Experiences in Quality Checking Medical Guidelines using Formal Methods

Perry Groot and **Arjen Hommersom** and **Peter Lucas**[1]
Michael Balser and **Jonathan Schmitt**[2]

Abstract. In health care, the trend of evidence-based medicine, has led medical specialists to develop medical guidelines, which are large nontrivial documents suggesting the detailed steps that should be taken by health-care professionals in managing the disease in a patient. In the Protocure project the objective has been to assess the improvement of medical guidelines using formal methods. This paper reports on some of our findings and experiences in quality checking medical guidelines. In particular the formalisation of meta-level quality criteria for good practice medicine, which is used in conjunction with medical background knowledge to verify the quality of a guideline dealing with the management of diabetes mellitus type 2 using the interactive theorem prover KIV. For comparison, analogous investigations have been performed with other techniques including automatic theorem proving and model checking.

1 Introduction

Computer-based decision support in health-care is a field with a long standing tradition, dealing with complex problems in medicine such as diagnosing disease and assisting in the prescription of appropriate treatment. The trend of the last decades has been to base clinical decision making more and more on sound scientific evidence, i.e; this has been called *evidence-based medicine* [41, 45]. In practice this has led organisations of medical specialists in particular areas to develop medical guidelines, i.e., structured documents suggesting the detailed steps that should be taken by health-care professionals in managing the disease of a patient, to promote standards of medical care. Ethical concerns about evidence-based medicine have been raised [11] and there is a potential risk that medical guidelines do harm when improperly developed [44]. However, guidelines have also shown to improve health-care outcomes [44] and may even reduce the costs of care up to 25% [8].

Researchers in Artificial Intelligence have picked up on the increasing use of medical guidelines and are working towards offering computer-based support in the development and deployment of guidelines using computer-oriented languages and tools [10, 30]. This has given rise to the emergence of a new paradigm for the modelling of complex clinical processes as a 'network of tasks', where a task consists of a number of steps, each step having a specific function or goal [15, 28]. Examples of languages that support task models, and which have been evolving since the 1990s, include PRO*forma* [16, 17], Asbru [37, 40], EON [42, 43], and GLIF3 [28]. In this work, medical guidelines are considered as real-world examples of structured documents, which can benefit from formalisation, although experience has shown that looking upon medical guidelines as formal objects is a nontrivial task [29].

One of the reasons for this is that medical guidelines should not be considered static objects as they are changed on a regular basis as new scientific evidence becomes available. Rapidly changing and evolving evidence makes it difficult to adjust guidelines in such a way as to keep them up to date. As a consequence, computer-based support of guideline development should also be concerned with the updating of guidelines, i.e., indicate where guidelines should be updated in light of new evidence.

In this article, we approach this problem by applying formal methods to checking the quality of medical guidelines. Here, we are mainly concerned with checking of *general* quality criteria of good practice medicine a guideline should comply to. This has been called the meta-level approach to quality checking of medical guidelines [24]. For example, a guideline should preclude the prescription of redundant drugs, or advise against a prescription of a treatment that is less effective than some alternative. Newly obtained evidence may invalidate properties of a guideline, because, for example, new patient management options have arisen or financial costs have decreased through new developments in drug therapy.

A solid foundation for the application of formal methods to the quality checking of medical guidelines can already be found in literature. In [15, 25] logical methods have been used to analyse properties of guidelines. We have shown in [24] that the theory of abductive diagnosis can be taken as a foundation for the formalisation of quality requirements of a medical guideline in temporal logic. This result has been used in verifying quality requirements of good practice medicine of alternative treatments [21].

The contribution of this paper, is that we formalise quality requirements of medical guidelines which include, besides separate treatments, also the temporal relations between separate treatments, by which we mean the order in which they are prescribed. Second, using our quality requirements and medical background knowledge, we interactively verify a guideline dealing with the management of diabetes mellitus type 2. More specifically, we model the guideline as a 'network of tasks' using the language Asbru and, additionally, verify meta-level properties for this model using KIV, an interactive theorem prover [6]. To the best of our knowledge, verification of a fully formalised guideline, as a network of tasks, using medical background knowledge has not been done before. The presented framework provides a sound formal foundation for further research in quality checking of medical guidelines and the temporal relations among different treatments involved.

[1] Institute for Computing and Information Sciences, Radboud University Nijmegen, e-mail:{perry,arjenh,peterl}@cs.ru.nl
[2] Institut für Informatik, Universität Augsburg, D-86135 Augsburg, e-mail: {balser,jonathan.schmitt}@informatik.uni-augsburg.de

The remainder of this paper is structured as follows. Section 2 gives an introduction to the Protocure project and the methodology employed within the project.[3] Section 3 gives an introduction to medical guidelines. Section 4 gives an overview of Asbru, the guideline representation language used throughout our work. Section 5 discusses in more detail the approach to formal verification of medical guideline. It discusses the main elements of a guideline a formal language should address and discusses the three types of knowledge involved: background knowledge, the treatment order in the guideline, and the quality requirements. Section 6 discusses in more detail how to formalise these three knowledge types in the context of diabetes mellitus type 2. Section 7 discusses in more detail how to translate everything into the KIV system. Section 8 gives the results with interactive verification with the theorem prover KIV.

2 Protocure: Improving medical guidelines by formal methods

The aim of the Protocure project has been to take the formalisation of guidelines one step further, by using guideline representation languages for modelling medical guidelines as formal objects and integrating them with formal methods for quality checking. The main objective of the Protocure project was the assessment of guideline improvement using formal methods, which has been done using the methodology shown in Figure 1 [2]. Initially, a medical guideline is selected, which is then gradually transformed into a formal representation. This transformation basically consists of two phases. Firstly, the guideline is modelled in the Asbru language, which is a language specifically designed for the modelling of medical guidelines. Asbru is described in detail in Section 4. Secondly, the Asbru model of the guideline is transformed in a formal language that can be used for verification. Formal languages, tools, and techniques that have been used within the Protocure project are (1) KIV, an interactive theorem prover that uses a variant of temporal logic, (2) Otter, an automatic theorem prover, and (3) SMV, a model checker that uses computation tree logic and linear temporal logic. These are described in more detail in forthcoming sections.

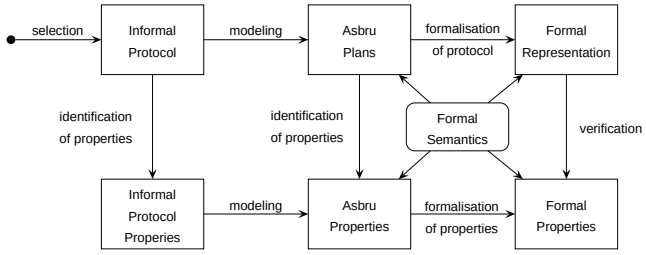

Figure 1. The process of guideline formalisation and verification as done in the Protocure project.

Closely related to the modelling of the guideline is the modelling of the properties one wants to check for the guideline under study. Several sources can be used to obtain such properties, which then also need to be translated into a formal language that will be used for verification. The simplest properties, so-called *structural properties* [12], are those properties that ensure that the Asbru model created is correct, e.g., reachability of all states. More complex properties deal with the medical intentions one wants to obtain when using a guideline. These can be derived from the guideline text or for example from quality indicators independently developed from the guideline [18]. Such properties need interpretation and were found to be harder to formalise. In this paper, we look, among others, at a specific type of such complex properties, namely meta-level quality requirements, which state requirements for general good medical practice.

[3] http://www.protocure.org

3 Medical guidelines

Guidelines, medical guidelines, or practice guidelines are all commonly used abbreviations for the full term 'clinical practice guideline'. An often cited definition of guidelines is the one by Field and Lohr [14]:

> Clinical practice guidelines are systematically developed statements to assist practitioner and patient decisions about appropriate health care for specific clinical circumstances.

Though 'protocol' is often synonymously used for 'guideline', a protocol gives detailed statements about *how* one should act in daily practice, whereas a guideline gives more general scientifically founded statements about *what* should be done. Protocols are often seen as more detailed, practice-oriented versions of a guideline [27]. In this work the focus is on medical guidelines.

An example of a fragment of a guideline is shown in Figure 2. It is part of the guideline for general practitioners about the treatment of diabetes mellitus type 2 [34]. General practitioners' guidelines are normally quite compact. Guidelines for medical specialists are often large – they can be as large as 100 pages – but even then they consists of sections similar to our example. Translating a guideline into a clear and structured fragment such as in Figure 2 can take a lot of effort; however, the formalisation of a guideline is not the main focus of the work presented, which is about verification of a formalised guideline.

- Step 1: diet.
- Step 2: if Quetelet index (QI) ≤ 27, prescribe a sulfonylurea drug; otherwise, prescribe a biguanide drug.
- Step 3: combine a sulfonylurea drug and biguanide (replace one of these by a α-glucosidase inhibitor if side-effects occur).
- Step 4: one of the following:
  - oral antidiabetic and insulin
  - only insulin

Figure 2. Tiny fragment of a clinical guideline on the management of diabetes mellitus type 2. If one of the steps $k = 1, 2, 3$ is ineffective, the management moves to step $k + 1$

The diabetes mellitus type 2 guideline provides practitioners with a clear structure of recommended actions to be taken for the control of the glucose level. This kind of information is typically found in medical guidelines in the sense that medical knowledge is combined with information about order and time of treatment (e.g., sulfonylurea in step 2), about patients and their environment (e.g., Quetelet index lower than or equal to 27), and finally which drugs are to be administered to the patient (e.g., a sulfonylurea drug). When verifying the quality of a guideline, the formal language used should at least address these elements. We come back to these elements in more detail in Section 5.1.

4 Medical guidelines in Asbru

Much research has already been devoted to the development of representation languages for medical guidelines. Most of them look at

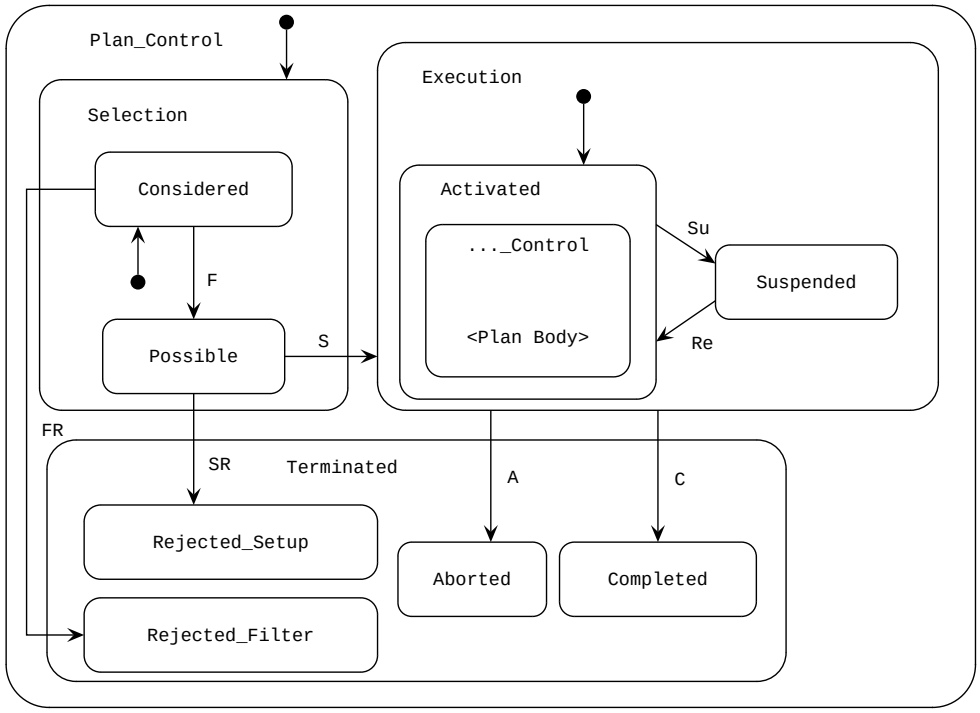

Figure 3. The plan state model, where Satisfied(*cond*) denotes that the environment satisfied the condition *cond* whereas Satisfiable(*cond*) denotes that, theoretically, the environment could still satisfy the condition *cond*, i.e., that no deadline has passed in case of time constraints.

guidelines consisting of a composition of actions, whose execution is controlled by conditions [27]. However, most of them are not formal enough for the purpose of our research as they often incorporate free-text elements which do not have a clear semantics. Exceptions to this are PRO*forma* [16, 17] and Asbru [37, 40]. The latter has been chosen in our research as a basis to formalise a medical guideline.

4.1 Introduction to Asbru

A medical guideline is considered in Asbru as a hierarchical *plan*. The main components of an Asbru plan are intentions, conditions, plan-body, and time annotations. Furthermore, a plan can have arguments and can alter the value of variables.

The *intentions* are the high-level goals of a plan. Intentions can be expressed in terms of achieving, maintaining, or avoiding a certain state or action. The states or actions to which intentions refer can be intermediate or final (overall). In total there are twelve possible forms of intentions built up by combining elements from the sets {achieve, maintain, avoid}, {intermediate, overall}, and {state, action}.

Conditions can be associated to a plan to define different aspects of its execution. The most important types of condition are: (1) filter and setup conditions,[4] which must be true before a plan can start, (2) abort conditions, which define when a plan must abort, and (3) complete conditions, which define when a started plan finishes successfully. Conditions can be 'over-ridable' (i.e., health personnel can manually satisfy the condition) or 'require confirmation' (i.e., conditions must be explicitly confirmed before they are satisfied).

The *plan-body* contains the actions, sub-plans, or both to be executed as part of the plan. The main types of plan-body are: (1) user-performed: an action has to be performed by a user, which requires interaction, which is not further modelled, (2) single-step: an action which can be either an activation of a sub-plan, an assignment of a variable, a request for an input value, or an if-then-else statement, (3) sub-plans: a set of plans to be performed in a given order, either sequentially, in parallel, in any-order, or unordered, and (4) cyclical plans: a repetition of actions over a time period. In case of sub-plans, it is also required to specify a waiting strategy to describe which of the sub-plans must be completed for the super plan to complete, e.g., all sub-plans should be executed (**wait-for all**).

Time annotations can be associated to various Asbru elements, e.g., intentions, conditions, plan activations. A time annotation specifies (1) in which interval things must start, (2) in which interval things must end, (3) their minimal and maximal duration, and (4) a reference time point.

4.2 The semantics of Asbru

To help in the understanding of Asbru we review here the semantics of Asbru in a semi-formal statechart notation [5]. In Asbru, plans are organised in a hierarchy, where a plan may include a number of sub-plans. The semantics of Asbru is defined in [3] by flattening the hierarchy of plans and using one top level control to execute all plans synchronously. Within each top level step, a step of every plan is executed. Whether a plan is able to progress depends on its conditions.

[4] filter conditions are conditions about values that cannot change value, e.g., *sex* = *male*, whereas setup conditions are conditions about values that may change, e.g., glucose level.

The plan state model shown in Figure 3 defines the semantics of the main plan hierarchy. The 'Plan_Control' is divided into a selection phase, an execution phase, and a termination phase. Each plan goes into the 'Considered' state when it receives a *consider* signal. In this state its *filter condition* is checked. If it evaluates to true, control advances to the state 'Possible'. Then the setup condition is checked and if it is passed, control advances to the execution phase. If the filter condition is not satisfied or the setup condition is not satisfiable anymore (i.e., it is not possible to satisfy the condition in the future, because a deadline has passed), the plan is rejected. The same happens, if the super-plan terminates. In the execution phase the plan waits for an external signal *activate*, to be sent by its super-plan.

In state 'Activated', the sub-plans are executed, which can be sequentially, in parallel, unordered, or in any order, and each order determines a different controlling statechart [3]. A plan can synchronise its sub-plans using the signals *consider* and *activate*. Additional control to propagate execution states of a sub-plan to its parent and vice versa is also present, e.g., the abortion of a mandatory sub-plan enforces the parent-plan also to abort. Sub-plans can either be completed successfully or aborted, e.g., in the case of emergency patient readings.

The complete technical definitions, in addition to the semantics of the other constructs that are not shown here, can be found in [5].

5 Verification of medical guidelines

5.1 Requirements for the verification of guidelines

To be able to verify quality criteria of medical guidelines using formal methods, we need to have a language that can be used to express quality criteria that can be related to the key elements in a guideline. In Section 3, we stated that the key elements in medical guidelines are (at least) order in time, patients, and interventions. Here, we discuss our choices for a language for the formal representation of those key elements, used in the remainder of the paper.

Time: As medical management is a time-oriented process, diagnostic and treatment actions described in guidelines are performed in a temporal setting. It has been shown previously that the step-wise, possibly iterative, execution of a guideline can be described by means of temporal logic [25]. This is a modal logic [13], where relationships between worlds in the usual possible-world semantics of modal logic is understood as time order. In this paper, we will use a variant of this logic, based on future-time linear temporal logic. The language of this logic is first-order logic augmented with the temporal operators listed in Table 1. The semantics of this language is given by a set D, representing the universe of discourse, a set of interpretations I_t for interpreting statements from the first-order logic, and a function *succ*, where $succ(t)$ is the set of zero or one successors of time points of t. First-order expressions φ at time t are interpreted using I_t in the domain D; for example, $t \vDash \varphi$ means that φ is satisfied at time t w.r.t. I_t and D [13].

Note that the **last** modality can only hold in models where at some point following the successor function, no successor exists. In all other models, **last** will never hold. Also note that some operators can be defined in terms of other operators, e.g., $\Box\,\varphi \equiv \neg \Diamond \neg \varphi$ and **last** $\equiv \bullet \bot$. A more expressive logic can be gained by including, for example, the **until** operator, where φ **until** ψ denotes that eventually ψ holds and before that φ holds. However, as such operators are not used in this paper, they have been omitted.

This logic allows one to look at guidelines formally at a particular abstraction level. In Section 8, we show this logic to be suitable for quality checking of medical guidelines; however, it is possible to add more fine-grained temporal operators if they are needed.

Patient groups: Although in practice a guideline is used for the management of a particular patient, recommendations in guidelines are always written with a certain *patient group* in mind – not just a single patient. Patient groups are groups of patients that share common characteristics about their current state or previous states. One can abstract from the actual situation of a patient by providing a logical language that refers to one or more situations, including the necessary common characteristics, without fixing all the details. Typical elements for describing the state of patients are symptoms, signs, and test outcomes. Here we have chosen to use predicate logic with equality and unique names assumption [32]. For example, the literal 'Condition(*hyperglycaemia*)' is used to represent the patient group of all patients that currently have the condition of hyperglycaemia. Subgroups of patient groups can be specified by using a conjunction with additional literals, e.g., 'Condition(*hyperglycaemia*) $\wedge$ QI $\leq$ 27' specifies the patient group of patients who have hyperglycaemia and also have a Quetelet index less than or equal to 27. We sometimes represent the conjunction also in set form, e.g., the latter conjunction becomes '$\{\text{Condition}(\textit{hyperglycaemia}), \text{QI} \leq 27\}$'.

Interventions and treatments: An intervention is the act of intervening, interfering, or interceding with the intent of modifying the outcome. In medicine, interventions include all medical actions that influence the state of a patient or his environment. A treatment is usually restricted to methods that provide a cure for an illness or disability, however, the terms intervention and treatment are often used synonymously. We have chosen to represent the domain of interventions by a countable set. Subsets of this set are interpreted as *treatments* in which each intervention of the set is applied. Interventions which are not an element of the treatment are assumed not to be applied. We abstract from medical management details such as changing drug dosages.

5.2 Verification approach

Medical guidelines give recommendations based on the best available evidence. Although diabetes mellitus type 2 is a complicated disease, the guideline fragment shown in Figure 2 is not. This indicates that much knowledge concerning diabetes mellitus type 2 is missing from the guideline. Verifying whether a guideline fulfils some property therefore additionally needs the specification of *background knowledge*.

The ideas that we use here to verify quality requirements for medical guidelines are inspired by previous work, where a distinction was made between the different types of knowledge that are involved in defining quality requirements [21]. We assume that there are at least three types of knowledge involved in detecting the violation of good medical practice:

1. Knowledge concerning the (patho)physiological mechanisms underlying the disease, and the way treatment influences these mechanisms. The knowledge involved could be for example causal or empirical in nature, and is an example of *object-knowledge*.
2. Knowledge concerning the recommended treatment in every step of the guideline and how the choice for each treatment is affected by the state of the patient, i.e., the order information from the medical guideline. This is also an example of *object-knowledge*.

Table 1. Used temporal operators; t stands for a time instance

Notation	Interpretation	Formal semantics
$\Box\,\varphi$	φ will always be true	$t \vDash \Box\,\varphi \Leftrightarrow \forall t' \geq t : t' \vDash \varphi$
$\Diamond\,\varphi$	φ will eventually be true	$t \vDash \Diamond\,\varphi \Leftrightarrow \exists t' \geq t : t' \vDash \varphi$
$\circ\,\varphi$	execution does not terminate and the next state satisfies φ	$t \vDash \circ\,\varphi \Leftrightarrow \exists\, t' \in \mathit{succ}(t) : t' \vDash \varphi$
$\bullet\,\varphi$	either execution terminates or the next state satisfies φ	$t \vDash \bullet\,\varphi \Leftrightarrow \forall\, t' \in \mathit{succ}(t) : t' \vDash \varphi$
last	the current state is the last	$t \vDash \textbf{last} \Leftrightarrow \mathit{succ}(t) = \varnothing$

3. Knowledge concerning good practice in treatment selection; this is *meta-knowledge*.

The first type of object-knowledge will be called *background knowledge*. The second type of object-knowledge is the order information from the medical guideline, which can be considered a network of tasks or a hierarchical plan. The plan prescribes treatment which influences the (patho)physiological mechanisms, which results in information about patient groups that can be used by the plan to make the best possible decision in subsequent step of the protocol. Incompleteness of background knowledge may lead to insufficient knowledge about a patient, which may result in a plan making a non-deterministic choice. Of course, the guideline should recommend the collection of data when possible if this data is crucial for decision making.

The third type of knowledge, the meta-knowledge, includes general knowledge about good practice medicine, for example, preferring a treatment over another if it uses a smaller number of drugs and has an equal effect on the patient. This knowledge will be formalised by *quality requirements*, i.e., (reasoning) patterns that specify the behaviour of treatment selection given certain patient data. These quality requirements can be used as proof obligations in the verification of medical guidelines.

In the following section, the three types of knowledge involved (background knowledge, medical guideline, and quality requirements) are described in more detail in the context of diabetes mellitus type 2 and a formalisation in terms of temporal logic as discussed in Section 5.1 is given. In Section 8 the quality requirements are verified with the interactive theorem prover KIV.

6 Formalisation diabetes mellitus type 2 guideline

6.1 Background knowledge

In diabetes mellitus type 2 various metabolic control mechanisms are deranged and many different organ systems may be affected. Glucose level control, however, is the most important mechanism. At some stage in the natural history of diabetes mellitus type 2, the level of glucose in the blood is too high (hyperglycaemia) due to decreased production of insulin by the B cells. Oral anti-diabetics either stimulate the B cells in producing more insulin (sulfonylurea) or inhibit the release of glucose from the liver (biguanide). Effectiveness of these oral diabetics is dependent on the condition of the B cells. Finally, as a causal treatment, insulin can be prescribed. The mechanisms have been formalised in terms of temporal logic in previous work [21], and is shown in Figure 4.

For example, axiom (1) denotes the physiological effects of insulin treatment, i.e., administering insulin results in an increased uptake of glucose by the liver and peripheral tissues. Axiom (8) phrases under what conditions you may expect the patient to get cured, i.e., when the patient suffers from hyperglycaemia and insulin production of his

(1) $\mathrm{Drug}(\mathit{insulin}) \rightarrow \circ\,(\mathit{uptake}(\mathit{liver}, \mathit{glucose}) = \mathit{up} \wedge \mathit{uptake}(\mathit{peripheral\text{-}tissues}, \mathit{glucose}) = \mathit{up})$

(2) $\mathit{uptake}(\mathit{liver}, \mathit{glucose}) = \mathit{up} \rightarrow \mathit{release}(\mathit{liver}, \mathit{glucose}) = \mathit{down}$

(3) $(\mathrm{Drug}(\mathrm{SU}) \wedge \neg \mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{exhausted}) \rightarrow \circ\, \mathit{secretion}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{up}$

(4) $\mathrm{Drug}(\mathrm{BG}) \rightarrow \circ\, \mathit{release}(\mathit{liver}, \mathit{glucose}) = \mathit{down}$

(5) $(\circ\, \mathit{secretion}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{up} \wedge \mathrm{Condition}(\mathit{hyperglycaemia}) \wedge \mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{subnormal} \wedge \mathrm{QI} \leq 27) \rightarrow \circ\, \mathrm{Condition}(\mathit{normoglycaemia})$

(6) $(\circ\, \mathit{release}(\mathit{liver}, \mathit{glucose}) = \mathit{down} \wedge \mathrm{QI} > 27 \wedge \mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{subnormal} \wedge \mathrm{Condition}(\mathit{hyperglycaemia})) \rightarrow \circ\, \mathrm{Condition}(\mathit{normoglycaemia})$

(7) $((\circ\, \mathit{release}(\mathit{liver}, \mathit{glucose}) = \mathit{down} \vee \circ\, \mathit{uptake}(\mathit{peripheral\text{-}tissues}, \mathit{glucose}) = \mathit{up}) \wedge \mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{nearly\text{-}exhausted} \wedge \circ\, \mathit{secretion}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{up} \wedge \mathrm{Condition}(\mathit{hyperglycaemia})) \rightarrow \circ\, \mathrm{Condition}(\mathit{normoglycaemia})$

(8) $(\circ\, \mathit{uptake}(\mathit{liver}, \mathit{glucose}) = \mathit{up} \wedge \circ\, \mathit{uptake}(\mathit{peripheral\text{-}tissues}, \mathit{glucose}) = \mathit{up} \wedge \mathit{capacity}(\mathit{b\text{-}cells}, \mathit{insulin}) = \mathit{exhausted} \wedge \mathrm{Condition}(\mathit{hyperglycaemia})) \rightarrow \circ\, (\mathrm{Condition}(\mathit{normoglycaemia}) \vee \mathrm{Condition}(\mathit{hypoglycaemia}))$

(9) $(\mathrm{Condition}(\mathit{normoglycaemia}) \oplus \mathrm{Condition}(\mathit{hypoglycaemia}) \oplus \mathrm{Condition}(\mathit{hyperglycaemia})) \wedge \neg(\mathrm{Condition}(\mathit{normoglycaemia}) \wedge \mathrm{Condition}(\mathit{hypoglycaemia}) \wedge \mathrm{Condition}(\mathit{hyperglycaemia}))$

Figure 4. Background knowledge $\mathcal{B}_{\mathrm{DM2}}$ of diabetes mellitus type 2. $\mathrm{Drug}(x)$ holds iff drug x is being administered at that moment in time. The $\oplus$ operator denotes the exclusive OR operator.

B cells are exhausted, an increased uptake of glucose by the liver and peripheral tissues results in the patient condition changing to normoglycaemia.

6.2 Asbru model

In Asbru, plans are hierarchically organised in which a plan refers to a number of sub-plans. The overall structure of the Asbru model of our running example (Figure 2), is shown in Figure 5. The top level plan 'Treatments_and_Control' sequentially executes the four subplans 'Diet', 'SU_or_BG', 'SU_and_BG', and 'Insulin_Treatments', which correspond to the four steps of the guideline fragment in Figure 2. The sub-plan 'Insulin_Treatments' is further refined by two sub-plans 'Insulin_and_Antidiabetics' and 'Insulin', which can be executed in any order.

The Asbru specifications of two plans in the hierarchy, namely 'SU_or_BG' and 'Insulin_Treatments' are defined as in Figure 6.

In the case of 'SU_or_BG' there is a relationship between the

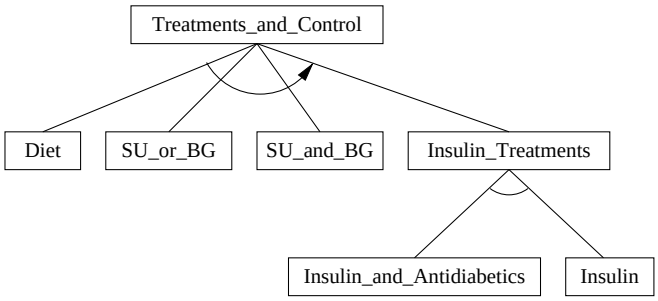

Figure 5. Asbru plan hierarchy of the diabetes mellitus type 2 guideline.

plan 'SU_or_BG'
  effects
    (QI $\leq$ 27 $\rightarrow$ SU $\in$ Drugs) $\wedge$
    (QI $>$ 27 $\rightarrow$ BG $\in$ Drugs)
  abort condition
    'condition = hyperglycaemia **confirmation required**'
  complete condition
    condition = hypoglycaemia $\vee$
    condition = normoglycaemia

plan 'Insulin_Treatments'
  body anyorder wait for one
    'Insulin_and_Antidiabetics'
    'Insulin'

Figure 6. Asbru specifications of two treatments recommended in the diabetes mellitus type 2 guideline.

Quetelet index (QI) and the drug administered. If the Quetelet index is less or equal than 27 then SU is administered, else BG is administered. The plan 'SU_or_BG' corresponds to step 2 in the guideline fragment of Figure 2, which completes if the patient condition improves, i.e., the patient no longer has hyperglycaemia. This is represented by the **complete condition**. The plan 'SU_or_BG' aborts when the condition of the patient does not improve, which is represented by the **abort condition**. It requires a manual confirmation to ensure that some time passes for the drugs to have an impact on the patient condition.

The plan 'Insulin_Treatments' consists of two sub-plans, which correspond to the two options of step 4 in the guideline fragment of Figure 2, i.e., either insulin is administered or insulin and antidiabetics are administered.

6.3 Quality requirements

Here, we give a formalisation of good practice medicine of medical guidelines. This extends previous work [21], which formalised good practice medicine on the basis of a theory of abductive reasoning of single treatments. The context of the formalisation given here is a fully formalised guideline, which consists, besides a number of treatments, of a control structure that uses patient information to decide on a particular treatment. This contrast with [21], which used a context of a singly chosen treatment.

Firstly, we formalise the notion of a *proper* guideline according to the theory of abductive reasoning. Let $\mathcal{B}$ be medical background knowledge, P be a patient group, N be a collection of intentions, which the physician has to achieve, and M be a medical guideline. Then M is called a *proper* guideline for a patient group P, denoted as $M \in Pr_P$, if:

(M1) $\mathcal{B} \cup M \cup P \not\models \bot$ (the guideline does not have contradictory effects), and

(M2) $\mathcal{B} \cup M \cup P \models \Diamond N$ (the guideline eventually handles all the patient problems intended to be managed)

Secondly, we formalise good practice medicine of guidelines. Let $\preceq_\varphi$ be a reflexive and transitive order denoting a preference relation with $M \preceq_\varphi M'$ meaning that M' is *at least as preferred* to M given criterion φ. With $\prec_\varphi$ we denote the order such that $M \prec_\varphi M'$ if and only if $M \preceq_\varphi M'$ and $M' \not\preceq_\varphi M$. When both $M \preceq_\varphi M'$ and $M' \preceq_\varphi M$ hold or when M and M' are incomparable w.r.t. $\preceq_\varphi$ we say that M and M' are *indifferent*, which is denoted as $M \sim M'$. If in addition to (M1) and (M2) condition (M3) holds, with

(M3) $O_\varphi(M)$ holds, where O_φ is a meta-predicate standing for an optimality criterion or combination of optimality criteria φ defined as: $O_\varphi(M) \equiv \forall M' \in \mathsf{Pr}_P : \neg(M \prec_\varphi M')$,

then the guideline is said to be *in accordance with good practice medicine* w.r.t. criterion φ and patient group P, which is denoted as $\mathsf{Good}_\varphi(M, P)$.

A typical example for O_φ is consistency of the recommended treatment order w.r.t. a preference relation $\preceq_\psi$ *over treatments*, i.e., $O_\varphi(M)$ holds if the guideline M recommends treatment T before treatment T' when $T' \prec_\psi T$ holds. For example, in diabetes mellitus type 2, a preference relation over treatments would be to minimise (1) the number of insulin injections, and (2) the number of drugs involved. This results, among others, in the following preferences: sulfonylurea drug $\sim$ biguanide drug, and insulin $\preceq_\psi$ insulin and antidiabetic $\preceq_\psi$ sulfonylurea and biguanide drug $\preceq_\psi$ sulfonylurea or biguanide drug $\preceq_\psi$ diet. A guideline M would then be in accordance with good practice medicine if it is consistent with this preference order $\preceq_\psi$, e.g., if M first recommends diet before a sulfonylurea or biguanide drug.

7 Specification in KIV

Previous sections have given the temporal logic formalisation of the background knowledge of diabetes mellitus type 2, the quality requirements, and the Asbru model of the medical guideline for diabetes mellitus type 2. In this section we discuss how these elements can be translated into KIV representations, so that they become amendable to verification.

7.1 Introduction to KIV

KIV is an integrated development environment to develop systems using formal methods [6]. The specification language of KIV is based on higher-order algebraic specifications. Reactive systems can be described in KIV by means of state-charts or parallel programs; here we use parallel programs. Parallel programs are modelled as follows. Let e denote an arbitrary (first-order) expression and v_d a dynamic variable (see below), then constructs for parallel programs include: $v_d := e$ (*assignments*), **if** ψ **then** ϕ_1 **else** ϕ_2 (*conditionals*), **while** ψ **do** ϕ (*loops*), **var** $v_d = e$ **in** ϕ (*local variables*), **patom** ϕ **end** (*atomic execution*), $\phi_1 \parallel \phi_2$ (*interleaved execution*), and $[p\#(e; v_d)]$ (*call to procedure* p *with value parameters* e *and reference parameters* v_d). The semantics of this extended language is defined in [1].

The correctness of systems is ensured by constructing proofs in an interactive theorem prover which is based on higher order logic

with special support for temporal logic, i.e., future-time linear temporal logic [4]. The logic of Table 1 is extended with static variables v_s, which are variables that are mapped to the same element in the universe of discourse at each time point. Dynamic variables v_d, such as program variables, may have different interpretations at different time points. In the upcoming sections, the use of static variables will be explicitly mentioned. A speciality of KIV is the use of primed and double-primed variables: a primed variable v_d' represents the value of this variable after a system transition, the double-primed variable v_d'' is interpreted as the value after an environment transition. System and environment transitions alternate, with v_d'' being equal to v_d in the successive state (cf. Figure 7 and Section 8.1).

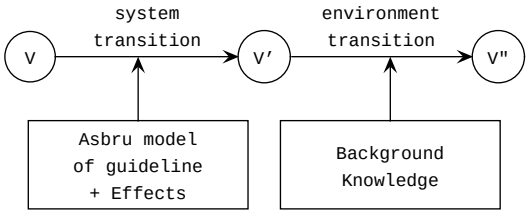

Figure 7. The relation between unprimed and primed variables as two distinct transitions: the system transition (including the Asbru model and its effects) and the environment transition (including the background knowledge).

7.2 Specification methodology in KIV

The guideline and patient can be looked upon as a system (guideline) that interacts with the environment (patient). KIV allows a clear distinction between system and environment transitions by using primed and double-primed variables. Therefore, the Asbru model is only allowed to map variables into primed variables, whereas the environment is only allowed to map primed variables into double primed variables. System and environment transitions alternate (Figure 7).

However, system transitions in Asbru may involve a large number of steps (e.g., signals, plan state changes) before the model reaches a stable state from which no further step can be made unless time progresses or the environment changes. Asbru is mainly a control oriented language and many control steps are not considered to take any real time at all. In an interactive theorem prover like KIV, this behaviour can be modelled by the introduction of two transition types, *micro-steps* and *macro-steps* [36]. Micro-steps are technical Asbru steps where time and environment are not allowed to change. Macro-steps are temporal steps in which interaction can occur with the environment (e.g., plan activations) and are only executed when there are no micro-steps possible. The variable 'Tick', controlled by the symbolic execution of the Asbru semantics, holds when a macro-step occurs.

In KIV, system descriptions are represented by means of a set of algebraic specifications. These algebraic specifications can be enriched with additional algebraic structures, which form a dependency structure between the different specifications. To maximise re-usability, several layers are used for representing our framework in KIV. The lowest layer in this dependency structure consists of standard data structures like Booleans and sets, which are typically obtained from libraries in KIV. On top of that, all data structures are represented necessary for representing the semantics of Asbru. The remaining layers consist of the structures dependent on the specific guideline under study. On top of the standard data structures, additional data structures are represented. For the diabetes case study, the data types

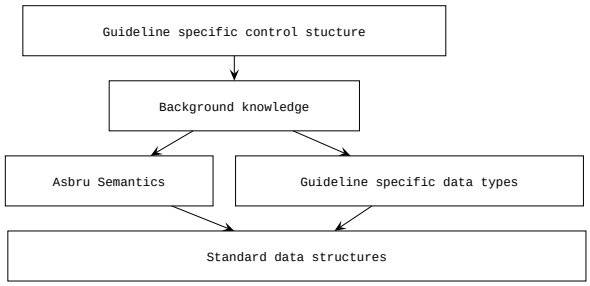

Figure 8. Dependency structure of Asbru specifications with $A \rightarrow B$ denoting that A depends on B

are modelled as enumeration types. On top of the asbru semantics and data structures the background knowledge is represented. The top layer consists of the control structure of the guideline, which is the structure of Figure 5 in the diabetes case study (cf. Figure 8).

7.3 Specification of background knowledge in KIV

The background knowledge is translated into algebraic specifications in KIV. All background knowledge axioms have been reformulated in terms of preconditions and postconditions. Every element that refers to the current point in time is interpreted as a precondition and each element that refers to the next point in time is interpreted as a postcondition. The values of these elements are stored in a data structure, denoted by 'Patient'. The patient is modelled by a sequence of pairs $[v, c]$, where v is the name of a variable and c a constant denoting the value of that variable, depending on the point in time. Updates to the patient record are done by appending a pair to the end of the sequence. Moreover, the most recent value of a variable v in a sequence s is given by the term $s[v]$. An example of the final translation can be found in Figure 9.

predicates
Knowledge : *patient* × *patient*;
axioms
BDM2-1:
Knowledge(*pre*, *post*) → (*insulin* ∈ *pre*[*treatment*] →
post[*uptake*(*liver*,*glucose*)] = *up* ∧
post[*uptake*(*peripheral-tissues*, *glucose*)] = *up*)
BDM2-8:
Knowledge(*pre*, *post*) → (*post*[*uptake*(*liver*,*glucose*)] = *up*
∧ *post*[*uptake*(*peripheral-tissues*,*glucose*)] = *up*)
∧ *pre*[*capacity*(*b-cells*,*insulin*)] = *exhausted*
∧ *pre*[*condition*] = *hyperglycaemia* →
post[*condition*] = *normoglycaemia*)

Figure 9. Background knowledge in KIV as a first order predicate using pre- and postconditions, i.e., *pre* and *post* are shorthand notations for patient data structures with *pre*[v] = c and *post*[v] = c referring to the condition $v = c$ of the patient in the current and next state respectively. The use of pre and $post$ variables is necessary to parameterise the background knowledge for arbitrary patient data structures. In addition, two translated rules from the background formalisation in [21] are shown with BDM2-i representing Axiom (i) (cf. Figure 4).

7.4 Specification of Asbru in KIV

As each Asbru plan has a strict format, an algebraic function 'mk-asbru-def' has been defined for the translation of Asbru plans into

KIV specifications. By calling 'mk-asbru-def' with the parameters that constitute a plan, translation of any guideline in Asbru becomes straightforward. The parameters consist of the various conditions that control plan state changes, the control type of sub-plans, a list of sub-plans, a retry value (for aborted plans), a wait-for condition (for mandatory sub-plans), and an optional wait-for flag (whether to wait for sub-plans). As there are quite a number of parameters, default values are provided to ease specification.

The Asbru semantics is implemented as a parallel program, parametrised with a given Asbru model. Temporal properties of this program are proven using symbolic execution and induction [1].

7.5 Specification of quality requirements in KIV

With the help of KIV, we have verified that the diabetes guideline is proper, i.e., that the guideline satisfies conditions (M1) and (M2) as defined in Section 6.3, which is discussed in detail in Subsections 8.1 and 8.2. Meta-level quality requirements are verified in KIV using a sequent $\Gamma \vdash \Sigma$ where the succedent Σ is some instantiation of (M3) and the antecedent Γ is a fixed structure that consists of the initial state of the patient and the Asbru model, the Asbru model, the effects of treatments, the background knowledge, and the environment assumptions. The sequent in Figure 10 is an example specification in KIV of the quality requirement that each patient is eventually cured from hyperglycaemia.

$$
\begin{array}{l}
\textit{/* Initial state of patient */} \\
\text{Patient}[\textit{condition}] = \textit{hyperglycaemia}, \\
\textit{/* Initial state of guideline */} \\
\text{AS}[\text{Treatments\_and\_Control}] = \textit{inactive}, \ldots, \\
\textit{/* Asbru model */} \\
[\textit{asbru}\#(\text{Treatments\_and\_Control}; \text{AS}, P)], \\
\textit{/* Effects */} \\
\Box\ (\text{AS}[\text{SU\_or\_BG}] = \textit{activated} \leftrightarrow \\
\quad \text{BG} \in \text{Patient}'[\textit{treatment}] \wedge \ldots), \\
\textit{/* Background knowledge */} \\
\Box\ \text{Knowledge}(\text{Patient}', \text{Patient}'') \\
\textit{/* Environment assumption */} \\
\Box\ (\text{AS}''[\text{Treatments\_and\_Control}] = \\
\quad \text{AS}'[\text{Treatments\_and\_Control}] \wedge \ldots) \\
\vdash \\
\textit{/* Property */} \\
\Diamond\ (\text{Patient}[\textit{condition}] = \textit{hypoglycaemia} \vee \\
\quad \text{Patient}[\textit{condition}] = \textit{normoglycaemia})
\end{array}
$$

Figure 10. Specification in KIV of the quality requirement that each patient is eventually cured from hyperglycaemia.

The *initial state* of the *patient* and the *Asbru model* are represented using additional data structures [35]. The patient data is represented in a data structure 'patient-data-history', which in Figure 10 is set to the patient group $\{\text{Condition}(\textit{hyperglycaemia})\}$. The initial state of the Asbru model is represented using a data structure 'AS' of type 'asbru-state', which keeps track of all plan states over time, and in which initially each plan is set to inactive. The *Asbru model* of the guideline describes the control structure, and its specification in KIV has already been discussed in Section 7.4. The *effects of treatments* specify in KIV the behaviour of plans in the Asbru model. This is a direct translation of the **effects** attribute used in the Asbru model, which specifies the expected behaviour of plans (cf. Section 6.2). In our diabetes case study the effects of plans are the administration of a certain drug as soon as the plan becomes activated, which may depend on the value of other variables like the Quetelet index (cf. Section 6.2). The *background knowledge* is represented in the sequent using the first-order predicate 'Knowledge' and has already been discussed in Section 7.3. The environment is in principle allowed to change every variable arbitrarily. The *environment assumptions* restrict the behaviour of the environment. These restrictions (1) forbid the environment to change some variable, (2) force the environment to deterministically change a variable (e.g., advancing a clock), and (3) guarantee certain variable assignments in a nondeterministic way (e.g., the existence of a value when a signal is sent).

8 Verification using KIV

8.1 Consistency of background knowledge

Property (M1) ensures that the formal model including the Asbru guideline and the background knowledge is consistent. The initial state is – in our case – described as a set of equations and it has been trivial to see that they are consistent. The guideline is given as an Asbru plan. The semantics of any Asbru plan is defined in a programming language where every program construct ensures that the resulting reactive system is consistent: in every step, the program either terminates or calculates a consistent output for arbitrary input values. The Asbru plan, thus, defines a total function between unprimed and primed variables in every step (Figure 7). The formula defining the effects maps the output variables of the guideline to input variables of the patient model. Again, it has been trivial to see that this mapping is consistent.

The background knowledge defines our patient model. We consider the patient to be part of the environment which is the relation between the primed and the double primed variables in every step. If the patient model ensures that for an arbitrary primed state there exists a double primed state, the overall system of alternating guideline and environment transitions is consistent: given an initial (unprimed) state, the guideline calculates an output (primed) state; the effects define a link between the variables of the guideline and the variables of the patient model; the patient model reacts to the (primed) output state and gives a (double primed) state which is again input to the Asbru guideline in the next step. In other words, the relation between the unprimed and the double primed state is the complete state transition. The additional environment assumptions referring to the Asbru environment do not destroy consistency as the set of restricted variables of the environment assumption is disjunct to the set of variables of the patient model.

It remains to ensure consistency of the background knowledge which we defined as a predicate 'knowledge'. Consistency can be shown by proving the property

$$\forall pre.\ \exists post.\ \text{'knowledge'}(pre, post)$$

which ensures that the relation is total. In order to prove that this property holds an example patient has been constructed. Verifying that the example patient is a model of the background knowledge has been fully automatic.

8.2 Successful treatment

In order to verify property (M2), i.e., the guideline eventually manages to control the glucose level in the patient's blood, a proof has been constructed. The verification strategy in KIV is symbolic execution with induction [1]. The plan state model introduced in [3]

defines the semantics of the different conditions of a plan and is implemented in KIV by a procedure called 'asbru', which is symbolically executed. Each plan can be in a certain state, modelled with a variable 'AS' (i.e., 'inactive', 'considered', 'ready', 'activated', and 'aborted' (or 'completed')) and a transition to another state depends on its conditions. In the initial state, the top level plan 'Treatments_and_Control' (abbreviated 'tc') is in 'inactive' state. After executing the first step, the plan is 'considered', after which execution continues as described in [3]. The execution is visualised in a proof tree (cf. Figure 11), where the bottom node is the start of the execution and splits if there is a case distinction.

Patients whose capacity of the B cells is 'normal' are cured with diet, while for other patients diet will not be sufficient. In this case, we assume that the doctor eventually aborts the diet treatment. We use induction to reason about the unspecified time period in which diet is applied. As an invariant,

$$Patient[\text{'capacity(B-cells,insulin)'}] \neq normal$$

is used. In the next step, the doctor has either aborted 'diet' or 'diet' is still active. In the second case, induction can be applied. When 'diet' is aborted, 'tc' sequentially executes the next plan, which is 'SU_or_BG' (cf. Figure 5).

The second treatment 'SU_or_BG' goes, as each Asbru plan, through a sequence of states, i.e., 'inactive', 'considered', 'ready', 'activated', and 'aborted', and thus becomes first 'considered' and after some steps becomes 'activated' (cf. Figure 11). In this case, either SU or BG is prescribed, depending on the Quetelet index QI. For a patient whose B cell capacity is 'subnormal', the background knowledge ensures that the condition of the patient improves. Thus, for the rest of the proof we can additionally assume that

$$Patient[\text{'capacity(B-cells,insulin)'}] \neq subnormal$$

After 'SU_or_BG' aborts, the third treatment ('SU_and_BG') is executed in similar fashion, where patients with nearly exhausted B cell capacity are cured. Thus, after aborting the first three treatments the precondition concerning the B cell capacity can be strengthened to

$$\begin{aligned} & Patient[\text{'capacity(B-cells,insulin)'}] \neq \text{'}normal\text{'} \\ \wedge\ & Patient[\text{'capacity(B-cells,insulin)'}] \neq \text{'}subnormal\text{'} \\ \wedge\ & Patient[\text{'capacity(B-cells,insulin)'}] \neq \text{'}nearly\text{-}exhausted\text{'} \end{aligned}$$

which, under the assumption that the only possible values of the capacity are normal, subnormal, nearly-exhausted, and exhausted, yields:

$$Patient[\text{'capacity(B-cells,insulin)'}] = exhausted$$

This statement together with the background knowledge ensures that the prescription of insulin, which is prescribed in both final treatments 'Insulin' and 'Insulin_and_Antidiabetics', finally cures the patient.

8.3 Optimality of treatment

With respect to property (M3), an optimality criterion of the guideline is that no treatments are prescribed that are not in accordance with good practice medicine (Section 6.3), i.e., some preference relation $\preceq$ between treatments exists and the guideline never prescribes a treatment T such that $T \preceq T'$ and T' cures the patient group under consideration.

In our case study the preference for treatments is based on the minimisation of (1) the number of insulin injections, and (2) the number

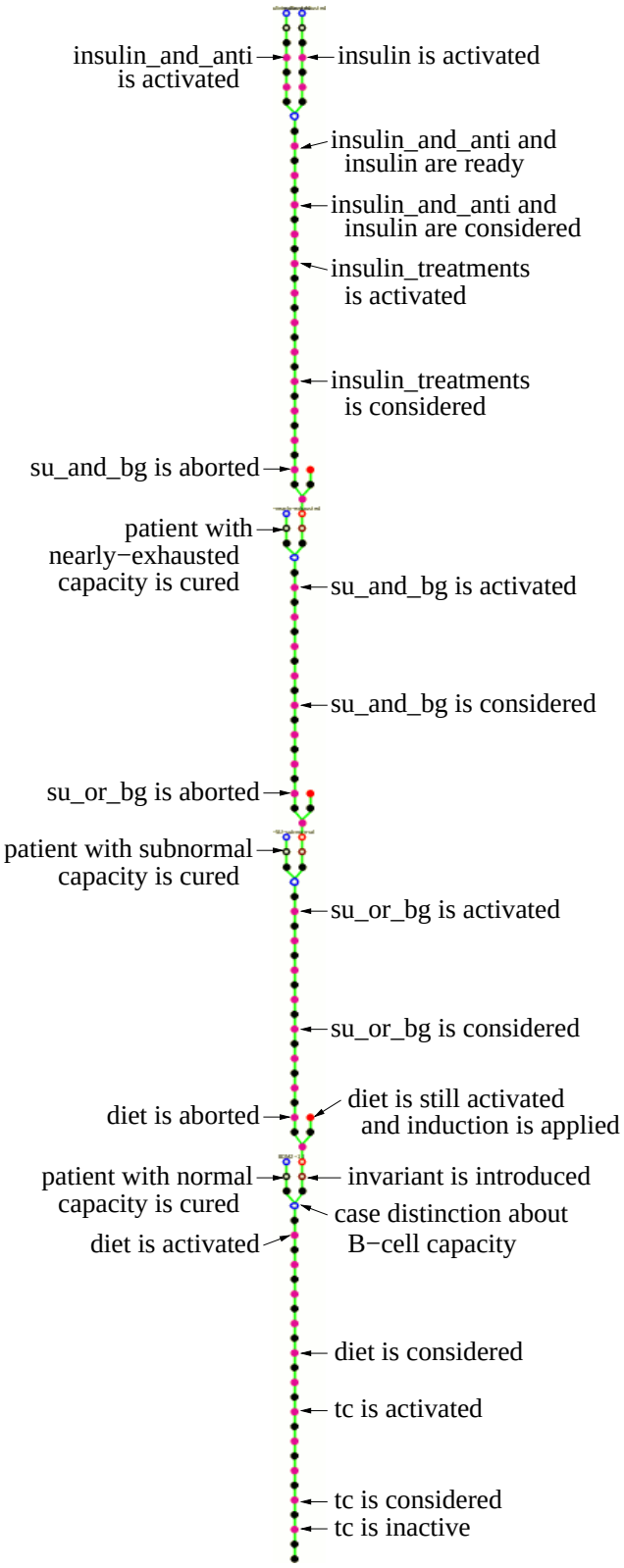

Figure 11. Overview of the proof that the guideline eventually manages all patient problems, which is explained in Section 8.2.

of drugs involved (cf. Section 6.3). We have defined this using a reflexive, transitive order $\leq$ such that for all treatments T, it holds that $insulin \leq T$ and $T \leq diet$. Furthermore, the treatments prescribing the oral anti-diabetics sulfonylurea and biguanide are incomparable. The proof obligation is then as follows:

$$\Box(\forall_T\colon Good_{\leq}(T, Patient) \rightarrow T \leq Patient[\text{'treatment'}])$$

where $Good_{\leq}(T, Patient)$ denotes that T is a treatment according to good practice medicine for $Patient$, as defined in [24]. To prove this, the following axiom was added to the system:

$$\Box Patient[\text{'QI'}] = Patient''[\text{'QI'}]$$

i.e., the Quetelet index does not change during the run of the protocol. This axiom is needed, because the decision of prescribing a treatment is not exactly at the same time as the application of the treatment and therefore the decision of prescribing this treatment could be based on a patient with a different Quetelet index than the patient that actually takes the drugs.

Proving this property in KIV was done in approximately 1 day using several heuristics for the straightforward parts. The theorem was proven using two lemmas for two specific patient groups. In total, it took approximately 500 steps, of which nearly 90% were done automatically, to verify this property.

8.4 Order of treatments

Finally, another instance of (M3) was proven. This property phrases that the order of any two treatments in the protocol is consistent with the order relation as we have defined in Subsection 6.3. In other words, in case a patient may receive multiple treatments, the less radical treatments are tried first. The formalisation of the property in KIV was done as follows:

$$\Box\forall_T(Tick \wedge T = Patient[\text{'treatment'}] \rightarrow \Box(\textbf{last} \vee (Tick \rightarrow \neg(T \leq Patient[\text{'treatment'}]))))$$

At each time, the current treatment is bound to a static variable (i.e., unchanged by symbolic execution) T, which can be used to compare against subsequent steps in the protocol. For any future steps, we require that either the protocol completes (**last** holds) or that activated treatments are not more preferred than T. The additional 'Tick' variable is needed in the formalisation to abstract from technical system steps.

This property also had a high degree of automation with roughly 800 steps in total. The reason for this slightly higher number of steps is due to nested temporal operators.

9 Discussion

As the interest in medical guidelines continues to grow, there is a need for criteria to asses the quality of medical guidelines. An important method for the appraisal of medical guidelines was introduced by the AGREE collaboration [9]. A solid foundation for the application of *formal methods* to the quality checking of medical guidelines, using simulation of the guideline [15, 31] and theorem proving techniques [25], can also be found in literature.

In [25], logical methods have been used to analyse properties of guidelines, formalised as task networks. In [24], it was shown that the theory of abductive diagnosis can be taken as a foundation for the formalisation of quality requirements of a medical guideline in temporal logic. This result has been used to verify quality requirements of good practice medicine of treatments [21]. However, in the latter work, the order between treatment depending on the condition of the patient and previous treatments was ignored. In this paper, we consider elements from both approaches by including medical background knowledge in the verification of complete networks of tasks. This required a major change to the previous work with respect to the formulation of quality criteria, because quality is now defined with respect to a complete network of tasks instead of individual treatments as presented in [24].

Compared to previous work concerning the verification of networks of tasks, the meta-level approach we have presented here has a number of advantages. In the meta-level approach, quality is defined independently of domain specific knowledge, and, consequently, proof obligations do not have to be extracted from external sources. One successful attempt of the latter was reported in [18], where quality criteria are formalised on the basis of instruments to monitor the quality of care in practice, i.e., medical indicators. Firstly, the question is whether these indicators, based on compliance with medical guidelines, coincide with the quality of the guideline itself. Secondly, it has been our experience that it is far from easy to find suitable properties in external sources, because these sources may not be completely applicable, e.g., typically, other guidelines may address different problem in the management of the same disease. Thirdly, many useful quality criteria of guidelines are implicit, making this approach fundamentally limiting. In this sense, the meta-level approach provides a more systematic method for the formulation of proof obligations and, thus, verification of medical guidelines.

In summary, in this study we have setup a general framework for the verification of medical guidelines, consisting of a medical guideline, medical background knowledge, and quality requirements. A model for the background knowledge of glucose level control in diabetes mellitus type 2 patients was developed based on a general temporal logic formalisation of (patho)physiological mechanisms and treatment information. Furthermore, we developed a theory for quality requirements of good practice medicine based on the theory of abductive diagnosis. This model of background knowledge and theory of quality requirements were then used in a case study in which we verified several quality criteria of the diabetes mellitus type 2 guideline used by the Dutch general practitioners. In the case study we use Asbru to model the guideline as a network of tasks and KIV for the formal verification.

In the course of our study we have shown that the general framework that we have setup for the formal verification of medical guidelines with medical background knowledge is feasible and that the actual verification of the proposed quality criteria can be done with a high degree of automation. We believe both the inclusion of medical background knowledge and task networks to be necessary elements for adequately supporting the development and management of medical guidelines.

10 Comparison with other formal verification techniques

Formal methods: Verification using symbolic calculation can be mechanised using the methods of several types of reasoning, such as model checking, constraint solving, theorem proving, etc. Figure 12 shows a range of formal methods ranging from cheap to incomplete to very expensive and complete (loosely based on a picture by Rushby). The work that is presented in this paper is

of the latter kind, which has certain advantages, e.g., it provides insight in the proof structure. For each case, it is relatively easy to inspect the proof tree and to find out the reason why a certain quality criterion holds. On the other hand, KIV is a tool with a very expressive logic, which may result in an additional overhead when verifying quality criteria of medical guidelines. Thus, it makes sense to look at cheaper methods for verification of medical guidelines. This is particularly important when guidelines are rapidly updated, where fully automated formal methods are most realistic. Below, work on model checking and automated theorem proving of medical guidelines is briefly discussed.

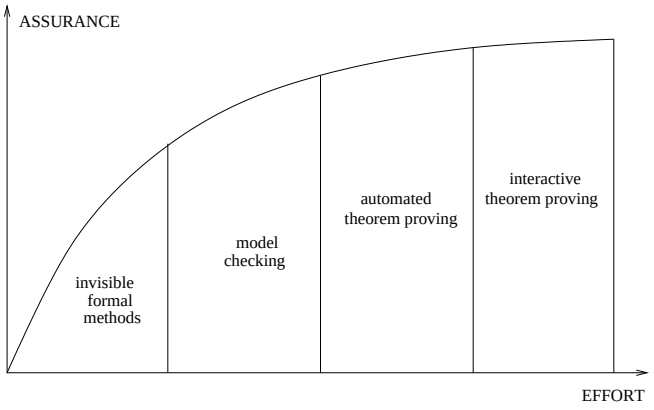

Figure 12. A spectrum of formal methods for formal verification allowing a tradeoff in the properties one can verify (assurance dimension) against the effort one needs to invest to obtain results (effort dimension).

Model checking: Model checking is an effective technique for verifying properties of a formal system. In model checking, a specification about a model, which is usually some form of transition system, is expressed as (temporal) logic formulas, and efficient algorithms traverse the states of the system to verify whether the specification holds or not. Extremely large state-spaces can be traversed in a short amount of time. The first model checkers verified the correctness of discrete state systems, but have been extended to also deal with real-time and probabilistic reasoning.

In the Protocure project, a mapping has been developed for automatically transforming guidelines in the Asbru language into SMV for model checking purposes [7]. As the mapping is made into SMV, this transformation abstracts from the notion of time. Hence, not every property can be verified using SMV [26]. Model checking has been found to be very useful when constructing the Asbru model. [12] defines a number of structural properties which should be fulfilled by a good quality Asbru model. By model checking these structural properties of the Asbru model, one can quickly check the model during development. Hence, model checking provides a good tradeoff between effort and assurance for these kind of properties, however, the framework as specified in [7] is unable to deal with more complex properties that deal for example with time.

In another study [19], model checking has been used to check the conformance of medical guidelines with medical protocols, which are local adaptations by hospitals of medical guidelines. A different view towards medical guidelines was followed in [19] compared to the program-like view presented in the current paper. As medical guidelines often omit many details, e.g., common sense reasoning about first informing a patient before treatment, guidelines are often under-constrained. In [19] a constraint-based approach is used for model checking the conformance of medical protocols. Additional background knowledge can be incorporated in the model checking approach by using modular model checking [22]. This allows one to verify a property with respect to a restricted part of the model. For example, one can restrict the model to those states that adhere to common sense medical practice, such as the fact that diagnosis usually occurs before treatment of the patient.

Automated theorem proving: Previously, it was shown that for reasoning about models of medical knowledge, for example in the context of medical expert systems [23], classical automated reasoning techniques (e.g., [33, 46]) are a practical option. In [20], we studied the use of automatic theorem proving techniques for quality checking medical guidelines. In this context, reasoning about Asbru plans is not feasible, however, simple treatment plans can be encoded directly in temporal logic. Translation of temporal logic yields a restricted first-order theory, e.g., the temporal formula $\mathbf{G}p$ can be interpreted as by $\forall_{t'}\colon (t \leq t' \rightarrow p)$. Such a formalisation is suitable for use in standard resolution-based theorem provers. Note that in practice, this is not a fully automated process, as the theorem prover needs to be guided in the use of (resolution-)strategies and sometimes it is helpful to define lemmas. Nonetheless, automated theorem provers require less interaction than interactive theorem provers. Furthermore, it is possible to add background knowledge to the system, whereas, adding background knowledge to a transition system will generally result in a state explosion making model checking infeasible.

ACKNOWLEDGEMENTS

We would like to thank all members of the Protocure project for providing a stimulating research environment.

REFERENCES

[1] M. Balser, *Verifying Concurrent Systems with Symbolic Execution – Temporal Reasoning is Symbolic Execution with a Little Induction*, Ph.D. dissertation, University of Augsburg, Augsburg, Germany, 2005.

[2] M. Balser, O. Coltell, J. van Croonenborg, C. Duelli, F. van Harmelen, A. Jovell, P. Lucas, M. Marcos, Misch. S., W. Reif, K. Rosenbrand, A. Seyfang, and A. ten Teije, 'Protocure: Supporting the development of medical protocols through formal methods', in *Computer-Based Support for Clinical Guidelines and Protocols*, eds., K. Kaiser, S. Miksch, and S. Tu, pp. 103–107. IOS Press, (2004).

[3] M. Balser, C. Duelli, and W. Reif, 'Formal semantics of Asbru - an overview', in *Proceedings of the International Conference on Integrated Design and Process Technology*, Passadena, (2002). Society for Design and Process Science.

[4] M. Balser, C. Duelli, W. Reif, and G. Schellhorn, 'Verifying concurrent systems with symbolic execution', *Journal of Logic and Computation*, **12**(4), 549–560, (2002).

[5] M. Balser, C. Duelli, W. Reif, and J. Schmitt, 'Formal semantics of asbru – v2.12', Technical report, University of Augsburg, (June 2006). Url: http://www.informatik.uni-augsburg.de/lehrstuehle/swt/se/publications/.

[6] M. Balser, W. Reif, G. Schellhorn, K. Stenzel, and A. Thums, 'Formal system development with KIV', in *Fundamental Approaches to Software Engineering*, ed., T. Maibaum, number 1783 in LNCS. 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 *Proceedings of 13th International SPIN Workshop on Model Checking of Software*, ed., A. Valmari, volume 3925 of *LNCS*, pp. 219–233. Springer-Verlag, (2006).

[8] P. Clayton and G. Hripsak, 'Decision support in healthcare', *International Journal of Biomedical Computing*, **39**, 59–66, (1995).

[9] AGREE Collaboration, 'Development and validation of an international appraisal instrument for assessing the quality of clinical practice guidelines: the agree project', *Qual Saf Health Car*, **12**, 18–23, (2003).
[10] 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), 1–27, (2004).
[11] D. Dickenson and P. Vineis, 'Evidence-based medicine and quality of care', *Health Care Analysis*, **10**, 243–259, (2002).
[12] G. Duftschmid and S. Miksch, 'Knowledge-based verification of clinical guidelines by detection of anomalies', *OEGAI Journal*, 37–39, (1999).
[13] E. Allen Emerson, 'Temporal and modal logic.', in *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B)*, 995–1072, (1990).
[14] *Clinical Practice Guidelines: Directions for a New Program*, eds., M. Field and K. Lohr, National Academy Press, Institute of Medicine, Washington D.C., 1990.
[15] J. Fox and S. Das, *Safe and Sound: Artificial Intelligence in Hazardous Applications*, AAAI Press, 2000.
[16] J. Fox, N. Johns, A. Rahmanzadeh, and R. Thomson, 'PRO*forma*: A method and language for specifying clinical guidelines and protocols', in *Medical Informatics Europe*, eds., J. Brender, J.P. Christensen, Scherrer. J.R., and P. McNair, pp. 516–520, (1996).
[17] J. Fox, N. Johns, A. Rahmanzadeh, and R. Thomson, 'PRO*forma*: a general technology for clinical decision support systems', *Computer Methods and Programs in Biomedicine*, **54**, 59–67, (1997).
[18] M. van Gendt, A. van Teije, R. Serban, and F. van Harmelen, 'Formalising medical quality indicators to improve guidelines', in *AIME*, number 3581 in LNAI, pp. 201–220. Springer Verlag, (2005).
[19] A. Hommersom, P. Groot, and P. Lucas, 'Checking guideline conformance of medical protocols using modular model checking', in *The 18th Belgium-Netherlands Conference on Artificial Intelligence*, pp. 173–180, (2006).
[20] 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).
[21] A.J. Hommersom, P.J.F. Lucas, and M. Balser, 'Meta-level Verification of the Quality of Medical Guidelines Using Interactive Theorem Proving', in *Logics in Artificial Intelligence: 9th European Conference*, volume 3229 of *Lecture Notes in Computer Science*, pp. 654–666, Lisbon, Portugal, (September 2004). Springer-Verlag.
[22] O. Kupferman and M.Y. Vardi, 'Modular model checking', *Lecture Notes in Computer Science*, **1536**, 381–401, (1998).
[23] P. J. F. Lucas, 'The Representation of Medical Reasoning Models in Resolution-based Theorem Provers', *Artificial Intelligence in Medicine*, **5**, 395–419, (1993).
[24] P.J.F. Lucas, 'Quality checking of medical guidelines through logical abduction', in *Proceedings of AI-2003, the 23rd SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, eds., F. Coenen, A. Preece, and A.L. Mackintosh, volume XX, pp. 309–321, London, (2003). Springer.
[25] M. Marcos, M. Balser, A. ten Teije, and F. van Harmelen, 'From informal knowledge to formal logic: A realistic case study in medical protocols', in *Proceedings of EKAW*, pp. 49–64. Springer, (2002).
[26] K. L. McMillan, *Symbolic Model Checking*, Kluwer Academic Publishers, 1993.
[27] S. Miksch, 'Plan management in the medical domain', *AI Communications*, **12**(4), 209–235, (1999).
[28] M. Peleg, A. Boxwala, O. Ogunyemi, P. Zeng, S. Tu, R. Lacson, E. Begnstam, and N. Ash, 'GLIF3: The evolution of a guideline representation format', in *Proc. AMIA Annual Symposium*, pp. 645–649, (2000).
[29] M. Peleg, L.A. Gutnik, V. Snow, and V.L. Patel, 'Interpreting procedures from descriptive guidelines', *Journal of Biomedical Informatics*, **39**(2), 184–95, (2006).
[30] 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).
[31] S. Quaglini, M. Stefanelli, A. Cavallini, G Micieli, C. Fassino, and C. Mossa, 'Guideline-based careflow system', *Artificial Intelligence in Medicine*, **20**(1), 5–22, (2000).
[32] R. Reiter, 'Equality and domain closure in first order databases', *Journal of ACM*, **27**, 235–249, (1980).
[33] J. A. Robinson, 'Automated Deduction with Hyperresolution', *International Journal of Computatational Mathematics*, **1**, 23–41, (1965).
[34] G.E.H.M. Rutten, S. Verhoeven, R.J. Heine, W.J.C. de Grauw, P.V.M. Cromme, and K. Reenders, 'NHG-standaard diabetes mellitus type 2 (eerste herziening)', *Huisarts Wet*, **42**, 67–84, (1999).
[35] J. Schmitt, M. Balser, and W. Reif, 'Complementary material to Deliverable D4.2b: Improved Verification System', in *Protocure II - Integrating formal methods in the development process of medical guidelines and protocols*, (2005).
[36] J. Schmitt, M. Balser, and W. Reif, 'Support for Interactive Verification of Asbru in KIV', Technical Report 2006-16, Universität Augsburg, Institut für Informatik, (June 2006).
[37] A. Seyfang, R. Kosara, and S. Misch, 'Asbru's reference manual, asbru version 7.3', Technical Report Asgaard-TR-20002-1, Vienna University of Technology, Institute of Software Technology, (2002).
[38] A. Seyfang, S. Miksch, P. Votruba, K. Rosenbrand, J. Wittenberg, J. von Croonenborg, W. Reif, M. Balser, J. Schmitt, T. van der Weide, P. Lucas, and A. Hommersom, 'D2.2a Specification of Formats of Intermediate, Asbru and KIV Representations', in *Protocure II - Integrating formal methods in the development process of medical guidelines and protocols*, (2004).
[39] A. Seyfang and J. Schmitt, 'D2.3b Asbru-to-KIV translator', in *Protocure II - Integrating formal methods in the development process of medical guidelines and protocols*, (2004).
[40] 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).
[41] S.E. Strauss, W.S. Richardson, P. Glasziou, and R.B. Haynes, *Evidence-based Medicine - How to Practice and Teach EBM*, Churchill Livingstone, 2005.
[42] S. Tu and M. Musen, 'A flexible approach to guideline modeling', in *Proceedings of American Medical Informatics Association Symposium (AMIA 1999)*, pp. 420–424, (1999).
[43] S. Tu and M. Musen, 'From guideline modeling to guideline execution: Defining guideline based decision-support services', in *Proceedings of American Medical Informatics Association Symposium*, pp. 863–867, Los Angeles, CA, (1999).
[44] S. Woolf, R. Grol, A. Hutchinson, M. Eccles, and J. Grimshaw, 'Potential benefits, limitations, and harms of clinical guidelines', *British Medical Journal*, **318**, 527–530, (1999).
[45] S.H. Woolf, 'Evidence-based medicine and practice guidelines: an overview', *Cancer Control*, **7**, 362–367, (2000).
[46] L. Wos, R. Overbeek, E. Lusk, and J. Boyle, *Automated Reasoning: Introduction and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1984.

Appendix A - Specification of Asbru in KIV

This appendix gives a bit more details about the specification of and reasoning about Asbru plans in KIV. More details about the representation is described in Protocure deliverables [38, 39] and the technical report [36].

The syntax of Asbru is defined with several algebraic specifications in KIV. Figure 13 gives an overview of the specifications and their dependency structure. The specifications with a box 'CUT' attached belong to the library specifications included in KIV and are not shown in detail. We discuss only some of the more important design choices in more detail below.

The 'asbru-clock-basic' specification defines the data type 'asbru-clock', which is a two-component counter, with the first component being either an integer or infinity, and the second component being a natural number. The first counter of the clock counts the time steps the system has gone through, i.e., the macro-steps (cf. Section 7.2). An integer is used as the absolute number is unimportant. This allows lemmas to be inserted at different time points without the difficulty with natural numbers that there exists some zero time point such that

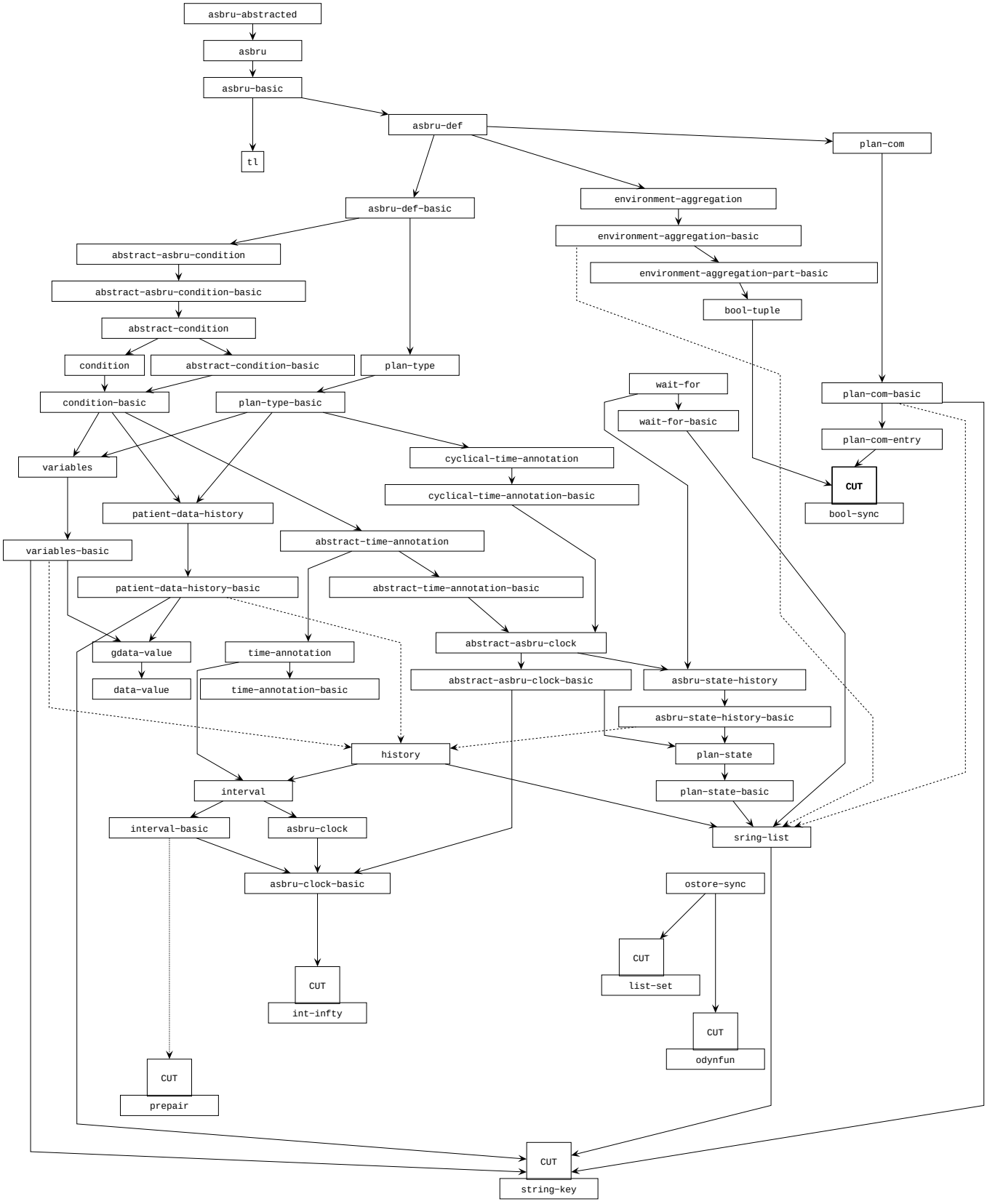

Figure 13. Definition of syntax of Asbru plans.

one cannot go back infinitely back in time. The second counter is a micro-step counter.

The 'asbru-clock' enriches the asbru clock, which adds functionality for moving back and forward in time. As micro-steps are technical steps that do not represent real time steps they are not related to concepts such as 'earlier' or 'later'. It is therefore not possible to address individual micro-steps, but only to a list of states that has been reached in between two macro-steps.

The 'interval-basic' creates a rudimentary time-interval using a pair of asbru-clocks.

The 'ostore-sync' specification adds the specification of the predicate 'sync'. This is needed to come around difficulties with concurrent access to data types within synchronous parallel execution. In general synchronous write access from more than one process to one variable is seen as a clash and the result of such a clash can be defined in a number of ways. For example, the result can be (1) chosen from the result of one of the processes, (2) arbitrary, (3) the results of both processes (e.g., when they access different fields in an array), or (4) an inconsistency leading to an abort of the program. The 'sync' predicate postpones the decision how to react to clashes and allows it to be specified on the case study level.

The 'history' specification is a generic specification with the type of the included dynamic function left undefined. This allows one to define generic simplification rules and reuse them for multiple specifications. In the Asbru specification the history construct is used for the variable history, the Asbru state history, and the patient data history. The selectors in the history are basically time points, but intervals have also been added to increase modularisation.

The most important data structures within the specification of Asbru are the 'asbru state', 'patient data', and 'patient'. The 'asbru state' stores all configurations of Asbru plans, i.e., their current state according to the semantics of the state-chart (cf. Figure 3). The 'patient data' stores all the known values about the patient. Note, that there is a difference between the 'patient' data structure and 'patient data' data structure, as the former contains information about the *actual condition* of the patient, while the latter represents the *knowledge* the medical staff has about the patient. The knowledge may be outdated as the values in the patient may have changed.

The plan states known by Asbru are defined in the specification 'plan-state-basic', which is enriched by 'plan-state' to included additional concepts to summarise some of the plan states, e.g., 'terminated' summarises the states 'completed', 'rejected', and 'aborted'. The synchronisation between plans is specified in 'plan-com', which gathers the signals that may be sent from a super-plan to its respective sub-plans. The signals are represented in internal variables to shield them from the environment which simplifies the sequents and their proofs as environmental non-interference does not have to be specified separately.

The interface to the Asbru specification is an algebraic type 'asbru-def' in KIV, which simply defines a structure of the form in Figure 14. Each Asbru plan is transformed into KIV using the algebraic function 'mk-asbru-def' by filling in the values used by the Asbru plan for its parameters.

```
asbru-def = mk-asbru-def
  (. .filter : asbru-condition;
   . .setup : asbru-condition;
   . .suspend : asbru-condition;
   . .reactivate : asbru-condition;
   . .complete : asbru-condition;
   . .abort : asbru-condition;
   . .type : plan-type;
   . .retry : bool;
   . .subplans : string-list;
   . .wait-for : wait-for;
   . .opt-wf : bool;
  );
```

Figure 14. Syntax of Asbru plans using 'mk-asbru-def'.

Appendix B - Symbolic execution of Asbru

This appendix gives a bit more details about reasoning about Asbru plans in KIV. More details about the symbolic execution is described in the Protocure deliverable [35] and technical report [36].

The proof method in KIV is based on a sequent calculus with rules of the form:

$$\frac{\Gamma_1 \vdash \Delta_1 \quad \ldots \quad \Gamma_n \vdash \Delta_n}{\Gamma \vdash \Delta}\, name.$$

Rules are applied bottom-up. Rule *name* refines a given conclusion $\Gamma \vdash \Delta$ with n premisses $\Gamma_i \vdash \Delta_i$. Furthermore, KIV uses rewrite rules to rewrite sub-formulas, which are of the form

$$name: \quad \phi \leftrightarrow \psi,$$

to replace a formula ϕ by an equivalent formula ϕ anywhere within a given sequent.

The idea of symbolic execution of arbitrary temporal formulas (e.g., Asbru plans) is to normalise the temporal formulas to the form $\tau \wedge \circ\phi$, which separates the possible first transitions from the temporal formulas describing the system in the next state. The general pattern of the normal form is given by

$$\tau_0 \wedge \textbf{last} \vee \bigvee_{i=1}^{n} (\exists X_i . \tau_i \wedge \circ\phi_i),$$

with X_i static variables occurring both in transition τ_i and system ϕ_i to capture the link between these formulas. The operator **last** is included as the system may also terminate. The rules in KIV to rewrite arbitrary temporal formulas to normal form are described in [1].

After normalisation, the sequent can be rewritten using the rules *dis l* and *ex l* to eliminate disjunction and quantification.

$$\frac{\phi, \Gamma \vdash \Delta \quad \psi, \Gamma \vdash \Delta}{\phi \vee \psi, \Gamma \vdash \Delta}\, dis\ l \qquad \frac{\phi[{}^{X_0}/_{X}], \Gamma \vdash \Delta}{\exists X.\phi, \Gamma \vdash \Delta}\, ex\ l$$

where X_0 is a fresh static variable with respect to the variables in free(ϕ)\{X}∪free(Γ, Δ). For the remaining premises

$$\tau_0 \wedge \textbf{last} \vdash \quad \tau_i \wedge \circ\phi_i \vdash$$

the two rules *lst* and *stp* can be applied

$$\frac{\tau[{}^{X,X,X}/_{A,A',A''}] \vdash}{\tau, \textbf{last} \vdash}\, lst \qquad \frac{\tau[{}^{X_1,X_2,A}/_{A,A',A''}], \phi}{\tau, \circ\phi \vdash}\, stp$$

where X, X_1, X_2 are fresh with respect to free(τ, ϕ). Note that rule *lst* deals with the situation when execution terminates and all free dynamic variables A - no matter if they are unprimed, primed, or double primed - are replaced by fresh static variable X. The result is a formula in pure predicate logic with static variables only, which can be proven with standard first-order reasoning. The rule *stp* advances the trace one step. The values of the dynamic variables A and A' in the old state are stored in fresh static variables x_1 and X_2. Double primed variables are unprimed variables in the next state. Finally, the leading next operators are discarded and the proof method continues with the execution of ϕ_i.

Table 2. Notation

Temporal Logic Operators and Statements (Sections 5 and 6)	
$\Box\,\varphi, \Diamond\,\varphi, \circ\,\varphi, \bullet\,\varphi$, **last**	See Table 1
$\mathcal{B}$	Background knowledge
T	Treatment
P	Patient group
N	Medical intentions
M	Medical guideline
Drug(x)	Holds if and only if drug x is administered at that point in time
SU	Sulfonylurea drug
BG	Biguanide drug
QI	Quetelet index
$T \preceq_\varphi T'$	Treatment T' is at least as preferred as treatment T
Good$_\varphi(T, P)$, Good$_\varphi(M, P)$	Treatment T, respectively, medical guideline M, is in accordance with good practice medicine for patient P and criteria φ

Asbru (Sections 4 and 6.2)	
considered, possible, activated, suspended, aborted, completed	Plan states
filter, setup, complete, abort	Conditions controlling execution
consider, activate	Synchronizing signals

Specification in KIV (Sections 7 and 8)	
v_s, v_d	A static, respectively, dynamic variable, which has a constant, respectively changing, interpretation on each time point
v'_d, v''_d	v'_d is the value of v_d after a system transition, v''_d is the value of v'_d after the environment transition, i.e., the value of v_d in the next state
Knowledge$(pre, post)$	For patient data structures *pre* and *post*, with *pre* denoting the current state and *post* the next state of the patient, the predicate Knowledge defines the relation that must hold between *pre* and *post*
$s[v]$	The value of variable v in algebraic sequence s
$s[v, c]$	Algebraic sequence s, where v is updated with value c
AS	The internal state of the Asbru program
Tick	A macro-step in the asbru execution

Appendix C - Notation

Table 2 provides a summary of the notation used in this paper.