Improving the Execution of Clinical Guidelines and Temporal Data Abstraction in High-Frequency Domains

Peter Votruba[1], **Andreas Seyfang**[1], **Michael Paesold**[1], **Silvia Miksch**[1,2]

Abstract. The execution of clinical guidelines and protocols (CGP) is a challenging task in high-frequency domains such as Intensive Care Units. On the one hand, sophisticated temporal data abstraction is required to match the low-level information from monitoring devices and electronic patient records with the high-level concepts in the CGP. On the other hand, the frequency of the data delivered by monitoring devices mandates a highly efficient implementation of the reasoning engine which handles both data abstraction and execution of the guideline.

The language Asbru represented CGPs as a hierarchy of skeletal plans and integrates intelligent temporal data abstraction with plan execution to bridge the gap between measurements and concepts in CGPs.

In this paper, we present our Asbru interpreter, which complies abstraction rules and plans into a network of abstraction modules by the system. This network performs the content of the plans triggered by the arriving patient data. Our approach evaluated to be efficient enough to handle high-frequency data while coping with complex guidelines and temporal data abstraction.

1 Introduction

In the field of medicine, the application of clinical guidelines and protocols helps to improve the quality of care by ensuring the optimal choice of treatment. Clinical guidelines are "systematically developed statements to assist practitioner and patient decisions about appropriate health care for specific clinical circumstances" [5]. Such guidelines are based on the best empirical evidence available at the moment [11]. A guideline describes the optimal care for patients and therefore, when properly applied, it is assumed that they improve the quality of care.

A precondition for the successful application of clinical guidelines and protocols is automatic abstraction of context-dependent time-annotated raw-data (e.g., percent of oxygen in blood at a certain second) to high-level medical concepts (e.g., sufficient oxygen saturation during an extended period of observation). This is performed by temporal data abstraction.

In domains, like medicine, where comprehensive knowledge is available and human life depends on plan execution, only approved and validated procedures are admitted and direct control over the planning process is crucial. The user community, namely the medical staff, would refuse a system which does not offer maximum transparency and coverage safety issues in the decision process.

[1] Institute of Software Technology, Vienna University of Technology, Austria {peter, seyfang, silvia}@asgaard.tuwien.ac.at, michael@paesold.at

[2] Department of Information & Knowledge Engineering, Danube-University Krems, Austria

Within the domain of medicine, intensive care poses additional challenges to plan execution. Here, the data arrives at a high rate (at least 1 measurement for many input channels per second) and it is error-prone due to measuring constraints (e.g., some sensors attached are to the skin of the patient, whose movements may lead to wrong measurements). The nature of the data raise the need for elaborate pre-processing of the data, including complex knowledge-based plausibility checks and correction heuristics which involve multiple channels and time windows and the synchronisation of low-frequency and high-frequency inputs. This processing together with the sometimes complex propagation of plan states needs to be performed in a timely manner to meet the real-time requirements of online patient monitoring.

The safety considerations in this domain also demand that the same system is used in online monitoring and validation of the guideline used therefore by running it on a large set of test data (previous recordings of patient data).

To meet these requirements, we integrate time-oriented, skeletal planning using the Asbru representation with real-time monitoring and temporal data-abstraction.

In the following, we discuss related work in Section 2. The language used to represent a clinical guideline in computer-executable form is Asbru. It is described in Section 3. In Section 4 we explore the details of the interpreter, illustrated by an example. The evaluation and future work is described in Section 5 followed by the conclusion.

2 Related Work

Asbru is one of a wide range of computer interpretable representation languages for clinical guidelines. We give a short overview of related approaches here. See [18] for a detailed comparison.

EON was developed as a component-based, extensible architecture, using Protégé for its internal representation [14]. It offers advanced decision criteria in the Protégé Axiom Language (PAL) or using temporal queries in an external database. The EON architecture was used in the ATHENA Decision Support System (DSS) [10].

GLARE [21] is a domain-independent system with tools for the full life-cycle of clinical guidelines. The representation formalism builds on a limited but comprehensible set of primitives and is designed to cope with different types of temporal constraints, particular attention was paid to the role of periodic events. Guidelines are stored in a database, an XML representation is planned.

The Guideline Interchange Format (GLIF3) [17] stresses the importance of sharing guidelines among different institutions and systems. Guidelines can be designed at three levels: a conceptual, a computable, i. e. verifiable, and an implementable specification in-

tended for incorporation into clinical systems. An execution engine for GLIF3, named GLEE, was implemented recently [22].

NewGuide [4], a component-based multi-level architecture, integrates the formalized model of medical knowledge contained in clinical guidelines with workflow management, formally grounded in Petri nets.

PRO*forma* [6] combines logic programming and object-oriented modelling, using R_2L as its representation language. One aim of the PRO*forma* project is to explore the expressiveness of a deliberately minimal set of modelling constructs. Similar to Asbru, effects of plans are implemented in a computer interpretable form.

There are many guideline modelling approaches today, but only few integrate strong data abstraction resources. This may be caused by the fact that in most settings, data is entered manually which allows to delegate the data abstraction task to the user by demanding qualitative high-level input instead of the original data. However, the integration of guideline execution into the clinical data flow becomes more and more important in order to apply decision support systems in clinical daily practice [12].

There have already been two attempts to implement an execution engine for Asbru. The first implementation created by Bosse [3] translated the guideline into a representation suitable to be executed in Clips. The implementation was customised for a single clinical protocol and was therefore abandoned after the end of the project.

A more general implementation is AsbruRTM [8]. It has been used to test Asbru guidelines in intensive care [9]. Unfortunately, AsbruRTM only supports a subset of the available plan types in Asbru Light and does not integrate advanced (temporal) data abstraction.

Spock [23] is a system for application of guidelines in Hybrid-Asbru, which is a semi-formal guideline language that combines formal structure with description text. Spock is therefore not suited for fully-automated execution, but it can support a human agent applying a guideline. Spock is integrated with the IDAN architecture [2] and can utilize its temporal abstraction capabilities for decision support.

None of the above systems closely integrate plan execution and the required data abstraction into the clinical data flow in a high frequency domain such as intensive care. We therefore designed a seamless framework for the abstraction of data, the execution of plans and monitoring the environment for relevant changes [19, 16] as described in Section 4. It is complemented by a range of tools for the authoring and visualisation of various aspects of the guideline and the abstraction rules [1].

3 Representing Plans and Abstractions in Asbru

Asbru is a time-oriented plan representation language that describes clinical guidelines as skeletal plans [20]. Skeletal plans are plan schemata at various levels of detail, capturing the essence of the procedure, but leave room for execution-time flexibility in the achievement of particular goals [7]. Several knowledge roles are attached to a plan: preferences, intentions, conditions, effects and a plan body, which describes the actions to be taken.

Asbru's distinguishing features are that (1) intentions, conditions, effects and world states are temporal patterns, which allow reasoning about the contained knowledge; (2) actions and states can be continuous (durative); (3) the language allows to model temporal uncertainties, different granularities, and repeated patterns in events, actions, plans and world states; (4) plans are executed in sequence, all plans or some plans in parallel, or unordered with or without mutual exclusion; (5) because of the advanced (temporal) data abstraction capabilities, diagnosis and treatment can be tightly integrated allowing each one to support the other one.

All conditions for the transition from one plan state to another are expressed in terms of temporal patterns. A temporal pattern consists of one or more parameter propositions or plan-state descriptions. Each parameter proposition contains a value description, a context, and a time annotation. The time annotation used allows a representation of uncertainty in starting time, ending time, and duration of an interval. Start and end are defined as shifts from a reference point. Reference points can be defined as sets of cyclical time points or references to parameter changes, allowing repeated temporal patterns.

Asbru differentiates between seven plan states: considered, possible and rejected represent the plan-selection phase, activated, suspended, completed and aborted represent the plan execution phase.

Asbru plan libraries are written in XML and consist of two major parts, the domain definition section and the plans section. The plan section contains the plan definitions as described above. The domain definition defines both quantitative and qualitative parameters. These can be directly input or abstracted from other parameters. There is a wide range of abstractions available. They can be grouped into value abstractions which exclusively deal with the value dimension of measurements, and value aggregations where temporal abstractions are applied on the measurement. The result of the abstraction is accessed in the plans – both in conditions and assignments.

4 The Asbru Interpreter

Conceptually, the Asbru Interpreter consists of three basic units: data abstraction, monitoring, and plan execution. In the data abstraction unit, various temporal or atemporal abstractions are applied to the patient data to gain information at higher conceptual levels. The provided quantitative or qualitative data is monitored to detect temporal patterns in the abstracted data. This information is used to control the selection and execution of plans. This data flow is not unidirectional, instead, the execution unit can interact with both monitoring and abstraction unit to adjust the monitored patterns and to adapt the abstraction process to the context given by the current plan states.

The interpreter has two different modes of operation: Batch mode and interactive mode. In batch mode, a large set of records is read to validate a guideline against patient data or to create complex abstractions of the data for later analysis. In interactive mode, data can be read from monitoring devices in addition to the user input.

4.1 System Architecture

Figure 1 shows the parts of the interpreter on the implementation level. The main parts are the Asbru Compiler and the Execution Manager. The three conceptual components mentioned above – data abstraction, monitoring, and plan execution – are seamlessly integrated in the module graph.

At program start-up, the Asbru plan library XML file is compiled into a directed graph of modules. For each time step, the Execution Manager enacts each of the modules in the network to process patient data, monitor temporal patterns, and execute the plans in the guideline. These modules are largely compatible with each other, which allows information extracted by any module to flow back into the abstraction or monitoring process. To handle complex networks with many inputs in high-frequency applications, the Execution Manager ensures that each module is enacted exactly when needed, allowing for small time steps by some modules without the overhead created by other modules which would not provide new information at that moment.

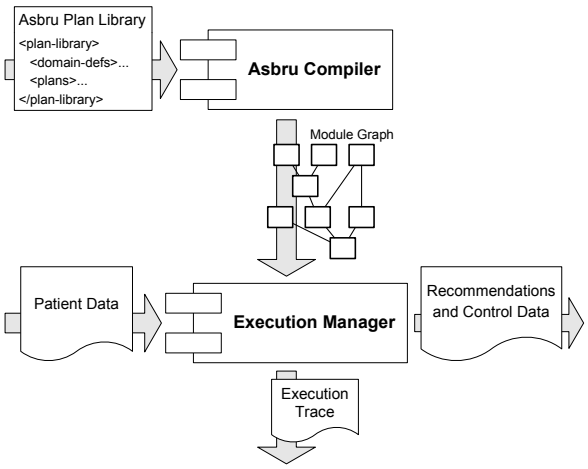

Figure 1. System architecture and data flow in the Asbru Interpreter. The Asbru plan library is compiled into a directed graph of modules by the Asbru Compiler. The Execution Manager uses this module network to process patient data and execute the plans representing the guideline. The output of the modules is provided in a uniform way, to be displayed in a graphical user interface. In addition, all state transitions are documented in a log file which is then translated to various table formats to ease the analysis of the execution path.

The output of the system can be displayed in a graphical user interface. In addition an interactive, graphical user interface currently under development will allow the stepwise control of plan execution. Using custom-built modules, the output can be integrated with the control of medical devices in a close-loop setting.

During operation, the interpreter writes an extensive log file documenting all abstraction steps and plan state changes. This is later transformed into various reports focussing on different aspects by easily customisable post-processing tools.

This is of particular importance when validating a clinical guideline against a set of patient data, which is the current main application of the interpreter.

4.2 Module Design

Each Asbru statement is translated to one or more modules. Some of these modules are simple, while others implement complex logic. At each time step, a module receives zero, one or more data points as input from its precursors in the abstraction graph and generates zero or one output data point. This output can be a simple time-stamped numeric value, or a complex structure such as the linear regression of a series of measurements, or the state of a plan together with synchronisation tokens for its children and parent plan. An important group of the modules – the temporal aggregation modules – produces output at a lower rate than its input, thus relieving its successors in the abstraction graph from the larger part of the data load by outputting high-level concepts such as "sufficient oxygen" only.

Modules can set alarms, to be triggered when a certain span of time is elapsed. Here we distinguish between *pre-alarms* and *post-alarms* depending on whether the alarm is triggered before or after processing the data for this time step. This distinction allows the implementation of both complex and convex temporal intervals. Alarms are set by various monitoring and value aggregation modules.

The available modules can be divided into the following categories.

Constant modules and system variables. These are the simplest category. They do not receive input. Constants modules are used to implement the constants found in an Asbru plan. System variables continuously produce values such as the current date.

Raw data modules. These modules interface the input channels and map the raw-data parameter definitions in Asbru. In real-time operation, they passively wait for arriving input data and return the next available value. In batch mode they read data from files.

Value abstraction modules. This group comprises the logical and arithmetic combination of inputs, and the mapping of quantitative values to qualitative categories.

Value aggregation modules. In order to map high-frequency, error-prone inputs to high-level concepts, it is mandatory to aggregate series of measurements and to derive the abstractions from them, and not from single measurements. Such aggregates can be descriptive statistics applied to moving time-windows, or more complex algorithms such as the *spread* [13].

Monitoring modules. These modules handle temporal patterns, such as parameter propositions, which control the state changes of Asbru plans. A parameter proposition has several states. Initially it is *not fulfilled*. As soon as an interval matches the time annotation, the parameter proposition becomes *fulfilled*. If the reference point is the symbolic value *now*, i. e. the current time point of evaluation, then the parameter proposition can become *no more fulfilled* in the future (provided that the condition or the context evaluate to true no more). A detailed discussion can be found in [16, 19]. Other Asbru features which are also covered by modules of this category are temporal constraints or constraint combinations.

Temporal abstraction modules. The patterns detected by monitoring modules and aggregates of the measurements often need one or more steps of temporal abstraction to detect complex patterns in the input data such as "five episodes of apoea followed by hyperoxemia during the previous hour".

Plan modules. Modules in this category represent Asbru plans or single plan steps. The network of parent and child plans is fully integrated with the other modules to from the module graph. Thus, the output from plan modules can be fed back into further abstraction steps.

The following actions are performed for each module: providing optional control data, attend to pre-alarms, process input data, attend to post-alarms and finally store the output of this module for use by other modules down the abstraction stream.

A plan module that represents a logical decomposition of plans needs to synchronise the plan modules representing its sub-plans. It does this by sending special data points to these child plan modules. Since the communication of plans with their children introduces cycles in the module graph, it is necessary to invoke plan modules more than once per data point received from the environment. Therefore, we divide each external *macro time step* into several internal *micro time steps*. This means that in-between processing two succeeding data points, the internal state propagation is propagated first. This does not introduce additional overhead as only the few concerned modules consume compute time in this phase.

The execution manager ensures that each module is only enacted upon new input. Therefore, a large number of complex low-frequency abstractions can be closely coupled with high-frequency modules. Further details are given in [16].

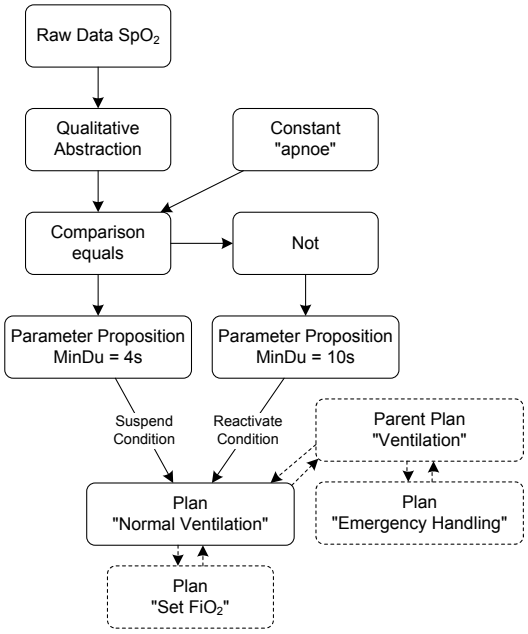

Figure 2. Sample module graph. The module graph maps directly to the described Asbru model. A raw data module has been created for the parameter `SpO2`, which is connected to a qualitative abstraction module. The resulting qualitative value is compared against a qualitative constant. This comparison module is connected parameter proposition module used for the suspend condition of the *normal ventilation* plan module. Another parameter proposition module is connected to the re-activate condition, which uses the negated output of the comparison module as input. The three other plan modules framed with a dashed line are shown to illustrate the context of this example, but will not be further explained in this paper.

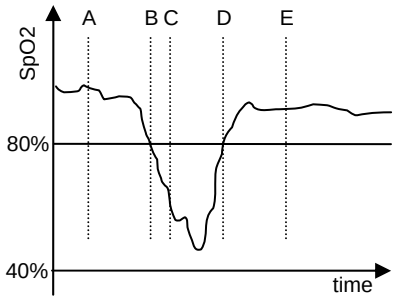

Figure 3. Sample input data. The graph shows the measurements of saturation of oxygen in blood (SpO2). The horizontal line shows the threshold between the qualitative region of apnoe (below 80%) and the other qualitative regions not relevant in our example. The dotted lines mark relevant time points described in the text.

4.3 Example: Ventilation in Neonates

From the field of artificial ventilation in neonates, we extract the following fragment of a protocol controlling the fraction of inspired oxygen based on measurements of partial pressure on oxygen in blood.

> *An external monitoring device measures the saturation of oxygen in blood SpO2). It delivers numeric values at a rate of 1 Hz. These values are abstracted to qualitative values. E.g., SpO2 below 80% is mapped to* apnoe*, while higher values are mapped to* decreased*,* normal *and* increased*. If the qualitative value of SpO2 equals* apnoe *for at least 4 seconds, then normal ventilation should be suspended. In this situation, the patient will receive emergency treatment by the medical staff. If the patient returns to less critical state, as defined by SpO2 being unequal* apnoe *for at least 10 seconds, normal ventilation is resumed."*

In Asbru, `SpO2` is a raw parameter. In addition, we introduce an abstracted qualitative parameter *SpO2-qualitative* with the possible values *apnoe*, *decreased*, *normal* and *increased*, where `apnoe` corresponds to $SpO2 < 80\%$. Furthermore, we create a plan called *normal-ventilation* with a suspend condition and a re-activate condition, both realised using parameter propositions. Each parameter proposition has reference point *now*. The first parameter proposition has the condition *SpO2-qualitative equal apnoe* and the time annotation *minimum-duration = 4 sec*. The second parameter proposition contains *SpO2-qualitative not-equal apnoe* and *minimum-duration = 10 sec*.

Figure 2 shows the relevant part of the module graph that is generated by the Asbru Compiler based on this specification. In terms of the overall concept, the monitoring task is performed by the parameter proposition modules, the data abstraction is provided by the modules feeding them, and the modules shown below the parameter proposition modules implement the plan execution.

In the following we discuss a typical series of events during abstracting and monitoring the input using the described modules. Figure 3 shows an excerpt from monitored saturation of oxygen in blood (SpO2).

Time point A represents one of many time steps during which the plan *normal ventilation* is activated and no changes are required. The value of `SpO2` is above the threshold (80%). Therefore, the comparison module does not create new output, after outputting *false* once at program start.

At time point B, the SpO2 value falls below the threshold. Therefore, the comparison module outputs `true`. The directly connected parameter-proposition module at the left detects a positive flank, i. e. a change from `false` to `true` in its input channel and sets an alarm to *current-time + 4s*.

At time point C (i. e. 4 seconds after B), the alarm set at time point B triggers and since the value of SpO2 did not change (i. e. no negative flank occurred), the parameter proposition module reports a found episode to the plan module. This means that the suspend condition of the plan gets fulfilled and the plan changes its state to `suspended`.

At time point D, the first parameter-proposition detects a negative flank (SpO2 is no more in the range of apnoe, therefore the value of the input changed from `true` to `false`) and changes its output to `no-more-fulfilled`. In this case, this input has no consequence for the state of the plan module. The second parameter proposition module detects a positive flank, as the comparison module outputs

false now which is inverted by the `not` module. Consequently, the parameter proposition module sets an alarm to *current-time + 10s*.

At time point E (i. e. 10 seconds after D), the previously set alarm for the second parameter proposition module triggers and since there was no negative flank for this module until then, this parameter-proposition reports now a found episode to the plan module. Therefore, the re-activate condition of the plan evaluates to `true` and the plan resumes (i. e. it changes its state back to `activate`).

5 Evaluation and Future Work

An extensive real-world guideline for breast cancer [15] was modelled in Asbru and test runs of the resulting model in the interpreter were successful. Running the interpreter on patient cases will allow the comparison between the expected outcome of applying the guideline and the actual outcome according to the model in Asbru.

This form of *validation* of the guideline is a very important link between the *verification* of the model, which takes today's verifiers to their limits because of the complexity of the model, and the discussion of prototypical execution paths with domain experts, which can only cover a small fraction of all possible paths.

Besides dealing with the low-frequency domain of breast cancer, practical tests with high-frequency data showed that the interpreter processes input from 10 channels and moderately complex abstractions thereof at a rate of more than 1 kHz on a standard PC. Most clinical data is recorded at 1 Hz, or 200 Hz. We therefore conclude that the computational performance is sufficient for real-time applications in clinical monitoring.

Future work will go into the construction of dedicated modules to interface equipment at intensive care units. Similar modules will allow the integration with other temporal abstraction systems. In addition, a graphical user interface to allow the interactive use of the interpreter by non-computer experts is under development.

6 Conclusion

Plan execution in real-world high-frequency domains such as intensive care units demand for tight integration of temporal data abstraction and plan execution to achieve the required intelligent reaction to unpredictable changes in the environment, i.e., the patient state.

While the knowledge in such domains is abstract, partly vague or incomplete, and often complex, the data arrives at a high rate and in a format that is far from the level in which the domain knowledge is specified. Translating the domain knowledge to such low levels by a knowledge engineer leads to well known short-comings regarding maintenance and assuring the correctness of the model.

We therefore designed an interpreter, which takes a high-level specification of skeletal plans and temporal data abstraction and compiles them into a network of abstraction modules. Using elaborate management of data flow, these modules process the data at a high rate, even in complex configurations.

Tests have demonstrated that the interpreter can handle a complete guideline, large sets of patient records, and high-frequency measurements. Applications to large sets of data are currently in progress.

ACKNOWLEDGEMENTS

We wish to thank the members of the Protocure II project, who provided us with essential feedback about using the Asbru Interpreter.

This work is part of the Protocure II project, which is supported by the European Commissions IST program, under contract number IST FP6-508794.

REFERENCES

[1] W. Aigner and S. Miksch, 'Supporting protocol-based care in medicine via multiple coordinated views', in *Proceedings of the Second International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2004)*, pp. 118–129. IEEE Computer Society Press, (2004).

[2] D. Boaz and Y. Shahar, 'A framework for distributed mediation of temporal-abstraction queries to clinical databases', *Artif Intell Med*, **34**(1), 3–24, (2005).

[3] T. Bosse, *An Interpreter for Clinical Guidelines in Asbru*, Master's thesis, Vrije Universiteit Amsterdam, 2001.

[4] P. Ciccarese, E. Caffi, L. Boiocchi, S. Quaglini, A. Kumar, and M. Stefanelli, 'A guideline management system', in *MedInfo 2004*, ed., M. Fieschi et al., pp. 28–32, Amsterdam, (2004). IOS Press.

[5] M. J. Field and K. H. Lohr, eds. *Clinical Practice Guidelines: Directions for a New Program*, Institute of Medicine, Washington DC, 1990. National Academy Press.

[6] J. Fox and S. Das, *Safe and Sound: Artificial Intelligence in Hazardous Applications*, MIT Press, 2000.

[7] P. E. Friedland and Y. Iwasaki, 'The concept and implementation of skeletal plans', *Journal of Automated Reasoning*, **1**(2), 161–208, (1985).

[8] C. Fuchsberger, *Entwicklung einer Ausführungseinheit für Asbru Light*, Master's thesis, Vienna University of Technology, 2003.

[9] C. Fuchsberger, J. Hunter, and P. McCue, 'Testing Asbru guidelines and protocols for neonatal intensive care', in *AIME 2005*, pp. 101–110. Springer, (2005).

[10] M. K. Goldstein, R. W. Coleman, S. W. Tu, R. D. Shankar, M. J. O'Connor, M. A. Musen, et al., 'Operationalizing clinical practice guidelines amidst changing evidence: Athena, an easily modifiable decision-support system for management of hypertension in primary care', *JAMIA*, **9**(6 Suppl 1), 11–16, (2002).

[11] R. Hanka, C. O'Brien, H. Heathfield, and I. E. Buchan, 'WAX ActiveLibrary: A tool to manage information overload', *Topics in Health Informatics Management*, **20**(2), 69–82, (1999).

[12] W. Horn, 'AI in medicine on its way from knowledge-intensive to data-intensive systems', *Artif Intell Med*, **23**(1), 5–12, (2001).

[13] S. Miksch, A. Seyfang, W. Horn, and C. Popow, 'Abstracting steady qualitative descriptions over time from noisy, high-frequency data', in *Artificial Intelligence in Medicine*, ed., W. Horn et al., pp. 281–290, Berlin, (1999). Springer.

[14] M. A. Musen, S. W. Tu, A. K. Das, and Y. Shahar, 'Eon: A component-based approach to automation of protocol-directed therapy', *Journal of the American Medical Informatics Association*, **3**, 367–388, (1996).

[15] Nationaal Borstkanker Overleg Nederland (NABON), *Guideline for the treatment of breast cancinoma*, Van Zuiden Communications B.V., 2002.

[16] M. Paesold, *Monitoring Temporal Patterns in Guideline Based Care*, Master's thesis, Vienna University of Technology, 2006.

[17] M. Peleg, A. Boxwala, O. Ogunyemi, et al., 'GLIF3: The evolution of a guideline representation format', in *Proc AMIA Annu Fall Symp*, pp. 645–649, (2000).

[18] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R. Greenes, R. Hall, P Johnson, N. Jones, A. Kumar, S. Miksch, S. Quaglini, A. Seyfang, E. Shortliffe, and M. Stefanelli, 'Comparing computer-interpretable guideline models: A case-study approach', *JAMIA*, **10**(1), (2003).

[19] A. Seyfang, *An Integrated System for Temporal Data Abstraction to Facilitate Guideline Execution and Knowledge-Based Data Analysis*, PhD dissertation, Vienna University of Technology, 2006. To appear.

[20] A. Seyfang, R. Kosara, and S. Miksch, 'Asbru 7.3 reference manual', Technical report, Vienna University of Technology, (2002).

[21] P. Terenziani, G. Molino, and M. Torchio, 'A modular approach for representing and executing clinical guidelines', *Artif Intell Med*, **23**(3), 249–276, (2001).

[22] D. Wang, M. Peleg, S. W. Tu, A. A. Boxwala, et al., 'Design and implementation of the GLIF3 guideline execution engine', *J Biomed Inf*, **37**, 305–318, (2004).

[23] O. Young and Y. Shahar, 'The Spock system: Developing a runtime application engine for hybrid-Asbru guidelines', in *AIME 2005*, pp. 166–170. Springer, (2005).