# Integrating Diagnosis and Treatment in a Flexible Way

**Andreas Seyfang** and **Silvia Miksch**
Institute of Software Technology,
Vienna University of Technology, Austria
E-mail: {seyfang, silvia}@ifs.tuwien.ac.at

**Abstract.** *There are many approaches to support diagnosis and decision support in the medical domain. There are also some systems modelling therapy plans. But currently there is only little connection between the two fields. Still, diagnosis and treatment are interdependent in various ways.*

*After describing the connections between these two tasks, we give an overview of the Asgaard framework. Finally, we demonstrate the representation of both diagnosis and treatment in Asbru—the plan representation language of the Asgaard system.*

## 1 Introduction

In the medical domain two developments take place. First, an increasing number of measuring devices deliver an increasingly complex stream of raw data which is often difficult to interpret. Second, both care providers and patient advocates call for ”optimal” care. This means that any actions taken by the medical staff should lead to minimal cost and best patient outcome—sometimes conflicting goals which are most often difficult to fulfill.

The first problem—the interpretation of data—is handled by temporal data abstraction. It transforms measured low-level data into abstract, high-level information. E.g., it abstracts the information ”the oxygen content of the patients blood is reduced” from a stream of numbers.

The second problem—finding the optimal treatment—is handled by therapy planning. In the USA and most European countries, medical organizations installed concensus boards issuing clinical guidelines and protocols for a broad range of situations or diseases. These guidelines ensure the flow of information from recent research to common practice, they help to save costs by informing physicians about cost effective alternatives, and they provide safe ground for physicians defending their decisions.

Data abstraction and therapy planning form a feedback cycle. On the one hand, data abstraction is always context dependent, i.e., one and the same value can mean different things under different contexts. This context is formed by the current therapeutic situation, i.e., it is supplied by executing therapy plans. On the other hand, the execution of therapy plans must be synchronized with the actual state of the patient. Data abstraction is the crucial foundation of such a synchronization process since the treatment plans use high-level concepts to describe the patient’s state. Since the synchronization of plans and patient states involves complex temporal reasoning, it is often considered a process of its own right, called patient monitoring.

In Section 2 we describe the issues in data abstraction and therapy planning. In Section 3 we show related work. In Section 4 we describe the Asgaard system and in Section 5 we give a practical example.

## 2 Related Work

Common methods of intelligent data analysis are time-series analysis, control theory, and probabilistic or fuzzy classifiers [2]. These approaches have a lot of shortcomings, which lead to applying knowledge-based techniques to derive qualitative values or patterns of current and past situations of a patient. The RÉSUMÉ project [12] performs temporal abstraction of time-stamped data without predefined trends. The system is based on a knowledge-based temporal-abstraction method. Bellazzi et al. [1] utilize Bayesian techniques to extract overall trends from cyclic data in the field of diabetes.

All these approaches are dealing mostly with low-frequency data, i.e., only few data items per day have to

be processed. Promising approaches for high-frequency data are the Time Series Workbench [4] which approximates data curves through a series of line-segments and VIE-VENT [5]. The temporal abstraction module of VIE-VENT focuses on the high-frequency domain of artificial ventilation of newborn infants. VIE-VENT incorporates fixed, but context-dependent temporal data abstraction schemata.

During the past 20 years, there have been several efforts to support complex guideline- and protocol-based care over time in automated fashion in automated fashion as explicit, well-defined plans. Examples of specialized architectures include EON [6], Careflow [9], PROforma [3], and the guideline interchange format (GLIF) [8].

A number of methods have been applied to represent clinical protocols and guidelines, including free text, flowcharts, decision tables, and medical logic modules [7]. However, the large domain knowledge available is often too uncertain and vague. To represent the implicit medical knowledge with these techniques explicitly is a nasty and intractable problem. Therefore, most protocol- based care systems basically consider protocols as composition of actions (which are to be performed) and conditions (which define when it is appropriate to perform the actions).

To overcome limitations observed in previous approaches, namely to arrive at a system combining temporal data abstraction with the execution of skeletal, time-oriented plans representing clinical treatment plans, the *Asgaard* framework was developed.

# 3 Our Solution

To handle the above described problems, the Asgaard framework [11] was developed. It outlines task-specific problem-solving methods to support both design and execution of skeletal plans. This project tries to build a bridge between the medical approaches and the planning approaches, addressing the demands of the medical staff on the one side and applying rich plan management on the other side. For the representation of plans, a time-oriented, intention-based, skeletal plan-representation language, called Asbru, was jointly developed by Peter Johnson, Silvia Miksch, and Yuval Shahar at the Stanford University in 1996. The current version of Asbru, 7.2, is based on XML (eXtensible Markup Language) which allows the use of a series of public domain tools for editing, parsing, and low-level verification. Details can be found at `http://www.ifs.tuwien.ac.at/asgaard/`.

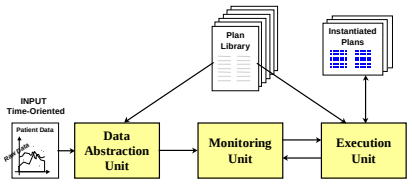


Figure 1: Runtime Modules of Asgaard.

Figure 1 shows those parts of the Asgaard framework which deal with the execution of plans including the data abstraction unit.

**Data Abstraction Unit:** The data abstraction unit receives the data from the input sources and processes it through a set of abstraction methods.

**Monitoring Unit:** The monitoring unit receives a list of parameter propositions (i.e., parts of conditions) which are relevant for future plan-state transitions from the execution unit and compares it to the high-level abstractions delivered by the data abstraction unit. If it detects a match, it informs the execution unit which performs the resulting state transitions.

**Execution Unit:** The execution unit handles the state transitions of the plans. It instantiates plans from the plan library and governs their life-cycle according to the state of the world as reported by the monitoring unit.

The data abstraction is specified in the header of the plan library. Each of the abstraction steps is described in one *parameter definition* including its parameters and the connections to other steps. This way, the data abstraction and the plan execution are seamlessly configured in a single file.

# 4 A Practical Example

In this section we present a practical example of the use of the data abstraction unit.

In artificial ventilation of neonates the setting for the fraction of inspired oxygen ($FiO_2$) must be adjusted according to the saturation of oxygen in the blood of the patient. This is measured by pulsoximetry, which delivers one measurement per second. These measurements show oscillations of various origins including technical issues of measurement. In contrast, frequent changes in the supply of oxygen are considered infavourable from the medical point of view. Any change in the $FiO_2$ takes

more than 5 minutes to take effects in the patients body, so more frequent changes are not admitted. Therefore, the measurements cannot be used to directly control the oxygen supply.

Bridging the gap between the high-frequency noisy measurements and the longer term perspective on the oxygen supply is performed by abstracting qualitative values using the spread algorithm. Mimicking the physicians practice, we observe the previous 5 minutes, i.e., we calculate a spread over 5 minutes. Based on it, we abstract a qualitative value of type `spo2-scale` which takes one of the values `substantially-below`, `slightly-below`, `normal`, `slightly-above`, and `substantially-above`. The following Asbru code defines this.

```
<qualitative-scale-def name="SpO2-scale">
  <qualitative-entry entry="substantially-below"/>
  <qualitative-entry entry="slightly-below"/>
  <qualitative-entry entry="normal"/>
  <qualitative-entry entry="slightly-above"/>
  <qualitative-entry entry="substantially-above"/>
</qualitative-scale-def>
```

The measurements are read in from the pulsoximeter which supplies them tagged as ”SPO2” in the stream of measurements. This association is defined in the tag *channel-name*. Minimum and maximum values ensure that any values which are not within this range of plausible values are marked invalid and are not processed further.

```
<parameter-def name="raw-data" type="percent">
  <raw-data-def channel-name="SPO2"
                mode="automatic" unit="%">
    <minimum-value>
      <numerical-constant unit="%" value="40"/>
    </minimum-value>
    <maximum-value>
      <numerical-constant unit="%" value="100"/>
    </maximum-value>
  </raw-data-def>
</parameter-def>
```

Based on the `raw-data`, a spread is calculated every second for the previous 5 minutes. If more than 70% of the values in this time interval are missing or invalid, the spread is not calculated and the result is marked invalid instead. This is because the result of a calculation based on too few values might not represent the entity which is hidden behind the missing values. In the following, we first show the definition of the spread and then the definition of the qualitative values based on this spread.

```
<parameter-def name="state-spread" type="spread">
  <spread-def>
    <time-window-size>
      <numerical-constant unit="min" value="5"/>
    </time-window-size>
    <step-width>
      <numerical-constant unit="s" value="1"/>
    </step-width>
    <minimum-of-valid-points>
      <numerical-constant unit="%" value="70"/>
    </minimum-of-valid-points>
    <parameter-ref name="raw-data"/>
  </spread-def>
</parameter-def>
<parameter-def name="state" type="SpO2-states">
  <qualitative-parameter-def>
    <limits unit="%">
      <context>
        <any/>
      </context>
      <limit-entry value="80.0"/>
      <limit-entry value="86.5"/>
      <limit-entry value="89.5"/>
      <limit-entry value="93.5"/>
      <limit-entry value="96.5"/>
      <limit-entry value="100.0"/>
    </limits>
    <parameter-ref name="state-spread"/>
  </qualitative-parameter-def>
</parameter-def>
```

Controlling the oxygen supply based on these qualitative values is not too straightforward and cannot be included in this paper. See [10] for the details of the complete setting to controll the $FiO_2$ based on the $SpO_2$ delivered by the pulsoximeter.

The following example illustrates the ordering of subplans i.e. treatment steps in the a plan. The plan I-RDS (infants' respiratory distress syndrome) starts by a subplan `initial-phase`. Then is takes one of three options, one at a time depending on their individual filter-condititions: `controlled-ventilation`, `permissive-hypercarpnia`, and `crisis-management`. If one of these fails, it tries another one, until finally the `controlled-ventilation` completes successfully. It then pursues by starting the subplan `weaning`.

Of course, this is again a simplyfied excerpt of what is necessary in a real clinical setting.

```
<plan name="I-RDS">
  <plan-body>
    <subplans type="sequentially">
      <wait-for>
        <all/>
      </wait-for>
      <plan-activation>
        <plan-schema name="initial-phase"/>
      </plan-activation>
      <subplans type="any-order"
                retry-aborted-children="yes">
        <wait-for>
          <static-plan-pointer
                plan-name="controlled-ventilation"/>
        </wait-for>
        <plan-activation>
          <plan-schema
                name="controlled-ventilation"/>
```

```
        </plan-activation>
        <plan-activation>
          <plan-schema
                name="permissive-hypercapnia"/>
        </plan-activation>
        <plan-activation>
          <plan-schema name="crisis-management"/>
        </plan-activation>
      </subplans>
      <plan-activation>
        <plan-schema name="weaning"/>
      </plan-activation>
    </subplans>
  </plan-body>
</plan>
```

# 5 Conclusion

Currently, the medical staff is suffering from information overload which range from the problems of data interpretation (like, arriving at trustable data, abstraction time-oriented qualitative information from quantitative data) to complex treatment management (like, intertwined diagnosis and treatment, exception handling, crisis management).

We presented a solution to overcome such difficulties. Our approach is part of the Asgaard project, which outlines task-specific problem-solving methods to support both design and execution of skeletal plans. Clinical protocols are becoming widespread in the medical domain to guarantee particular health care standards with appropriate costs. Our Asgaard system represents protocols as time-oriented, skeletal plans. It provides tools for the various problem-solving methods needed during design and execution time of protocols. An integrated framework for all the aspects involved is provided and a common representation language – Asbru – is used. Therefore, the Asgaard system ensures the beneficial use of computer support for all the tasks involved in the design, application, and improvement of protocols.

The system presented supports the data interpretation on the one hand, and simplifies the complex treatment management, which is based on the derived qualitative descriptions, on the other hand. These result in an easy to use and to comprehend treatment recommendations.

**Acknowledgments.** This project is supported by ”Fonds zur Förderung der wissenschaftlichen Forschung – FWF” (Austrian Science Foundation), P12797-INF.

# References

[1] R. Bellazzi, C. Larizza, P. Magni, S. Montani, and G. De Nicolao, ‘Intelligent analysis of clinical time series by combining structural filtering and temporal abstractions’, in *Artificial Intelligence in Medicine*, ed., Horn, W. et al., pp. 261–270, Berlin, (1999). Springer.

[2] M. Berthold and D.J. Hand, *Intelligent Data Analysis: An Introduction*, Springer, Berlin, 1999.

[3] J. Fox, N. Johns, and A. Rahmanzadeh, ‘Disseminating medical knowledge: the proforma approach’, *Artificial Intelligence in Medicine*, **14**, 157–181, (1998).

[4] J. Hunter and N. McIntosh, ‘Knowledge-based event detection in complex time series data’, in *Artificial Intelligence in Medicine*, ed., Horn, W. et al., pp. 271–280, Berlin, (1999). Springer.

[5] S. Miksch, W. Horn, C. Popow, and F. Paky, ‘Utilizing temporal data abstraction for data validation and therapy planning for artificially ventilated newborn infants’, *Artificial Intelligence in Medicine*, **8**(6), 543–576, (1996).

[6] 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 Information Association*, **3**(6), 367–88, (1996).

[7] E. Pattison-Gordon, J. J. Cimino, G. Hripcsak, S. W. Tu, J. H. Gennari, N. L. Jain, and R. A. Greenes, ‘Requirements of a sharable guideline representation for computer applications’, Technical Report Report No. SMI-96-0628, Stanford University, (1996).

[8] 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 2000 AMIA Annual Symposium (AMIA 2000)*, ed., O. M.J, (2000).

[9] S. Quaglini, M. Stefanelli, G. Lanzola, V. Caporusso, and S. Panzarasa, ‘Flexible guideline-based patient careflow systems’, *Artificial Intelligence in Medicine*, **22**(1), 65–80, (2001).

[10] A. Seyfang, S. Miksch, W. Horn, M. Urschitz, C. Popow, and C. Poets, ‘Using time-oriented data abstraction methods to optimize oxygen supply for neonates’, in *Artificial Intelligence in Medicine*, ed., Quaglini, S. et al., Berlin, (2001). Springer. forthcoming.

[11] 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**, 29–51, (1998).

[12] Y. Shahar and M. A. Musen, ‘Knowledge-based temporal abstraction in clinical domains’, *Artificial Intelligence in Medicine*, **8**(3), 267–298, (1996).