

DELIVERABLE SUMMARY SHEET

Project Number: *IST-2001-33049*

Project Acronym: **PROTOCOLURE**

Title: *Improving medical protocols by formal methods*

Deliverable N°: D3

Due date: 31/3/2002

Delivery Date: 3/4/2002

Short Description:

Objectives

The goal of this deliverable was to determine a number of interesting protocol properties.

Procedure

We distinguish properties at the implementation level and properties at the conceptual level. Properties at the implementation level are protocol independent. They are defined during in-depth studies of the protocol-representation language Asbru. Properties at the conceptual level are protocol-dependent. They are defined based on discussions with domain experts.

Within intensive working session (studying Asbru for the first part and collaboration with domain experts for the second part) we could filter out challenging ones and we conclude that we will concentrate on the domain-specific properties for our further verification work.

Results

In this deliverable, we described properties to verify in clinical protocols. We distinguish properties at the conceptual level and properties at the implementation level. While implementation level properties are more general, specific properties at the conceptual level directly reflect the need for verification in a practical application. We therefore will select properties to verify during this project from the group of protocol-dependent properties.

The properties not verified during this project remain as open challenges for a future project.

Partners owning: all

Partners contributed: all

Made available to: public

Desirable/Required Properties of Main Asbru Elements

Silvia Miksch and Andreas Seyfang

Vienna University of Technology
Institute of Software Technology and Interactive Systems (IFS)
Favoritenstrasse 9-11 / 188, A-1040 Vienna, Austria

email: {silvia, seyfang}@asgaard.tuwien.ac.at
url: <http://www.ifs.tuwien.ac.at/~{silvia, seyfang}>

Introduction and Motivation

Safety and transparency aspects are very critical issues not only in the medical domain. A clinical protocol is only accepted by the medical staff, if you “prove” that it is verified and valid as well as all assumptions and consequences are evident. Some available techniques to check the correctness of clinical protocols may be helpful, however, they are not sufficient to verify clinical protocols. The main reason for this is that clinical protocols incorporate several domain-specific properties, which are essential for the verification processes.

One important issue is to define appropriate properties to verify, which is the main goal and the content of this deliverable. We distinguish two types of properties:

- (a) properties at the *implementation* level, which are protocol-representation language dependent, but independent of the particular clinical/medical protocols themselves; and
- (b) properties at the *conceptual* level, which are independent of the language used to represent protocols, but highly oriented towards the medical content of the protocol.

The Acquisition Process

The acquisition of the properties at the two levels described above was performed in different co-operative working sessions.

The implementation-oriented properties were designed during in-depth studies of the various language-dependent elements – in our case, the protocol-representation language Asbru –, such as conditions, effects, and intentions of the Asbru language. Therefore, the properties at implementation level say something about the correctness of the Asbru specification. This defining process was done independently from any medical domain and the medical examples used within the two articles (compare Appendix A) are used mainly for illustration of the overall concepts. Therefore, these examples are not only based on the two reference protocols chosen in the first workpackage, but on general medical examples explaining the language-dependent parts.

The properties at the conceptual level were designed and acquired in close co-operation with the medical experts involved in the project and additional medical specialists asked for more in-depth advice (compare Appendix B). The overall objective of this process was to define language-independent properties. However, the acquisition process was somehow guided by the Asbru language, too. For example, the idea of intentions is an Asbru language concept, the safety parts are oriented on the very detailed definition of conditions, etc. in Asbru, or the complex plan layout influenced the sequencing properties.

According to the acquisition procedure discussed above, this deliverable consists of two halves: Appendix A dealing with the properties at the implementation level and Appendix B covering the properties at the conceptual level. In the following two sections we will illustrate the main contributions of the two Appendices (in other words the content of the articles with in these two Appendices) and the results of the different discussions and working sessions to arrive at appropriate properties according to the particular goals.

Appendix A: Properties at the implementation level

This part covers two journal articles, which are/will be published in the journal: Artificial Intelligence in Medicine and a list of informal and partly vague properties, which was used as a starting point for the working sessions.

(1) Duftschmid, G.; Miksch, S.: *Knowledge-Based Verification of Clinical Guidelines by Detection of Anomalies, Artificial Intelligence in Medicine, Special Issue: Workflow Management and Clinical Guidelines in Medicine, pp. 23-41, 22(1), 2001.*

The focus of this work is to present an approach, which helps to ensure the semantical correctness of clinical guidelines on three levels: the plan itself, all its knowledge roles and all its subplans. The verification process examines the components of every Asbru plan and all its subplans for the existence of several anomalies, which indicate violations of corresponding specifications. The outcomes of the verification process are *legal* or *meaningful* plans, instead of insisting on complete or totally correct plans.

The following properties at the implementation level were discussed during a Protocure meeting with all the partner participating in the Protocure project, which can be summarised as “internal coherence” from the verification point of view.

- [1] Every condition must have a chance to be satisfied.
- [2] Every valid sequence of plan states must be reachable.
- [3] Every plan must be able to complete.
- [4] The target state of a transition should be unambiguous.
- [5] Child plan should not be stopped by parent.
- [6] No state of a plan should be skipped.
- [7] Every plan must have a chance to execute its body.
- [8] A plan must not loop in its execution.
- [9] Every condition, checked at a “wait-state” must be defined in a way, that it makes sense to wait for its satisfaction.
- [10] A condition must only refer to states of other plans, which are reachable.
- [11] A condition should not contain redundant parameter propositions.

These properties were classified as possible candidates and also the interactive verification within the KIV system should be possible, however according to their potential, these properties were classified as low priority candidates.

(2) Duftschmid, G.; Miksch, S., Gall, W.: *Verification of Temporal Scheduling Constraints in Clinical Practice Guidelines, to appear in Artificial Intelligence in Medicine, 2002.*

In this paper, the authors focus on the detection of flaws within temporal scheduling constraints. Temporal scheduling constraints are important elements of therapy management, and are frequently incorporated in clinical practice guidelines. We present a suitable verification method that is based on calculating the minimal network of temporal constraints on the execution of guideline activities.

The method presented serves three purposes:

- (a) it checks whether temporal scheduling constraints are consistent with scheduling constraints implied by control flow operators and the hierarchical structuring of a guideline;
- (b) it yields suggestions for an equivalent, yet more explicit representation of non-minimal constraints; and
- (c) it can be used by the guideline interpreter to assemble feasible time intervals for the execution of each guideline activity.

During a meeting with all the partners of the Protocure project, the overall idea of this approach was discussed. However, due to the time constraints, we agreed that these kinds of properties are challenging, but out of the scope for the current phase of the Protocure project.

(3) *A List of Informal and Partly Vague Properties*

The following is a list of informal and partly vague properties which were collected in the past and these properties were another starting point for the definition of appropriate properties at the implementation level. A prerequisite to understand this list is that the reader is familiar with the various Asbru elements, which can be found in:

Seyfang, A.; Kosara, R.; Miksch S.: Asbru's Reference Manual, Asbru Version 7.3, Vienna University of Technology, Institute of Software Technology and Interactive Systems, Vienna, Technical Report, Asgaard-TR-2002-1, 2002. url: http://www.ifs.tuwien.ac.at/asgaard/asbru/asbru_7_3/

Additionally, this list is based on the whole Asbru language, whereas the current project is only focusing a subset of the whole Asbru language, which we call informally *Asbru light* and is described in detail in deliverable D2 .

[1] *Usage of Parameters:*

Parameters that are asked or observed but which are never used, or the other way around (parameters are used, but never asked/observed/got). This are "gaps" in the protocol.

[2] *Relation of Effects to Intentions:*

An Effect can make the Intention (partial) true or (partial) false.

[3] *Contribution of Subplans:*

A subplan has to contribute to the Intention of the plan.

[4] *Relation of Control Structure to Conditions:*

Determine the relation between the control structure (e.g., Do-All-Sequential) and conditions (e.g., Filter-Conditions).

For Example:

- If in a Do-All-Sequential and the Filter-Condition fails then there is an error in the plan.
- If in a Do-Some-Any-Order, the actions of one subplan should not influence Conditions of other subplans.

[5] *Combinations of Conditions and other Asbru elements:*

The combination of particular conditions imply another Asbru element.

For Example:

- if Filter & Setup & Complete-Condition (& Some-Terminology) hold
then the Intentions holds
- Variants of these:
if (Filter & Setup) & ~Suspended & ~Aborted & Complete & Some-Terminology
(domain ontologies) hold
then Overall-Intentions holds
- Complete-conditions should be consistent (or derivable) with the Filter/Setup-conditions and the execution of the plan (derivable: Filter & Setup -> [plan]Complete -> Intention).

[6] *Plan Library Completeness Properties:*

For each Setup-Condition should be a plan or set of plans that satisfied the Setup-Condition (by Intentions/Compete-Conditions/Effects). i.e. there must be a plan to make the Setup-Conditions true.

- Several relation could hold:
 - Plan or set of plans are "stronger" (do more) then the Setup-condition.
soundness: {plan} -> Setup
 - Plan or set of plans are "weaker" (do less) then the Setup-condition.
completeness: Setup -> {plan}
 - Plan or set of plans do exactly the same.
 - or more difficult: "partly" soundness/completeness.

[7] *Relation of Effects to the Behavior:*

Effects represent the “Actual Behavior”, whereas Intentions represent “Expected Behavior”:

- Intention may be true, which represents the “Expected Behavior”
- Effects must be true, which represents the “Actual Behavior”

[8] *Relation of Effects to the Intentions:*

The Overall Intentions are a subset of Effects, then the Effects with Likelihood = 1.0 represents the Overall-Intention.

These list were acquired and discussed in detail during working sessions and we conclude, that we will not be interested in static analysis as described in [1] or dynamic dimensions as illustrated by [6].

To summarise the work and the discussions of the properties at the implementation level as mentioned in the sections (1) to (3) above, we categorised properties dealing with coherence and correctness as follows.

(a) *Termination properties*

These are properties, which specify a particular point to terminate the execution of each protocol. In other words, infinite loops in the protocol must not exist, because they refer to an error in the protocol.

(b) *Plan body implies intentions*

The intentions of a plan express the purpose of the execution of this plan in a more abstract form. The conditions and the plan body define, which steps are taken to meet these intentions. In a correct plan, the plan body and the conditions imply the intentions.

(c) *Redundancies of plans*

Finding redundancies in subplans or conditions which never trigger would be an interesting task for the future.

(d) *Correct replacement of subplans*

Given an existing plan library, one wants to prove, that certain changes did not change the overall behaviour (or certain parts of it). This plays an important role in the adaptation of global clinical protocols for the needs of a particular site, e.g., a hospital.

(e) *Internal Coherence*

Redundancies to examine include subplans which are never used during execution, conditions which never trigger, and time annotations which do not define a constraint

Appendix B: Properties at the conceptual level / protocol-dependent properties

The second part of properties are the ones at the protocol-dependent level, which is more the conceptual level. In Appendix B, different protocol properties are listed, which were acquired during in-depth sessions with the domain experts. This group of properties are oriented on the two reference protocols: (a) American Academy of Pediatrics' practice guideline for the Management of Hyperbilirubinemia in the Healthy Term Newborn and (b) The Dutch College of General Practitioners' (NHG) Standard for Type 2 Diabetes Mellitus.

The properties described focus on aspects like the particular clinical parameters that are handled in the protocols or on the actions that the healthcare providers are required to perform. Ten properties at the conceptual level could be identified. Almost all properties are illustrated with at least one example.

Conclusion

The goal of this deliverable was to determine a number of interesting protocol properties. We could distinguish two types of properties: properties at the *implementation* level and properties at the *conceptual* level. A large number of such properties are presented in the three papers in the Appendix. Within intensive working session (studying Asbru for the first part and collaboration with domain experts for the second part) we could filter out challenging ones and we conclude that we will concentrate on the domain-specific properties for our further verification work.

Appendix A:

Properties at the Implementation Level

(1) *Duftschnid, G.; Miksch, S.: Knowledge-Based Verification of Clinical Guidelines by Detection of Anomalies, Artificial Intelligence in Medicine, Special Issue: Workflow Management and Clinical Guidelines in Medicine, pp. 23-41, 22(1), 2001.*

(2) *Duftschnid, G.; Miksch, S., Gall, W.: Verification of Temporal Scheduling Constraints in Clinical Practice Guidelines, to appear in Artificial Intelligence in Medicine, 2002.*

Knowledge-based verification of clinical guidelines by detection of anomalies

G. Duftschmid^{a,*}, S. Miksch^b

^a*Department of Medical Computer Sciences, University of Vienna, Spitalgasse 23, A-1090 Vienna, Austria*

^b*Vienna University of Technology, Institute of Software Technology (IFS),
Favoritenstrasse 9-11, A-1040 Vienna, Austria*

Received 31 May 2000; received in revised form 21 September 2000; accepted 8 November 2000

Abstract

As shown in numerous studies, a significant part of published clinical guidelines is tainted with different types of semantical errors that interfere with their practical application. The adaptation of generic guidelines, necessitated by circumstances such as resource limitations within the applying organization or unexpected events arising in the course of patient care, further promotes the introduction of defects. Still, most current approaches for the automation of clinical guidelines are lacking mechanisms, which check the overall correctness of their output. In the domain of software engineering in general and in the domain of knowledge-based systems (KBS) in particular, a common strategy to examine a system for potential defects consists in its verification. The focus of this work is to present an approach, which helps to ensure the semantical correctness of clinical guidelines in a three-step process. We use a particular guideline specification language called Asbru to demonstrate our verification mechanism. A scenario-based evaluation of our method is provided based on a guideline for the artificial ventilation of newborn infants. The described approach is kept sufficiently general in order to allow its application to several other guideline representation formats. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Clinical guidelines and protocols; Verification; Medical plan management

1. Introduction

Within the last decade, public and private dissatisfaction about the perceived health and economical consequences of inappropriate medical care has significantly increased [5]. These perceptions stem from many sources including ceaselessly escalating health care costs, wide variations in medical practice patterns, and recognition that the obvious effect

* Corresponding author. Tel.: +43-1-40400-6696; fax: +43-1-40400-6697.

E-mail addresses: georg.duftschmid@akh-wien.ac.at (G. Duftschmid), silvia@ifs.tuwien.ac.at (S. Miksch).

of some health services does not justify the efforts invested. Market pressures have further contributed to complicate the situation by driving medical organizations to increase productivity and to reduce costs, all without adversely affecting patient care.

One method that has been proposed as an answer to these problems is to adopt standard practice guidelines. The main goals that are pursued by the application of guidelines are the improvement of patient care's quality and the reduction of treatment costs. It could be shown that these desired effects can actually be reached, provided that the guidelines are properly formulated and followed [8].

In Section 2 we will give an overview of current approaches for the computerization of clinical guidelines and point out some techniques for their verification. After a short introduction of the Asbru language for the representation of guidelines in Section 3, we will present our verification method in Section 4. A scenario-based evaluation of our method will be provided in Section 5.

2. Related approaches to improve medical care

2.1. Automation of clinical guidelines

Being confronted with the laborious and inherently error-prone manual management of paper-based clinical guidelines, the interest in a computerized handling of the problem started to grow within the medical community. First attempts to automate guideline-based care were able to demonstrate a number of advantages over manual methods [13,30].

A common strategy among guideline applications is the so-called prescriptive approach, where an active interpretation of a preselected guideline is generated by the system, e.g. [6,11,20,25,29,31,32]. Another approach is the critiquing approach, where the system critiques the physician's plan rather than recommending a complete one of its own [2,33]. Finally, several approaches are based on the hypertext browsing of guidelines via the world wide web, e.g. [3,13].

2.2. Verification of clinical guidelines

One prerequisite for a broad acceptance and an efficient application of guidelines in the clinical domain is the guarantee of a high level of quality and reliability. However, as has been shown in numerous studies most clinical guidelines embody different kinds of semantical errors that compromise their practical value: guidelines are often vague and incomplete, they show serious omissions and unintentional ambiguities as well as unclear definitions, incompleteness, and inconsistency [15,16,18,26,27,29].

The situation is further complicated by the fact that the process of correcting such errors within a guideline is not a one-time matter. Rather this task is of repetitive nature as guidelines are subject to change. Changes may be effectuated by the evolution of a guideline to implement medical progress in the treatment of individual diseases. Another reason for the modification of a guideline, which will become effective even more frequently is addressed in [7–9]: generic, site-independent guidelines, designed to be

sharable between institutions, have to be adapted in most cases when applied by an individual organization according to the specific properties of the organization and of the patient to be treated. These adaptations may become necessary before the execution of a guideline due to resource or organizational limitations as well as particular patient characteristics. They may also become essential during the execution of a guideline due to unexpected events that arise in the course of patient care. In each of these cases, it has to be checked that the alterations done to the guideline do not affect its logical flow.

In contrast to the intensive efforts to develop new guidelines, the issue of providing mechanisms to ensure their overall correctness has been widely neglected thus far. However, as is already known from the domain of software engineering, an early identification of flaws within algorithmic knowledge is critically important [1]. In the domain of software engineering in general and in the domain of knowledge-based systems (KBS) in particular, a common strategy to examine a system for potential defects consists in its verification. We use the term *verification* as proposed in [12]: verification is defined as the sum of all processes, which attempt to determine whether a KBS does or does not satisfy its purely formal specifications. As will be shown in Section 4.2.2, we are concerned with specifications derived from formalizable concepts.

As already indicated within [5,14], approaches concerning the semantical verification of clinical guidelines are still rather rare. In the following we will give an overview of the few approaches concerning guideline verification, which have been published yet.

2.2.1. Decision-table techniques

One method for the verification and simplification of guidelines described in the literature consists in the logical analysis of guidelines by applying decision-table techniques [26,27]. Verification here is limited to two different properties, which are *completeness* and *consistency* of a guideline: a guideline is said to be complete when an action is defined for every possible value-combination of parameters used within the guideline. It is considered consistent, if each of its rules, consisting of a certain condition and an assigned action, is unique. If the latter property is violated, the authors distinguish between the three variants of *redundant*, *contradictory* or *conflicting* guidelines.

2.2.2. Examination of related tasks

Quaglini et al. [23] described how a guideline may be examined for logical correctness. As a formal representation model, they organize guidelines as sets of hierarchical tasks, which amongst others include activation conditions and subtasks. Subtasks may either be in AND-relation or in XOR-relation, which means that their parent task completes either after all subtasks have completed (AND), or after one and only one subtask has completed (XOR). To show a guideline's correctness Quaglini and co-workers check for *completeness* and *coherence*: Merging Shiffman and Greenes's concepts of completeness and conflict [27], Quaglini and co-workers consider a guideline complete, if the activation-conditions of each set of subtasks in XOR-relation are defined in a way that for every combination of variable values there is exactly one task activated by that combination. Checking a guideline for coherence means to look for conjunctive subtasks, which exclude each other because of incompatible activation conditions.

2.2.3. Condition checking based on semantic constraints

In [17] a system called Commander is described, which helps to verify a clinical guideline by checking its conditions similar to [27]. In contrast to Shiffman and Greenes, the authors ignore the *consistency* property in their verification process and exclusively concentrate on examining the *completeness* of a guideline. Hereby, they incorporate semantic constraints to reduce the combinations of variable values to consider.

2.2.4. Dynamic testing methods

In [28] an approach for the dynamic analysis of the ONCOCIN knowledge-base (KB) is described, the latter being used for the representation of cancer therapy guidelines. By using the ScriptGen system, the oncologist is given the possibility to generate certain sets of test cases, which allow for the testing of selected paths within a guideline. Hereby, instances are searched for in the KB whose behavior deviates from that dictated by the specification, which is the corresponding cancer therapy guideline. Typical defects are rules that fire erroneously, rules that contain errors in parameter domains and missing rules.

2.2.5. Verification approaches in the domain of KBS

Looking for general verification methods from the domain of KBS that may be useful, we found some parallels to our case: a substantial number of verification approaches in the field of KBS is based on the detection of flaws in rule-bases, e.g. [19,21,22,34]. As we will demonstrate in Section 4.2.1, a relationship can be established between a rule-base and a clinical guideline. This principally enables the reuse of work done for the verification of rule-based systems. However, existing work is far from being sufficient for a profound verification of clinical guidelines: techniques, which are reusable in our case, are too generic to allow the incorporation of those guideline-specific properties, which we consider essential for our verification processes.

3. The Asgaard/Asbru project

Most existing approaches for the automation of clinical guidelines suffer from at least one significant limitation: they require a complete modeling of the guideline up to its smallest grained components during design time and do not support site- or patient-specific adaptation in response to a changed environment or unexpected events. These approaches are lacking the types of knowledge, which would be essential for the purpose of guideline adjustment, or do not provide them in a structured format.

The time-oriented, intention-based plan-representation language *Asbru* [16], developed within the *Asgaard* project [24], offers such support. Clinical guidelines coded in the *Asbru* language are organized within a *plan-specification library*. In the following the term *plan* is used to designate a clinical guideline, coded in the *Asbru* language. By providing a formal description of a plan's *intention*, *Asbru* satisfies an essential demand for efficient guideline transformation [8]: the structured definition of a plan's intention allows it to be adapted while remaining faithful to its original aim.

Our verification approach, presented in Section 4, further supports *Asbru*'s ability of plan modification: before any plan can be released that has been altered based on

compatible intentions, it must be verified amongst others that it is free of logical inconsistencies. The subject of determining compatibility of intentions represents another interesting research issue but is not addressed in this paper.

3.1. Components of Asbru

In Asbru a clinical guideline is modeled as a set of hierarchical plans. Each plan is identified by a unique name and consists of a set of arguments, including a *time annotation* that represents the temporal scope of the plan, and five elementary components. These components are *preferences*, *intentions*, *conditions*, *effects* and a *plan body*. Each plan may contain any number of subplans within its plan body, which may themselves be decomposed into sub-subplans. In the following, we will call such a tree structure of plans, starting from one root plan down to its leaf plans, a *plan hierarchy*. During execution time, the system interpreter attempts to decompose each plan into its subplans until an undecomposable leaf plan is found. Such atomic plans are called *primitive plans* and are transferred to a suitable agent for their execution.

Intentions, world states, actions/plans and effects are durative. Therefore, plan states and corresponding transition criteria are incorporated in Asbru to cope with the time-oriented environment. In this paper we will particularly focus on the verification of these transition criteria, which are expressed by means of the *condition* component.

3.2. Plan states and state-transition criteria

A set of eight different *plan states* is used to describe the actual state of a plan during plan selection and plan execution. Seven different conditions build the *state-transition criteria*, controlling transitions between neighboring plan states. Fig. 1 shows Asbru's *model of plan states* (MPS), a deterministic, finite-state automaton. It illustrates the sequence of possible states of a plan and the corresponding conditions shown above the arrows.

Asbru provides seven different conditions:

1. *filter-preconditions* need to hold initially if the plan is applicable, but can not be achieved;
2. *setup-preconditions* need to be achieved to enable a plan to start;
3. *activate-condition* determines if the plan should be started manually or automatically;
4. *suspend-conditions* determine when an activated plan has to be interrupted;
5. *abort-conditions* determine when an activated or suspended plan is terminated unsuccessfully;
6. *complete-conditions* determine when an activated plan is terminated successfully;
7. *reactivate-conditions* determine when a suspended plan can be continued.

The plan states, which are stopping the execution of a plan successfully or unsuccessfully (rejected, completed, aborted and suspended states), may be propagated in both directions of the plan hierarchy: the parent plan always propagates these plan states to its children, whereas a child plan propagates them to its parent plan if the child belongs to its parent's *continuation set* (see Section 4.2.1).

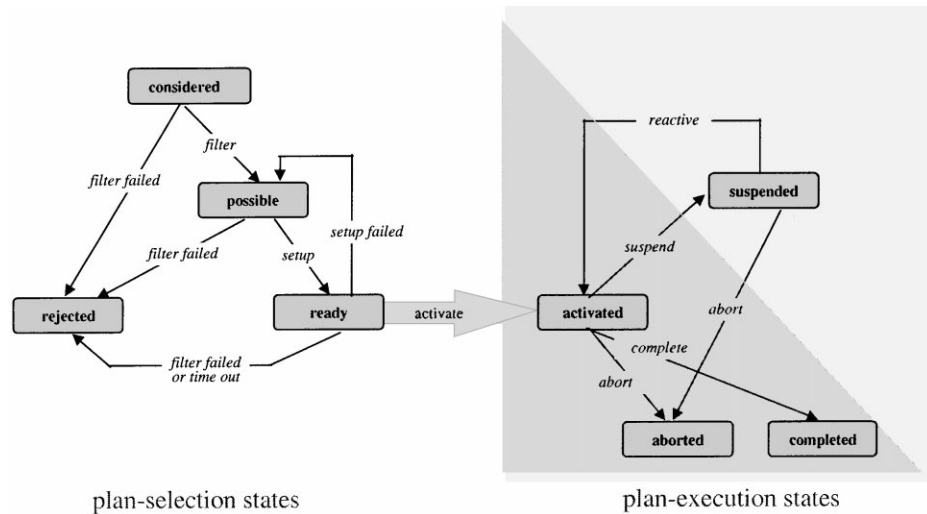


Fig. 1. Model of plan states — states and transition criteria during plan-selection (left) and plan-execution phases (right).

4. Verification of Asbru plans

We developed a partial verification method that aims at the identification of flaws within a guideline [4]. Hereby, we reuse existing verification work as the basis of domain-independent flaws, e.g. [21]. We further extend it by incorporating guideline-specific knowledge for the detection of flaws, which are characteristic for the domain of guideline-based care. The required knowledge is assumed to be available in a suitable KB component (see Section 4.2.3).

4.1. Methodology

Our verification approach examines the components of every Asbru plan and all its subplans for the existence of several anomalies, which indicate violations of corresponding specifications. The concept of anomalies is adopted from [22], where anomalies are defined as symptoms of probable errors. Our goal is to arrive at *meaningful* plans instead of totally correct plans. A plan is called meaningful, if it does not contain any anomalies, which would violate one of our specifications. We distinguish three levels of anomalies according to their locality (see Table 1).

The described approach is based on the hierarchical organization of clinical guidelines within the Asbru language. This type of structuring is also common in a wide range of other guideline representations, e.g. [6,11,23,31,32]. Therefore, our approach may also be applicable to numerous guideline models, other than the Asbru language.

The properties examined by our verification method are of static nature, which means that we will not have to execute any Asbru guideline in order to verify them.

Table 1

Three different levels of plan-verification: detect anomalies within single components (level 1), single plans (level 2) and whole plan hierarchies (level 3)

Decomposition	Detect anomalies within three levels			
	Level 1	Level 2	Level 3	
Plan A			}	
Subplan A _a		}		
Component A ₁	✓			}
...	✓			
Component A _m	✓			
...				
Subplan A _x		}		
Component X ₁	✓			}
...	✓			
Component X _n	✓			

4.1.1. Algorithm

The three levels are verified in a bottom-up fashion: first, level 1 is examined, which means that every single component of a plan is checked for anomalies within its own scope. The goal of checking level 2 is to detect anomalies, which result from dependencies between two or more components of a single plan. In level 3, the whole plan hierarchy is finally checked for anomalies that may originate from dependencies between two or more plans of the hierarchy. As guideline adaptations, in response to organizational or contextual issues, will primarily involve the exchange of whole plans, verification level 3 will be of particular importance in these cases. Fig. 2 shows an exemplary implementation of our method that is initiated by sending the message *verifyHierarchy()* to all root-plans of the library.

We have shown in [4] that, whereas the complexity of checking levels 1 and 2 grows linearly with the number of plans, it is exponential for checking level 3 in the general case.

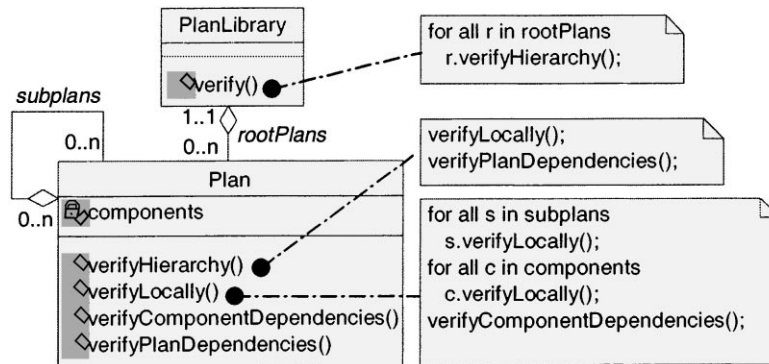


Fig. 2. Two classes with methods, relevant for three levels verification.

We have demonstrated, however, that level 3 can be verified with polynomial effort under the assumption that the size of examined component sets is limited by a suitable constant.

4.2. Verification of plan conditions

Any kind of guideline or plan, regardless of how it is modeled, needs a mechanism to control the sequence of its proposed actions. This mechanism is usually implemented through conditions. In the following, we will practically apply our verification method to illustrate the examination of Asbru conditions. Our considerations will be based on the existence of the following three features, incorporated in the Asbru language:

- conditions to control state transitions;
- a generic Model of Plan States;
- a hierarchical organization of plans.

4.2.1. Mapping plans to rule bases

Preece et al. gave a detailed summary of anomalies, that can occur in rule bases [21]. In the following we will show, how Asbru's Model of Plan States and every single plan can be transformed to a RB. We can then refer to [21] in formulating the specifications for plan conditions. Some of the anomalies we address, especially those which are more general in nature, are directly derived from [21]. These existing anomalies have been complemented with additional ones, which result from specifics of the Asbru language and do not have an equivalent [21].

After some definitions, we will first describe the complete and generic RB MPS for the Asbru's Model of Plan States (see Fig. 1). Its rules specify in which sequence the conditions of a plan are considered. The RB MPS is preset by the Asbru language, assumed to be initially verified and cannot be changed by the user. Therefore, it does not have to be verified further.

The second RB we present is called *PC*, and contains the plan conditions of all Asbru plans. It can be divided into rule bases PC_i for each plan, thus $PC = PC_1 \cup \dots \cup PC_n$. All rules in *PC* have in common that they may only contain consequences, which are elements of *ConditionSet* (see below). As the rule bases PC_i are to be implemented by the user, they will be the target of our verification method.

4.2.1.1. Definitions. Assume *pl* and *pa* are parameters for entities “plan” and “patient”.

PlanSet = set of all plans in the Asbru library.

PatientSet = set of all patients, to whom a plan may be applied.

ConditionSet = {filter(*pl*, *pa*), setup(*pl*, *pa*), activate(*pl*, *pa*), suspend(*pl*, *pa*), reactivate(*pl*, *pa*), abort(*pl*, *pa*), complete(*pl*, *pa*)} / * correlates with arrows in Fig. 1 */.

FinalStateSet = {rejected(*pl*, *pa*), aborted(*pl*, *pa*), completed(*pl*, *pa*)}.

Rule $R = L1 \wedge \dots \wedge Ln \rightarrow M$; antec¹(R) = { $L1, \dots, Ln$ }; conseq²(R) = M .

SS_i = set of all subplans of plan *i*.

¹ Abbreviation for *antecedent*.

² Abbreviation for *consequent*.

$CS_i = \text{continuation set}$ of plan i . This is a subset of SS_i , containing all subplans, relevant for the completion of plan i : if a plan has subplans, some of them may need to complete as a prerequisite for the completion of plan i itself. The set of relevant subplans is defined in the parent plan.

A hypotheses H can be *inferred* from a RB for some environment E , if H is a logical consequence of supplying E as input to RB , formally: $\text{infer}(H, RB, E) \text{ iff } (RB \cup E) \vdash H, E \in \text{PatientSet}$.

A hypotheses H is *inferable* from a RB if there is some environment E such that H can be inferred from RB for E , formally: $\text{inferable}(H, RB) \text{ iff } (\exists E) \text{ infer}(H, RB, E), E \in \text{PatientSet}$.

A rule $R \in RB$ *fires* for some environment E , if the antecedent of R is a logical consequence of supplying E as input to RB , formally: $\text{fire}(R, RB, E) \text{ iff } (\exists \sigma) (RB \cup E) \vdash \text{antec}(R)\sigma, E \in \text{PatientSet}$.

A rule $R \in RB$ is *fireable* if there is some environment E such that R fires for E , formally: $\text{fireable}(R, RB) \text{ iff } (\exists E) \text{ fire}(R, RB, E), E \in \text{PatientSet}$.

4.2.1.2. Rule bases MPS and PC. Table 2 shows the RB MPS . The order of the rules is important to determine which and when rules will be fired. The choice of rule ordering is oriented towards the state-transition criteria (compare Fig. 1).

Table 3 gives an example for a simplified version of a certain PC_k , a RB for all conditions of the plan $GDM\text{-}TYPE\ II$ to treat noninsulin-dependent gestational diabetes mellitus in patients with normal blood-glucose levels.

In order to get a complete model for the execution control of a certain plan i , we have to unify its conditions base PC_i with the generic RB for Asbru's Models of Plan States, formally $MPS \cup PC_i$.

To handle the anomalies, which may occur within a RB $MPS \cup PC_i$, it is first necessary to outline the specific properties of this RB .

- We mentioned that we are going to map all conditions of Asbru plans to rules of the form $L1 \wedge \dots \wedge Ln \rightarrow M$. As these rules may only contain conjunctive literals, we will have

Table 2
RB MPS representing the generic model of plan states

R1	Considered(pl, pa) /* starting state for each plan */
R2	Considered(pl, pa) $\wedge$ filter(pl, pa) $\rightarrow$ possible(pl, pa)
R3	Considered(pl, pa) $\wedge$ $\neg$ filter(pl, pa) $\rightarrow$ rejected(pl, pa)
R4	Possible(pl, pa) $\wedge$ setup(pl, pa) $\rightarrow$ ready(pl, pa)
R5	Possible(pl, pa) $\wedge$ $\neg$ filter(pl, pa) $\rightarrow$ rejected(pl, pa)
R6	Ready(pl, pa) $\wedge$ $\neg$ filter(pl, pa) $\rightarrow$ rejected(pl, pa)
R7	Ready(pl, pa) $\wedge$ $\neg$ setup(pl, pa) $\rightarrow$ possible(pl, pa)
R8	Ready(pl, pa) $\wedge$ activate(pl, pa) $\rightarrow$ activated(pl, pa)
R9	Activated(pl, pa) $\wedge$ abort(pl, pa) $\rightarrow$ aborted(pl, pa)
R10	Activated(pl, pa) $\wedge$ complete(pl, pa) $\rightarrow$ completed(pl, pa)
R11	Activated(pl, pa) $\wedge$ suspend(pl, pa) $\rightarrow$ suspended(pl, pa)
R12	Suspended(pl, pa) $\wedge$ reactivate(pl, pa) $\rightarrow$ activated(pl, pa)
R13	Suspended(pl, pa) $\wedge$ abort(pl, pa) $\rightarrow$ aborted(pl, pa)

Table 3

RB PC_k representing the conditions of plan *GDM-TYPE II*

Female(pa) $\wedge$ Pregnant(pa) $\rightarrow$ filter(GDM-TYPE II, pa)
Test_available(glucose-tolerance-test, pa) $\rightarrow$ setup(GDM-TYPE II, pa)
Activate(GDM-TYPE II, pa) /* automatic activation */
Delivered(pa) $\rightarrow$ complete(GDM-TYPE II, pa)
State(blood-glucose, high, pa) $\rightarrow$ suspend(GDM-TYPE II, pa)
State(blood-glucose, normal, pa) $\rightarrow$ reactivate(GDM-TYPE II, pa)
Insulin-indicator(pa) $\rightarrow$ abort(GDM-TYPE II, pa)

to find correct substitutions for the disjunctions, allowed within Asbru conditions. This will be done by splitting a rule at each disjunction, thereby creating a new rule with identical consequent for each disjunction. As an example, rule $(L1 \wedge L2) \vee (L3 \wedge L4) \rightarrow M$ will be split into two rules $L1 \wedge L2 \rightarrow M$ and $L3 \wedge L4 \rightarrow M$.

- The set of possible consequences of rules we have to consider is small: as we mentioned before, only the rule bases PC is modifiable by the user and only the conditions $C \in ConditionSet$ should be addressed by rules within it. This means that PC should only contain rules R with $conseq(R) = C$. As Asbru's Model of Plan States defines seven different conditions, each PC_i will usually contain around seven rules with different consequences, even though the actual number of rules may be slightly lower or higher: it may be lower, as the conditions are optional plan elements. It may be slightly higher, as a condition may include disjunctions and would then be split into several rules. Evidently, it will be easy to guarantee that PC only contains rules referring to valid Asbru conditions: one might provide some sort of input support to the user or even apply manual verification due to the small number of rules.
- The amount of rules we have to consider when checking anomalies within one plan is limited to the scope of the plan's hierarchy: plans of different hierarchies are independent, no anomalies can result from relationships between their rules.
- The possible sequences (rule ordering), in which the rules of any PC_i are considered, is preset by the rule bases MPS. In contrary to "general" RB, we can, therefore include the order of inference into our considerations. This allows us to include some dynamic aspects of Asbru plans into our static verification process: although, we will not actually execute a plan to verify it, RB MPS gives us the opportunity to take the control flow of a plan during execution into account.

4.2.2. Anomalies

In the following, we will list all anomalies concerning Asbru conditions and the corresponding specifications they violate. The anomalies are organized according to the levels in which they may occur (compare Table 1). The underlying specifications may be seen as an extension of existing verification work, achieved through the integration of Asbru specific language features such as its Model of Plan States (see Fig. 1).

Apart from their potential occurrence in original, generic guidelines, anomalies may be caused by inadequate specialization of guidelines: anomalies may be introduced by adaptation of guidelines to the specific characteristics of a receiving patient, e.g. when tuning certain thresholds of single conditions for a particularly sensitive patient (level 1).

This may also damage the correct interaction with other conditions within the same plan (level 2). Level 3 anomalies may be caused by guideline modifications due to organizational limitations: here a typical strategy will be to exchange whole, unsuitable plans with others that pursue compatible goals. Such replacements may interfere with the correct interaction with other plans of the same hierarchy.

4.2.2.1. Level 1: unsatisfiable conditions. Any single condition being part of a plan must have a chance to be satisfied during execution of the plan in order to have an influence on the plan's behavior. Referring to the above rule bases, we equivalently specify that each rule of PC must be fireable.

Formally: $\text{fireable}(R, PC) \forall R \in PC$.

The violation of this specification is a special case of Preece and co-workers *redundant rule* [21]. It may for example originate from medically implausible conditions (e.g. domain or type violations), corresponding to the *illegal attribute values anomaly* described in [19]. Another possibility would be a condition that contains a conjunction of incompatible parameter values.

Example: $\text{male}(pa) \wedge \text{pregnant}(pa) \rightarrow \text{filter}(\text{PlanX}, pa)$.

4.2.2.2. Level 1: redundant parameter-value pairs within conditions. Asbru conditions may contain conjunctions and disjunctions of parameter-value pairs. Each of them should check for some additional data, redundant tests would not make sense. This means we should avoid conditions containing several identical or entailing parameter-value pairs.

A conjunction of entailing parameter-value pairs within a condition is equivalent to the *redundant literal* anomaly defined in [21]. We stated that each rule is of the form $R = L1 \wedge \dots \wedge Ln \rightarrow M$.

Then we request formally: $\neg ((Li \rightarrow Lj) \wedge (i \neq j) \forall (R \in PC; 1 \leq i, j \leq n))$.

We mentioned before that a condition consisting of disjunctive parameter-value pairs would be split into several rules in RB PC , one split for each disjunction. We will, therefore have to check within each PC_i that there is no pair of rules subsuming each other, corresponding to the *subsumed rule* anomaly defined within [22].

Formally: $\neg ((\exists \sigma) (R \rightarrow R'\sigma) \forall (R, R' \in PC_i; 1 \leq i \leq n))$.

Below, a possible representative of this anomaly is shown.

Example: $\text{higher}(\text{cholesterol}, 200, pa) \wedge \text{higher}(\text{cholesterol}, 220, pa) \rightarrow \text{filter}(\text{PlanX}, pa)$.

4.2.2.3. Level 2: unreachable, valid sequence of plan states. Any sequence of plan states, which defines a valid path according to Asbru's Model of Plan States, must also statically be possible within a single plan. Consequently, all conditions belonging to a valid path of plan states must be satisfiable. Otherwise, one or more states of the plan would not be reachable.

For every rule contained in every PC_i , we demand that there must exist a patient for whom the rule is fireable, considering all rules in PC_i , which have fired before for the same patient. For each rule R in PC_i , the set of rules, which must fire before considering R is defined through MPS . We can, therefore specify that for each plan i there must exist an environment E , such that each consequence in MPS can be inferred by supplying E as input to $MPS \cup PC_i$.

Formally: $(\exists \sigma) \text{inferable}(\text{conseq}(R)\sigma, \text{MPS} \cup \text{PC}_i) \forall (R \in \text{MPS}; 1 \leq i \leq n)$.

If any two conditions of a valid state path happen to be incompatible, the plan state of the later checked condition cannot be reached, and the plan contains an anomaly.

Example: plan state *ready* is not inferable for PlanX below.

$R_x: \text{male}(pa) \rightarrow \text{filter}(\text{PlanX}, pa);$

$R_y: \text{pregnant}(pa) \rightarrow \text{setup}(\text{PlanX}, pa).$

4.2.2.4. Level 2: ambiguous state transition. Each time a state transition takes place during the execution of a plan, it must be clear which will be the target state of the transition, as a plan may only have one active state at a time. Conditions, that fork the state path within the Model of Plan States (e.g. the *complete*–, *suspend*– and *abort*–condition in Fig. 1) are the source of potential problems: if more than one of these conditions becomes true at the same time, the successor plan state will be ambiguous.

An intuitive solution would be to request *mutual exclusion* from forking-conditions. However, this strategy is probably unrealistic: we would impose a too harsh restriction on a plan designer by demanding that each set of forking-conditions had to be mutually exclusive. Especially, if the conditions are complex, it will be annoying for the designer to be forced to design the conditions in a way that they exclude each other. A more relaxed strategy would be to demand that forking-conditions should be independent in a way that their concurrent satisfaction is at least not statically foreseeable. In other words we could specify that the antecedents of forking-conditions must not entail each other. As this demand does not rule out a concurrent satisfaction of two forking-conditions, we must provide a heuristic to determine the proper target-state in a conflict situation of this kind.

For the purpose of the definition of the following anomaly, we will interpret all successor states of a state path fork as *semantic constraint expressions*, meaning that their concurrent occurrence does not make semantic sense (see Section 4.2.3).

$\text{ConstraintsSet} = \{(\text{possible}(pl, pa), \text{rejected}(pl, pa)), (\text{ready}(pl, pa), \text{rejected}(pl, pa)), (\text{suspended}(pl, pa), \text{aborted}(pl, pa)), (\text{suspended}(pl, pa), \text{completed}(pl, pa)), (\text{completed}(pl, pa), \text{aborted}(pl, pa)), (\text{activated}(pl, pa), \text{aborted}(pl, pa))\}.$

In order to avoid statically predictable, ambiguous state transitions, we can request that there must not be any pair of rules R and R' in $\text{MPS} \cup \text{PC}_i$, such that the antecedent of R entails the antecedent of R' , and their consequents infer a semantic constraint expression from ConstraintsSet .

Formally: $\neg ((\exists \sigma) (\text{antec}(R)\sigma \rightarrow \text{antec}(R')\sigma) \wedge (\{\text{conseq}(R)\sigma, \text{conseq}(R')\sigma\} \in \text{ConstraintsSet}) \forall (R, R' \in \text{MPS} \cup \text{PC}_i; 1 \leq i \leq n))$.

This corresponds to the *ambivalent rule pair* anomaly in [22]. Below, a possible representative of this anomaly is shown, where it is not clear whether PlanX should be suspended or aborted if the patient's bilirubin level is greater or equal 15.

Example: $R_x: \text{higher}(\text{bilirubin}, 5, pa) \rightarrow \text{suspend}(\text{PlanX}, pa);$

$R_y: \text{higher_equal}(\text{bilirubin}, 15, pa) \rightarrow \text{abort}(\text{PlanX}, pa);$

$R_z: \text{lower_equal}(\text{bilirubin}, 1, pa) \rightarrow \text{complete}(\text{PlanX}, pa).$

4.2.2.5. Level 3: inability to complete. For the determination of a plan's ability to complete, it is necessary to check whether the *complete*–condition and all predecessor conditions can be satisfied for all plans, belonging to the plan's *continuation set*.

Using the abbreviation CS_i for the *continuation set* of plan i , we specify: for each plan i there must exist an environment E such that the consequence of rule $R10 \in MPS$ can be inferred by supplying the *same* E as input to $MPS \cup PC_k$ for all $k \in CS_i$.

Formally: $((\exists E \in PatientSet, \sigma) \text{ infer}(conseq(R10)\sigma, MPS \cup PC_k, E)) \forall (1 \leq i \leq n; k \in CS_i; R10 \in MPS)$.

We distinguish two scenarios that prevent a plan's completion:

- *Incompatible conditions within the plan itself*: this kind of anomaly has to be checked for in level 2 and is already covered by the unreachable, valid sequence of plan states anomaly.
- *Incompatible conditions within two or more plans that belong to the continuation set of the same plan*: all plans, belonging to the continuation set of a certain PlanX, have to complete as a prerequisite for the completion of PlanX. Therefore, no incompatibility must occur in a set of conditions, required to reach the complete state for all of these plans.

Example: $CS_{PlanA} = \{PlanAa, PlanAb\} /^* \text{ continuation set of PlanA } /^*$.

R_x : *blood-group*(pa, A) $\rightarrow$ *filter*(*PlanAa*, pa);

R_y : *blood-group*(pa, B) $\rightarrow$ *filter*(*PlanAb*, pa).

The *filter* – preconditions of plans PlanAa and PlanAb, corresponding to rules R_x and R_y , are incompatible and both plans belong to the *continuation set* of their parent PlanA. Therefore, PlanA will not be able to complete.

4.2.2.6. Level 3: termination enforced by parent. As explained in Section 3.2, the plan states rejected, aborted, suspended and completed may be propagated from a plan to its subplans, thereby stopping the latter. This kind of overruling a plan's actual state should not be the ordinary case, and it should, therefore be assured, that the relevant conditions avoid such a situation. As state propagation might be deliberately applied in some cases, the violation of this specification is not considered an error, but a warning. In terms of our KB we specify:

SS_i = set of all subplans of plan i ;

$RS = \{R3, R5, R6, R9, R10, R13\} \subseteq MPS$: rules with a consequent $\in FinalStateSet$;

$R11 \in MPS$: rule with consequent *suspended*(pl, pa).

Then we demand for each plan i : whenever the consequent of a rule R from FSR can be inferred for a certain environment E , then it must also be possible to infer the consequent of a rule R' from FSR for each of plan i 's subplans for the same environment E . The same holds for rule $R11$ from MPS .

Formally: $\text{infer}(conseq(R)\sigma, MPS \cup PC_i, E) \rightarrow \text{infer}(conseq(R')\sigma, MPS \cup PC_k, E) \forall (1 \leq i \leq n; R, R' \in FSR; k \in SS_i; E, \sigma)$.

$\text{infer}(conseq(R11)\sigma, MPS \cup PC_i, E) \rightarrow \text{infer}(conseq(R11)\sigma, MPS \cup PC_k, E) \forall (1 \leq i \leq n; E) \forall (1 \leq i \leq n; R11 \in FSR; k \in SS_i; E, \sigma)$.

In the anomaly below, the *abort* – conditions of parent PlanA (rule R_x) does not entail the *abort* – conditions of child PlanAa (rule R_y). Consequently, there is a chance that the parent will override its child's true state with the state aborted.

Example: R_x : *higher*(*bilirubin*, 5, pa) $\rightarrow$ *abort*(*PlanA*, pa);

R_y : *higher*(*GOT*, 22, pa) $\rightarrow$ *abort*(*PlanAa*, pa).

4.2.3. Functionality of the verification knowledge base

We assume that our verification method has access to a domain-specific KB, which can be queried for the following information:

- *Incompatibility of conditions.* The KB defines which findings cannot occur simultaneously for the same patient. The concept of incompatibility is equivalent to the *semantic constraint expression* used in [21]: a semantic constraint expression is an expression $\{L1, \dots, Ln\}$, which is interpreted as meaning that the simultaneous truth of $L1 \wedge \dots \wedge Ln$ would not make semantic sense. For example, the set $\{\text{blood-group-A}(x), \text{blood-group-B}(x)\}$ says that, for all x , x cannot have blood-groups A and B at the same time.
- *Entailment of conditions.* Entailment is not only restricted to conditions concerning the same parameters. In the medical domain there is a high number of dependencies between different parameters, that may be the source of non-trivial entailment (e.g. parameter *gender* and pregnancy-related parameters).
- *Attributes of medical parameters.* For each medical parameter that might be used in an Asbru plan, the KB contains information about its value domain and its type.

5. Scenario-based evaluation

After the theoretical foundation of our verification method has been illustrated, we will in the following describe a scenario of its application. Object of the analysis will be an exemplary guideline for the artificial ventilation of newborn infants, shown in Fig. 3.

The top-level plan is called infants respiratory distress syndrome therapy (*I-RDS-therapy*). It consists of four subplans that are decomposed into further subplans. Fig. 4 shows an excerpt of these plans, coded in Asbru and including different anomalies.

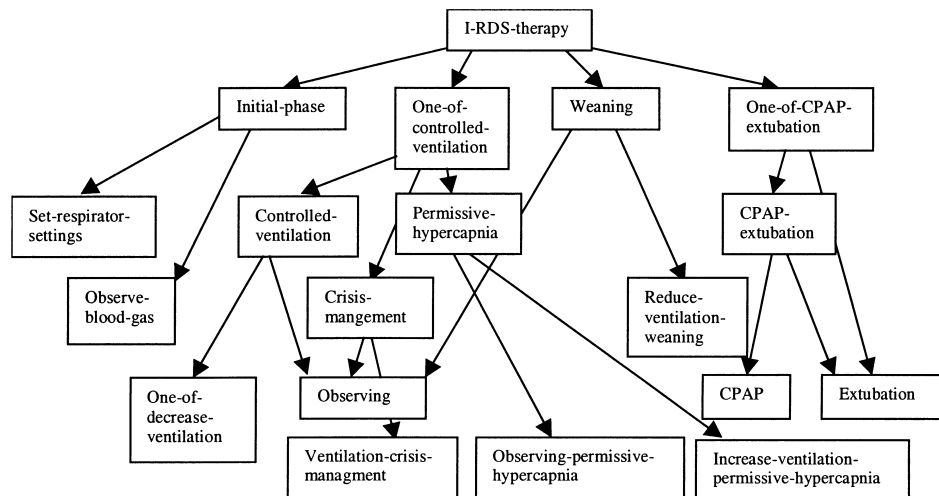


Fig. 3. Plan hierarchy of guideline for artificial ventilation of newborn infants, suffering from infant respiratory distress syndrome (I-RDS).

```

(PLAN one-of-controlled-ventilation
  (COMPLETE-CONDITIONS (PIP (< 20) I-RDS *now*))
  (DO-SOME-ANY-ORDER
    (controlled-ventilation)
    (permissive-hypercapnia)
    (crisis-management)
    CONTINUATION-CONDITION controlled-ventilation))

(PLAN controlled-ventilation
  (INTENTION:INTERMEDIATE-STATE
    (MAINTAIN STATE(BG) NORMAL controlled-ventilation *))
  (SETUP-PRECONDITIONS (PIP (<= 30) I-RDS *now*))
  (ABORT-CONDITIONS ACTIVATED
    (OR (PIP (> 30) controlled-ventilation *)
        (PIP (> 40) controlled-ventilation *)))
  (COMPLETE-CONDITIONS
    (FiO2 (<= 50) controlled-ventilation
      [[_, _], [_, _], [180 MIN, _], *self*]))
    (f (<= 60) controlled-ventilation
      [[_, 2 WEEKS], [5 WEEKS, _], [_, _], BIRTH]))
  (DO-ALL-SEQUENTIALLY
    (one-of-increase-decrease-ventilation)
    (observing)))

(PLAN permissive-hypercapnia
  (INTENTION:INTERMEDIATE-STATE
    (ACHIEVE STATE(BG) NORMAL controlled-ventilation
      [[_, _], [_, _], [_, 30 Min], *self*] 0.9))
  (SETUP-PRECONDITIONS
    (PIP (>= 30) controlled-ventilation *now*))
  (SUSPEND-CONDITIONS (PIP (< 5) I-RDS *now*))
  (COMPLETE-CONDITIONS (PIP (< 25) I-RDS *now*))
  (DO-ALL-SEQUENTIALLY
    (increase-ventilation-permissive-hypercapnia)
    (observing-permissive-hypercapnia)))

```

Fig. 4. Fragment of I-RDS plans.

Anomalies, detected in checking-level 1:**Redundant parameter-value pairs in ABORT-CONDITIONS of****plan** controlled-ventilation :

```
(PIP (> 40) controlled-ventilation *)
entails
(PIP (> 30) controlled-ventilation *)
```

Anomalies, detected in checking-level 2:**Target state of transition ambiguous in****plan** permissive-hypercapnia :

```
(SUSPEND-CONDITIONS (PIP (< 5) I-RDS *now*))
entails
(COMPLETE-CONDITIONS (PIP (< 20) I-RDS *now*))
```

Anomalies, detected in checking-level 3:**Plan one-of-controlled-ventilation may stop****child plan** controlled-ventilation :

```
(COMPLETE-CONDITIONS (PIP (< 20) I-RDS *now*))
does not entail
(COMPLETE-CONDITIONS
  (FiO2 (<= 50) controlled-ventilation
    ([_, _], [_, _], [180 MIN, _], *self*))
  (f (<= 60) controlled-ventilation
    ([_, 2 WEEKS], [5 WEEKS, _], [_, _], BIRTH)))
```

Fig. 5. Exemplary output of verifying plan-hierarchy *I-RDS-therapy*.

Fig. 5 shows an exemplary output for the analysis of the above plan-hierarchy that may be generated by the verifier.

6. Conclusion

Within this paper a method has been presented, which allows for the verification of clinical guidelines represented in a computer-readable format, in a three levels process. The ability of automated verification strongly supports the adaptation of generic guidelines to organization- or patient-specific characteristics, as the latter process is susceptible to the

introduction of flaws in the original guideline's logic. The verification of a clinical guideline is done by checking it for the occurrence of a certain set of predefined anomalies. In the identification process of these anomalies we oriented on one particular language for the representation of clinical guidelines, called Asbru. In this work we have focused on anomalies concerning the Asbru language element *condition*.

Our approach is reusable in two ways: first, it is applicable for the verification of numerous guideline-representation formats, other than Asbru. This is due to the fact that our method is based on a hierarchical organization of guidelines and the usage of conditions to control the guideline's execution flow. Both concepts are common in most current approaches. Second, our approach can be ported to other domains than guideline-based care, as the Asbru language is suitable for several areas of planning [16]. For the reuse of our verification approach in those other areas, it would only be necessary to provide the corresponding, domain-specific knowledge to adapt our KB component.

Another advantage of our method is given by the fact that it allows a significant limitation of the computational effort required. Instead of examining the whole guideline library for the detection of an anomaly, the search can be limited to a single guideline hierarchy. Besides the computational effort also the verification's complexity is reduced. This is reached by means of the information hiding process of decomposing a guideline into its components and performing stepwise, local verification. A final benefit of our organization of the verification process into three separate levels is revealed by the fact that it prepares us for an incremental verification process: adding a new, locally verified plan (levels 1 and 2) to an already verified plan hierarchy for example only requires the repetition of level 3 checks within the extended hierarchy.

After outlining the advantages of our approach we will also comment on its weaknesses: our verification method is designed for the detection of static anomalies within the guideline code. Although, we include dynamic aspects of Asbru plans into the verification process by considering the Asbru MPS, our method does not support the analysis of executing guidelines. A second limitation of our method, which is, however typical for all verification approaches that rely on anomaly detection, lies in its inability to guarantee a totally correct guideline. This shortage originates from the fact that the set of anomalies to be considered are identified in a heuristical process. Therefore, one can never be sure that really all anomalies that may possibly occur within a guideline have actually been handled.

However, even though we cannot provide a complete list of all different anomalies, it is undoubtedly useful to know that a certain range of anomalies will be uncovered by the described verification approach. By applying this strategy, we find ourselves in conformity with the domain of error-based testing, where the goal of running tests is not to show that the program is free from all errors but rather that the program is free from certain well-defined types of errors [10].

Acknowledgements

The authors thank Klaus Hammermüller, Robert Kosara, Andreas Seyfang and Yuval Shahar, who contribute to the development of the project. We are also grateful to Michael Balser, Wolfgang Reif, Annette ten Teije and Frank van Harmelen for

their valuable suggestions. The Asgaard project is supported by “Fonds zur Förderung der wissenschaftlichen Forschung” (Austrian Science Fund), P12797-INF.

References

- [1] Adrion W, Branstad M, Cherniavsky J. Validation, verification and testing of computer software. *Comput Rev* 1982;14(2):159–92.
- [2] Advani A, Lo K, Sahar Y. Intention-based critiquing of guideline-oriented medical care. In: *Proceedings of the AMIA Annual Symposium 98 Orlando, FL, 1998*, p. 483–7.
- [3] Barnes M, Barnett G. An architecture for a distributed guideline server. In: *Proceedings of the 19th Annual Symposium on Computer Applications in Medical Care*. Philadelphia: Hanley and Belfus, 1995. p. 233–7.
- [4] Duftschmid G. Knowledge-Based Verification of Clinical Guidelines by Detection of Anomalies, PhD Thesis. Vienna University of Technology, Vienna, 1999.
- [5] Field M, Lohr K. Clinical Practice Guidelines: Directions for a New Program. Institute of Medicine, Washington (DC): National Academy Press, 1990.
- [6] Fox J, Johns N, Rahmzadeh A. Protocols for medical procedures and therapies: a provisional description of the PROForma language and tools. In: *Proceedings of the Sixth Conference on Artificial Intelligence in Medicine Europe (AIME)*, Grenoble, France, 1997, p. 21–38.
- [7] Fridsma D. Representing the work of medical protocols for organizational simulation. In: *Proceedings of the AMIA Annual Symposium*. Orlando, 1998. p. 305–8.
- [8] Fridsma D, Gennari J, Musen M. Making generic guidelines site-specific. In: *Proceedings of the 20th Annual Symposium on Computer Applications in Medical Care*. Philadelphia: Hanley and Belfus, 1996. p. 597–601.
- [9] Fridsma D, Thomsen J. Representing medical protocols for organizational simulation: an information processing approach. *Comput Math Org Theory* 1998;4(1):71–95.
- [10] Hamlet R. Special section on software testing. *Commun ACM* 1988;31:662–7.
- [11] Herbert S, Gordon C, Jackson-Smale A, Renaud S. Protocols for clinical care. *Comput Methods Progr Biomed* 1995;48:21–6.
- [12] Laurent JP. Proposals for a valid terminology in KBS validation. In: *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI-92)*, Vienna, Austria, 1992, p. 829–34.
- [13] Liem E, Obeid J, Shareck P, Sato L, Greenes R. Representation of clinical practice guidelines through an interactive world-wide-web interface. In: *Proceedings of the Annual Symposium on Computer Applications in Medical Care (SCAMC-95)*. New Orleans (LA), 1995.
- [14] McCormick K, Moore S, Siegel R. Clinical practice guideline development: methodology perspectives. Rockville (MD): AHCPR Publication No. 95-0009, Agency for Health Care Policy and Research, 1994.
- [15] McDonald C, Overhage J. Guidelines you can follow and trust: an ideal and an example. *J Am Med Assoc* 1994;271(11):872–3.
- [16] Miksch S, Shahar Y, Johnson P, Asbru: A task-specific, intention-based and time-oriented language for representing skeletal plans. In: *Proceedings of the Seventh Workshop on Knowledge Engineering: Methods and Languages (KEML-97)*. UK: Milton Keynes, 1997.
- [17] Miller DWJ, Frawley SJ, Miller PL. Using semantic constraints to help verify the completeness of a computer-based clinical guideline for childhood immunization. *Comput Methods Progr Biomed* 1999;58(3):267–80.
- [18] Musen M, Rohn J, Fagan L, Shortliffe E. Knowledge engineering for a clinical trial advice system: uncovering errors in protocol specification. *Bull du Cancer* 1987;74(291):296.
- [19] Nguyen T, Perkins W, Laffey T, Pecora D. Knowledge base verification. *Artif Intell Mag* 1987;2(2):69–75.
- [20] Ohno-Machado L, Gennari J, Murphy S, Jain N, Tu S, Oliver D, Pattison-Gordon E, Greenes R, Shortliffe E, Barnett G. The guideline interchange format: a model for representing guidelines. *J Am Med Assoc* 1998;5(4):357–72.
- [21] Preece A, Batarekh A, Shinghal R. Verifying rule-based systems. *Knowledge Eng Rev* 1992;7(2):115–41.
- [22] Preece A, Shinghal R. Foundation and application of knowledge base verification. *Int J Intell Syst* 1994;9(8):683–702.

- [23] Quaglini S, Saracco R, Stefanelli M, Fassino C. Supporting tools for guideline development and dissemination. In: *Proceedings of the Sixth Conference on Artificial Intelligence in Medicine Europe (AIME)*, Grenoble, France, 1997, p. 39–50.
- [24] Shahar Y, Miksch S, Johnson P. The Asgaard project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines. *Artif Intell Med* 1998;14:29–51.
- [25] Sherman E, Hripesak G, Starren J, Jender R, Clayton P. Using intermediate states to improve the ability of the Arden syntax to implement care plans and reuse knowledge. In: *Proceedings of the Annual Symposium on Computer Applications in Medical Care (SCAMC-95)*, New Orleans, LA, 1995, p. 238–42.
- [26] Shiffman R. Representation of clinical practice guidelines in conventional and augmented decision tables. *J Am Med Inform Assoc (JAMIA)* 1997;4:382–93.
- [27] Shiffman R, Greenes R. Improving clinical guidelines with logic and decision-table techniques. *Med Decision Making* 1994;14:245–54.
- [28] Shwe M, Tu S, Fagan L. Validating the knowledge base of a therapy planing system. *Methods Inform Med* 1989;28:36–50.
- [29] Tierney W, Overhage J, Takesue B, Harris L, Murray M, Vargo D, McDonald C. Computerizing guidelines to improve care and patient outcomes: the example of heart failure. *J Am Med Inform Assoc (JAMIA)* 1995;2:316–22.
- [30] Tu S, Kemper C, Lane N, Carlson R, Musen M. A methodology for determining patients' eligibility for clinical trials. *Methods Inform Med* 1993;32:317–25.
- [31] Tu S, Musen M. The EON model of intervention protocols and guidelines. In: *Proceedings of the AMIA Annual Fall Symposium (Formerly SCAMC)*. Washington (DC), 1996, p. 587–91.
- [32] Uckun S. Instantiating and monitoring skeletal treatment plans. *Methods Inform Med* 1996;35:324–33.
- [33] Van der Lei J, Musen M. A model for critiquing based on automated medical records. *Comput Biomed Res* 1994;24(2):344–78.
- [34] van Harmelen F. Applying rule-base anomalies to KADS inference structures. *Decision Support Syst* 1998;21(4):271–80.

Verification of temporal scheduling constraints in clinical practice guidelines

Georg Duftschmid¹⁾, Silvia Miksch²⁾, Walter Gall¹⁾

1) University of Vienna, Department of Medical Computer Sciences

Spitalgasse 23, A-1090 Vienna, Austria

{georg.duftschmid,walter.gall}@akh-wien.ac.at, www.akh-wien.ac.at/imc/

2) Vienna University of Technology, Institute of Software Technology

Favoritenstraße 9-11/188, A-1040 Vienna, Austria

silvia@ifs.tuwien.ac.at, www.ifs.tuwien.ac.at/

Address requests for reprints and correspondence to:

Georg Duftschmid

University of Vienna, Department of Medical Computer Sciences,

Spitalgasse 23, A-1090 Vienna, Austria,

Tel.: +43-1-40400 / 6696

Fax.: +43-1-40400 / 6697

Email: georg.duftschmid@akh-wien.ac.at

Verification of temporal scheduling constraints in clinical practice guidelines

Abstract

The computerization of clinical practice guidelines is a significant scientific challenge for the medical informatics community. One frequently reported factor hindering this objective is the existence of deficiencies within guideline knowledge. In this paper, we focus on the detection of flaws within temporal scheduling constraints. Temporal scheduling constraints are important elements of therapy management, and are frequently incorporated in clinical practice guidelines. We present a suitable verification method that is based on calculating the minimal network of temporal constraints on the execution of guideline activities. Our method serves three purposes: (1) it checks whether temporal scheduling constraints are consistent with scheduling constraints implied by control flow operators and the hierarchical structuring of a guideline; (2) it yields suggestions for an equivalent, yet more explicit representation of non-minimal constraints; (3) it can be used by the guideline interpreter to assemble feasible time intervals for the execution of each guideline activity. We evaluate our approach by applying it to a guideline specified in the Asbru language. For this purpose, we implemented a prototype verifier. Although we concentrate on the guideline representation language Asbru as the demonstration medium of our method within this paper, our approach can be reused to verify several alternative guideline-representation formats.

Keywords:

Clinical practice guidelines, medical plan management, verification, temporal constraint satisfaction

1 Introduction

The computerization of clinical practice guidelines and the variety of tasks associated with this objective have been the subject of numerous research projects performed over the last decade in the medical informatics community [5, 9, 18, 19, 24, 25]. In comparison with the extensive research efforts invested in this subject, few guidelines have actually been employed in an automated version in clinical practice. One frequently reported factor that complicates the implementation of such applications is the existence of deficiencies within guideline knowledge [12, 17, 32]: As safety and transparency are critical issues in the medical domain, guidelines that might lead to inappropriate treatment of the patient as a result of ambiguous or even incorrect instructions are not acceptable for implementation.

In software engineering in general and in knowledge-based systems in particular, a common strategy to ensure the correctness of a system is its verification. As clinical guidelines encode a multitude of different types of knowledge (e.g., *goals*, *conditions*, *effects*), all of which might be used incorrectly, the application field for verification is broad in this domain. One such type of knowledge is the concept of *temporal scheduling constraints* within a guideline. Temporal scheduling constraints are important elements of therapy management, and are frequently incorporated in clinical practice guidelines, either explicitly or implicitly [14]. They are especially important when the guidelines are used in critical contexts, for instance when physicians have to schedule different drugs regimens and the drugs need to be taken in specific temporal windows. As the correct timing of guideline activities is a basic necessity for the safe application of a guideline, temporal scheduling constraints constitute a relevant subject for verification.

This paper will present an approach that permits verification of guidelines during their design phase by checking whether the following three types of scheduling constraints are consistently applied:

1. Temporal scheduling constraints
2. Scheduling constraints implied by the guideline's control flow
3. Scheduling constraints implied by a hierarchical structuring of a guideline

As we will show, scheduling constraints of the above mentioned three types have been integrated to a considerable extent in the majority of current guideline representation formats. We will refer to the Asbru language as the demonstration medium of our approach, as it provides the most powerful functionality in temporally constraining guideline execution. However, our work is reusable within

other guideline representations to the degree that they integrate temporal scheduling constraints.

The presented approach is based on calculation of the minimal temporal network and serves three purposes: (1) it allows identification of inconsistent temporal scheduling constraints; (2) it examines temporal scheduling constraints for their minimality and yields the most explicit equivalent representation as a suggestion for the adaptation of non-minimal temporal scheduling constraints; and (3) it can be used by the guideline interpreter to determine feasible time intervals for the execution of each guideline activity.

The paper is organized as follows: In section 2 we will discuss existing work in regard of guideline verification. In section 3 we will examine the extent to which scheduling constraints are integrated in current guideline representation formats: After providing an overview on their implementation in several important formats in section 3.1, we will explicitly examine the plan specification language Asbru in this regard in section 3.2. The verification of scheduling constraints within a clinical guideline will be handled in section 4: In section 4.1 we will show how our verification task corresponds to a simple temporal problem. The properties to be checked for a guideline in the course of our verification process will be described in section 4.2. In sections 4.3 and 4.4 we will examine the possible origins of inconsistencies and non-minimalities of temporal scheduling constraints. Two factors that complicate verification will be discussed in sections 4.5 and 4.6, namely Asbru's operator for unordered, sequential plan execution and the integration of multiple time lines. The method we use to verify temporal scheduling constraints will be described in section 5. It detects inconsistent and non-minimal temporal scheduling constraints and yields a suggestion for the modification of non-minimal temporal scheduling constraints. After presenting some results on evaluating our approach in section 6, a conclusion will be presented in section 7.

2 Related work on the verification of guidelines

Compared with the number of presented approaches for the computerization of clinical guidelines and the different associated tasks that have been tackled, the verification of clinical guidelines is poorly reported in the literature. Shiffman and Greenes [27] used logical analysis and the application of decision table techniques to verify and simplify clinical practice guidelines. Guidelines are regarded as sets of condition-action pairs, and verification consists of proving their *completeness* and *unambiguousness*. A guideline is considered incomplete if it does not provide an action for each possible value combination of parameters used within conditions. It is considered ambiguous if it prescribes different actions for identical value combinations of parameters. Shiffman [26] refined the

to appear in *Artificial Intelligence in Medicine*, 2002.

proposed method by showing how decision tables, representing the guideline's logic, may be augmented by additional information such as cost, risk, evidence or literature citations. Miller et al. [16] further improved Shiffman and Greenes' approach by considering only those value combinations that are semantically possible. Quaglini et al. [22] described how a guideline may be checked for *completeness* and *coherency*. Whereas completeness is defined in an equivalent manner as Shiffman and Greenes have done, checking a guideline for coherency means looking for conjunctive actions, which exclude each other because of incompatible activation conditions. Duftschmid and Miksch [4] proposed a three-step method to verify conditions within a clinical guideline, modeled as a hierarchy of Asbru plans. In that study, a guideline is checked for the occurrence of anomalies, which represent violations of certain properties postulated from a legal guideline's condition set. According to their scope, three types of anomalies are distinguished, which are anomalies *within a single condition*, *within a single plan*, and *within a hierarchy of plans*.

All the above mentioned approaches concern the verification of condition-based clinical guidelines only and do not consider temporal information. Guarnero et al. [8] described an approach for the representation of clinical guidelines, which allows the specification of minimum and maximum durations for individual guideline actions. They use a temporal reasoning system known as LaTeR for examining the consistency of these duration intervals with delay intervals, which may be defined between pairs of actions to be executed in sequence.

Other related work can be found in the domain of temporal constraint propagation, which can be distinguished according to the type of constraints reasoned about:

- Qualitative constraints: Allen's interval algebra [2] and Vilain and Kautz's point algebra [34] are two fundamental representatives.
- Quantitative constraints: This class of constraints was originally studied by Dechter et al. [3].
- Combination of qualitative and quantitative constraints: Two important approaches that integrate both types of constraints were presented by Kautz and Ladkin [11], and Meiri [13].

To tackle our problem we concentrated on the work on combined qualitative and quantitative constraints as well as purely quantitative constraints. We did this for the following reason: In our guideline representation language both types of constraints are possible. Temporal scheduling constraints are quantitative. Scheduling constraints induced by control flow and hierarchical structuring are qualitative. Therefore, work on combined qualitative and quantitative constraints is relevant for our problem. As we will see in sections 4.3.2 and 4.3.3, the spectrum of qualitative constraints we may expect is reduced to the relations "=" and " $\leq$ " between two time points. These

constraints may easily be translated to quantitative ones using the functions described by Meiri [13]. Thus, translating qualitative into quantitative constraints and working only with the latter type was also a reasonable option for us.

Several authors have focused on the propagation of combined qualitative and quantitative constraints in temporal networks: Kautz and Ladkin [11] use separate networks for both types of constraints and solve them independently. In a final step, they circulate information between the two networks using functions for the translation of qualitative into quantitative constraints and vice versa. Meiri's approach [13] equally relies on translation functions between qualitative and quantitative constraints. It differs from Kautz and Ladkin's approach in the way the two types of constraints are connected in the constraint network: Instead of using separate networks, Meiri stores both types of constraints in a single network and performs constraint propagation therein. Rit's approach of propagating temporal constraints in scheduling problems [23] is similar to ours, as it is based on the same structure for representing quantitative temporal scheduling constraints, which is termed *Set of Possible Occurrences* (SOPO). Here the goal is to solve the so-called *constrained occurrences problem*, where a network of SOPOs connected by qualitative constraints is compressed to those occurrences which are compatible with the constraints between them. While inconsistencies are detected, the approach does not ensure minimality of SOPOs. The paper further suggests a two-dimensional graphical representation and propagation of constraints. Although this visualization allows an exact graphical representation of an event with an uncertain start, end, and duration, it suffers from a paucity of intuitive comprehensibility. Stillman et al. [30] further extend Rit's work by allowing parameterized qualitative as well as quantitative constraints to be defined between SOPOs. Vere [33] describes several constraints that must hold within a quantitatively defined *temporal window* of an activity and presents an algorithm for the propagation of qualitative constraints between several windows of sequential and consecutive activities. However, temporal windows here are simpler than our temporal scheduling constraints, as a fixed duration is assumed and uncertainty in the finishing time point is not supported.

Our main point of orientation was the work on the propagation of purely quantitative constraints presented by Dechter et al. [3]. We relied on their work to keep our approach simple: The extra effort of using a combined approach did not seem justifiable merely to support the constraints "=" and " $\leq$ " in their original form within our constraint network. Instead, we chose to replace these constraints by their quantitative counterparts and use a simple and highly efficient approach for the propagation of purely quantitative constraints. The approach presented by Dechter et al. covers all of our requirements, except for unordered sequential execution of guideline activities. This type of control

to appear in Artificial Intelligence in Medicine, 2002.

flow cannot be tackled, as Dechter et al. only support binary constraints. If unordered sequential activities are to be considered, approaches that permit the representation of disjunctive non-binary temporal constraints such as the frameworks described by Stergiou and Koubarakis [29] or Staab [28] must be applied. However, the problem of verifying a guideline that includes unordered sequential activities (and thus disjunctive non-binary temporal constraints) is NP hard, as opposed to an effort of $O(n^3)$ otherwise. As Asbru is currently the only representation format that considers unordered sequential activities, we chose to integrate this type of control flow in the verification process only through partial checks. Hereby, the efficiency of our approach is not compromised.

3 Integration of scheduling constraints in guideline representation formats

In this section, we will first give an overview how scheduling constraints are integrated within six important current guideline approaches. As it builds the demonstration medium of our verification method, we will then inspect the guideline specification language Asbru in detail for its integration of scheduling constraints.

3.1 Overview

In the following, we will examine how the guideline representation formats GLIF, PROforma, DILEMMA / PRESTIGE, EON, the Arden Syntax, and Asbru implement (1) temporal scheduling constraints, (2) scheduling constraints implied by the guideline's control flow, and (3) scheduling constraints implied by hierarchical structuring of a guideline.

3.1.1 Temporal scheduling constraints

In its original version [19], the GuideLine Interchange Format (GLIF) included temporal constraints on patient data elements only. Their purpose was to define the required actuality of a data element to be considered within the guideline. However, the current version 3 includes a structured grammar based on the Arden Syntax logic grammar [10]. Among other features, it permits the specification of temporal expressions [21]. This enhancement of its temporal expressiveness is intended to be utilized in the form of duration constraints on actions and decisions.

PROforma [5] models guidelines as plans, which may consist of one or more tasks. Each plan may define temporal constraints on the enactment of tasks, which are specified in the form of time durations between tasks. Further, preconditions may be stated for each task; satisfaction of the preconditions is a prerequisite for the execution of a task. These preconditions are expressed with the

standard *PROforma* syntax, which also includes temporal constructs.

The DILEMMA approach [9] and its successor, the PRESTIGE project [7], allow temporal expressions to be included in conditions, which control the transitions between the different states their guidelines may adopt. Thus, it should be possible to determine a guideline's starting time point by adding temporal constraints to its eligibility criteria.

The EON approach [18] permits the scheduling of guideline steps through its "start_constraint" attribute, which may be a temporal expression. In addition, activities may have duration constraints. By incorporating the RÉSUMÉ system, EON further allows the specification of complex, temporal expressions including temporal abstraction. However, the latter functionality is only used to extend EON's expressiveness when referring to patient data.

Sherman et al. [25] describe how guidelines may be implemented within the Arden Syntax. In their approach, a guideline is represented as a set of Medical Logic Modules (MLMs), which are synchronized by means of so-called *intermediate states*: By storing a specific coded value in the database, an MLM triggers its successor MLMs, which define this particular event in their *evoke slot*. The Arden Syntax allows the specification of a time delay within the *evoke slot*, which builds an offset from the triggering event and has to be awaited before the MLM is actually started [10]. The *time delay* statement may consequently be used to specify minimum duration constraints between individual MLMs.

An approach that emphasizes efficient handling of temporal expressions within guideline knowledge is the *Asbru* language [15], which was created as part of the *Asgaard* project [24]. *Asbru* provides the most powerful functionality in specifying temporal scheduling constraints in comparison with the above mentioned approaches. As we will demonstrate in greater detail in section 3.2.1, *Asbru* allows uncertain specification of a guideline activity's duration and its starting and finishing time points. Furthermore, each temporal scheduling constraint may be based on a local reference time point, which permits the creation of multiple time lines.

3.1.2 Scheduling constraints implied by the guideline's control flow

The integration of scheduling constraints based on a guideline's control flow was already indirectly postulated by Pattison-Gordon et al. in their paper dealing with the requirements of shareable guideline representation [20]. Their demand for an explicit representation of *temporal sequence* addresses the ability to express different variants of control flow within computerized guidelines, such as the sequential, parallel and iterative execution of guideline actions. All of the above mentioned guideline representation techniques have implemented this functionality. We will describe

to appear in *Artificial Intelligence in Medicine*, 2002.

in section 4.3.2 the kinds of scheduling constraints that may result from a guideline's control flow.

3.1.3 Scheduling constraints implied by hierarchical structuring of a guideline

Ohno-Machado et al. [19] specified the hierarchical structuring of guidelines, referred to by them as the *decomposability of actions*, as a representational requirement of the GLIF model. All other representation formats, except the approach based on linking MLMs by intermediate states, equally rely on a hierarchical organization of guidelines. We will describe in section 4.3.3 the kinds of scheduling constraints that may result from the hierarchical structuring of a guideline.

3.2 The plan representation language Asbru

Within the *Asgaard* project, a set of tasks and computational models is investigated with the purpose of supporting the execution of clinical guidelines [24]. As a foundation for the associated task-specific problem-solving methods, the time-oriented, intention-based language *Asbru* has been developed. It uses plans as the basic structure for representing guidelines [15]. Among the full set of Asbru language elements, containing knowledge classes such as *preferences*, *intentions*, *conditions* and *effects*, the *time annotation* and *plan body* constructs are of particular interest in our context.

3.2.1 Time annotations

The general purpose of a time annotation is to constrain the temporal occurrence of plan elements (including plans themselves): The plan element has to start in a certain time interval, which we term the *starting interval* (SI). It has to finish in a time interval, which we term the *finishing interval* (FI). Finally, its duration has to be within certain limits, which we call the *duration interval* (DI).

A time annotation is defined by seven parameters, consisting of the endpoints of the above three intervals and a reference point:

- SI is specified by [*earliest starting shift* (ESS), *latest starting shift* (LSS)].
- FI is specified by [*earliest finishing shift* (EFS), *latest finishing shift* (LFS)].
- DI is specified by [*minimum duration* (minDu), *maximum duration* (maxDu)].
- The upper and lower limits of SI and FI are offsets to the *reference point* (Ref), whereas DI is unrelated to Ref.

Consequently, a complete time annotation is specified through the data structure [[ESS, LSS], [EFS, LFS], [minDu, maxDu], Ref]. For the intervals' domains, the obvious limits are used: DI must be a subset of $[0, \infty)$, whereas SI and FI are subsets of $(-\infty, \infty)$. A graphical

representation of a time annotation on one- and two-dimensional scales is shown in Figure 1: 1D visualization has the disadvantage that it does not allow an unambiguous placement of DI, a problem that can be solved by 2D visualization [23]. Despite this limitation, we will use 1D visualization in the remaining part of this paper for two reasons: First, we found it to be much more intuitive and second, an unambiguous placement of DI is not essential for our purpose.

*** Insert Figure 1 here ***

By defining three time intervals from which the time annotation's starting and ending time points and duration may originate, Asbru permits representation of temporal uncertainty. As each time annotation can further define its local reference point, multiple time lines are permitted. To support incomplete temporal information, parameters may be left unrestricted, denoted by "_". Logically, this corresponds to setting the parameter to the respective bound of its interval's domain, i.e. an unrestricted minDu is set to 0, ESS to $-\infty$, and LFS to ∞ .

Note that none of the intervals is redundant. Any change in an interval's parameters has an influence on the time annotation's characteristics, as long as it stays within the bounds ($[ESS^-, LSS^+]$, $[EFS^-, LFS^+]$, $[minDu^-, maxDu^+]$) given by ($[EFS - maxDu, LFS - minDu]$, $[ESS + minDu, LSS + maxDu]$, $[MAX(0, EFS - LSS), LFS - ESS]$). As we will describe in section 4.1, a time annotation may be regarded as a *temporal constraint network* with three variables for *Ref*, the *start* and the *finish* of an event, and the three intervals building distance constraints between them. In this interpretation, the above restriction on the intervals' bounds corresponds to the properties of a *minimal network*, which provides the most explicit encoding of the involved constraints [3].

If at least one parameter of SI or FI is defined together with a reference point, we say that a *time annotation of a plan element is linked to the time line*. This means that the plan element may not start or finish before or after a certain point in time. If only DI is defined, however, the time annotation is completely detached from the time line. The plan element may be executed at any time, as long as its duration stays within the specified limits.

3.2.2 Plan body

In Asbru, a clinical guideline is modeled as a set of hierarchical plans. As indicated by the term *hierarchical*, each plan may contain any number of subplans within its plan body, which may themselves be decomposed into sub-subplans, and so on. In the remaining part of this paper, such a tree structure of plans, starting from one root plan down to its leaf plans, will be referred to as a *plan hierarchy*; the term will be used as a synonym for *a clinical guideline coded in the Asbru language*. Each plan corresponds to a single guideline activity (e.g., the activity *weaning* is represented as one

to appear in *Artificial Intelligence in Medicine*, 2002.

of several plans within a plan hierarchy for the artificial ventilation of neonates). Note that the root plan of a plan hierarchy constitutes the top-level activity and is usually named after the guideline itself (e.g., plan *artificial ventilation of neonates*).

The control flow within a plan hierarchy is explicitly specified and allows sequential (with defined or undefined order), parallel, cyclic, and arbitrary execution of plans. During run-time, a plan is decomposed into its subplans until a non-decomposable plan – known as action – is found. Actions are executed by the user or by an external call to a computer program. A set of predefined actions exists in order to carry out interaction with the user or to retrieve information from the patient's medical records.

In order to constrain the possible time intervals of its execution, a plan may be associated with a time annotation that defines a *temporal scheduling constraint* for the plan. The resulting data structure will be termed *plan activation* (PA) in the remaining part of this paper. The time interval during which a plan is actually executed will be known as its *execution interval* (EI). Within a PA, a plan may only be associated with *one* time annotation; disjunctions of several time annotations are not allowed.

As an example, the plan activation (GDM-II [[0 WEEKS, 8 WEEKS], [24 WEEKS, _], [18 WEEKS, _], CONCEPTION]) means: "Plan *GDM-II* for the treatment of gestational diabetes mellitus type II should be executed in a time interval, *starting between 0 and 8 weeks after the conception, finishing no earlier than 24 weeks after the conception, and lasting at least 18 weeks.*" There are no restrictions on the latest finishing point and the maximum duration.

Example 1. A plan hierarchy consisting of 11 PAs in 4 levels. Control flow is specified by 5 different operators.

```
(P1 [[_,_],[_],390],[50,370],Ref)
  do-seq-ordered ((P2 [[_,_],[_],_],[70,100],_),
                 (P3 [[_,_],[_],_],[120,150],_),
                 (P4 [[_,_],[_],_],[130,160],_)))

(P2 do-parallel ((P5 [[_,_],[_],_],[90,_,_]),
                 (P6 [[40,_,_],[_],_],[_,_],Ref))))

(P3 do-cyclic ((P7 [[_,_],[_],_],[20,_,_],_)
               retry=[10,_] exec=[5,_])))

(P4 do-arbitrary ((P8 [[_,_],[_],_],[120,160],_),
                  (P9 [[_,_],[_],_],[20,_,_],_)))

(P8 do-seq-unordered ((P10 [[_,_],[_],_],[90,_,_],_),
                     (P11 [[_,_],[_],_],[80,_,_],_)))
```

Example 1 shows a plan hierarchy that covers all three kinds of scheduling constraints examined in this paper, namely (1) temporal scheduling constraints, (2) scheduling constraints implied by the guideline's control flow, (3) scheduling constraints implied by a hierarchical structuring of a guideline: the plan hierarchy consists of eleven plans in four layers, each of which is associated with a specific time annotation and thus constitutes a PA. The execution of plans within the hierarchy is further influenced by five different operators for flow control. The semantics of these operators and the additional parameters of the cyclic operator will be explained in section 4.3.2. The actual guideline actions (such as recommendations to the user) reside in the plan bodies of the leaf plans. As they are not essential for our task, they are omitted in Example 1.

■

4 Verification of scheduling constraints

The PAs within an Asbru plan hierarchy, building a set of temporal scheduling constraints for the involved plans constitute the target of our verification approach. We will show in section 4.1 how our task of verifying PAs can be mapped to a simple temporal problem. In section 4.2 we will then define the properties *consistency* and *minimality*, for which PAs are checked in the verification process. In sections 4.3 and 4.4 we will examine how inconsistencies and non-minimalities may arise within a single PA or within a network of several PAs. Asbru's operator for unordered, sequential execution of plans does not fit into our framework, and will therefore be separately handled in section 4.5. In section 4.6, we will address the problem of multiple time lines within different PAs, which complicates verification.

4.1 Analogy to a simple temporal problem

Our verification task can be mapped to a simple temporal problem: according to Dechter et al., a *simple temporal problem* (STP) is a temporal constraint satisfaction problem in which all involved (quantitative) constraints consist of a single interval [3]. This is a special case of the general *temporal constraint satisfaction problem*, where constraints are disjunctions of intervals.

The constraint network of an STP can be represented by a directed *constraint graph* (CG): Nodes represent variables $X_1, \dots, X_n$ for the time point of the occurrence of certain events, whereas an edge $i \rightarrow j$ indicates that an interval $[a_{ij}, b_{ij}]$ for the temporal distance between the two corresponding events is specified, representing the constraint $a_{ij} \leq X_j - X_i \leq b_{ij}$. Within the constraint network of an STP, no more than one interval per edge is allowed, which means that disjunctions of constraints are ruled out.

to appear in *Artificial Intelligence in Medicine*, 2002.

Each constraint can equally be expressed by a pair of inequalities $X_j - X_i \leq b_{ij}$ and $X_i - X_j \leq -a_{ij}$. This constitutes the basis of the directed *distance graph* (DG), which has the same node set as the CG. Each edge $i \rightarrow j$, however, is labeled by a single weight a_{ij} representing the linear inequality $X_j - X_i \leq a_{ij}$. As an example, on the right side of Figure 2 the weight on the edge from P.S to P.F constrains the distance $P.F - P.S$ to be lower or equal to $maxDu$.

A PA with a fully specified time annotation constrains the start, finish and duration of a plan. As it does not allow disjunctions of time annotations, it satisfies the corresponding demand of an STP. In order to represent a PA as the constraint network of an STP, we decompose it into the three nodes *Start of Plan (P.S)*, *Finish of Plan (P.F)* and *Reference Point (Ref)*. Figure 2 shows the CG and DG corresponding to a PA.

*** Insert Figure 2 here ***

4.2 Properties to be checked

We verify the PAs within an Asbru plan hierarchy by checking them for the following two properties:

- Consistency

We examine whether PAs are consistent with those scheduling constraints within a guideline, which are induced by control flow operators or hierarchical structuring. To cite an example, our goal is to check whether the PAs within a plan hierarchy, such as the one depicted in Example 1, are consistent. The concept of consistency we use is based on the absence of negative cycles within the DG of an STP, as defined by Dechter et al. [3]: We consider a PA *to be consistent with respect to its associated time annotation* if and only if its DG representation does not contain negative cycles. A negative cycle is defined as a cyclic path within the DG, for which the sum of weights a_{ij} (c.f., section 4.1) is negative. In this paper, a PA that is *consistent with respect to its associated time annotation* will be referred to by the shorter term, *consistent PA* only.

- Minimality

We examine whether each PA is *minimal with respect to its time annotation*. Using the abbreviated version *minimal PA* in the remaining part of this paper, we consider a PA to be *minimal* if and only if the constraint network, given by the three intervals of the associated time annotation, is minimal. The concept of minimality of a constraint network is defined the same way as Dechter et al. [3] have done: A constraint network is minimal if it is *tighter* than any

equivalent constraint network. Two networks are *equivalent* if they represent the same solution set, which in our case corresponds to two PAs that allow the same set of EIs for their plans. A network T is *tighter* than a network S if all constraints of T are tighter than the corresponding constraints in S, i.e. constraint T_{ij} is tighter than S_{ij} for all pairs i,j . A constraint T_{ij} , limiting the distance between two variables is tighter than a constraint S_{ij} , if every pair of values allowed by T_{ij} is also allowed by S_{ij} . In other words, a PA is minimal if its time annotation represents the most explicit encoding of the induced constraints.

Inconsistencies are treated as errors that have to be cleared by the user. Non-minimalities, however, only represent sub-optimal specifications of PAs that commonly occur within guidelines: In fact, most PAs will be non-minimal when initially defined by the user, as the latter usually does not want to be concerned with checking each PA for minimality. When derived from real-world constraints, PAs may even be voluntarily kept non-minimal. Therefore, they are only indicated to the user; their correction is not enforced.

4.3 Inconsistencies

As the presence of a negative cycle indicates that an STP is inconsistent, we will examine in the sequel how cycles may arise within one PA or within a network of several PAs. For each identified type of cycle we will examine how it may become negative and thereby introduce an inconsistency. We will start by inspecting a single PA for negative cycles in section 4.3.1. In the examination of a network of PAs, we will distinguish three causes of linking PAs: (1) the specification of control flow in section 4.3.2; (2) parent–child relations in section 4.3.3; and (3) links to the time line in section 4.3.4.

4.3.1 Single plan activation

If we take a look at the DG of a single fully specified PA (see right side of Figure 2), we find that cycles are already identifiable here. First we have three cycles, each enclosing two of the three nodes: ESS and LSS build a cycle between Ref and P.S, EFS and LFS between Ref and P.F, and finally minDu and maxDu form a cycle between P.S and P.F. A trivial kind of inconsistency will occur here when at least one of the following inequalities holds:

$$ESS > LSS, EFS > LFS \text{ or } minDu > maxDu$$

Second, we receive two cycles that span all three nodes of a PA: that comprising ESS, LFS, and minDu, and that comprising LSS, maxDu, and EFS. All other cycles that span all three nodes are

to appear in *Artificial Intelligence in Medicine*, 2002.

concatenations of two two-node cycles and may be correspondingly decomposed. Inconsistencies resulting from the latter two cycles will occur if at least one of the following inequalities holds:

$$\min Du > LFS - ESS \text{ or } EFS - LSS > \max Du$$

Looking for relative positionings of SI and FI that can be excluded a priori as being inconsistent, the latter two inequalities give us the following trivial insight: considering that the minimum value for $\min Du$ is 0, the first inequality tells us that an FI, that is in relation *before* to SI, is always inconsistent. The second leaves any position of SI relative to FI open, as $\max Du$ is unbound on the upper end.

4.3.2 Specification of control flow

As mentioned in section 3.2, the Asbru language provides operators that allow the user to explicitly define the order in which plans are to be executed. We will now examine what kind of constraints between PAs are imposed by the semantics of these operators for flow control.

*** Insert Figure 3 here ***

Sequential execution of plans with defined order

Asbru provides an operator for sequential, ordered execution of plans with common semantics: Every plan, activated as the $(i+1)^{\text{th}}$ member of a set of sequential plans with defined order, may not be started before the plan corresponding to the i^{th} member is completed. In other words, a sequential ordered execution of plans P1 and P2 implies the qualitative constraint $P1.F \leq P2.S$ between the finish of P1 and the start of P2. This translates to the quantitative constraint $[0, \infty)$ between time points P1.F and P2.S [13]. Figure 3 (a) visualizes this circumstance in the CG and DG. The maximum distance of ∞ is omitted in the DG, as it does not represent a constraint.

Parallel execution of plans

The Asbru operator for parallel execution of plans has the following semantics: Parallel plans have to start concurrently, whereas no restriction is made for their finishing time point. This entails the qualitative constraint $P1.S = P2.S$ between the starting time points of parallel plans P1 and P2, which translates to a maximum (and therefore also minimum) distance of 0 between these points. Figure 3 (b) is a graphical representation of this constraint for two parallel plans. The edges are presented in diagonal orientation to optically distinguish them from *parent-child* relationships.

Arbitrary execution of plans

The operator for arbitrary execution of plans does not impose any constraint on the order: Plans may be executed in parallel, overlapping or sequential fashion. Therefore, the distance between them cannot be restricted in any way. For reasons of completeness, Figure 3 (c) visualizes the unconstrained distance between the start and the finish of two arbitrary executed plans.

Cyclic execution of plans

The Asbru language offers several ways to specify a repetitive execution of a single plan. All these alternatives are extensions of the following scheme: A regular PA is enhanced by a retry-delay and a lower (*minExec*) as well as an upper limit (*maxExec*) for the number of plan executions. The retry-delay is given by an interval that defines the minimum (*minDelay*) and maximum waiting period (*maxDelay*) between two plan executions. Figure 3 (d) shows the DG of the cyclic execution of a single plan. The upper graph gives a static view of the cyclic plan, whereas the lower graph shows one particular instance in detail. As the retry-delay and number of plan executions can be defined by the user, they are susceptible to inconsistencies. We mentioned in section 4.3 that a PA is inconsistent if $minDu > maxDu$. Similarly, a cyclic PA is inconsistent if $(minDu \cdot minExec) + (minDelay \cdot (minExec - 1)) > (maxDu \cdot maxExec) + (maxDelay \cdot (maxExec - 1))$.

4.3.3 Parent–child relations

As stated in section 3.2, Asbru organizes guidelines in a tree structure of plans known as a *plan hierarchy*. A plan hierarchy may consist of several levels, and each non-leaf plan has a *parent* $\rightarrow$ *child* relationship with its subplans on the next lower level. Using the obvious approach of parent–child synchronization, Asbru does not allow child plans to start before their parent or to finish after their parent. For a parent plan P and a child plan C, this entails the qualitative constraints $P.S \leq C.S$ and $C.F \leq P.F$ between their starting and finishing time points. This translates to a minimum distance of 0 between P.S and C.S and a minimum distance of 0 between the C.F and P.F [13].

*** Insert Figure 4 here ***

Figure 4 (a) visualizes this circumstance in the CG and DG. Clearly, the PAs of plans P and C will be inconsistent if $C.minDu > P.maxDu$.

If we combine the constraints originating from control flow with those originating from parent–child relationships, we receive less obvious possibilities for inconsistencies: Figure 4 (b) depicts a scenario where plan P1 has two child plans P2 and P3, which are executed *sequentially* in order P2, P3. Plan P2 itself is parent of the *cyclic* plan P4. The scenario in Figure 4 (b) contains three different cycles. Cycle 1 between P2 and P4, shown in dark, may induce an inconsistency between the PAs of P2 and

to appear in *Artificial Intelligence in Medicine*, 2002.

P4, if $P4.[(minDu \cdot minExec) + (minDelay \cdot (minExec - 1))]$ $> P2.maxDu$. Cycle 2 between P1, P2 and P3, represented in medium gray, may yield an inconsistency if $P2.minDu + P3.minDu > P1.maxDu$. Cycle 3 is depicted in light gray and circumvents the minimum duration of P2 by passing over P4. It will be inconsistent if $P3.minDu + P4.[(minDu \cdot minExec) + (minDelay \cdot (minExec - 1))]$ $> P1.maxDu$. Note that the third cycle is not rendered obsolete by the other two: The total minimum duration of all instantiations of P4 can be lower than the maximum duration of P2 (hereby avoiding an inconsistency in cycle 1), but higher than the minimum duration of P2 at the same time. Together with the minimum duration of P3, it may then exceed the maximum duration of P1, causing an inconsistency in cycle three.

4.3.4 Links to the time line

We have seen in sections 4.1 and 4.3.1, how links to the time line can create cycles within a single PA. Now we will demonstrate that such cycles may also span several PAs. Depending on the involved control flow operators, different kinds of cycles are possible. Therefore, we will examine in the following for each control flow operator, how links to the time line within the involved PAs may induce inconsistencies:

Sequential execution of plans with defined order

We assume a set of sequential plans with defined order, where an early executed plan P_m has a link to the time line of the "earliest" type (ESS or EFS defined) and a later executed plan P_n has one of the "latest" type (LSS or LFS defined) with $m < n$. Then these links form a cycle within the corresponding DG, which further involves the minimum durations of the intermediate plans, the minimum durations of 0 between the finish and the start of neighboring plans and potentially the minimum durations of P_m and P_n .

*** Insert Figure 5 here ***

We omit cycles in the opposite direction, as they involve the unconstrained maximum distances of ∞ between the finish and the start of consecutive plans. Figure 5 shows the possible scenarios with grayed minimum duration constraints and the conditions for each cycle to become inconsistent.

Parallel execution of plans

We assume a set of parallel plans including plans P_i and P_j , which have links to the time line of the opposite type, i.e. one of the "earliest" type and the other of the "latest" type. We then distinguish four types of cycles induced by these links. Figure 6 shows the scenarios with grayed duration

constraints and the conditions for each cycle to become inconsistent.

*** Insert Figure 6 here ***

Arbitrary execution of plans

As the minimum and maximum distances in the DG between two arbitrarily executed PAs are $+\infty$, a cycle involving two or more such plans (of the same parent) cannot become negative. Inconsistencies can only originate from cycles that involve the links between a single arbitrarily executed plan and its parent or children plans.

Cyclic execution of plans

If a cyclic PA defines an SI and/or an FI, these refer to the start of the first execution and/or to the finish of the last execution of the plan. As already shown, the total minimum and maximum durations between the start of the first and the finish of the last executions are $(minDu \cdot minExec) + (minDelay \cdot (minExec - 1))$ and $(maxDu \cdot maxExec) + (maxDelay \cdot (maxExec - 1))$. As we thus treat all executions of a cyclic PA as a whole and its links to the time line can only refer to the first and last executions, only a single PA is involved here. Therefore, a cyclic PA linked to the time line can only be part of a network of PAs through the links to its parent or children plans; it cannot generate a network of PAs itself. Therefore, when examining a network of PAs for inconsistencies, cyclic plans only have to be considered as part of those cycles that include the cyclic plan's links to its parent and/or children plans. However, each single cyclic plan may itself become inconsistent, as shown in section 4.3.2.

4.4 Non-minimalities

In this section, we will examine how non-minimalities may arise within one PA or within a network of several PAs. Although we do not allow inconsistent PAs within a plan hierarchy, we do not force the user to define minimal PAs during the design of a plan hierarchy. Ensuring manually that each PA is minimal would be annoying for her. Instead, we point out non-minimal PAs to the user during plan verification, and give her the option to adapt it. Providing the minimal variant of a PA as a suggestion on how to modify it does not require extra effort: The algorithm we use for consistency checking is based on the computation of minimal PAs within a plan hierarchy.

4.4.1 Single plan activation

A single PA that is consistent by not containing a negative cycle as described in section 4.3.1, is not

to appear in *Artificial Intelligence in Medicine*, 2002.

necessarily minimal. It will be minimal when it satisfies the following additional demands that can be deduced from the definition of a time annotation's bounds ($[ESS^-, LSS^+]$, $[EFS^-, LFS^+]$, $[minDu^-, maxDu^+]$), as given in section 3.2.

- $ESS \leq EFS$ and $LSS \leq LFS$
- $MAXIMUM(0, EFS - LSS) \leq minDu \leq MINIMUM(EFS - ESS, LFS - LSS)$
- $MAXIMUM(EFS - ESS, LFS - LSS) \leq maxDu \leq LFS - ESS$

The first point allows SI to be in any of the relations $\{equal, overlaps, meets, before\}$ to FI. Depending on SI and FI, the second and third points yield the bounds for DI. On the left side of Figure 7 (a), a PA that does not satisfy the above demands is shown: One objection to minimality consists in its EFS being smaller than ESS. The PA on the right side of Figure 7 (a) represents an equivalent minimal version of the same PA that satisfies the above demands. We use the 1D representation introduced in Figure 1 here, as it allows a graphical visualization of the difference between non-minimal and minimal PAs.

*** Insert Figure 7 here ***

4.4.2 Network of plan activations

Non-minimality of a PA may also be induced by links to other PAs in a common network. The left side of Figure 7 (b) shows a plan hierarchy containing PAs that are rendered non-minimal by two types of scheduling constraints: One is implied by a control flow operator for sequential execution and the other by a parent–child relation.

4.5 Handling Asbru's operator for unordered, sequential plan execution

The Asbru language includes an operator for sequential execution of plans, where the order in which the plans are executed is not specified. Accordingly, an unordered sequential execution of plans P1 and P2 presumes that P1 is executed either before or after P2. In terms of the DG, this means that there is a minimum distance of 0 either between the finish of P1 and the start of P2, or between the finish of P2 and the start of P1.

The disjunctions within the last two sentences already indicate that an unordered, sequential execution of plans exceeds the scope of STPs. It also does not fit into the framework of general temporal constraint satisfaction problems (TCSPs), as defined by Dechter et al. [3]: Although their TCSP framework allows disjunctions within constraints, the latter may only be of the binary type. This means that each disjunct within one disjunction must exclusively refer to the same pair of

variables (e.g., $a \leq X_j - X_i \leq b \vee c \leq X_j - X_i \leq d$). However, the unordered sequential execution of plans P1 and P2 introduces the 4-ary constraint $0 \leq P2.S - P1.F \leq \infty \vee 0 \leq P1.S - P2.F \leq \infty$.

To incorporate the unordered sequential execution of plans, we need an approach that can handle non-binary constraints, such as the framework proposed by Stergiou and Koubarakis [29]. They consider temporal constraints of the form $X_l - Y_l \leq r_l \vee \dots \vee X_n - Y_n \leq r_n$, where $X_1, \dots, X_n, Y_1, \dots, Y_n$ are variables and $r_1, \dots, r_n$ are constants. This covers the representation of unordered sequential plans, e.g. for plans P1, P2 and P3 we would define the three constraints $(P1.F - P2.S \leq 0) \vee (P2.F - P1.S \leq 0)$, $(P1.F - P3.S \leq 0) \vee (P3.F - P1.S \leq 0)$ and $(P2.F - P3.S \leq 0) \vee (P3.F - P2.S \leq 0)$.

However, the problem of determining the consistency of a set of constraints that includes disjunctions of non-binary constraints is NP hard. Therefore, we choose to apply partial verification for unordered sequential plans only, by checking whether there are any easily identifiable objections to their consistency:

- We check whether there is any objection to the assumption that the set of plans can be executed in sequential order at all. This is the case if two or more of the corresponding PAs are *overlapping*, which means that they force the plans to be executed at least partially in parallel.

Whether two PAs are overlapping can be deduced from their parameters LSS and EFS:

$$Overlapping(PA_1, PA_2) \text{ iff } ((LSS(PA_1) < EFS(PA_2)) \wedge (LSS(PA_2) < EFS(PA_1)))$$

If two or more PAs of a set of unordered sequential plans are overlapping, we will receive an inconsistency (i.e. a negative cycle within their DG), no matter how we arrange them.

- The constraint on the EI of a child plan with respect to its parent's EI (see section 4.3.3) clearly also holds for unordered, sequential plans. The corresponding constraints are inserted in the matrix of weights accordingly (see section 5).
- The total minimum durations of a set of unordered, sequential plans must not exceed their parent's maximum duration. To perform the corresponding check, we can execute an additional verification run, where we treat the set of unordered, sequential plans as ordered, sequential plans, considering only their DIs but ignoring their SIs and FIs.

4.6 Verification with multiple time lines

As we have seen in section 4.3.4, our verification process involves checking whether particular constraints among two or more PAs, which are linked to the time line, are consistent. This purpose is

to appear in *Artificial Intelligence in Medicine*, 2002.

rendered difficult by Asbru's ability to support multiple time lines within one plan hierarchy:

Asbru allows the reference point of a PA to refer to patient-specific time points (e.g., birth, conception, ...). The actual calendar date for such a time point can only be determined during the execution of a plan, after it has been assigned to a patient. Statically, its relation to another PA's reference point is a priori unknown, which means that the distance between them is set to $(-\infty, +\infty)$. This, in due course, can prevent certain "obvious" inconsistencies from being detected:

Let us assume that we have to check, in the context of a guideline for the treatment of gestational diabetes, whether plans P1 and P2 can be executed sequentially in order P1, P2, when activated by $(P1 \ [[_], [_], [2, _], [_], [_], \text{delivery}])$ and $(P2 \ [[_], 2], [_], [_], [_], \text{conception}])$ (see upper diagram in Figure 8REFFORMATVERBINDEN). As the unknown distance between the two reference points turns the sum of weights of the induced cycle to $+\infty$, a sequential execution of P1 and P2 is found to be consistent (see lower left diagram in Figure 8).

*** Insert Figure 8 here ***

In such a case, verification can be optimized if additional knowledge on the relation between the reference points of different PAs is available. Let us assume that we include such knowledge in our checking process, i.e. for the same pregnancy, the event `delivery` always happens *after* the event `conception`. This translates to a constraint of $[\varepsilon, +\infty)$ on the distance between them, where ε represents the smallest supported time unit. When using this additional knowledge, a sequential execution of P1 and P2 is found to be inconsistent (see lower right diagram in Figure 8).

No additional knowledge is needed when checking PAs with identical reference points, as they are trivially related. The same holds for PAs, whose reference points are specified as time points of the calendar.

5 Verification method

Like Dechter et al., we use Floyd-Warshall's *all-pairs-shortest-paths algorithm* to detect inconsistencies within a network of PAs:

```

for  $i, j = 1$  to  $n$  do  $d_{ij} \leftarrow a_{ij}$ ;
for  $i = 1$  to  $n$  do  $d_{ii} \leftarrow 0$ ;
for  $k = 1$  to  $n$  do
  for  $i, j = 1$  to  $n$  do
     $d_{ij} \leftarrow \min(d_{ij}, d_{ik} + d_{kj})$ ;

```

Applied to the network's DG, the algorithm calculates the minimal network represented as a matrix of minimum distances d_{ij} in time $O(n^3)$, where n is the number of nodes. Inconsistencies (i.e. negative cycles) can simply be detected by examining the sign of the diagonal elements d_{ii} within the minimum distance matrix. The matrix of weights a_{ij} is set up by inserting the defined constraints originating from the individual PAs, parent-child relations and control flow operators. Instead of the value ∞ for unconstrained distances, we use the constant $INF = (n \cdot MAX(a_{ij})) + 1$, as no path can have more than n arcs.

The algorithm further permits the identification of non-minimal PAs and provides their minimal versions as a suggestion on how to adapt them. We make use of this feature by pointing out non-minimal PAs to the user during the verification process and leave her the choice of accepting the minimal version, modifying the PA differently or leaving it as it is. An additional benefit of calculating the minimal network is the fact that it may be used to assemble solutions, i.e. assignments to the parameters of all PAs, that are consistent with the network's constraints in time $O(n^2)$. Finding a solution of the network is an essential task for the guideline interpreter to determine the EIs of all plans.

*** Insert Table 1 here ***

To demonstrate the set-up process of our method, Table 1 (a) shows the matrix of weights a_{ij} , after inserting the constraints of plan P2 and its subplans of Example 1 (we use here only this part of the entire plan hierarchy, in order to keep the example simple). The value 701 corresponds to INF. Figure 9 provides an obvious depiction of the weights corresponding to all other values: The value 0 at cells (P5.S, P2.S), (P2.F, P5.F), (P6.S, P2.S), and (P2.F, P6.F) originates from parent-child relations. The value 0 at cells (P5.S, P6.S) and (P6.S, P5.S) originates from the parallel execution of plans P5 and P6. The value -40 at cell (P6.S, Ref) originates from the ESS of P6. All other values originate from the minimum or maximum durations of plans P2 and P5.

Table 1 (b) shows the matrix of minimum distances d_{ij} , which results from applying Floyd-Warshall's algorithm to the matrix of weights in Table 1 (a). It demonstrates that the PAs within P2 and its subplans are consistent, as it does not contain a negative diagonal elements d_{ii} . It further shows that none of the three PAs is minimal: For P2 the values of ESS, LSS, EFS and minDu, corresponding to cells (P2.S, Ref), (Ref, P2.S), (P2.F, Ref), and (P2.F, P2.S) can be tightened. For P5 all values except minDu and LFS can be tightened. For P6 all values except ESS and LFS can be tightened.

6 Evaluation of the verification method

Four existing clinical guidelines have been represented in the Asbru language so far. These include the following: a guideline for the observation and treatment of gestational diabetes mellitus used at the Stanford Medical School, which is based on the California Diabetes and Pregnancy Program "Sweet Success"; a guideline that supports the artificial ventilation of neonates used at the neonatal intensive care unit of the University of Vienna Medical School [6]; a guideline for the treatment of sinusitis used by general practitioners in the Netherlands [31]; and a guideline for the management of hyperbilirubinemia in newborns, issued by the American Academy of Pediatrics [1]. Although these four guidelines comprise numerous scheduling constraints implied by control flow and hierarchical structuring, PAs are barely utilized. Therefore, they do not represent interesting subjects for an automatic verification of their temporal scheduling constraints. For the demonstration of our approach, we will thus verify the PAs within the fictive guideline shown in Example 1. This guideline is an interesting test case, as it covers all three kinds of scheduling constraints examined in this paper, including several variants thereof, namely (1) temporal scheduling constraints: different variants of incomplete time annotations are used, some of which are linked to the time line; (2) scheduling constraints implied by the guideline's control flow: the full spectrum of Asbru's control flow operators is covered; (3) scheduling constraints implied by a hierarchical structuring of a guideline: the plan hierarchy is arranged in four different levels.

*** Insert Figure 9 here ***

To visualize the different cycles emerging from this guideline, we depict the corresponding DG in Figure 9. It does not include the two unordered sequential plans P10 and P11, as the corresponding constraints on their control flow cannot be represented within a DG.

As an XML-DTD (Document Type Definition) exists for the current version of Asbru, in the final implementation of our verifier we intended to employ XML as the import format of Asbru plans. The prototype verifier, which we currently use and present in this paper, is confined to a manual specification of the plan hierarchy within the verifier. Its verification algorithm, however, is based on the method described in section 5. The application contains two text areas, where the left side shows the plan hierarchy to be analyzed and the right side depicts the results of the verification process.

*** Insert Figure 10 here ***

Figure 10 shows the result of verifying the guideline of Example 1. It tells us that the PAs within the plan hierarchy are inconsistent for two reasons: First, an inconsistent path corresponding to a negative cycle within Figure 9 is present, which involves the minimum durations of plans P4, P7 and

P5, the latest finishing shift of P1, the earliest starting shift of P6, and several links induced by control flow operators and parent-child relations. Second, the total minimum duration of the unordered sequential plans P10 and P11 exceeds the maximum duration of their parent plan P8.

*** Insert Figure 11 here ***

Figure 11 shows the result of verifying the guideline after the two inconsistencies were cleared by setting the latest finishing shift of plan P1 to 420 and the minimum duration of plan P10 to 70. As described in section 4.4, the verifier provides the minimal version of each PA as a suggestion on how to adapt it.

7 Conclusion

Scheduling constraints, expressed either explicitly by control flow operators (e.g., for sequential or parallel execution) or implicitly by a hierarchical modeling, are basic elements of most current formats for the computerized representation of clinical guidelines. Another type of scheduling constraint, which is frequently found in conventional guidelines, is given by temporal specifications (e.g., "determine the newborn's blood glucose level within 30 minutes after birth", "execute a controlled ventilation for at least 3 hours"). The importance of the latter kind of constraint is reflected by a growing number of integrations within current guideline specification formats. However, like other types of knowledge, temporal scheduling constraints may be used inconsistently within a guideline and may thus be an obstacle for the guideline's computerization.

In this paper we have addressed the problem by presenting an approach for the verification of temporal scheduling constraints within clinical practice guidelines, represented in the Asbru language. The approach is based on the calculation of the minimal temporal network, and serves three purposes: (1) it allows the identification of inconsistent temporal scheduling constraints; (2) it detects non-minimal temporal scheduling constraints and yields suggestions for their representation in an equivalent, but more explicit form; and (3) it can be used by the guideline interpreter to determine feasible EIs for each guideline activity.

The computation of the minimal network can be done in $O(n^3)$ time for a guideline with n individual activities and therefore constitutes an efficient method for its verification. Efficiency is an issue for us insofar as it allows our verification process to be applied interactively during the design phase of a guideline, e.g. after each definition of a new temporal scheduling constraint, without annoying the user by long answering times.

to appear in *Artificial Intelligence in Medicine*, 2002.

A limitation of our approach consists in its inability to fully verify temporal constraints on the execution of unordered sequential guideline activities. Although we apply several checks that can detect basic types of inconsistencies in this case, a full verification would require a technique that can handle disjunctions of non-binary constraints. Even though suitable approaches for this task have been described in the literature [28, 29], the problem of fully verifying a guideline that includes unordered sequential activities is NP hard. As Asbru is currently the only representation format that considers unordered sequential activities, we chose to integrate them in the verification process only through partial checks. Hereby, the efficiency of our approach is not compromised.

Our approach is reusable in two ways: (1) It is applicable for the verification of several guideline representation formats other than Asbru, such as GLIF, EON, PROforma, or DILEMMA / PRESTIGE. This is due to the fact that our approach assumes the existence of certain basic operators for flow control within a guideline representation format (such as sequential, parallel or cyclic execution) and its organization in a hierarchical structure. These concepts are realized in most current approaches [5, 9, 18, 19]. Our considerations on temporal scheduling constraints (originating from Asbru's PAs in this paper) have to be adapted when reusing our approach for a different representation, according to the expressiveness of the respective format in this area. (2) Our approach can be ported to domains other than guideline-based care for the following reasons: First, the Asbru language is suitable for several areas of planning [15]. Second, our verification method is a domain-independent, general approach for processing temporal constraints [3].

Acknowledgments. The authors thank Michael Balser, Frank van Harmelen, Peter Johnson, Robert Kosara, Wolfgang Reif, Andreas Seyfang, Yuval Shahr, and Annette ten Teije, who contribute to the execution of the project. The project is supported by the "Fonds zur Förderung der wissenschaftlichen Forschung" (Austrian Science Fund), P12797-INF.

References

- [1] American Academy of Pediatrics, Practice parameter: management of hyperbilirubinemia in the healthy term newborn., *Pediatrics* 94 (1994) 558-565.
- [2] J. F. Allen, Maintaining knowledge about temporal intervals, *Communications of the ACM* 26 (1983) 832-843.
- [3] R. Dechter, I. Meiri and J. Pearl, Temporal constraint networks, *Artificial Intelligence* 49 (1991) 61-95.

- [4] G. Duftschmid and S. Miksch, Knowledge-based verification of clinical guidelines by detection of anomalies, *Artificial Intelligence in Medicine* 22 (2001) 23-41.
- [5] J. Fox, N. Johns and A. Rahmanzadeh, Disseminating medical knowledge: the PROforma approach, *Artificial Intelligence in Medicine* 14 (1998) 157-181.
- [6] JP. Goldsmith and EH. Karotkin, *Assisted ventilation of neonates* (Saunders, Philadelphia, 1993).
- [7] C. Gordon, I. Herbert and P. Johnson, Knowledge representation and clinical practice guidelines: the DILEMMA and PRESTIGE projects, in: Brender-J, Christensen-J, Scherrer-JR and McNair-P, eds., *Proceedings of Medical Informatics Europe 96*; Copenhagen, Denmark (R. Brompton Hospital London UK. Medical Informatics Europe '96: Human Facets in Information Technologies. IOS Press Amsterdam Netherlands, 1996).
- [8] A. Guarnero, M. Marzuoli, G. Molino, P. Terenziani, M. Torchio and K. Vanni, Contextual and temporal clinical guidelines, in: *Proceedings of the AMIA-98 Annual Symposium* (1998) 683-687.
- [9] S. Herbert, C. Gordon, A. Jackson-Smale and Salis Renaud, Protocols for clinical care, *Computer Methods and Programs in Biomedicine* 48 (1995) 21-26.
- [10] G. Hripcsak, Writing Arden Syntax medical logic modules, *Computers in Biology and Medicine* 24 (1994) 331-363.
- [11] H. A. Kautz and P. B. Ladkin, Integrating metric and qualitative temporal reasoning, in: *Proceedings 9th National Conference on Artificial Intelligence (AAAI-91)* (AT&T Bell Labs. Murray Hill NJ USA. AAAI Press Menlo Park CA USA, 1991).
- [12] C. McDonald and J. Overhage, Guidelines you can follow and trust: An ideal and an example, *Journal of the American Medical Association (JAMA)* 271 (1994) 872-873.
- [13] I. Meiri, Combining qualitative and quantitative constraints in temporal reasoning, *Artificial Intelligence* 87 (1996) 343-385.
- [14] S. Miksch, Plan management in the medical domain, *AI Communications* 12 (1999) 209-235.
- [15] S. Miksch, Y. Shahar and P. Johnson, Asbru: A task-specific, intention-based, and time-oriented language for representing skeletal plans, in: *Proceedings of the 7th Workshop on Knowledge Engineering: Methods & Languages (KEMML-97)*, Milton Keynes, UK (1997).
- [16] D.W. Miller, Jr, S.J. Frawley and P.L. Miller, Using semantic constraints to help verify the

to appear in *Artificial Intelligence in Medicine*, 2002.

completeness of a computer-based clinical guideline for childhood immunization, *Computer Methods and Programs in Biomedicine* 58 (1999) 267-280.

- [17] M. Musen, J. Rohn, L. Fagan and E. Shortliffe, Knowledge engineering for a clinical trial advice system: uncovering errors in protocol specification, *Bulletin du Cancer* 74 (1987) 291-296.
- [18] M. Musen, S. Tu, A. Das and Y. Shahar, EON: A component-based approach to automation of protocol-directed therapy, *Journal of the American Medical Informatics Association (JAMIA)* (1996) 367-388.
- [19] L. Ohno-Machado, J. Gennari, S. Murphy, N. Jain, S. Tu, D. Oliver, E. Pattison-Gordon, R. Greenes, E. Shortliffe and G. Barnett, The GuideLine Interchange Format: A model for representing guidelines, *Journal of the American Medical Association (JAMA)* 5 (1998) 357-372.
- [20] E. Pattison-Gordon, J. Cimino, G. Hripcsak, S. Tu, J. Gennari, N. Jain and R. Greenes, Requirements of a sharable guideline representation for computer applications. Stanford Technical Report SMI-96-0628, Stanford University, 1996.
- [21] M. Peleg, A. A. Boxwala, O. Ogunyemi, Q. Zeng, S. Tu, R. Lacson, E. Bernstam, N. Ash, P. Mork, L. Ohno-Machado, E. H. Shortliffe and R. A. Greenes, GLIF3: the evolution of a guideline representation format, in: *Proceedings of the AMIA-2000 Annual Symposium*, Los Angeles, CA (2000) 645-649.
- [22] S. Quaglini, R. Saracco, M. Stefanelli and C. Fassino, Supporting tools for guideline development and dissemination, in: *Proceedings of the 6th Conference on Artificial Intelligence in Medicine Europe (AIME)* (1997) 39-50.
- [23] J.F. Rit, Propagating temporal constraints for scheduling, in: *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86)*; Los Altos, CA (Morgan Kaufmann, 1986) 383-388.
- [24] 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* 14 (1998) 29-51.
- [25] E. Sherman, G. Hripcsak, J. Starren, R. Jender and P. Clayton, Using intermediate states to improve the ability of the Arden Syntax to implement care plans and reuse knowledge, in: *Proceedings of the Annual Symposium on Computer Applications in Medical Care (SCAMC-*

95) (1995) 238-242.

- [26] R. Shiffman, Representation of clinical practice guidelines in conventional and augmented decision tables, *Journal of the American Medical Informatics Association (JAMIA)* 4 (1997) 382-393.
- [27] R. Shiffman and R. Greenes, Improving clinical guidelines with logic and decision-table techniques, *Medical Decision Making* 14 (1994) 245-254.
- [28] S. Staab, On non-binary temporal relations, in: Prade-H, ed. *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI 98)* (Comput. Linguistics Lab. Freiburg Univ. Germany. Wiley Chichester UK, 1998).
- [29] K. Stergiou and M. Koubarakis, Backtracking algorithms for disjunctions of temporal constraints, *Artificial Intelligence* 120 (2000) 81-117.
- [30] J. Stillman, R. Arthur and A. Deitsch, Tachyon: A constraint-based temporal model and its implementation, *SIGART Bulletin* 4 (1993).
- [31] S. Thomas, R.M.M. Geijer, J.R. van der Laan and Tj. Wiersma, *NHG-Standaarden voor de huisarts II* (Productie Wetenschappelijke uitgeverij Bunge, Utrecht, 1996).
- [32] W. Tierney, J. Overhage, B. Takesue, L. Harris, M. Murray, D. Vargo and C. McDonald, Computerizing guidelines to improve care and patient outcomes: the example of heart failure, *Journal of the American Medical Informatics Association (JAMIA)* 2 (1995) 316-322.
- [33] S. Vere, Planning in time: Windows and durations for activities and goals, *IEEE Transactions on PAMI* 5 (1983) 246-267.
- [34] M. Vilain and H. Kautz, Constraint propagation algorithms for temporal reasoning, in: *Proceedings AAAI-86: Fifth National Conference on Artificial Intelligence*; Univ (BBN Labs. Cambridge MA USA. American Assoc. Artificial Intelligence Menlo Park CA USA, 1986) 377-382.

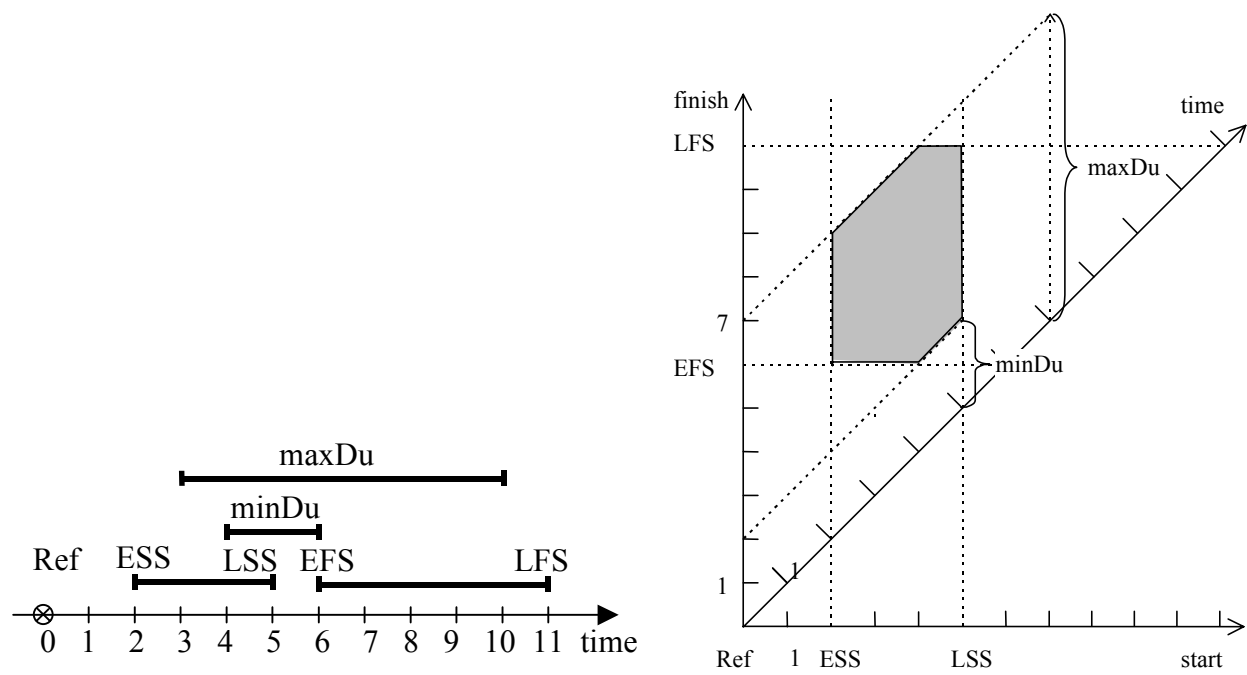


Figure 1

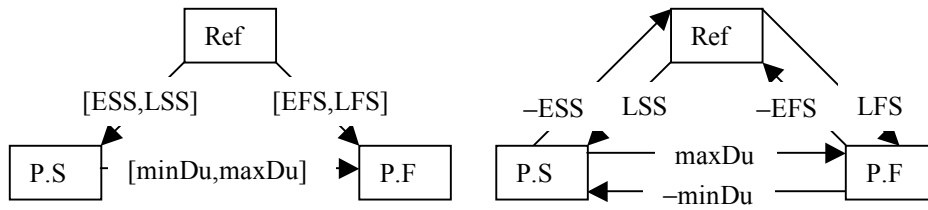


Figure 2

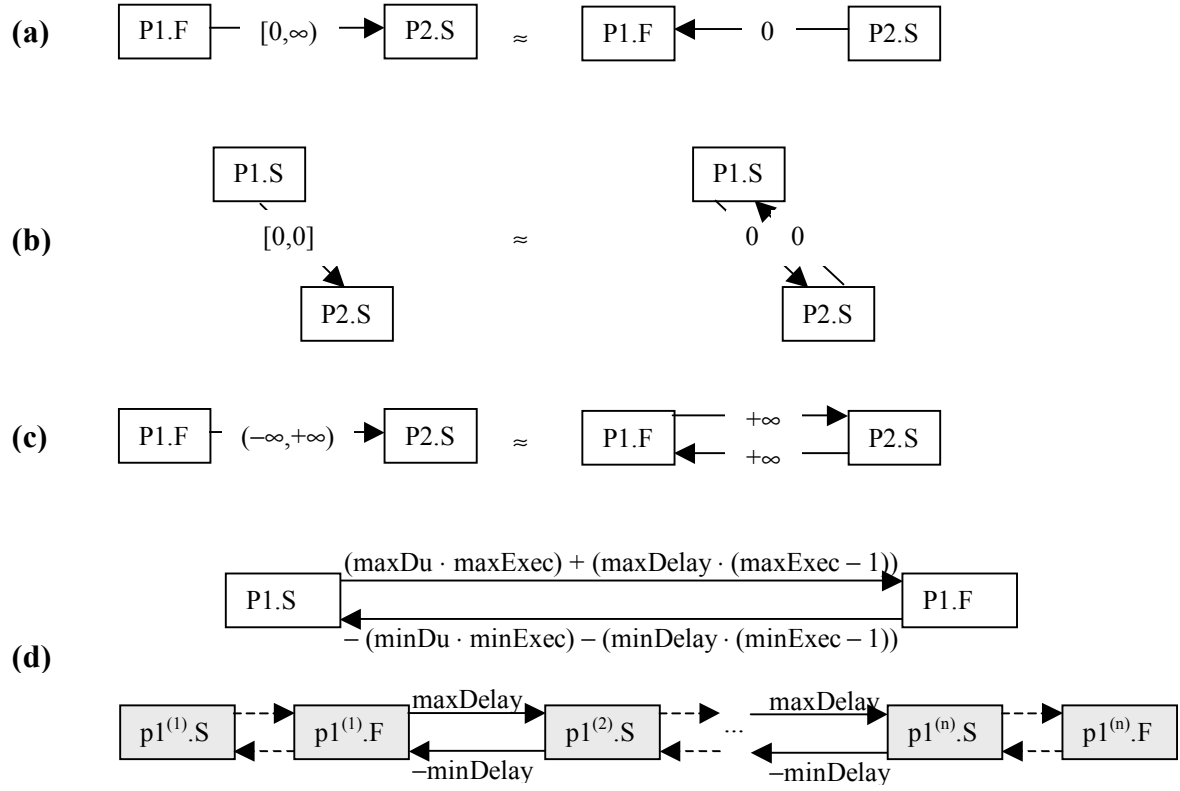
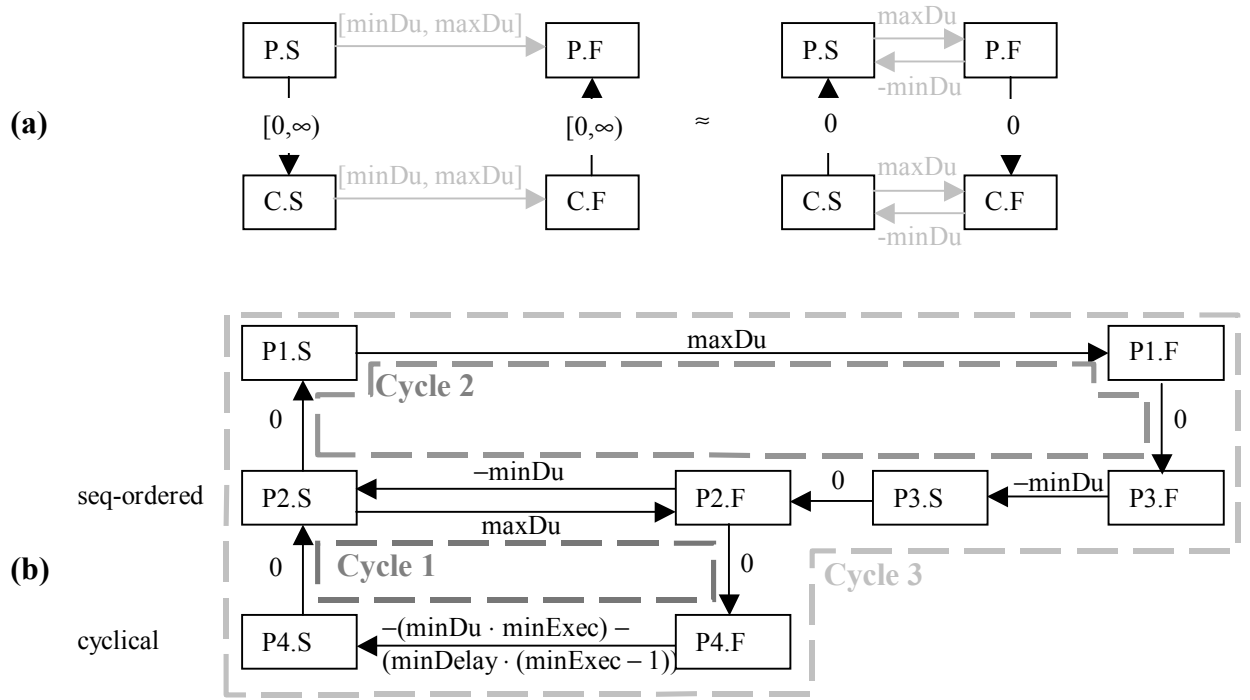


Figure 3



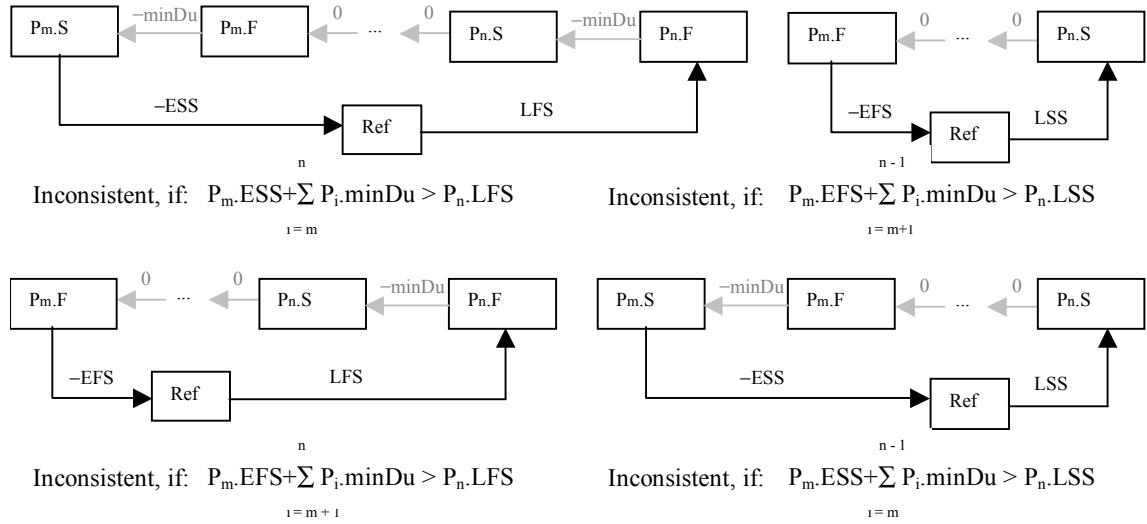


Figure 5

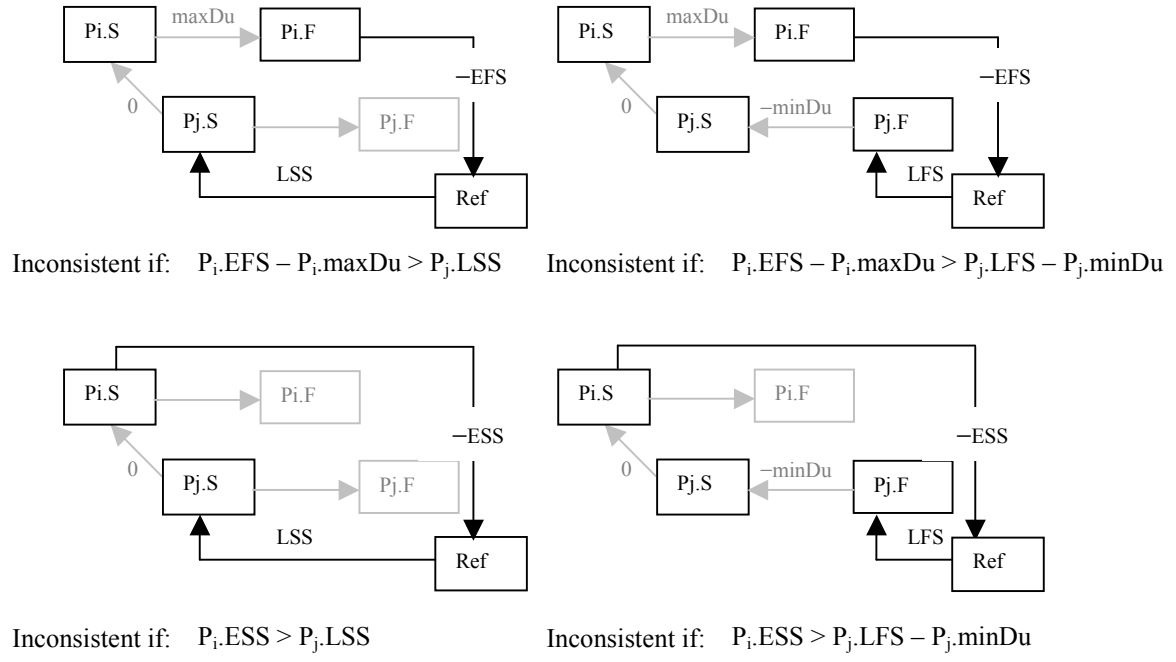


Figure 6

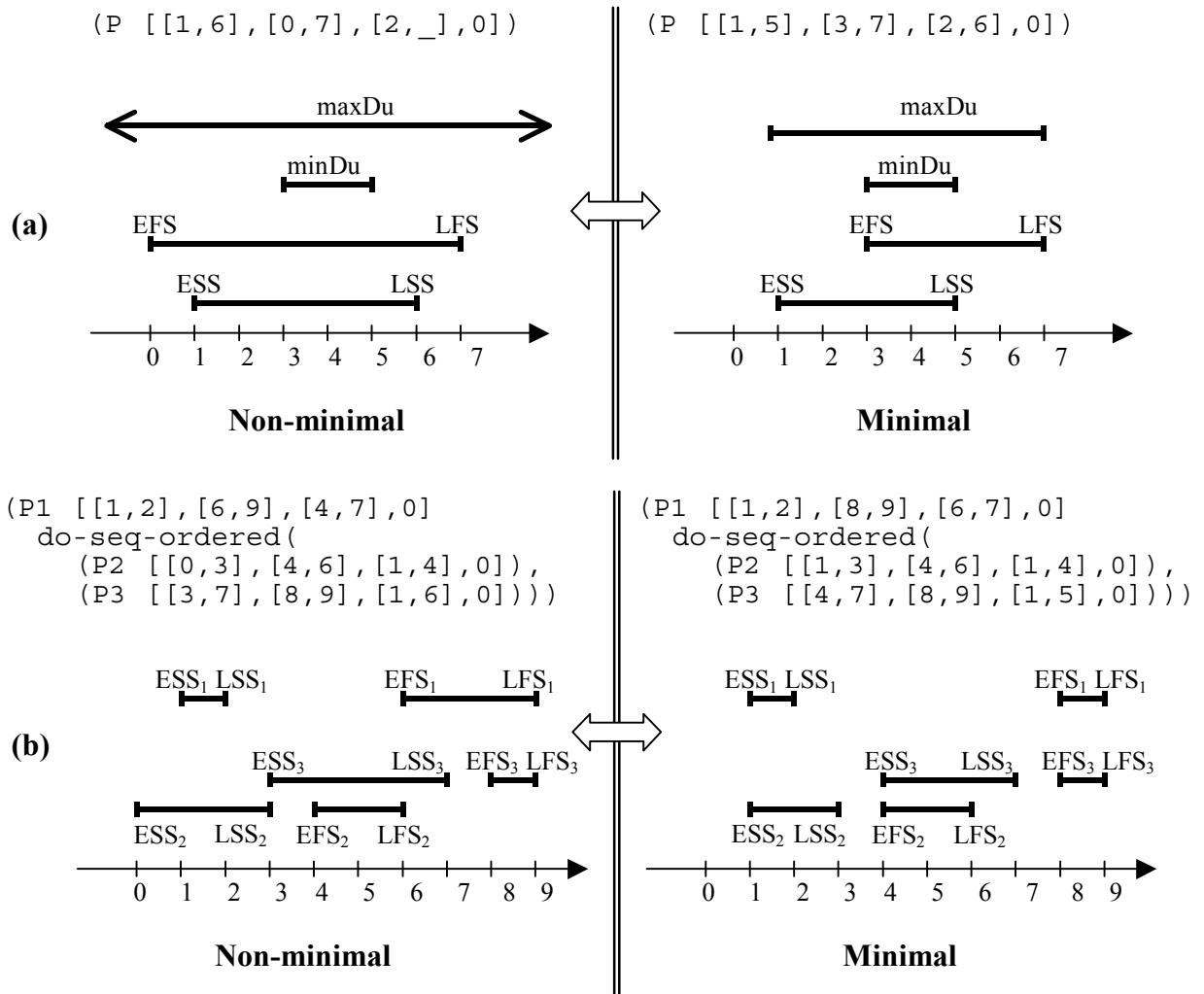


Figure 7

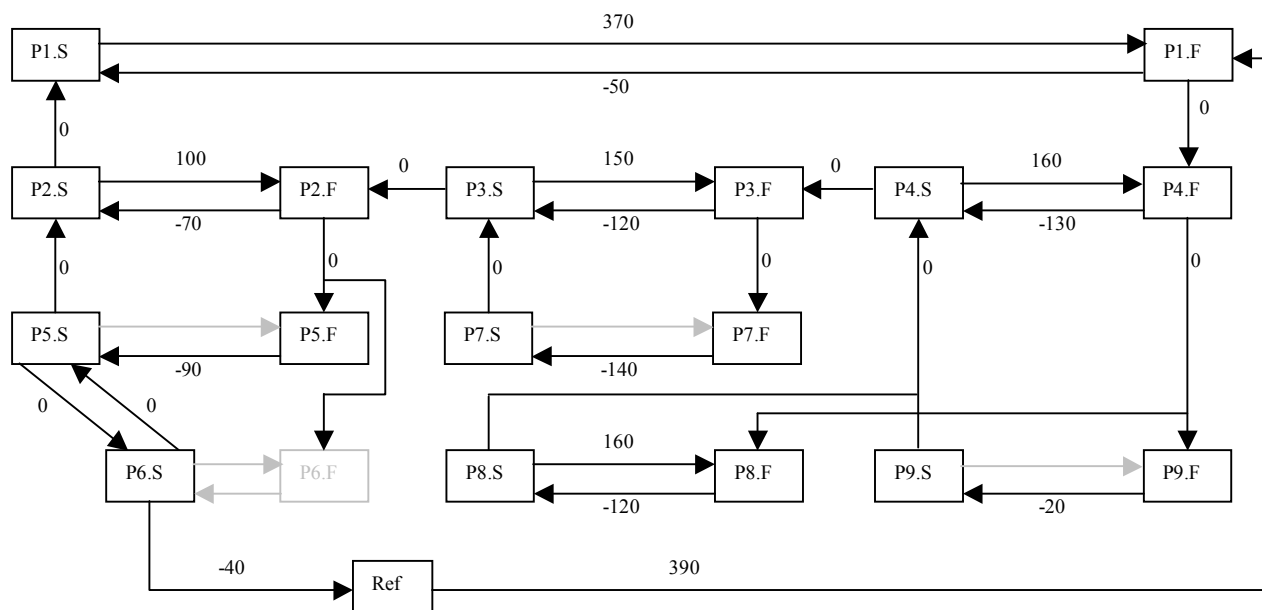


Figure 9

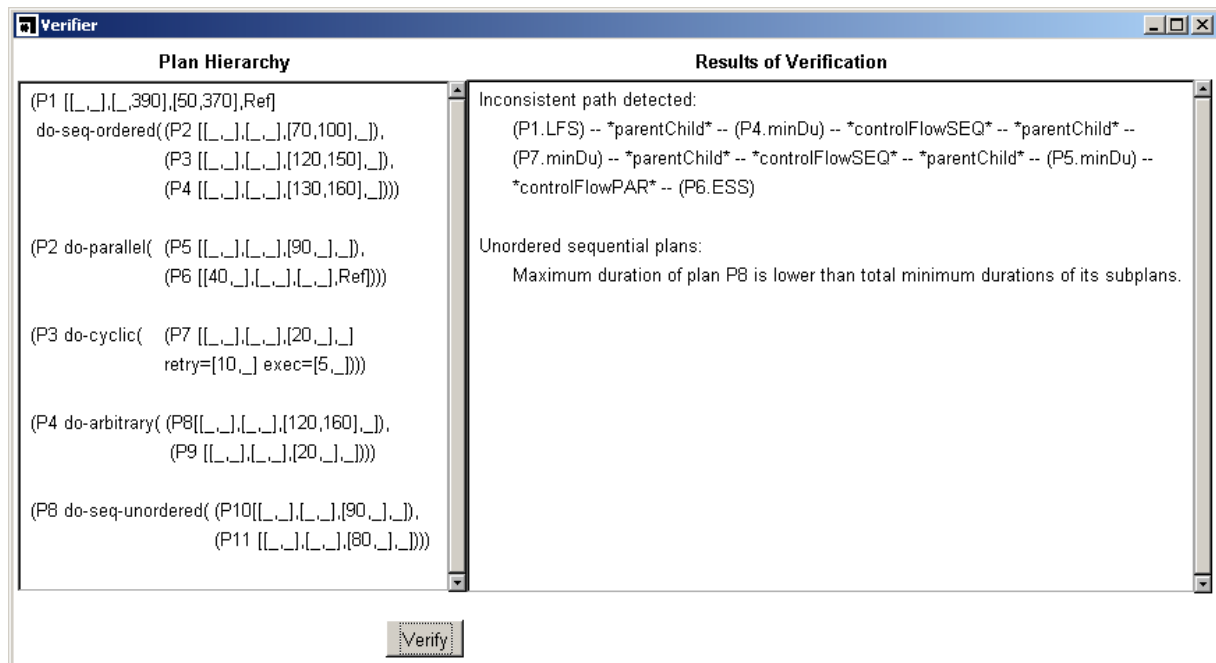


Figure 10

to appear in *Artificial Intelligence in Medicine, 2002*.

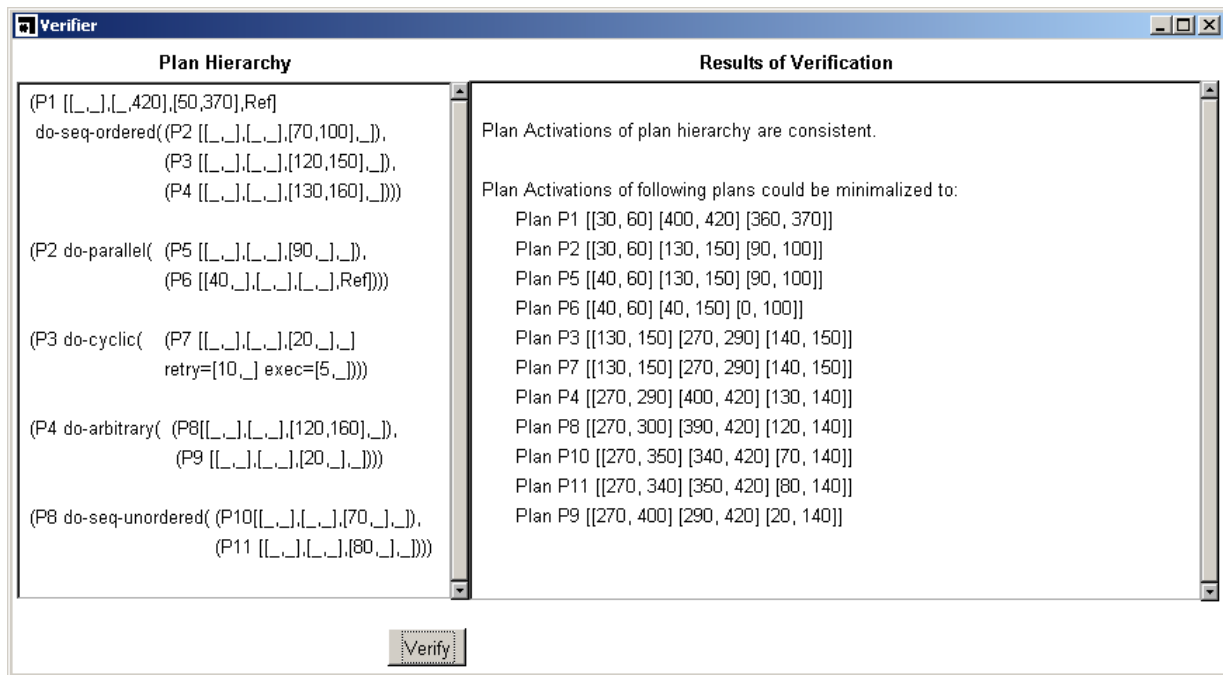


Figure 11

	P2.S	P2.F	P5.S	P5.F	P6.S	P6.F	Ref
(a)	P2.S	0	100	701	701	701	701
	P2.F	-70	0	701	0	701	0
	P5.S	0	701	0	701	0	701
	P5.F	701	701	-90	0	701	701
	P6.S	0	701	0	701	0	701
	P6.F	701	701	701	701	0	701
	Ref	701	701	701	701	701	0
(b)	P2.S	0	100	10	100	10	100
	P2.F	-90	0	-90	0	-90	0
	P5.S	0	100	0	100	0	100
	P5.F	-90	10	-90	0	-90	10
	P6.S	0	100	0	100	0	100
	P6.F	611	701	611	701	611	0
	Ref	611	701	611	701	611	701

Table 1

to appear in *Artificial Intelligence in Medicine*, 2002.

Figure 1. One-dimensional (left) and two-dimensional (right) graphical representation of time annotation $[[2, 5], [6, 11], [2, 7], 0]$. In 1D visualization, the duration bars must be considered to be "floating" above SI and FI; their position depicted here is only one possible scenario. In 2D visualization, the time annotation is represented as a shaded polygon, which also allows an unambiguous depiction of DI.

Figure 2. Constraint graph (left) and distance graph (right) of plan activation (P $[[ESS, LSS], [EFS, LFS], [minDu, maxDu], Ref]$).

Figure 3. Constraint implied by (a) sequential, ordered execution of plans P1 and P2; (b) parallel execution of plans P1 and P2; (c) arbitrary execution of plans P1 and P2. (d) Static view of the cyclic plan P1 with instance p1 below. Instance p1 is executed n times, where $minExec \leq n \leq maxExec$. Within p1, minimum and maximum durations are represented as dashed edges.

Figure 4. (a) Constraints between parent plan P and child plan C, shown as vertical edges. The gray horizontal edges represent the duration constraints. They are charted to make cycles obvious. (b) Cycles involving parent-child relations and the control flow.

Figure 5. Inconsistent cycles introduced by sequential plans with defined order linked to the time line.

Figure 6. Inconsistent cycles introduced by parallel plans linked to the time line.

Figure 7. Non-minimal (left side) and minimal (right side) versions of (a) two equivalent single PAs; (b) two equivalent networks of PAs. DIs are omitted.

Figure 8. Within a guideline for treating gestational diabetes, it is checked whether plans P1 and P2 may be executed sequentially in order P1, P2, when activated by $(P1 \text{ } [[_ , _], [2, _], [_ , _], delivery])$ and $(P2 \text{ } [[_ , 2], [_ , _], [_ , _], conception])$. Without additional knowledge, this scenario is found to be consistent. When the additional knowledge " $conception < delivery$ " is included, the scenario is found to be inconsistent.

Figure 9. DG of the plan hierarchy, shown in Example 1. Unconstrained durations are shown in gray.

Figure 10. Result of verifying the plan hierarchy shown in Example 1.

Figure 11. Result of verifying the corrected plan hierarchy.

Table 1. Matrix of (a) weights a_{ij} ; (b) minimum distances d_{ij} ; for plan P2 and subplans of Example 1. Compare with the corresponding part of DG in Figure 9.

Appendix B:

Properties
at the Conceptual Level /
Protocol-Dependent Properties

Properties at the conceptual level / protocol-dependent properties

In addition to the properties that are defined at the implementation level, which heavily depend on the characteristics of the language used to represent protocols—in our case, Asbru, we can define other properties at the conceptual level, more dependent on the particular protocols at hand. In this part we discuss different proposals for properties that can be useful to verify medical protocols at this level. As we will see, these properties focus on aspects like the particular clinical parameters that are handled in the protocols or on the actions that the healthcare providers are required to perform.

We can distinguish different categories of protocol-dependent properties:

1. *Correctness* properties. They aim at ensuring that the protocol is free from faults.
2. *Safety* properties. Their goal is ensuring that hazardous situations will not occur.
3. Properties about (details of) *actions*. Here we can refer to different aspects of clinical actions, such as the duration or doses of a treatment, or even to interactions between actions.
4. Properties about *traces of actions*. Sometimes it might be desirable to express properties about actions in terms of their occurrence in time, e.g. sequences of required/forbidden actions.
5. Properties about particular *groups of patients*. They reflect the additional properties that might be necessary to manage patients with special characteristics.
6. Properties about *interactions with the management of related diseases*. Protocols for chronic diseases are characterised by the existence of interactions with the management of other related conditions, which might need additional attention.
7. Properties about *indicators*. In some cases protocols could be checked using some sort of indicator of the expected outcomes.

It is important to note that the above categories are not necessarily mutually exclusive. Thus, it is possible to define a property about safety conditions required for the management of a concrete patient group. For instance, in the framework of the Diabetes protocol we can define safety properties stating more strict criteria for the control of patients with complications. Thus, patients with microalbuminuria need a more strict blood pressure control.

Lastly, emphasising that the definition of these conceptual properties requires a deep understanding of both the disease and the concrete protocol dealing with it. Therefore, the participation of medical experts in this task is very important. When this has not been possible, we have resorted to additional information in other protocols and/or research publications.

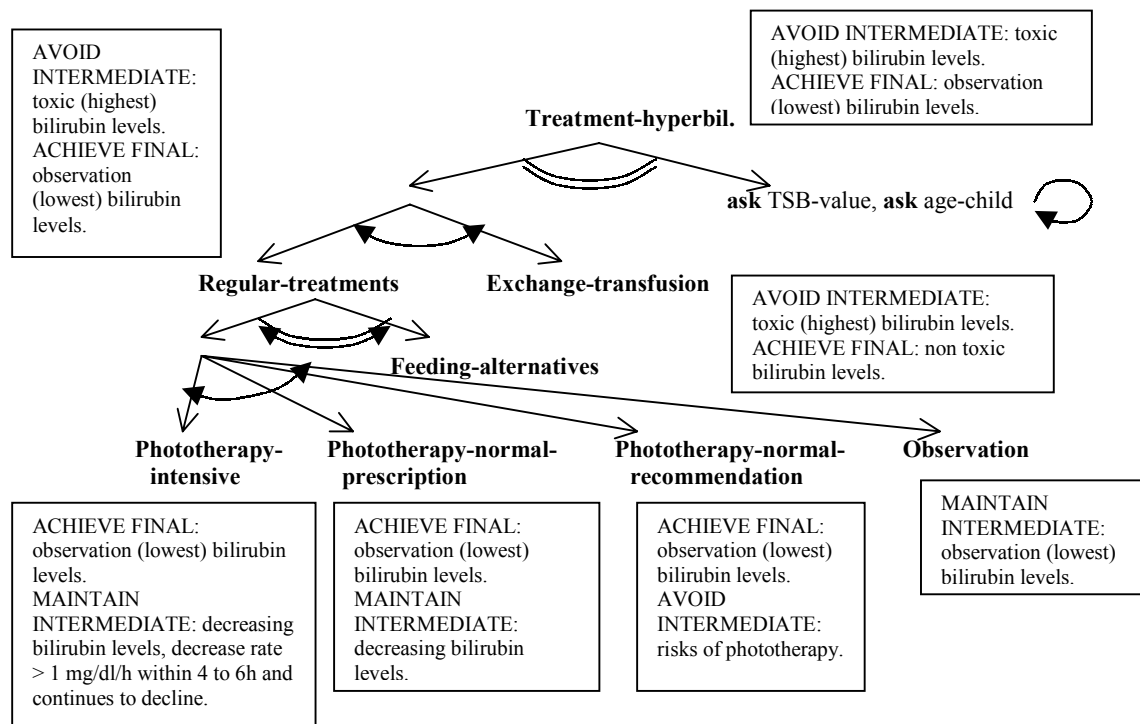
In the following paragraphs we describe the different protocol properties that have been considered promising from the medical point of view. The different properties are presented along with some significant examples we have identified in the two reference protocols, whenever it has been possible. Although this identification process has been driven to a great extent by the reference protocols, we are confident that the listed properties could in principle be applied to any other protocol.

Property 1: *Verification of correctness of intentions.*

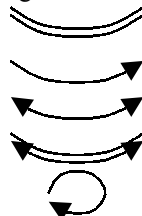
Type: correctness.

Rationale: The idea is ensuring that the intention of a plan follows from both the intentions of its subplans and the way these are composed. Although this property could be seen as an Asbru-dependent one, the fact that it relies on expert knowledge for a proper definition of intentions makes it more amenable to verification as a protocol-dependent property.

Example–Intentions of Jaundice treatments: The following figure shows the hierarchical decomposition of the treatment part in the Asbru version of jaundice protocol. The Treatment-hyperbilirubinemia plan consists of a part performing either Regular-treatments or Exchange-transfusion, in parallel with a cyclical measurement of bilirubin levels. The Regular-treatments plan will be used to carry out any of the phototherapy treatments below it, or simply an observation. In case of necessity, an exchange transfusion will be applied. The intentions of the different plans/clinical actions are shown in the boxes next to each plan label. Notice that these intentions, which were not part of the original protocol, have been added with the help of our medical partners.



Legend:



Parallel: parallel composition.

Sequentially: sequential composition.

Any-order: any sequential composition.

Unordered: any composition (everything is possible).

Cyclical: cyclical repetition.

Property 2: *Verification of correctness of a treatment.*

Type: correctness, actions (treatments).

Rationale: In some cases the medical expert has an idea, according to “good practice” criteria, of what a correct treatment should look like. Here we would verify that the protocol complies with these criteria. Concerning the kind of criteria, they could make reference to different treatment aspects (e.g. dosage) or to the expected outcome of a treatment. Note: Safety considerations concerning treatments are included in the next paragraph.

Example 1–Correct doses: When using 2 doses per day of insulin for Diabetes treatment, our specialist considers a good practice to distribute the morning and evening doses according to the ratio 2/3-1/3.

Example 2–Correct treatment application: In Jaundice treatment, in some protocols it is specified that during phototherapy the baby has to be put under the lamp with the maximum surface of the skin exposed to the light (i.e. no clothes at all).

Property 3: *Verification of safety of a treatment.*

Type: safety, actions (treatments).

Rationale: Again, good practice sometimes establishes the limits for the safety of a treatment, beyond which it could be considered harmful. The idea is verifying that the protocol complies with these criteria. These criteria will in general be related to the different aspects of a treatment.

Example 1–Maximal doses: In the Jaundice treatment, according to our specialist, it is not advisable to administer phototherapy for a time longer than 10 days.

Example 2–Safe dose increases: In the Diabetes treatment, according to our specialist, it is not advisable to increment drug/insulin doses too quickly unless the patient is in a critical situation. The idea is avoiding the negative effects of treatment overshoot, i.e. hypoglycemia episodes. More precisely, the dose should be titrated (i.e. increased by the smallest amount) in intervals of at least 2 to 4 weeks, unless the blood glucose is that high (e.g. >15 mmol/l) that a faster pace of dose titration is needed.

Property 4: *Verification of safety of a combination of treatments.*

Type: safety, actions (combination of treatments).

Rationale: The idea is verifying that applying the protocol does not result in a combination of treatments known to be dangerous for the patient, according to good practice.

Example: No example has been found for the moment.

Property 5: *Verification of different safety conditions.*

Type: safety.

Rationale: The general aim is trying to avoid non-desired situations, like side effects, no effect or overshoot of drugs. However, this can be too strong in general, because e.g. a drug with some inconvenient side effect can still be acceptable if it serves to the clinical goal. Notes: An alternative could be verifying more specific safety conditions, e.g. referring to patients with a certain type of complications. In any case, before concentrating on this type of properties, we have to analyse the possibilities of verification by formal methods.

Example: No example has been found for the moment.

Property 6: *Verification of correctness/safety properties about the management of complications of a patient group.*

Type: correctness/safety, groups of patients.

Rationale: Here the idea is verifying some properties necessary for the management of patients with special characteristics that require additional attention.

Example–Periodicity of controls in patients with complications: In the Diabetes treatment, it is advisable to monitor patients with complications more often (correctness property). For example, according to experts, individuals who have suffered a foot ulcer should have their feet inspected at least every 3 months

Property 7: *Verification of correctness/safety properties about the management of related diseases.*

Type: correctness/safety, interactions with related diseases.

Rationale: Here the idea is verifying properties referring to different types of interactions with the management of related diseases.

Example–Safe treatment in patients with hypertension: According to our expert, in the Hypertension treatment of patients with Diabetes it is not advisable to give beta-blockers to individuals who are unaware of hypoglycemia (safety property). The reason is that beta-blockers can make the patients (even more) unaware of hypoglycemia.

Property 8: *Verification of sequences of actions done by a physician.*

Type: traces of actions.

Rationale: The idea here is checking the actions performed by a physician to solve a particular patient case against the recommendations of the protocol for that case. This verification can be useful in both directions. First, if the protocol is used as golden standard, we will be able to determine whether the doctor complies with the standard recommendations. Second, if the physician's actions are seen as the standard behavior that the protocol should reflect, we will be validating the protocol. This second is the most interesting with the purpose of improving the quality of the protocol.

Example–Sequences of actions for Jaundice management: Here we will use the solutions of an expert to a collection of case data of jaundiced children, which we have used in previous work. The case data are of the form:

“...”

Another source could be another protocol for the management of jaundice in healthy infants, which includes a case problem with the corresponding solution.

Property 9: *Verification of correctness/safety properties expressed in terms of patterns of actions.*

Type: correctness/safety, traces of actions (patterns).

Rationale: The aim is verifying that the application of the protocol leads to sequences of actions that are considered correct according to good practice, or that it prevents sequences that are not desirable.

Example 1–Correct treatment: In Jaundice treatment, according to other existing protocols, in order to determine the evolution of the disease, once a phototherapy treatment has been applied, bilirubin measurements should be performed and not blanching skin tests (correctness property).

Example 2–Correct treatment: In Jaundice treatment, also according to other protocols, the baby should be hydrated before and during phototherapy (correctness property).

Property 10: *Verification of correctness/safety properties expressed in terms of indicators.*

Type: correctness/safety, indicators.

Rationale: The aim is verifying that the application of the protocol results in actions that comply with indicators of expected outcomes defined either in the protocol itself or by external sources.

Example–Indicators for Jaundice management: There exists a series of clinical indicators for Jaundice treatment that have been defined by the MAJIC Steering Committee (MAJIC stands for Making Advances against Jaundice in Infant Care). As an example, we can cite indicator 5:

“Measures whether infants with elevated bilirubin levels not responding to phototherapy receive an exchange transfusion”

The previous paragraphs describe the protocol properties that have been considered important from the medical point of view. Table 1 lists the different properties and summarises the category or categories in which they fall.

	correctness	safety	actions	traces of actions	groups of patients	interactions with related diseases	indicators	examples?
Property 1	×							YES
Property 2	×		×					YES
Property 3		×	×					YES
Property 4		×	×					NO
Property 5		×						NO
Property 6	×	×			×			YES
Property 7	×	×				×		YES
Property 8				×				YES
Property 9	×	×		×				YES
Property 10	×	×					×	YES

Table 1: Summary of protocol properties.

Although we have not been able to find examples of all of them in our reference protocols, we believe that these properties are potentially interesting for protocol verification in general. However, this fact, together with obvious time restrictions, limits the utilisation of the complete list of properties in our assessment project. Therefore, a choice had to be made. For this choice we have taken into account not only the potential applications of the verification properties but also our knowledge of the aspects to be verified. According to these criteria, we have decided to focus on the following properties:

- Property 1: verification of correctness of intentions
- Property 2: verification of correctness of a treatment
- Property 3: verification of safety of a treatment
- Property 8: verification of sequences of actions done by a physician

DELIVERABLES TABLE

Project Number: *IST-2001-33049*

Project Acronym: **PROTOCURE**

Title: *Improving medical protocols by formal methods*

Del. No.	Revision	Title	Type ¹	Classifi- cation ²	Due Date	Issue Date
D0	1	Project presentation	O*	Pub.	28/2/2002	26/2/2002
D1	1	Reference protocols and their Asbru models	R	Pub.	28/2/2002	28/2/2002
D2	1	Formal semantics of main Asbru elements	R	Pub.	31/3/2002	28/3/2002
D3	1	Desirable/required properties of main Asbru elements	R	Pub.	31/3/2002	3/4/2002
D3'		KIV formalisation	D	Pub.	31/5/2002	
D4		Verification of properties on the reference protocols	R	Pub.	30/9/2002	
D5		Evaluation of results	R	Pub.	30/11/2002	

¹ R: Report; D: Demonstrator; S: Software; W: Workshop; O: Other – Specify in footnote

² Int.: Internal circulation within project (and Commission Project Officer + reviewers if requested)

Rest.: Restricted circulation list (specify in footnote) and Commission SO + reviewers only

IST: Circulation within IST Programme participants

FP5: Circulation within Framework Programme participants

Pub.: Public document

* Report, webpage.