hypotheses are:

It ought to be the case that Alison does the paperwork.
If Alison does the paperwork, then Alison leaves the office late.

Let us denote Alison does the paperwork by p and Alison leaves the office late by q:

Then, we can prove It ought to be the case that Alison leaves the office late (i.e., **OB***q) as follows:*

(C1) **OB**p (*hypothesis*)
(C2) p⇒q (*hypothesis*)
(C3) **OB**(p⇒q) (*deontic inference rule, from (C2)*)
(C4) **OB**(p⇒q) ⇒ (**OB**p ⇒**OB**q) (*deontic axiom*)
(C5) **OB**p ⇒ **OB**q (*Modus ponens, from (C3) and (C4)*)
(C6) **OB**q (*Modus ponens, from (C1) and (C5)*)

There is a number of variants of the SDL inference system and there are interesting *logical and philosophical considerations for each of them.*

2.6.1 *Fuzzy deontic logic*

While there is a number of approaches to fuzzy modal logic (see the next chapter for a recall of fuzzy logic), there is a very limited literature on fuzzy deontic logic. One version

of fuzzy deontic logic was introduced and discussed by (Gounder & Esterline, 1998). In their framework, given the statements

p = Person p receives a driver's license.
q = Person p is 18 or older.
r = Person p is an employee of company c.
s = Person p is over 80 years old.
t = Person p is under 20 years old.
u = Company c gives its employees a bonus.
v = The employees of company c arrive at work not more than ten minutes late.

one can consider the following interesting deontic statements:

$$\mathbf{OB}(\mathrm{p} \Rightarrow \mathrm{q})$$

$$\mathbf{OB}\mathrm{p} \Rightarrow \mathbf{OB}\mathrm{q}$$

$$\mathrm{r} \Rightarrow \mathbf{OB}(\neg \mathrm{s} \wedge \neg \mathrm{t})$$

$$\mathrm{u} \Rightarrow \mathbf{OB}\mathrm{v}$$

These statements need not be crisp, but can be in a permissible range which is given in the fuzzy truth value in the interval [0, 1] and one obligation may lead to another obligation.

Concerning reasoning in this theory, as said in (Gounder & Esterline, 1998), since fuzzy logics work with numerical measures, axiomatic systems are not appropriate. Instead, fuzzy versions of the semantic properties exist and can be shown to correspond to some of the axioms for the crisp systems in special ways that support dependency among assertions in a modal domain.

2.7 The frame problem

A final issue that must be discussed, in the context of knowledge representation using formal logic, is the "frame problem," as originally recognized and named in (McCarthy & Hayes, 1969). Put most simply, this is the problem of representing the effects of action without having to represent explicitly a large number of intuitively obvious non-effects (i.e., properties that are not affected by the action). The frame problem also has a wider epistemological importance and it considers whether it is possible, in principle, to limit the scope of the reasoning required to derive the consequences of an action. The name "frame problem" was derived from a common technique used in producing animated cartoons

where the currently moving parts of the cartoon are superimposed on the "frame", which depicts the non changing background of the scene.

While the frame problem is a major issue for certain approaches to logic-based AI, we don't see it as an objection to scalably deploying logic-based technology to draw inferences based on large spatiotemporal knowledge bases (nor to other scalable real-world inference applications). Rather, we see it as an objection to embedding logical inference engines in overly simplistic cognitive architectures. We believe the frame problem can be bypassed via judicious inference control heuristics, including some that are implicit in the ideas discussed in the previous section, and some others that will be discussed here.

2.7.1 *Review of the Frame Problem*

To elaborate the frame problem a little more fully, suppose we have the following knowledge:

- Alison is in her office and she wears a blue suit.
- If Alison moves from her office, then she is in the loby.

If we represent the above in classical first-order logic, using some suitable formalism for representing time and action (e.g., CTL logic, or an appropriate subset of PLN, both discussed below), we can derive that after Alison moves from her office she will be in the lobby. However, we will not be able to derive that Alison's suit is still blue. Namely, the knowledge given above does not rule out the possibility that the color of Alison's suit changes when she moves out of her office. A straightforward solution for this is to add rules that explicitly describe the non-effects of each action (e.g., "when Alison moves from her office, the color of her suit does not change"). Such formulae are called *frame axioms*. However, this is not a satisfactory solution. Namely, since *most* actions do not affect *most* properties of a situation, in a domain comprising M actions and N properties we will, in general, have to write out MN frame axioms which would make any reasoning process impractical.

In (Shanahan & Baars, 2005) there is a more detailed account on the frame problem, including a brief description of Dennett' memorable example:

> *. . . consider the challenge facing the designers of an imaginary robot whose task is to retrieve an object resting on a wagon in a nearby room. But the room also contains a bomb, which is timed to explode soon. The first version of the robot successfully works out that it must pull the wagon out of the room. Unfortunately, the bomb is on the wagon. And although the robot knows the bomb is on the wagon, it fails to notice that pulling the wagon out brings the bomb along too. So the designers produce a second version of the robot. This model works out all the consequences of its actions before doing anything. But the new robot gets blown up too, because*

it spends too long in the room working out what will happen when it moves the wagon. It had just finished deducing that pulling the wagon out of the room would not change to color of the room's walls, and was embarking on a proof of the further implication that pulling the wagon out would cause its wheels to turn more revolutions than there were wheels on the wagon—when the bomb exploded. So the robot builders come up with a third design. This robot is programmed to tell the difference between relevant and irrelevant implications. When working out the consequences of its actions, it considers only the relevant ones. But to the surprise of its designers, this version of the robot fares no better. Like its predecessor, it sits in the room "thinking" rather than acting. "Do something!" they yelled at it. "I am," it retorted. "I'm busily ignoring some thousands of implications I have determined to be irrelevant. Just as soon as I find an irrelevant implication, I put it on the list of those I must ignore, and." the bomb went off.

It is obvious that the human brain incorporates a solution to the frame problem and does not suffer from overwhelming information when deriving new conclusions. When we say that Alison left her office, we don't need to state explicitly that her suit hasn't change its color, that Bob's cat hasn't changed its sex, that the Sun continues to shine, etc. Such information is taken for granted by common sense. In mathematical logic, however, nothing is taken for granted and in classical logic it is necessary to represent explicitly all the things that change and all the things that do not change by some action.

2.7.2 *Working around the Frame Problem*

Perhaps the best-known attempt to work around the frame problem, within the scope of logic-based AI, begins from the observation that the inference process in classical logic is *monotonic*, meaning that the set of conclusion can only grow when we add new premises (we do not retract some conclusions if we are presented with some new premise). But, it would be nicer if one could infer that Alison's suit is, generally, of the same color when she leaves her office and, in addition, it would be suitable, to add some exceptions, stating otherwise ("If Alison spilled coffee on her suit, then she has to change her suit before she leave her office"). In other words, one would like to be able to declare the general rule that an action can be assumed not to change a given property of a situation *unless* there is evidence to the contrary. Such reasoning is possible within non-monotonic logics. Despite the fact that there is also a number of problems with the frame problem when addressed by non-monotonic logics, it can be considered that they provide a satisfactory solution. In artificial intelligence, there are also some other approaches, that handle different incarnations of the frame problem with more or less success. The frame problem is still making an influence on issues in cognitive sciences, philosophy, psychology, etc.

Propositional and first-order logic as defined are monotonic. This means that the set of facts that can be derived from S increases when S increases. However, again, this is not appropriate for some sorts of common-sense reasoning. For example, if we are given a fact that Tweety is a bird, we by default derive the fact that Tweety flies. But, if we are given an additional fact that Tweety is a penguin, then we retract our conclusion and derive the new one – that Tweety does not fly. There is a family of logics following this motivation and trying to model common-sense reasoning, summarized for instance in (Reiter, 1980; Delgrande & Schaub, 2003)

However, nonmonotonic logic is not the only route for circumventing the frame problem; for instance in our own work with PLN, we have taken a significantly different approach, to which we will return in the final chapter of this book.

http://www.springer.com/978-94-91216-10-7

Real-World Reasoning: Toward Scalable, Uncertain Spatiotemporal, Contextual and Causal Inference

Goertzel, B.; Geisweiller, N.; Coelho, L.; Janičić, P.; Pennachin, C.

2011, IX, 269 p. 59 illus., 1 illus. in color., Hardcover
ISBN: 978-94-91216-10-7
A product of Atlantis Press