Formal Methods for Verification of Clinical Practice Guidelines

Arjen HOMMERSOM [a,1], Perry GROOT [a], Michael BALSER [b], and Peter LUCAS [a]

[a] *Radboud University Nijmegen, The Netherlands*
[b] *Universität Augsburg, Germany*

Abstract. Formal methods play an important role in the development of software and hardware systems. In recent years, there has been a growing interest to apply these methods in the area of medical guidelines and protocols. This paper summarises these efforts, compares the approaches and discusses the role of formal methods in this area.

Keywords. Verification, (Interactive) Theorem Proving, Model Checking

Introduction

Formal methods are commonly defined as mathematics-based techniques for the specification, development, and verification of software and hardware systems [49,69]. A specification of a system is a mathematical description that describes *what* the system should do, i.e., the requirements or properties that should hold for an implementation of a system, written as well-formed statements in a formal logic. Given the specification, formal verification is the act of proving or disproving the correctness of an implementation with respect to the specification in a mathematically rigorous way. Many of such techniques have been developed. However, we will focus this chapter on those techniques that have been applied in the context of clinical guidelines or protocols, which are for a large part based on logic.

Formal methods can be exploited in relationship to medical guidelines in several ways. One choice is to consider the guideline as the 'system' that is being developed. Then, verification involves checking whether this guideline adheres to certain correctness or quality criteria, which is the main topic of this chapter. However, guidelines could also be considered as the golden standard for other types of systems. For example, in developing local protocols, the requirements are often derived from the more general guideline, if available. Similarly, if we look at the actual clinical practice as a system, e.g., formalised in terms of an electronic patient record, then verification may involve comparing medical actions against

[1]Corresponding Author: Arjen Hommersom, Radboud University Nijmegen, Toernooiveld 1, 6525 ED Nijmegen, the Netherlands; E-mail: arjenh@cs.ru.nl.

recommendations given by a guideline. We will only briefly address such other possibilities as they are discussed more elaborately in a Chapter 7 of this book.

This chapter is structured as follows. In Section 1, we review, from a formal perspective, some of the different specification languages that have been proposed for modelling guidelines. Questions regarding formal semantics and expressiveness are addressed. In Section 2, we address the different type of properties that have been investigated in literature. Next, in Section 3, we look at some actual verification studies and methodologies. In Section 4, we briefly discuss formal methods in relation to protocol development and compliance checking, as described above. Finally, in Section 5, we discuss the role of formal methods in its relation to medical guidelines and especially the impact it may have on medical practice.

1. Guideline Specification

Several methods have been developed to support the modelling of guidelines. Specialised guideline modelling languages have been developed of which some have a formal semantics. Other authors have proposed the use of general-purpose logical languages for the specification of guidelines as many languages that have been proposed in artificial intelligence have rich structures and are likely to be expressive enough for specification of aspects of guidelines. Furthermore, as they are often based on logic, they have a formal semantics. On the other hand, they lack some intuitive primitives elements of specialised guidelines modelling languages such as "decisions" or (medical) "actions".

1.1. General-purpose Formal Languages

Description logic [3] is a family of well-known knowledge representation formalisms, which has been used for the development of a wide range of applications. An important advantage is that they constitute a decidable fragment of first-order logic, and thus provide a clear syntax and semantics and make automatic verification possible. In [53] it is proposed to model certain aspects of practice guidelines in this logic, i.e., taxonomic relations (*chest x-ray is a type of x-ray*), mereologic order (*identification is part of the interview*), and temporal order (*physical examination precedes chest x-ray*). Examples of the use of description logics in the clinical domain can be found in e.g., [54].

Logic-based formalisation of medical guidelines has also been suggested in the context of agent modelling. In [12], guidelines are considered a set of social integrity constraints, formalised using standard logic with additional operators **H** (indicating a ground fact), **E** (indicating an expectation), and **EN** (indicating a *negative* expectation). For example, the following:

$$\mathbf{H}\,(enter(Patient, emergency\_ward), T_0) \rightarrow$$
$$\mathbf{E}\,(examine(Physician, Patient), T_1) \wedge T_1 > T_0$$

denotes that in whenever a patient enters the emergency ward, then it is expected that a physician will examine the patient at some later time instance T_1. The underlying idea is to use these constraints in order to prevent agent behaviour that

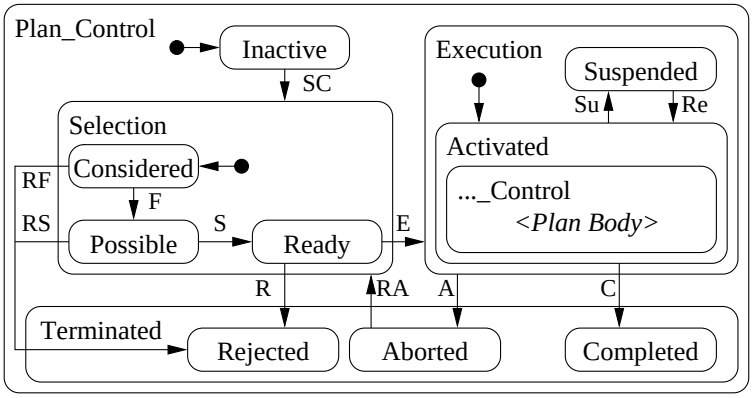

Figure 1. Plan state model of a single Asbru plan.

is not compliant with the guideline. This is particularly useful in hospitals where integrity constraints could be used for checking if the hospital staff is compliant with the guideline. The semantics of this language has been formalised as an abductive proof procedure [1].

Temporal reasoning is an important aspect of medical guidelines. Therefore, it is not surprising that temporal logic has been proposed as a suitable formalism. For example, computation tree logic has been applied for modelling a guideline for the purpose of formally studying refinement of guidelines to protocols (cf. Chapter 7 and [30]).

1.2. Guideline Modelling Languages

A number of languages have been developed to write down clinical guidelines in a computer-interpretable format (so called computer-interpretable clinical guidelines, CIGs). In [38], the languages Asbru, EON, GLIF, GUIDE, PRODIGY, and PRO*forma* are compared, based on a common case study. The paper identifies a number of common components and a number of significant differences between languages.

1.2.1. Semantics

For the syntax and semantics of guideline languages three levels can be considered:

1. the dynamic behaviour of guideline components, and
2. conditions usually given in a so called expression language,
3. temporal abstraction of conditions.

For Asbru, the dynamic behaviour of plan execution (level 1) is defined in a formalism called Structural Operational Semantics (SOS) [44]. The abstract execution model of a single plan is illustrated in Figure 1. Each arrow in the state transition system represents a single SOS rule: Filter and setup conditions (F and S) are used to control the applicability of a plan, abort and complete conditions (A and C) are used to monitor execution. The sub plans in the plan body can be

organised using different body types (e.g., sequential, any-order, or parallel). The current state of a plan – especially if a plan has been rejected, aborted, or completed – is propagated according to the plan hierarchy to its super and sub plans. If a plan is mandatory, it must be completed, otherwise it may also be rejected or aborted. Details can be found in [4,5]. Evaluation of conditions (level 2), however, is not considered in detail. Asbru offers a variety of possibilities to abstract from patient data (see [56,55,61]). In most cases, conditions are simple and can be translated one-to-one to a first order formula. For more complex conditions, a full formal semantics still needs to be defined. As a speciality of Asbru, conditions can be monitored over time according to so called time annotations (temporal abstraction, level 3). A revised and complete semantics of time annotations has been published in [52].

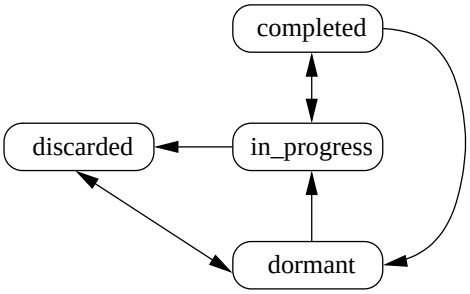

Figure 2. State and transitions of PRO*forma* Task component.

Similar to Asbru, an operational semantics for PRO*forma* is given in [63,24]. The dynamic behaviour of "Task" components is again described in an operational style. Figure 2 illustrates the underlying state transition system. The transition system has recently been simplified from 11 to 4 states without restricting expressiveness of the PRO*forma* language, as some of the states were found to be redundant to describe the core semantics of the language.

The semantics is described rather informally for GLIF [68], EON [65], GLARE [2,25], and GUIDE [46]. The descriptions usually give a flavour of a semantics in terms of 'states' (e.g., completed and aborted states), message passing, etc, although these terms are not defined in detail. While this is enough for actually building guidelines, the semantics must be more precise when it comes to formal verification. Facilities for testing and making sure that the model is unambiguous and syntactically correct is available for most approaches. An overview of these facilities can be found in [16].

Verification, in particular a verification technique called model checking (cf. Section 3.3), is nevertheless applied to guidelines written in GLARE by translating GLARE clinical guidelines to the formal language Promela [25]. This translation can be considered as defining the formal semantics of GLARE. As a drawback, in order to fully understand the execution model of GLARE, one needs to fully understand the translation as well as the Promela semantics.

1.2.2. Temporal Constraints

Medical guidelines do not simply describe a process that is to be executed instantly. Instead, the process evolves in time where reasoning about time is done explicitly. As this is such a prominent feature of guideline representation languages, this topic deserves some more detail. As an example, consider the administration of a drug three times a day for five days in a row. Different guideline modelling languages offer different constructs for expressing temporal constraints. The temporal constraints normally define only certain minimum and maximum boundaries instead of fixed time periods; for example, a drug is not taken every day at exactly the same time but should be taken between 07:00 and 10:00 a.m.

In the following paragraphs, the constructs for expressing temporal constraints in Asbru and in GLARE are introduced.

Whereas temporal constraints in GLARE refer to the start and end point of actions, Asbru time annotations incorporate arbitrary conditions, e.g., conditions that refer to the condition of the patient. This is more expressive; however, it adds complexity to the verification problem.

Time Annotations in Asbru. An example property containing a temporal constraint is the following: "Intensive phototherapy should produce a decline in the TSB (total serum bilirubin) level of 1 to 2 mg/dl within 4 to 6 hours". This can be expressed with so called time annotations in Asbru:

$$\Delta_{\mathrm{TSB}} \leq -1\mathrm{mg/dl} \quad [\_, 6h]\ [4h, \_]\ [\_, \_] \quad \mathrm{enter}(pti, \mathrm{activated})$$

i.e., a condition Δ_{TSB} must occur 4 to 6 hours after the treatment plan 'pti' (phototherapy intensive) has been activated. In general, this is the pattern for Asbru time annotations:

$$condition \quad [ESS, LSS]\ [EFS, LFS]\ [minDu, maxDu] \quad RP$$

A starting interval is defined by an earliest starting shift (*ESS*) and a lasted starting shift (*LSS*). The earliest- and latest finishing shift (*EFS* and *LFS*) define the finishing interval. These intervals are relative to a reference point (*RP*), that describes a time point that is the offset to the shifts. In addition, a minimum and maximum duration (*minDu* and *maxDu*) can be defined. A complete formal semantics of Asbru time annotations is given in [52].

Temporal Constraints in GLARE. GLARE's temporal language allows one to model temporal constraints between actions, and is quite similar to Asbru's time annotations. In GLARE, it is possible to define the minimum and maximum *duration* of actions, the minimum and maximum *delay* between actions and repetition/periodicy constraints. Consider actions A and B. Then the following examples illustrate the types of relations that can be expressed:

- the duration of A is between 10 and 20 minutes.
- the end of A is equal to the start of B.
- B starts between 10 and 20 minutes after the end of A.
- A is repeated once every week for two weeks, until condition C does not hold anymore.

A more detailed description of time in the GLARE framework can be found in [2].

2. Specification of Guideline Properties

To be able to specify properties, a representation language is required. Similar to the previous section, general and specific languages have been proposed. First, we discuss the type of properties that have been investigated. Then, we focus on the general and more specific representation languages that have been used in literature. Finally, we focus some of the methodological issues related to the specification of properties.

2.1. Type of Properties

A wide range of properties of medical guidelines and protocols that have been checked can be found in literature. In order to make the distinction more comprehensible, it is useful to make a distinction between two classes of properties, namely:

1. *Intra-guideline consistency* (this term is proposed in [60]): this deals with consistency issues within a single guideline. Inconsistencies include structural defects, e.g., inconsistencies between temporal structure and temporal specifications; ambiguities, such as conflicting recommendations for the same patient group; incompleteness, when a recommendation is missing for a relevant patient group.
2. *Intentions / Goals*: these approaches deal with correctness criteria in terms of the process or result of executing the guideline. Note that we will use the term intentions and goals synonymously here.

The first type of properties are typically stated informally or in the language that the guideline is modelled in (e.g., logic). Hence, the focus on the representation of properties will be on the second type. These type of properties are derived from various sources. Shahar argues for explicitly specifying the design rationale in the guideline itself [57]. In practice, however, other sources have to be used, e.g., recommendations from other guidelines (inter-guideline consistency), indicators [66], expert opinions, and general criteria relative to additional medical knowledge.

2.2. Temporal Logic

2.2.1. Introduction

Several temporal logics have been developed, in particular tense logics since the 1960s. Differences between logics result from different models of time and expressiveness. In linear temporal logics (e.g., Linear Temporal Logic (LTL) [45]), models form a linear trace, while in branching logics models typically (e.g., Computation Tree Logic (CTL) [8,14,21]) form a tree.

A model of a medical guideline, is a Kripke structure M over a set of atomic propositions AP, which formally is defined as a four tuple $M = (S, S_0, R, L)$ where S is a finite set of states, $S_0 \subseteq S$ is the set of initial states, $R \subseteq S \times S$ is a total transition relation, and $L : S \rightarrow 2^{AP}$ is a function that labels each state with the set of atomic propositions true in that state. A *path* in the model

$$
\begin{array}{lcl}
M, s \models p & \Leftrightarrow & p \in L(s) \\
M, s \models \neg f_1 & \Leftrightarrow & M, s \not\models f_1 \\
M, s \models f_1 \wedge f_2 & \Leftrightarrow & M, s \models f_1 \text{ and } M, s \models f_2 \\
M, s \models \mathbf{E} g_1 & \Leftrightarrow & \text{there is a path } \pi \text{ from } s \text{ such that } M, \pi \models g_1 \\
M, \pi \models f_1 & \Leftrightarrow & s \text{ is the first state of } \pi \text{ and } M, s \models f_1 \\
M, \pi \models \neg g_1 & \Leftrightarrow & M, \pi \not\models g_1 \\
M, \pi \models g_1 \wedge g_2 & \Leftrightarrow & M, \pi \models g_1 \text{ and } M, \pi \models g_2 \\
M, \pi \models \mathbf{X} g_1 & \Leftrightarrow & M, \pi^1 \models g_1 \\
M, \pi \models \mathbf{F} g_1 & \Leftrightarrow & \text{there exists a } k \geq 0 \text{ such that } M, \pi^k \models g_1 \\
M, \pi \models g_1 \mathbf{U} g_2 & \Leftrightarrow & \text{there exists a } k \geq 0 \text{ such that } M, \pi^k \models g_2 \text{ and} \\
 & & \text{for all } 0 \leq j < k, M, \pi^j \models g_1
\end{array}
$$

Figure 3. Semantics of temporal logic with f_1 and f_2 representing state formulas and g_1 and g_2 representing path formulas.

M from a state s is an infinite sequence $\pi = s_0 s_1 s_2 \ldots$ such that $s_0 = s$ and $R(s_i, s_{i+1})$ holds for all $i \geq 0$. With π^i we denote the suffix of π starting at s_i, i.e., $\pi^i = s_i s_{i+1} s_{i+2} \ldots$.

CTL uses atomic propositions, propositional connectives, *path quantifiers* and *temporal operators* for describing properties of *computation trees*, i.e., the tree that is formed by designating a state in the Kripke structure as the initial state and then unwinding the structure into an infinite tree according to the transition relation R with the initial state as root. This leads to two types of formulas: *state formulas*, which are true in a specific state, and *path formulas*, which are true along a specific path. A path formula is build up by applying one of the temporal operators to state formulas. In this chapter, the temporal operators used are **X**, **G**, **F**, and **U**. With $\mathbf{X}\varphi$ being true if φ holds in the next state, $\mathbf{G}\varphi$ if φ holds in the current state and all future states, $\mathbf{F}\varphi$ if φ holds in the current state or some state in the future, and $\varphi \mathbf{U} \psi$ if φ holds until eventually ψ holds. A state formula can be built inductively from atomic propositions, propositional connectives, and if f and g are path formulas, then $\mathbf{E}f$ and $\mathbf{A}f$ are state formulas. The path quantifiers **A** and **E** are used to specify that all or some of the paths starting at a specific state have some property.

The semantics of CTL is defined with respect to a Kripke structure M. Given a state formula f, the notation $M, s \models f$ denotes that f holds in state s of the Kripke structure M. Assuming that f_1 and f_2 are state formulas and g_1 and g_2 are path formulas, the relation $\models$ is defined inductively as shown in Figure 3. The remaining syntax consisting of $\vee, \rightarrow, \mathbf{G}, \mathbf{A}$ can be defined as usual, i.e., $f_1 \vee f_2 \equiv \neg(\neg f_1 \wedge \neg f_2)$, $f_1 \rightarrow f_2 \equiv \neg f_1 \vee f_2$, $\mathbf{G}g \equiv \neg\mathbf{F}\neg g$, and $\mathbf{A}f \equiv \neg\mathbf{E}\neg f$.

In contrast to CTL, LTL provides operators for describing events along a single computation path. Each formula is of the form $\mathbf{A}f$, with f being a path formula, which is either an atomic proposition or inductively defined as $\neg f$, $f \vee g$, $f \wedge g$, $\mathbf{X}f$, $\mathbf{F}f$, $\mathbf{G}f$, $f\mathbf{U}g$ with f, g path formulas. This language can be evaluated on Kripke structures as presented in Figure 3.

2.2.2. Expressiveness and Complexity

It is a well-known fact that the expressiveness of LTL and CTL is incomparable. For example, the following CTL statement

EF *normotension*

i.e., *'normotension' (normal blood pressure) may eventually occur* is not expressible in LTL. On the other hand, the following LTL formula

F G *normotension*

i.e., *eventually 'normotension' will always hold* is not expressible in CTL. Note that the formula **AF AG** *normotension* is stronger and expresses that there exists some state after which *all* patient groups have a normal blood pressure *at the same time*, which is more restrictive than the previous formula.

Both for theorem proving as well as model checking, reasoning using CTL is generally easier than reasoning in LTL, for example, LTL is in a higher complexity class. However, the discussion whether to use CTL or linear-time temporal logic (LTL) for model checking is far from being settled, as LTL is usually more intuitive and better suited as a specification language. For example, in [67], the advantages of linear-time frameworks is identified in terms of expressiveness, compositionality, property-specific abstractions, uniformity, and the use of bounded model checking.

2.3. Clinical Goal Representation

Several ontologies for intentions have been proposed in the context of medical guidelines.

2.3.1. Ontology of Goals in Breast Cancer

On the basis of a corpus of examples of clinical goal statements in breast cancer, an ontology was developed in [23]. As the authors describe, when clinical processes are designed and enacted, this should allow for the possibility of urgent changes to the care plan or "plan repair". In order to do this, a reason for each service has to be made explicit to be capable of recovering when goals are not achieved. They make a distinction between knowledge and action goals, where knowledge goals deal with acquiring information and deciding between alternative hypothesis about the world and action goals deal with achieving some state of the world and enacting tasks. A conclusion of this work is that this delivers a more balanced classification of types of goals compared to other ontologies. However, further work is expected to be needed in order to make a final scheme in the context of breast cancer.

2.3.2. Asbru Intentions

Although most guideline representation languages allow for representation of goals and intentions, representation of intentions is most developed in the Asbru language [38]. These intentions are considered "temporal patterns of provider actions and patient states, at different levels of abstraction, that should be maintained,

achieved, or avoided" [57]. Furthermore, a distinction is made to whether the intention refers to a clinical state or action and whether the intention holds during enactment of the clinical process (*intermediate*) or after it has been completed (*overall*). The intention may also contain a rich temporal structure.

In the verification methods using Asbru, these intentions are typically formalised in temporal logic. For example, in [7], the Asbru intention "achieve overall state: α" is formalised as:

$$\mathbf{AG}\,(\mathit{current\_plan} = \mathit{completed} \rightarrow \mathbf{AF}\,\mathbf{AG}\,\alpha)$$

From a formal point this raises some questions. For example, in this example, it is possible, due to the use of the **F** operator, that α holds much later than the current plan. Moreover, one could argue that the similar looking, but non-equivalent, LTL formula:

$$\mathbf{G}\,(\mathit{current\_plan} = \mathit{completed} \rightarrow \mathbf{F}\,\mathbf{G}\,\alpha)$$

is the right formalisation. The point that we would like to make here is that formal languages are particularly useful to discuss such subtle differences. Even though in the original Asbru specification, the property seems uncomplicated, questions can be raised with respect to the intended semantics.

2.4. Methodology for the Specification of Properties

A major problem in the specification of properties as we have presented so-far is that properties from sources other than the original guideline differ in terminology. This is especially common in medicine where terms typically have multiple synonyms. Furthermore, properties may address aspects which are not even contained in the guideline. These properties can only be verified if the guideline is enriched by the additional aspects.

The problem of attaching the terminology of guidelines to the terminology found in properties is commonly recognised in the literature [26]. This problem could be further addressed using ontologies to standardise terminology as found in some of the guideline representation languages.

A structured approach to bridge the gap between informal properties and a temporal formula matching the aspects and terminology of the guideline has been proposed in [62]. This paper introduces a stepwise approach to formalise properties. The original informal goal is *reduced* in a first step to the scope of the guideline. In a second step, the goal is *normalised* to determine the expected behaviour and the timing constraints, i.e., start and end points between which the behaviour should be observed. After the normalisation step, it is rather easy to correctly *formalise* the property in a formalism called Goal Definition Language (GDL). A final step, the *attachment*, maps concepts and terminology of the property definition to concepts of the guideline.

This process does not solve the problem of mapping the different terminology of properties and guidelines. However, it gives structure to the process of attaching properties to guidelines and makes sure that medical domain experts are able to perform and understand the different steps. Assuming the domain experts are aware of both the ontology of the property as well as the ontology found in the guideline, this methodology enables the validation of properties.

3. Verification

Properties can be verified *on-the-fly*, i.e., properties define runtime constraints which are monitored during guideline execution, or *prior to execution*. Runtime constraints are always evaluated *for a given case* while verification prior to execution has to take into account *every possible case*. The latter is thus much more complex. This chapter is focussed on verification prior to execution.

There are roughly two verification approaches, namely model checking [13, 47] which explores a (finite) model and theorem proving which explores logical derivations of a theory. There has been a particular focus on the use of theorem provers for reasoning about programs, which can be traced back the well-known Boyer-Moore theorem prover in the early 1960s [10,9]. In AI, theorem provers have for example been used to verify knowledge-based systems (e.g., [22]), as the knowledge representation is often based on logic.

3.1. Interactive Theorem Proving

The European project Protocure[2] has had a major impetus on the use of formal methods for the verification of medical guidelines. In [64], the results of this work is summarised. The guideline that is used deals with the treatment of jaundice and diabetes and is modelled in Asbru. This model was then, partly manually and partly mechanically, translated to temporal logic and given to an interactive theorem prover, called KIV[3] [6,27]. Indicators and intentions mentioned in the guideline were then used as correctness criteria in order to verify these guidelines. This work was subsequently extended in several ways. First, the semantics of a part of the Asbru language was incorporated in the KIV theorem prover, making it possible to translate a guideline model completely automatically [51], including complex time annotations that can occur in guidelines (for more details see Section 1.2.2). Second, the addition of background knowledge in order to verify more general quality criteria was investigated [29]. A complete description of the latter work can be found in [50].

Interactive theorem proving systems are sometimes called "proof assistants" as they do not construct proofs themselves, but rather support the construction of a proof by a user. In mathematics, proof assistants such as Mizar, HOL, and Coq are popular; in these systems almost all proof steps have to be performed manually. KIV was designed for use in program verification and attempts at providing more proof steps automatically; however, it does not exhaustively investigate large search spaces which keeps the amount of time it spends on calculations under control. The main advantage of such techniques is that it can, in principle, handle problems of arbitrary complexity, hence it is especially suitable if the model of the guideline is detailed and contains many complex constructs. By abstracting parts of the guideline, more automated techniques, such as automated theorem proving or model checking become feasible. These have also been applied to medical guidelines and are discussed below.

[2] `http://www.protocure.org` [accessed February 2008]
[3] `http://www.informatik.uni-augsburg.de/swt/kiv` [accessed February 2008]

3.2. Resolution-based Theorem Proving

It was shown that for reasoning about models of medical knowledge, for example in the context of medical expert systems [33], classical automated reasoning techniques (e.g., [48,70]) are a practical option. In [31], the use of automatic theorem proving techniques for quality checking medical guidelines was studied. Translation of temporal logic yields a restricted first-order theory. Such a formalisation is suitable for use in standard resolution-based theorem provers. Typically, automated theorem provers require little or no interactions compared to interactive theorem provers. Resolution-based theorem proving facilities have been proven successful for many complex problems in algebra [43] and logic [32]; however, it does put a certain limit to the complexity of the guideline that can be verified.

3.3. Model Checking

With model checking [15], temporal properties can be automatically verified for a given state transition system. In principle, model checking is automatic, however, the application is limited to finite state transition systems. During the last years, tools have been refined, additional methods to automatically reduce the state space have been introduced, and computers, in general, have become more and more powerful such that nowadays, systems with a very large number of states can be automatically verified. Popular model checkers are SMV[4] [15], which uses Binary Decision Diagrams [35], SPIN[5] [28], an explicit state model checker, and others.

Model checking has become very helpful in software engineering for analysing reactive system designs. A medical guideline can be viewed as a concurrent system and model checking can be applied. It is necessary to transform the medical guideline to the input language of the model checker. Transformation can be automated by writing a suitable compiler. After the guideline has been transformed, temporal properties expressed either in CTL or in LTL can be verified. An interesting aspect of model checking is that, if the property does not hold, a counter example is provided which helps in improving the medical guideline or property. However, if verification does not terminate, the guideline model must be abstracted to reduce the state space. This abstraction must be provided manually such that, in general, model checking still requires expert knowledge.

In [7], the Cadence SMV model checker has been used to verify temporal properties of Asbru medical guidelines. An Asbru guideline is automatically translated to the SMV input language. A large subset of the Asbru language is translated, however, complex conditions and certain details of temporal constraints are currently abstracted. As a consequence, only a restricted set of properties can be verified. In the paper a selection of structural properties is considered. Verification of these properties has revealed a number of errors in the Asbru model.

GLARE medical guidelines have also been verified with model checking [25]. A guideline is translated to the input language of the SPIN model checker. This translation is fully automatic.

[4] `http://www.cis.ksu.edu/santos/smv-doc` [accessed February 2008]
[5] `http://www.spinroot.com/` [accessed February 2008]

3.4. Other Techniques

Several other techniques have been proposed primarily for checking that the guideline model is internally consistent. They may be used as means to validation (i.e., check that the model represents the guideline) or verification (i.e., to check that the guideline is correct).

Rule-based and Decision Table. In [59,58], guidelines are represented as a decision table and completeness and ambiguousness are investigated of the guideline. In [36], guidelines are looked upon as rules, similar to modelling as done in for example *Arden Syntax*. Verification involves checking that the guideline is complete, i.e., for every possible situation an advice is given using a tool called "Commander". In their study of a guideline for childhood immunisation, they were able to identify a number of missing immunisation rules.

Coherence Analysis. In [18,20], Asbru models are translated to first-order logic and rich structural properties are being investigated in order to check the coherence of the model. Similar to the structure, the coherence of temporal constraints that have been put on the (Asbru) model is discussed in [19]. If problems with coherence can be traced back to the guideline, this can be considered a form of verification; however, the verification mostly deals with the formal model rather than checking the correctness of the original guideline.

Petri Nets. A different method to simulate dynamic systems is by modelling the system as a so called Petri net [41,42], which is also known as a place/transition net or P/T net. A Petri net is a mathematical representation of discrete distributed systems that graphically depicts the structure of a distributed system as a directed bipartite graph with annotations using place nodes, transition nodes, and directed arcs. A whole range of tools is available for using Petri nets to model and simulate complex dynamic systems.[6] Recently, this technique has been applied to analyse biological systems [40,37]. In [40], Petri nets have been used to model malaria parasites invading host erythrocytes. Petri nets also form the basis for the GUIDE guideline representation language, where they have been used for simulation of the health care processes [46].

3.5. Verification of Temporal Constraints

Checking temporal constraints is not impossible in a theorem proving and model checking approach. In fact, reasoning using temporal constraints been done using interactive theorem proving [51]. However, in many other cases other techniques are employed. One approach to deal with temporal constraints involves monitoring temporal constraints *during executing* of guidelines. This is possible in various guideline modelling frameworks. A more challenging approach is to verify the constraints *prior to execution*.

The temporal constraints in GLARE can be mapped to STP-trees , an exension of the "standard" STP (Simple Temporal Problems, see [17]) to cope with

[6] `http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html` [accessed February 2008]

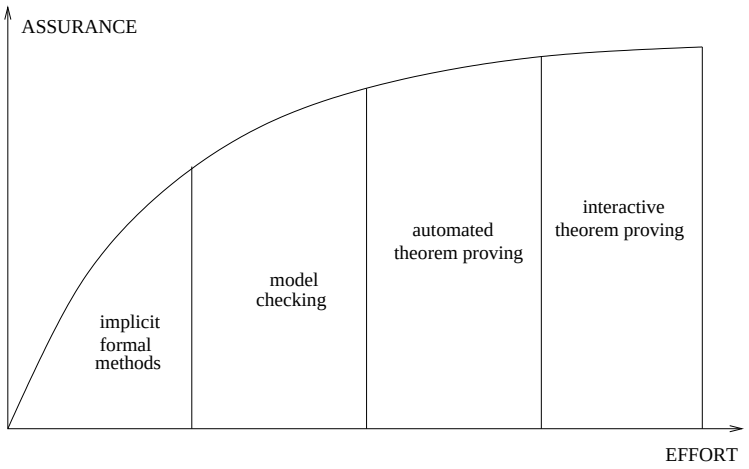

Figure 4. A spectrum of formal methods for formal verification allowing a tradeoff in the properties one can verify (assurance dimension) against the effort one needs to invest to obtain results (effort dimension).

(possibly periodical) repeated actions. Algorithms exist to automatically verify the *consistency* of a given set of constraints prior to the execution of guidelines [2].

4. Applications of Formal Methods

4.1. Verification

Figure 4 shows a range of formal methods ranging from cheap and incomplete to very expensive and complete (loosely based on a picture by Rushby [49]). Many of the techniques discussed in Section 3.4 can be considered implicit formal methods (e.g., type checkers, parsers), as they are largely automatic and could be, for example, integrated in guideline authoring software. As the picture suggests, some other techniques, such as model checking and theorem proving require significant amount of work and is unlikely to be done by guideline developers. Nonetheless, as we have shown in this chapter, promising results have been reported in literature. Given the fact that guidelines have a large impact in medical practice, costs associated with the use of formal methods might be justified. We elaborate on this point somewhat further in the final section.

4.2. Adaptation

Guideline adaptation is a process in which existing guidelines are adapted so that they can be used within a different care setting. Several reasons may exist for adapting the recommendations of an existing guideline to suit a local context, e.g., cultural differences, constraints on resources, end-user involvement, etc, though it is often the case, that the adaptation is faithful to the original guideline [34,30,39]. It is possible to employ formal methods in order to make sure that the adaptations do not violate the original guideline. A review of adaptations as done in practice can be found in Chapter 7 of this book.

4.3. Compliance

More general to the idea of using formal methods for checking that an adaptation is compliant to the guideline is checking that the actual clinical process is compliant to the guideline. If the clinical process is recorded as in, for example, an electronic patient record, then verification can be done on this basis. The use of formal methods can then be described as *critiquing*, i.e., to find differences between the actual actions and a set of 'ideal' actions as described by a clinical guideline. This topic is further discussed in Chapter 9 of this book.

5. Discussion

In this chapter, we have given an overview on the use of formal methods in the analysis of medical guidelines. In guideline specification, we found that there are quite a number of specification languages that have been proposed, but only a limited amount of work has been done in providing a formal semantics of these languages, which renders only a few of them suitable for the use of formal methods. Research in the area of specification of properties has mainly focused on (1) the acquisition of properties, and (2) design of an ontology of properties. Languages that are used are either informal, which again makes it problematic to use them in formal methods, or are closely based on a standard temporal logic. Verification has moved forward the last few years from the ad-hoc use of formal models in order to analyse certain aspects of guidelines to more systematic approaches using techniques that are now widely used in the formal methods community such as theorem proving and model checking.

In medicine, safety is extremely important, witnessed by the fact that ensuring safety is the primary preoccupation of regulatory agencies. Nonetheless, mistakes are made in hospitals; in fact, it was found that every year, in the Netherlands, 30,000 people are harmed and 1,700 people die in hospitals due to causes that can be avoided [11]. As medical errors have such far reaching consequences, there is good reason to investigate in the use of formal methods in medicine. Guidelines play an important role and errors in these guidelines may contribute to medical errors and mistakes. It is therefore clear that making sure that the guidelines are of the highest possible quality is essential.

We believe the benefits of using formal methods on top of other techniques to improve guidelines speak for themselves. First, formal verification and especially interactive verification is very helpful in analysing the language itself as formal methods force one to formalise the semantics. For example, this resulted in a number of problems with the Asbru language, which were detected while verifying properties of a medical guideline. As a consequence, the formal semantics of time annotations in Asbru has been significantly improved by verifying properties of the language itself [52]. The same holds for medical guidelines itself: much can be gained by formalising medical guidelines in practice. Informal text is interpreted differently by different readers and it is difficult to keep all parts of an informal medical guideline consistent, as a guideline is typically written by various authors. However, the true challenge remains to introduce a standardised formal language

into the practice of guideline development. Only after this challenge has been met, the true potential of formal verification can be seen.

6. Research Agenda

As mentioned in the discussion, significant progress has been made in the last few years in the area of formal methods and clinical guidelines, just considering the amount of work that has been produced. However, much work still has to be done. Some of the issues that could be further investigated are mentioned here.

First, it would be convenient to introduce a standardised, machine readable format into the guideline development process. Otherwise a gap remains between informal text of the medical guideline and the machine readable model which is the basis for further analysis. The machine readable format must be easy to understand and yet expressive enough for a large variety of medical guidelines. Currently, a number of standardised languages exist for writing down medical guidelines, yet none of these languages have been used by guideline developers on a large scale, nor do they take into account the special features of formal verification. Similarly, a detailed formal semantics should underly the machine readable format and should be used as a standard for building tools such as editor, interpreters, compilers, etc.

Formal verification of properties is difficult and time consuming. While interactive verification can only be performed by logicians, automatic methods have potential to be applied by guideline designers. This raises the question in which situation a certain technique should be employed. Guidelines for guideline developers could improve the practical usefulnes as well as the visibility of the research that is being done.

Finally, and what is possibly most challenging is that there seems to be a gap between the work in this area that has been done so far and the medical community. It seems to be notoriously difficult to get medically relevant results, which might be due to the fact that only very few medical doctors are involved in this research. For example, it is relevant to know whether or not a guideline is "safe" or "correct", which are concepts that are difficult to grasp. However, in order to make a real impact in medicine, such difficult questions will have to be answered.

References

[1] M. Alberti, M. Gavanelli, E. Lamma, P. Mello, P. Torroni, and G. Sartor. Mapping deontic operators to abductive expectations. *Computational & Mathematical Organization Theory*, 12(2):205–225, 2006.

[2] L. Anselma, P. Terenziani, S. Montani, and A. Bottrighi. Towards a comprehensive treatment of repetitions, periodicity and temporal constraints in clinical guidelines. *Artificial Intelligence Medicine*, 38(2):171–195, 2006.

[3] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider. *The Description Logic Handbook - Theory, Implementation and Applications*. Cambridge University Press, 2003.

[4] M. Balser, C. Duelli, and W. Reif. Formal semantics of Asbru - an overview. In H. Ehrig, B. Kraemer, and A. Ertas, editors, *Proceedings of the 6th Biennial world Conference on Integrated Desgin and Process Technology*, pages 1–8, USA, 2002. Society for Desgin and Process Science.

[5] M. Balser, C. Duelli, W. Reif, and J. Schmitt. Formal semantics of Asbru – v2.12. Technical report, University of Augsburg, June 2006.

[6] M. Balser, W. Reif, G. Schellhorn, K. Stenzel, and A. Thums. Formal system development with KIV. In T. Maibaum, editor, *Fundamental Approaches to Software Engineering*, number 1783 in LNCS, pages 363–366. Springer-Verlag, 2000.

[7] S. Bäumler, M. Balser, A. Dunets, W. Reif, and J. Schmitt. Verification of medical guidelines by model checking – a case study. In A. Valmari, editor, *Proceedings of 13th International SPIN Workshop on Model Checking of Software*, volume 3925 of *LNCS*, pages 219–233. Springer-Verlag, 2006.

[8] M. Ben-Ari, Z. Manna, and A. Pnueli. The temporal logic of branching time. *Acta Inform*, 20, 1983.

[9] R. S. Boyer, M. Kaufmann, and J. S. Moore. The Boyer-Moore theorem prover and its interactive enhancement. *Computers and Mathematics with Applications*, 29(2):27–62, 1995.

[10] R. S. Boyer and J. S. Moore. Proving theorems about LISP functions. *Journal of the Association for Computing Machinery*, 22(1):129–144, 1975.

[11] M.C. de Bruijne, M. Zegers, L.H.F. Hoonhout, and C. Wagner. *Onbedoelde schade in Nederlandse ziekenhuizen: dossieronderzoek van ziekenhuisopnames in 2004.* Instituut voor Extramuraal Geneeskundig Onderzoek (NIVEL), 2007.

[12] F. Chesani, A. Ciampolini, P. Mello, M. Montali, P. Torroni, M. Alberti, and S. Storari. Protocol specification and verification by using computational logic. In F. Corradini, F. de Paoli, E. Merelli, and A. Omicini, editors, *WOA 2005: Dagli Oggetti agli Agenti. 6th AI*IA/TABOO Joint Workshop "From Objects to Agents": Simulation and Formal Analysis of Complex Systems*, pages 184–192. Pitagora Editrice Bologna, November 2005.

[13] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[14] E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Logic of Programs: Workshop*, number 131 in LNCS, Yorktown Heights, NY, May 1981. Springer.

[15] E.M. Clarke, O. Grumberg, and A.D. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, London, England, 2001.

[16] P.A. de Clercq, J.A. Blom, H.H.M. Korsten, and A. Hasman. Approaches for creating computer-interpretable guidelines that facilitate decision support. *Artificial Intelligence in Medicine*, 31:1–27, 2004.

[17] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1–3):61–95, 1991.

[18] G. Duftschmid and S. Miksch. Knowledge-based verification of clinical guidelines by detection of anomalies. *OEGAI Journal*, pages 37–39, 1999.

[19] G. Duftschmid, S. Miksch, and W. Gall. Verification of temporal scheduling constraints in clinical practice guidelines. *Artificial Intelligence in Medicine*, 25:93–121, 2002.

[20] G. Duftschmid, S. Miksch, Y. Shahar, and P. Johnson. Multi-level verification of clinical protocols. In *Proceedings of the Workshop on Validation and Verification of Knowledge-Based Systems*, Trento, Italy, 1998.

[21] E.A. Emerson and E.M. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In *Automata, Languages and Programming*, number 85 in LNCS, pages 169–181, 1980.

[22] D. Fensel, A. Schonegge, R. Groenboom, and B. Wielinga. Specification and verification of knowledge-based systems. In B.R. Gaines and M. A. Musen, editors, *Proceedings of the 10th Banff knowledge acquisition for knowledge-based systems workshop*, pages 1–20, Department of Computer Science, University of Calgary, 1996.

[23] J. Fox, A. Alabassi, E. Black, C. Hurt, and T. Rose. Modelling clinical goals: a corpus of

examples and a tentative ontology. *Stud Health Technol Inform*, 101:31–45, 2004.
[24] J. Fox and S. Das. *Safe and Sound: Artificial Intelligence in Hazardous Applications*. AAAI Press, 2000.
[25] L. Giordano, P. Terenziani, A. Bottrighi, S. Montani, and L. Donzella. Model checking for clinical guidelines: an agent-based approach. In *AMIA 2006*, pages 171–195, Washington, 2006. American Medical Informatics Association.
[26] P. Groot, A. Hommersom, P. Lucas, R. Serban, A. ten Teije, and F. van Harmelen. The role of model checking in critiquing based on clinical guidelines. In R. Bellazzi, A. Abu-Hanna, and J. Hunter, editors, *11th Conference on Artificial Intelligence in Medicine*, number 4595 in LNAI, pages 411–420. Springer-Verlag Berlin Heidelberg, 2007.
[27] D. Haneberg, S. Bäumler, M. Balser, H. Grandy, F. Ortmeier, W. Reif, G. Schellhorn, J. Schmitt, and K. Stenzel. The user interface of the kiv verification system - a system description. *ENTCS special issue*, 2007. To appear.
[28] G.J. Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley Professional, 2004.
[29] A.J. Hommersom, P. Groot, P.J.F. Lucas, M. Balser, and J. Schmitt. Verification of medical guidelines using background knowledge in task networks. *IEEE Transactions on Knowledge and Data Engineering*, 19(6):832–846, 2007.
[30] A.J. Hommersom, P. Groot, P.J.F. Lucas, M. Marcos, and B. Martínez-Salvador. *Computer-based medical guidelines and protocols: A primer and current trends*, chapter A Constraint-based Approach to Medical Guidelines and Protocols. IOS Press, 2008.
[31] A.J. Hommersom, P.J.F. Lucas, and P. van Bommel. Automated theorem proving for quality-checking medical guidelines. In *Proceedings of CADE-20 Workshop on Empirically Successful Classical Automated Reasoning (ESCAR)*, 2005.
[32] T. Jech. OTTER experiments in a system of combinatory logic. *Journal of Automated Reasoning*, 14(3):413–426, 1995.
[33] P.J.F. Lucas. The Representation of Medical Reasoning Models in Resolution-based Theorem Provers. *Artificial Intelligence in Medicine*, 5:395–419, 1993.
[34] M. Marcos, B. Martínez-Salvador, A.J. Hommersom, P. Groot, P.J.F. Lucas, A. Jovell, and S. Blancafort. Deliverable 5.1: Case-study in transformations for protocol development, 2006.
[35] K. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, Boston, 1993.
[36] D.W. Miller, S.J. Frawley, and P.L. Miller. Using semantic constraints to help verify the completeness of a computer-based clinical guideline for childhood immunization. *Computer Methods and Programs in Biomedicine*, 58(3):267–280, March 1999.
[37] M. Peleg, D. Rubin, and R.B. Altman. Using petri net tools to study properties and dynamics of biological systems. *J Am Med Inform Assoc.*, 2005.
[38] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R.A. Greenes, R. Hall, P.D. Johnson, N. Jones, A. Kumar, S. Miksch, S. Quaglini, A. Seyfang, E.H. Shortliffe, and M. Stefanelli. Comparing computer-interpretable guideline models: a case-study approach. *Journal of the American Medical Informatics Association*, 10(1):52–68, 2003.
[39] M. Peleg, D. Wang, A. Fodor, S. Keren, and E. Karnieli. *Computer-based medical guidelines and protocols: A primer and current trends*, chapter Adaptation of practice guidelines for clinical decision support: a case study of diabetic foot care. IOS Press, 2008.
[40] M. Peleg, I. Yeh, and R.B. Altman. Modelling biological processes using workflow and petri net models. *Bioinformatics*, 18(6), 2002.
[41] J.L. Peterson. Petri nets. *ACM Computing Surveys*, 9(3):223–252, 1977.
[42] J.L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall, PTR Upper Saddle River, NJ, USA, 1981.
[43] J.D. Phillips and P. Vojtěchovskiý. Linear groupoids and the associated wreath products. *Journal of Symbolic Computation*, 40(3):1106–1125, 2005.
[44] G.D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, 1981.
[45] A. Pnuelli. A temporal logic of concurrent programs. *Theoretical Computer Science*, 13:45–60, 1981.
[46] S. Quaglini, M. Stefanelli, A. Cavallini, G. Micieli, C. Fassino, and C. Mossa. Guideline-

based careflow systems. *Artificial Intelligence in Medicine*, 20(1):5–22, 2000.

[47] J-P. Queille and J. Sifakis. Specification and verification of concurrent systems is CESAR. In *International Symposium on Programming, LNCS 137*, pages 337–351. Springer Verlag, 1982.

[48] J.A. Robinson. Automated Deduction with Hyperresolution. *International Journal of Computatational Mathematics*, 1:23–41, 1965.

[49] J. Rushby. Formal methods and their role in the certification of critical systems. Technical report, CSL-95-1, March 1995.

[50] J. Schmitt, M. Balser, S. Bäumler, A. Dunets, N. Moebius, F. Lautenbacher, W. Reif, M. Marcos, Martínez-Salvador, A.J. Hommersom, P. Groot, P.J.F. Lucas, R. Serban, A. ten Teije, F. van Harmelen, A. Seyfang, S. Miksch, K. Rosenbrand, J. Wittenberg, and J. van Croonenborg. Deliverable d4.2c: Formal verification of selected guideline properties, 2006. Available at `http://www.protocure.org`.

[51] J. Schmitt, A. Hoffmann, M. Balser, W. Reif, and M. Marcos. Interactive verification of medical guidelines. In *14th Symposium on Formal Methods (FM'06)*, volume 4085 of *LNCS*, pages 21–27. Springer, 2006.

[52] J. Schmitt, W. Reif, A. Seyfang, and S. Miksch. Temporal dimension of medical guidelines: The semantics of asbru time annotations. In *ECAI 2006 WS – AI techniques in healthcare: evidence-baded guidelines and protocols*, 2006.

[53] S. Schulz and U. Hahn. A description logic approach to clinical guidelines and protocols. In *Symposium on Computerized Guidelines and Protocols*, 2004.

[54] S. Schulz, Markó. K., and B. Suntisrivaraporn. Complex occurrents in clinical terminologies and their representation in a formal language. In *Proc. of the First European Conference on SNOMED CT (SMCS'06)*, Copenhagen, Denmark, 2006.

[55] A. Seyfang and S. Miksch. Advanced temporal data abstraction for guideline execution. In *Proceedings of the Symposium on Computer-based Support for Clinical Guidelines and Protocols*, pages 88–102, 2004.

[56] A. Seyfang, S. Miksch, W. Horn, M. Urschitz, C. Popow, and C. Poets. Using time-oriented data abstraction methods to optimize oxygen supply for neonates. In *Artificial Intelligence in Medicine*, pages 217–226, 2001.

[57] Y. Shahar, S. Miksch, and P. Johnson. The asgaard project: A task-specific framework for the application and critiquing of time-orientied clinical guidelines. *Artificial Intelligence in Medicine*, 14:29–51, 1998.

[58] R.N. Shiffman. Representation of clinical practice guidelines in conventional and augmented decision tables. *Journal of the American Medical Informatics Association*, 4:382–393, 1997.

[59] R.N. Shiffman and R.A. Greenes. Improving clinical guidelines with logic and decision-table techniques: Application in hepatitis immunization recommendations. *Med Decis Making*, 14:245–254, 1994.

[60] O. Sokolsky, B.G. Silverman, and I. Lee. Consistency of clinical guidelines. Position Statement (`http://www.glif.org/workshop/position_stmt/ConsistencyPosition.pdf`), 2000.

[61] M. Stacey and C. McGregor. Temporal abstraction in intelligent clinical data analysis: A survey. *Artificial Intelligence in Medicine*, 39(1):1–24, 2007.

[62] R. Stegers, A. ten Teije, and F. van Harmelen. From natural language to formal proof goal: Structured goal formalisation applied to medical guidelines (extended abstract). In S. Staab and V. Svatek, editors, *Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management (EKAW'06)*, LNAI. Springer-Verlag, 2006.

[63] D. Sutton and J. Fox. The syntax and semantics of the PRO*forma* guideline modeling language. *Jounal of the American Medical Informatics Association*, 10(5):433–443, 2003.

[64] A. ten Teije, M. Marcos, M. Balser, J. van Croonenborg, C. Duelli, F. van Harmelen, P. Lucas, S. Miksch, W. Reif, K. Rosenbrand, and S. Seyfang. Improving medical protocols by formal methods. *Artificial Intelligence in Medicine*, 36(3):193–209, 2006.

[65] S.W. Tu and M.A. Musen. From guideline modeling to guideline execution: Defining guideline-based decision-support services. In *Proc. AMIA Symposium*, pages 863–867, Los

Angeles, CA, 2000.

[66] M. van Gendt, A. ten Teije, R. Serban, and F. van Harmelen. Formalising medical quality indicators to improve guidelines. In *Proceedings of the Tenth European Conference on Artificial Intelligence in Medicine (AIME'05)*, volume 3581 of *LNAI*. Springer Verlag, 2005.

[67] M.Y. Vardi. Branching vs. linear time: Final showdown. *Lecture Notes in Computer Science*, 2031:1–22, 2001.

[68] D. Wang, M. Peleg, S.W. Tu, A.A. Boxwala, O. Ogunyemi, Q. Zeng, R.A. Greenes, V.L. Patel, and E.H. Shortliffe. Design and implementation of the glif3 guideline execution engine. *Journal of Biomedical Informatics*, 37(5):305–318, October 2004.

[69] J.M. Wing. A specifier's introduction to formal methods. *IEEE Computer*, 23(9):8–24, 1990.

[70] L. Wos, R. Overbeek, E. Lusk, and J. Boyle. *Automated Reasoning: Introduction and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1984.