

Verification of Medical Guidelines by Model Checking – A Case Study

Simon Bäumlér, Michael Balsér, Andriy Dunets,
Wolfgang Reif, and Jonathan Schmitt

Lehrstuhl für Softwaretechnik und Programmiersprachen,
Institut für Informatik, Universität Augsburg,
Augsburg, 86135 Germany

{baeumlér, balsér, dunets, reif, schmitt}@informatik.uni-augsburg.de
<http://www.informatik.uni-augsburg.de/lehrstuehle/swt/se/>

Abstract. This paper presents a case study on how to apply formal modeling and verification in the context of quality improvement in medical healthcare. The aim is to verify quality requirements of medical guidelines and clinical treatment protocols that are used to standardize patient care both for general practitioners and hospitals. This research is supported by the European Commission’s IST program and brings together experts from computer science, artificial intelligence in medicine, hospitals, and the Dutch Institute for Healthcare Improvement (CBO). We present the process of formal modeling and verification of guidelines using the modeling language Asbru, temporal logic for expressing the quality requirements, and model checking for proof and error detection. The approach is illustrated with a case study on a guideline from the American Association for Pediatrics on “Jaundice in healthy Newborns”¹.

Keywords: Model checking, verification, formal methods, Asbru, abstraction, medical guidelines.

1 Introduction

Over the last decade, the approach of evidence-based medicine has increased the application of clinical guidelines in medical practice. Medical guidelines provide clinicians with healthcare recommendations based on valid and up-to-date empirical evidence. Usually they consist of “systematically developed statements to assist hospital staff with appropriate healthcare decisions” [12]. Application of guidelines improves the quality of medical treatment and it has been proven that adherence to guidelines and protocols may reduce healthcare costs up to 25%.

Many practical guidelines and protocols still contain ambiguous, incomplete or even inconsistent elements. Recent efforts have tried to address quality improvement of guidelines [21]. Our general approach to verification of guidelines is based

¹ This work has been partially supported by the European Commission’s IST program, under contract number IST-FP6-508794 Procure II.

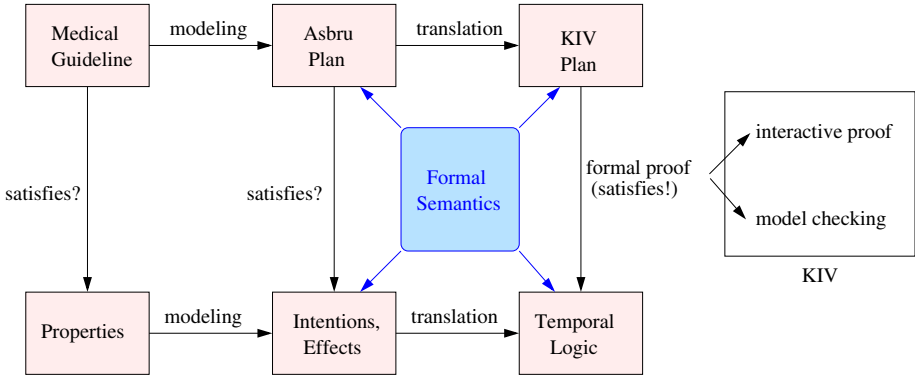


Fig. 1. Formalization and verification of protocols in the Protocure project

on the observation that guidelines can be viewed as parallel programs. Therefore the classical formal methods for the quality assurance of software can be applied for the case of medical guidelines, especially because guidelines are highly-structured, systematic documents that are amenable for formal verification.

Because most parts of a guideline consist of informal plain text an appropriate representation language with clear and well-defined semantics is required. For this purpose we use Asbru[19]. Asbru is a temporal, skeletal plan-representation language which was especially designed for the medical domain. The most important advantage of using Asbru as a modelling language is its formal semantics[3]. Figure 1 shows the flow of documents in the overall verification process. The original guideline is depicted in the upper left corner of Fig. 1. It is modeled as Asbru plan using the knowledge-representation language Asbru.

The Asbru model is the basis for further tasks, e.g. to build decision support systems. For these tasks it is necessary to ensure the quality of the model. Thus, we are interested in tools to efficiently debug the model, e.g. to ensure its consistency. [10] defines a number of structural properties which should be fulfilled by a good quality Asbru model. Some of these properties can be checked by syntactic analysis. Other properties require formal analysis. Furthermore, we are interested in the formal verification of more complex medical properties such as medical indicators. Complex, infinite state properties in general require interactive theorem proving. For structural and simple medical properties we aim for efficient techniques which can be automatically applied. For this, we automatically translate the model into a formal representation for an interactive theorem prover KIV[2]. In order to apply model checking, we further translate the model into the input language of SMV model checker[18]. In this paper, we focus on model checking of properties.

Simultaneously to the above transformation, in Figure 1, a number of interesting properties have been identified while analyzing both the original protocol and its Asbru model. We distinguish between *Medical Properties* and *Structural Properties* (see 3.2).

To evaluate our approach we have considered the medical guideline for “Jaundice in healthy newborns”, a medical guideline from the American Association of Pediatrics, that covers various features of Asbru. We will use the jaundice protocol in the following sections as running example for our paper.

The identified properties do not depend on timing constraints. Therefore, it has been possible to abstract away from time which reduces the complexity of the model. For the verification of these properties, we have chosen SMV as a model checker, because to our knowledge this is one of the most efficient tools to verify large models without complex timing constraints. For real-time properties, the use of timed model checkers, e.g. UPPAAL[22], will be of interest.

We have used Cadence SMV version 10-11-02 with default settings on a computer with a 3 GHz Pentium processor and 2 GB of RAM.

Our main contributions are: (i) a validated formal model of a concrete medical guideline where the quality has been assured by automatic techniques, (ii) tool support for automatic verification of all of the properties from [10], (iii) case study to assess the possibilities of light-weight model checking techniques to verify structural or simple medical properties of medical guidelines; this case study could serve as a reference case study for other model checkers and other automatic techniques in the field of medical guidelines. We do not present new strategies for model checking SMV models in general.

The paper is organized as follows. Section 2 gives a short overview of the Jaundice guideline and the Asbru language with its formal semantics. Section 3 gives a description of a concrete infinite state model of the jaundice protocol and describes its reduction to a finite state model using an abstraction. In section 4 we summarize our experiences from this case study and describe possible improvements of the current process planned for future work.

2 Asbru: A Knowledge Representation Language for Protocols

We describe Asbru and its use by a simple example. Details on Asbru can be found in [19].

2.1 The Jaundice Protocol

Jaundice, or hyperbilirubinemia, is a common disease in newborns which is caused by increased bilirubin levels in blood. Under certain circumstances, high bilirubin levels may have destructive neurological effects and thus must be accurately treated. Often jaundice disappears without treatment, but sometimes a phototherapy is needed to reduce the level of total serum bilirubin(TSB). In a few cases, however, jaundice is a sign of a severe disease, which must be treated appropriately.

The jaundice reference guideline[1] is a 10 pages document which contains various notations: the main text; a list of factors to be considered when assessing a jaundiced newborn; two tables - one for the management of the healthy term

newborns and another for the treatment options for jaundiced breast-fed ones; and a flowchart describing the steps in the protocol. The Protocol consists of two parts performed sequentially: diagnosis and treatment. Treatment is performed if disease symptoms are detected. During the application of the protocol, as soon as the possibility of a more serious disease is uncovered, the recommendation is to exit without any further action. The further treatment is not considered in the guideline.

2.2 Modeling the Jaundice Protocol in Asbru

Medical guidelines are represented as hierarchical skeletal **plans**, i.e. plans with subplans. Figure 2 shows the hierarchy of plans representing the Asbru model of jaundice protocol. It is made up of about 40 plans. Two phases in the protocol control flow clearly emerge: diagnostics and treatment parts which are executed sequentially. Three “Check-for-...” plans model two check-ups at specific time intervals and a continuous monitoring of the TSB level. We focus here on the treatment phase, which is more interesting from the verification point of view. It consists of two parallel plans, namely the actual treatment and a cyclical plan asking for the input of new TSB and age values every 12 to 24 hours. Depending on the current bilirubin level, either the *regular-treatments* or an *exchange-transfusion* can take place. The plan-body of regular-treatments plan contains two subplans which are executed in parallel without any ordering. The

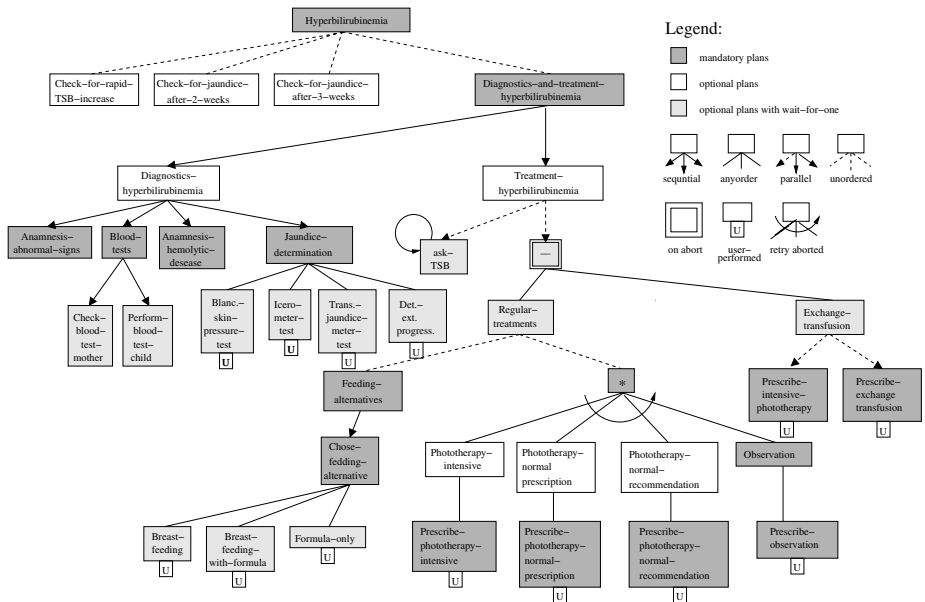


Fig. 2. Asbru plans modeling jaundice protocol

*regular-treatments** subplan (which is abbreviated with * in Figure 2) represents a group of therapies, which are executed sequentially without any order. All of the therapy plans are optional with the exception of the *observation* plan which must complete for the successful completion of the parent plan. Any of these therapies can be restarted, in case if it is eventually aborted.

Figure 3 shows an example of two Asbru plans from the jaundice guideline. An Asbru model of a plan contains the definitions of different descriptive elements like intentions, conditions, plan body and control structures. In the following we describe these main elements.

The **intentions** are the high-level goals of a plan. Intentions can be expressed in terms of achieving, maintaining or avoiding certain states or actions. The states or actions to which intentions refer can be intermediate or final.

```

plan regular-treatments
intentions ...
conditions ...
plan-body type=unordered, wait-for all
  feeding-alternatives
  /* implicit subplan regular-treatments*: */
  do type=any-order, retry-aborted-subplans=yes, wait-for observation
    phototherapy-intensive
    phototherapy-normal-prescription
    phototherapy-normal-recommendation
    observation

plan phototherapy-intensive
intentions
  achieve-overall-state: (bilirubin=observation)
  maintain-intermediate-state:
    (and(TSB-decrease=yes in [[4h,-],[-,6h],[-,-]] SELF)
      (TSB-change>1 in [[4h,-],[-,6h],[-,-]] SELF))
conditions
  setup-condition: (or(bilirubin=phototherapy-intensive in NOW)
    (normal-phototherapy-failure))
  abort-condition: (or(and(bilirubin!=phototherapy-intensive)
    (not normal-phototherapy-failure))
    (intensive-phototherapy-failure))
  intensive-phototherapy-failure:
    (and(bilirubin=phototherapy-intensive in NOW)
      (or(and(TSB-decrease=yes in [[4h,-],[-,6h],[-,-]] SELF)
        (TSB-change<1 in [[4h,-],[-,6h],[-,-]] SELF))
        (TSB-decrease=no in [[4h,-],[-,6h],[-,-]] SELF)))
plan-body
  prescribe-intensive-phototherapy

```

Fig. 3. Regular-treatments and Phototherapy-intensive plans

Thus, the intention label “maintain-intermediate-state” means that always during the execution of the plan a certain condition must be satisfied. Generally there are twelve possible forms of intention: *[achieve/maintain/avoid] [intermediate/overall] [state/action]*. Most of the medical properties we considered in the verification are gained from the intentions (see Sec. 3.2). For example, one of the intentions of the phototherapy-intensive plan (see Fig. 3) is to maintain a certain intermediate state, i.e. in all cases in 4 to 6 hours after the activation of the plan the bilirubin level decreases. As all intentions of the jaundice guideline this is an universal property.

Every **plan-body** contains the actions to be performed by the plan and/or subplans to be executed as part of the plan. A wide variety of **control** structures can be used to specify the execution order of the actions in the plan-body. There are the following types of plan-bodies in Asbru:

- user-performed: an action to be performed by the user, which requires user interaction and thus is not modeled further
- single step: an action which can be either an activation of a subplan, an assignment of a variable, or request for an input value
- subplans: a set of steps to be performed in a given order. The possible execution orders are: **sequential**, **parallel**, in any possible sequential order (**anyorder**) and in parallel without any restrictions on the synchronization (**unordered**)
- cyclical plan: a repetition of actions over time periods

When a plan-body contains subplans it is possible to define the completion of some (or all) subplans as a necessary precondition for the successful completion of the parent plan. For example **wait-for-all** type of the plan-body means, that the successful completion of parent plan requires successful completion of all of its subplans. Similarly **wait-for-one** or **wait-for someplan** can be defined.

The *regular-treatments* plan (Fig. 3) is a good example for a more complicated structure of the plan-body. It has an unordered plan-body with wait-for-all option and two subplans: *feedings-alternatives* and *regular-treatments**. The implicit plan *regular-treatments** consists of several different therapies executed in any order (see Fig. 2 and Fig. 3). Its subplan *phototherapy-intensive* (Fig. 3), for instance, describes one of the therapies. Its plan-body simply contains the activation of the subplan *prescribe-intensive-phototherapy*.

A variety of **conditions** can be associated with a plan, which influence control of an execution of the plan. The most important types of conditions are the following: filter-, setup-, activate-, abort-, and complete-condition. The meaning of these conditions is described more closely in the section 2.3. Conditions can not only specify a set of satisfying current states² but also they can be monitored over time, if they are formulated using time annotations, e.g. in 4 to 6 hours after the activation of plan the bilirubin level change decrease is greater than 1.

² An Asbru state is composed of the state of execution of all plans and the state of the patient. Further we have the Asbru history which is defined as a mapping from the Asbru clock to an Asbru state and allows to specify time annotated conditions.

Time annotations can occur in conditions. They specify the time period where a parameter condition used in the time annotation is monitored. A time annotation is defined by the following eight entities: reference point (*REF*), earliest starting shift (*ESS*), latest starting shift (*LSS*), earliest finishing shift (*EFS*), latest finishing shift (*LFS*), minimum duration (*MinDu*), maximum duration (*MaxDu*) and parameter proposition *ParamProp*. These components are combined in the data structure:

$$(\textit{ParamProp} \textbf{in} [[\textit{ESS}, \textit{LSS}], [\textit{EFS}, \textit{LFS}], [\textit{MinDu}, \textit{MaxDu}]] \textit{REF})$$

Reference points like *NOW* (current time) and *SELF* (time of activation of this plan) are commonly used. Consequently a time annotation defines a set of time intervals (also called *set of possible occurrences*). A time annotation is *TRUE* if and only if there exist a time interval in the set of possible occurrences where the condition *ParamProp* is evaluated to *TRUE* in all time points within this interval.

As example consider the following time annotation from the phototherapy-intensive plan:

$$(\textit{TSB-decrease} = \textit{yes} \textbf{in} [[4\textit{h}, -], [-, 6\textit{h}], [-, -]] \textit{SELF})$$

This time annotation monitors the bilirubin level on the time interval between 4 to 6 hours after the plan start. It is evaluated to *TRUE* if there is a new bilirubin measurement with a value smaller than the latest measured value before the plan started. It has the following meaning: there exist time interval (or point as special case of interval) with earliest starting at 4h after the plan activation and latest finishing at 6h after activation where *TSB-decrease=yes* is true. The predicate *TSB-decrease=yes* is evaluated to *TRUE* on the given interval if and only if for all time point t_0 on this interval the bilirubin value is smaller than bilirubin value at the time of plan activation.

2.3 Formal Semantics of Asbru

We use the formal semantics of Asbru defined in [3]. The semantics follows two goals: first it should document Asbru and be understandable for users; on the other hand it should be formal enough. We use the example of *regular-treatments** plan to explain the semantics here.

The operational semantics of Asbru is defined using statecharts. It uses the formal semantics of statecharts defined in [9]. Asbru plans are modeled as statecharts which run in parallel and communicate via shared variables and signals. Asbru conditions are monitored over time. The evaluated conditions trigger transitions of the statecharts. The evaluation of conditions depends on the data inputs from the environment usually describing dynamics of patient. Shared variables like patient parameters or state of other plans can also influence the evaluation of Asbru conditions. For example the abort-condition of phototherapy-intensive plan (see Fig. 3) triggers the abort of the plan as soon as it fails to reduce bilirubin level in 4 to 6 hours after the plan activation.

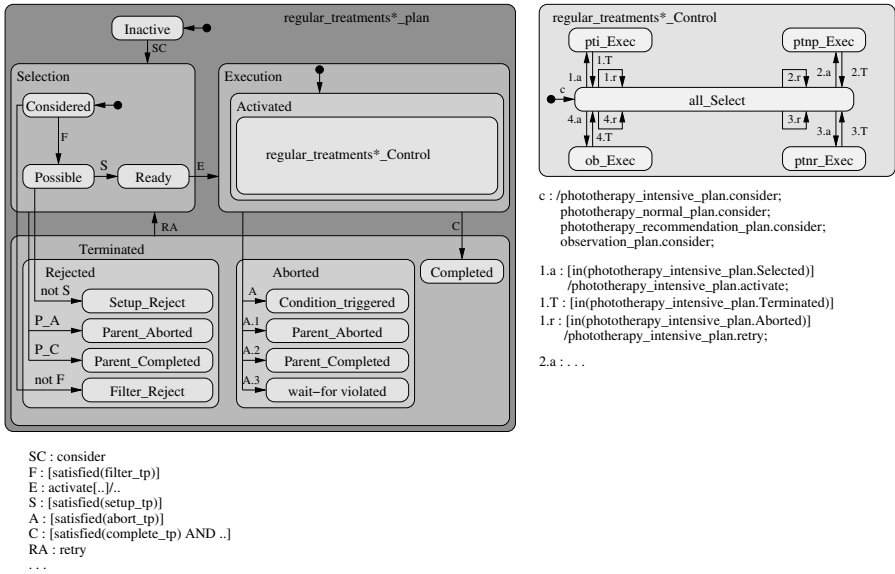


Fig. 4. Statecharts modeling regular-treatments* plan

The *regular-treatments** plan is an implicit subplan of the *regular-treatments* plan (see Fig. 2 and Fig. 3). The behavior of the *regular-treatments** plan is defined by the statechart in Figure 4. This statechart is divided into a *Selection* phase and an *Execution* phase. Initial state of the plan is *Inactive*. An external signal *consider* triggers the selection phase (transition SC). In the state *Considered* the condition *filter_tp* is checked. In case this condition is satisfied, the plan changes to the state *Possible* and so on. In the state *Activated* the subplans are executed. The execution of subplans is controlled by the *regular_treatments*_Control* statechart (Fig. 4), which models an anyorder control of subplans. It is responsible for the generation of the *consider*-, *activate*- or *retry*-signals, which control the execution of subplans. All subplans are selected in parallel (transition *c*) and executed such that at most one subplan is active at the same time (transitions *i.a*). If the activated plan terminates (transitions *i.T*) another one can be activated. If several subplans reach state *Selected* simultaneously, one of them is activated nondeterministically. If parameter flag “*retry-aborted*” is set, then the transition *i.r* initiates a restart.

The original infinite state model of the system is a composition of statecharts running in parallel and reacting on environment inputs. Interaction with the environment happens in micro- and macro-steps. One macro-step consists of many micro-steps which describe reactions of the system on certain environment input. When a system achieves a stable state the corresponding macro-step is completed and in the first micro-step of the next macro-step new input from the environment is read. This model corresponds to the assumption that the system, which models the guideline, always reacts quick enough to changes of the environment. It is also intuitively the proper modeling for medical plans,

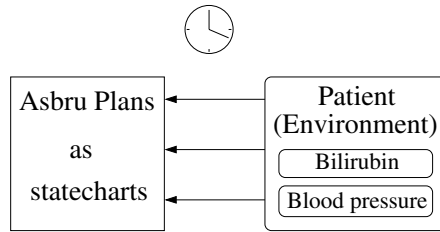


Fig. 5. Original infinite state model of medical guideline

i.e. control of plans does not take time to activate or cancel some plans. The notion of time is defined using macro-steps. Time passes only at the begin of every macro-step.

In Asbru we have no explicit model of the patient, i.e. we do not model specific patient behavior. We assume that the patient has chaotic behavior, i.e. parameter values blood pressure, bilirubin level in blood, etc., change arbitrarily over time. This allows us to investigate whether the medical protocol reacts adequately in all possible cases. Consequently the infinite state concrete model is the composition of statecharts modeling medical plans, the environment and the Asbru clock modeling current time, see Figure 5. This model resembles somehow the model of parallel processes communicating using shared variables.

3 Model Checking Process

The crucial point in model checking and verification in general is computing an optimal abstraction of the examined system. Much research has concentrated on tackling the state explosion problem. A variety of abstraction techniques have been developed, for example [15], [16], [4], [14] and [8]. The basis for these investigations is an important observation: there are various aspects of the concrete model that have no impact on the checked property and can be abstracted in such a way that the size of the model is drastically reduced, but the property is still safely verified, i.e. the satisfaction of a property over an abstract model implies satisfaction over the concrete model. Methods that derive an abstract model directly from some high-level description of the system are needed.

3.1 Abstract Finite State Model

Generally, it is a hard task to construct a correct abstract finite state model for the generic Asbru model completely automatically, since Asbru is a very expressive language.

The infinite parameters describing the patient (or environment) can be abstracted to finite state variables using data abstraction [4], [17]. The more problematic issue is time, which usually requires some kind of history variable. All plans, in order to proceed, must know whether their setup-, filter-, abort- or complete-condition is satisfied or not. Some of these conditions contain time

annotations, as for example the abort-condition of *phototherapy-intensive* plan does (see Fig. 3). In order to evaluate time annotated conditions an Asbru plan must access its history.

Our goal is an abstraction which can be constructed automatically for the given Asbru model. In order to construct a finite state model an appropriate abstraction which eliminates time and history is needed. We use a simple abstraction that maps all time annotations to atomic propositions whose logical value is randomly assigned in every macro step. Those random inputs can eventually generate behavior in the abstract model that is not present in the concrete model. This abstraction preserves only ACTL³ properties, as it is an over-approximation, which adds extra behavior to the abstract model. Nevertheless, this is not a problem for us, because most interesting properties we aim to verify are intentions, which are ACTL properties.

The main weakness of this abstraction is the generation of false negatives during the verification of properties. On the other hand the important advantage is its automatic generation. By this abstraction we shift the complexity of time and history to the environment. According to the statechart semantics inputs from the environment happen in the first micro-step of every macro-step and provide the required information about the patient needed for the controls of plans, e.g. up-to-date value of blood pressure or information about change of the bilirubin level in blood over the period of 6 hours after the start of the plan.

For the generation of the SMV model we translate the statecharts from the Asbru semantics into an equivalent flat state transition system, which can be directly encoded in the SMV input language. This part models the control flow of the guideline. On the other hand, data flow and time is modeled using the abstraction techniques described above.

3.2 Properties

The results from the verification of properties should help to improve the quality of medical guideline. Structural properties specify the general correctness requirements, which must be satisfied by every Asbru protocol, regardless of its content. For example, every plan should eventually terminate, every plan must have a chance to execute or be able to complete. Our experiences from the jaundice case study has shown that verification of structural properties helps to discover errors produced during the translation of informal medical guideline into the formal Asbru model. The following structural properties have been considered: termination (Asbru plans should always terminate), every plan can eventually be activated (completed), there are no redundant conditions (i.e. every condition can eventually have influence on the control flow of plans) and all wait states are eventually quitted. These properties have been formalized as CTL formulas and are automatically generated for every Asbru plan as SMV specifications. Most of them are originally not ACTL properties, but their verification can be indirectly accomplished by the verification of the corresponding

³ The logic ACTL is the set of all well-formed state formulas from CTL[11] containing no existential operators (EX and EU).

```

term_pti_plan: SPEC AG(!ptip_state = inactive ->
    AF(ptip_state = completed |
        ptip_state = rejected |
        ptip_state = aborted))
satisf_abort_pti_plan: SPEC AG(!(ptip_state=activated &
    ptip_abort_condition))
reach_activated_pti_plan: SPEC AG(!(ptip_state=activated))
wait_possible_pti_plan: SPEC AG(ptip_state = possible ->
    AF(ptip_setup_condition |
        ptip_is_terminated))

```

Fig. 6. SMV specification of structural properties for Phototherapy-intensive plan

ACTL properties, as described in Section 3.3. For example, the corresponding ACTL properties (in SMV syntax) for the plan *phototherapy intensive* (*pti*) are depicted in Figure 6. The property `term_pti_plan` formulates a termination property, i.e. every plan that was previously selected always terminates in the future. By the `satisf_abort_pti_plan` property we try to verify whether the abort-condition of the phototherapy-intensive plan is redundant or not. In case we find a non-spurious counter-example for `satisf_abort_pti_plan` we know that abort-condition is not redundant, i.e. it has an influence on the plan execution. Similar properties can be formulated for all other Asbru conditions. Reachability of important states is tested by properties like `reach_activated_pti_plan`. The property `wait_possible_pti_plan` tests whether wait state *possible* is eventually quitted.

In contrast to structural properties medical properties address high level aspects of medical protocols, such as relevant clinical parameters or general safety requirements concerning actions of physicians or overall intentions of the guideline. As an example, when treating jaundice, it is required that 6 hours after application of phototherapy the bilirubin level must drop significantly. In the jaundice case study we considered only plan intentions as conceptual properties. For instance, plan *phototherapy-intensive* has two intentions, which can be specified as ACTL properties, as Figure 7 shows. With the *intermediate state* in the second property we mean only stable states, i.e. states in which the reaction of the plan on the environment inputs is completed. Therefore, the variable *tick*

```

--achieve overall state: bilirubin = observation
SPEC AG(ptip_state = completed -> AF AG bilirubin = observation)

--maintain intermediate state: tsb_decrease = yes & tsb_change>=1
SPEC AG((ptip_state = activated & tick) ->
    (pti_tsb_decrease_yes_signal &
    !pti_tsb_change_less_one_signal))

```

Fig. 7. SMV specification of medical properties for Phototherapy-intensive plan

is used to describe the *activated* state of the *pti* plan where it is *stable*, i.e. no transitions of the corresponding statechart are activated.

3.3 Verification Process

The abstraction we use is described in 3.1. It allows us to generate the SMV model fully automatically. On the other hand it can introduce unrealistic behavior, which has an impact on the verification process. Figure 8 illustrates the general scheme of verification. Due to property preservation considerations we examine only ACTL properties although it is also indirectly possible to verify ECTL properties. The medical properties we considered are the intentions of the plans, which are formulated as ACTL formulas.

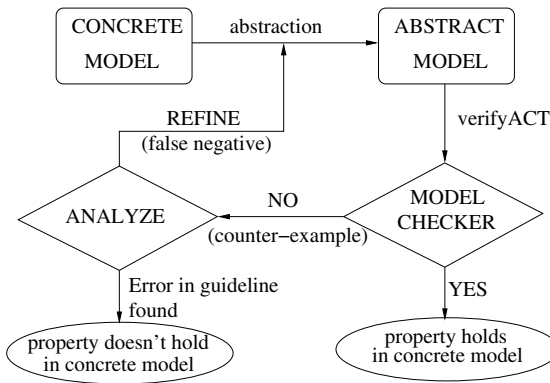


Fig. 8. Verification scheme

IF ACTL property is proved to be false the corresponding counter example is generated. In the next step we have to analyze this trace to find out whether it is a real bug in the concrete model or just some unrealistic trace added by the over-approximation. We also have to verify ECTL properties since most implementation level (structural) properties are existential properties, e.g. satisfiability of Asbru conditions or non-redundancy of plans. If, for instance, a ECTL formula $EF\phi$ must be verified, we first verify the ACTL formula $AG\neg\phi$. If it is *true* then original formula is *false*. On the other hand, if a counter example is found then we analyze whether it is realistic one. If the found counter example trace is realistic then the original formula is *true* and the generated counter example is the trace that satisfies the original formula.

3.4 Results and Experiences from Verification of Jaundice

The abstracted model of the jaundice guideline was constructed in the SMV language and model checking was used to verify different properties. In particular we verified structural and medical properties of approximately 30 Asbru

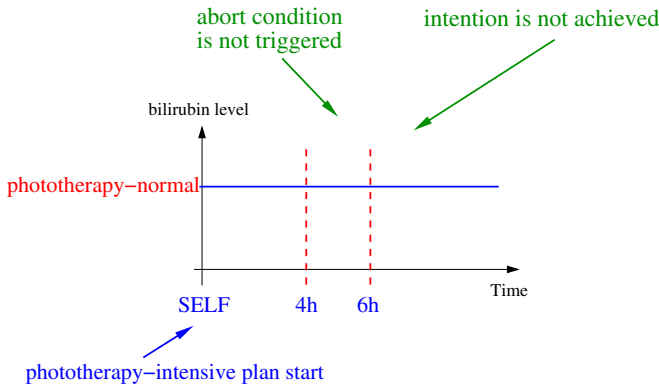


Fig. 9. Counter example visualization

plans from the jaundice hierarchy of plans. Automatic generation of the SMV model consisting of 3000 lines of code and including 360 structural and 40 medical properties has accelerated the whole process. Checking one property takes about 2 minutes on average whereby 2×10^6 BDD-nodes were allocated. Complete verification of circa 400 properties lasted 3.5 hours and 4×10^6 BDD-nodes were allocated.

In the whole verification process the manual effort of analyzing the counter examples and refining the abstract model was rather small and acceptable. The verification of structural properties uncovered several nontrivial modelling errors and therefore helped immensely to gain more confidence in the formal Asbru model.

During the process of formalization and verification of the medical properties, a number of errors and ambiguities were discovered. We have found special cases of treatment that violate plan intentions. These special cases have been overlooked by the Asbru modelers and were consequently not appropriately considered in the Asbru model. Figure 9 illustrates a possible execution sequence that violates the intention of the phototherapy-intensive plan, see Figure 3. The intention postulates that in 4 to 6 hours after plan start the bilirubin level must decrease in the other case plan must abort. As we see in Figure 9 bilirubin level does not decrease in the corresponding time interval and the plan does not abort. The reason for the violation of the plan intention is that the abort-condition was too weak and did not consider all possible cases. Using model checking verification method we have discovered many other similar “forgotten” cases in the overall plan hierarchy containing 30 plans.

4 Summary and Outlook

In this paper we have described the automated verification of medical guidelines using the jaundice case study as an example. The simple abstraction we applied yielded surprisingly good results in the jaundice case study. In fact only few

refinements and fine tuning were needed while verifying properties. This simple abstraction allows us to construct the SMV model fully automatically, which is an important advantage as we plan to apply this method on further case studies. The approach as a whole is not completely automatic but partly an interactive one (see Fig. 8) because our abstraction is over-approximation. Due to the high expressiveness of the Asbru modeling language it is practically not possible to construct a correct abstraction for the general case of an Asbru-model automatically without any user interaction. Nonetheless the degree of automation of this process is very high.

The first errors we have found were consequence of too coarse abstraction. In some cases our first verification experiments have shown that the used abstraction is too coarse. Therefore, to avoid the interactive component *ANALYZE* in the process, see Figure 8, we plan to construct the correct abstraction. We see this as promising direction for further work on verification of Asbru. Further we plan to use the KIV theorem prover to show the correctness of the constructed abstraction by proving the corresponding bisimulation equivalence. Another profitable improvement can be a visualization of model checking results. The graphical interpretation of counter examples can make verification more efficient as it makes the interpretation of traces easier.

Our approach to verify simple properties of medical guidelines by model checking were surprisingly successful. Therefore, it is promising to also apply other automatic techniques to the verification of more complex properties, e.g. real-time properties, and larger guidelines.

Currently, we are applying our method on a second guideline, which describes the treatment of breast cancer. This guideline is considerably larger but our first experiences with it are very promising.

References

1. American Academy of Pediatrics, Provisional Committee for Quality Improvement and Subcommittee on Hyperbilirubinemia. Practice parameter: management of hyperbilirubinemia in the healthy term newborn *Pediatrics*, 94:558-565, 1994.
2. M. Balsler, W. Reif, G. Schellhorn, K. Stenzel, A. Thums. Formal system development with KIV. In T.Maibaum, editor, *Fundamental Approaches to Software Engineering*, number 1783 in LNCS. Springer, 2000.
3. M. Balsler, C. Duelli, W. Reif. Formal Semantics of Asbru - An Overview *In Proc. of the 6th World Conference on Integrated Design and Process Technology(IDPT-02)*, June 2002.
4. E.M. Clarke, O. Grumberg, D.E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512-1542, September 1994.
5. E.M. Clarke, O. Grumberg, D. Peled. Model Checking. *MIT Press*, 2000.
6. P. Cousot, R. Cousot. Abstract interpretation : A unified lattice model for static analysis of programs by construction or approximation of fixpoints. *ACM Symposium of Programming Language*, pages 238-252, 1977.
7. D. Dams. Abstract Interpretation and Partition Refinement for Model Checking. *PhD Thesis*, Eindhoven University of Technology, 1996.

8. D. Dams, R. Gerth, O. Grumberg. Abstract interpretation of reactive systems. *ACM Transactions on Programming Languages and Systems*, 19(2):253-291,1997.
9. A. Pnueli, B. Josko, H. Hungar, W. Damm. A Compositional Real-time Semantics of STATEMATE Designs. *Lecture Notes in Computer Science*, pages 186-238, Springer Verlag, Berlin, Proceedings COMPOS'97.
10. G. Duftschmid, S. Miksch. Knowledge-based verification of clinical guidelines by detection of anomalies *OEGAI Journal* 1999, pages 37 – 39.
11. E. Allen Emerson. Temporal and Modal Logic. *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics 1990*, J. van Leeuwen, ed., North-Holland Pub. Co./MIT Press, Pages 995-1072.
12. M.J. Field, K.N. Lohr. *Clinical Practice Guidelines: Directions for a New Program*. National Academy Press, Washington D.C., USA, 1992.
13. J. Fox, N. Johns, C. Lyons, A. Rahmanzadeh, R. Thomson, P. Wilson. PROforma: a general technology for clinical decision support systems. *Computer Methods and Programs in Biomedicine*, 54:59-67, 1997.
14. P. Godefroid, M. Huth, R. Jagadeesan. Abstraction-based Model Checking using Modal Transition Systems. *Proceedings of CONCUR'2001*, Aalborg, August 2001. *Lecture Notes in Computer Science*, vol. 2154, pages 426-440, Springer-Verlag.
15. O. Grumberg. Abstractions and Reductions in Model Checking. *Nato Science Series*, Vol. 62, Marktobendorf summer school, 2001.
16. S. Graf, H. Saidi. Construction of abstract state graphs with PVS. In *Computer aided verification*, volume 1254 of LNCS, pages 72-83, June 1997.
17. D.E. Long. Model checking, Abstraction, and Compositional Reasoning. *PhD Thesis*, Carnegie Mellon University, 1993.
18. K.L. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem*. Kluwer Academic, 1993.
19. Y. Shahar, S. Miksch and P. Johnson. The Asgaard project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines. *Artificial Intelligence in Medicine* 1998, pages 29 – 51.
20. R. Milner. *A Calculus of Communicating Systems*. Springer, 1980.
21. A. ten Teije, M. Marcos, M. Balsler, J. van Croonenborg, C. Duelli, F. van Harmelem, P. Lucas, S. Miksch, W. Reif, K. Rosenbrand, A. Seyfang, J. Coltel, A. Jovell. Supporting the development of medical protocols through formal methods. In *Proc. of the Symposium on Computerized Guidelines and Protocols (CGP-04)*, IOS Press, 2004.
22. K.G. Larsen, P. Peterson, Wang Yi. UPPAL in a nutshell. *Journal of Software Tools for Technology Transfer*, 1(1-2):134-152, 1997.