

Asgaard's Contribution to the Guideline Representation Comparison

Andreas Seyfang and Silvia Miksch
Institute for Software Technology
Vienna University of Technology, Austria

February 10, 2001

Preface

Let us first consider three pairs of terms, which will highlight most of the particularities of Asgaard.

Discourse model vs. process model. These are two ways to look at the patient and her course of disease: The first way is to consider the situation of the patient consulting the doctor, starting with some information acquisition, followed by diagnosis, and finished by prescribing some drugs or treatment and maybe ordering the patient to come again. The second way is to consider the treatment of a patient as a single process, consisting of different treatments interleaving with repeated consultations with the physician. While the reasoning strategies of many physicians are situation-based, Asgaard adopts the process-model perspective, because it makes dependencies explicit.

Diagnosis vs. treatment. To cure a patient, both diagnosis and treatment are indispensable. However, the focus of most protocols and guidelines currently available lies more on diagnosis than on treatment. Asgaard models diagnosis as the abstraction of (qualitative) parameters and treatment as skeletal plans. The case studies provided meet the characteristics mentioned before. Therefore, more diagnostic parts are modeled than such concerning treatment.

Guideline vs. protocol. The terms "guideline" and "protocol" are not well-defined. We adopt the following definitions. The Institute of Medicine (IOM) defined clinical guidelines as "systematically developed statements to assist practitioner and patient decisions about appropriate health care for specific clinical circumstances". We define a clinical protocol as a more detailed version of a clinical guideline referring to a certain class of therapeutic interventions. Asgaard's focus is on protocols, not on guidelines.

This means that our strength lies in the modeling of detailed flows of actions, including all the uncertainty involved. Unfortunately, there are no protocols in the strict sense, but only guideline available today, so it is indispensable to consult domain experts to arrive at complete protocols. For the present two guidelines, this was not possible.

1 A Short Introduction to Asgaard and Asbru

The Asgaard framework [20] 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 developed [19]. Asbru is used to define skeletal plans which are instantiated during plan execution.

1.1 Asgaard's Problem-Solving Methods

Asgaard's problem-solving methods correspond to the tasks of the plan management [11]. These tasks are to be accomplished in a highly interleaved way. Some of them are mostly done during *design time* and others during *execution time* of clinical protocols.

1.1.1 Methods Mostly Done at Design Time

Plan Visualization: is a graphical way to design and browse the topological and the temporal aspects of the interconnected plans;

Advanced Plan Editing: is an editor guided by the knowledge roles to give full access to expert users, who tune features of the knowledge roles;

Plan Verification: examines the correctness of interrelated clinical plans by a three-level detection of anomalies (method semantics);

Plan Validation: compares the intended states against the prescribed actions and intended plans (domain semantics);

Plan-Scenario Testing: applies scenarios of plans to test their functionalities and their course of activities.

1.1.2 Methods Mostly Done at Execution Time

Plan Selection: chooses applicable clinical plans from the plan library according to the patient's state, the plan's overall intentions, and plan effects;

Plan Instantiation: adjusts the parameters of a clinical plan according to the patient data record and the medical environment;

Data Abstraction: transforms information obtained from sensors or user input into a format suitable for the monitoring module, which consists of three tasks: (a) data validation, (b) calculation of derived values, and (c) transformation of time-stamped data into qualitative information.

Monitoring: synchronizes the execution of plans with the state of the real world by communicating the results of data abstraction to the plan execution, thereby sharing some activities with the latter.

Plan Execution: maps plans and actual situations in the medical environment which is done on three distinct layers: (a) plan synchronization, (b) plan adaptation, and (c) replanning.

Plan Critiquing: consists of two parts: (a) recognition of intentions: Why is the executing agent executing a particular set of actions, especially if those actions deviate from the plan's prescribed actions; (b) critique of the executing agent's actions: Is the executing agent deviating from the prescribed actions or intended plan? Are the deviating actions compatible with the author's plan and state intentions?

Plan Evaluation: examines retrospectively, if the executed plans achieved the desired effects according to the patient's state, intentions, executed actions, and plan effects;

Execution Visualization: visualizes, which plans have been executed (when and how) together with the patient status;

Plan Rationale/History: is logging events, states, intentions and performed actions; generates on-line help information and explanations about plan selection, instantiation, execution states, success and failure of plans integrating the domain-specific annotations.

The Asgaard project is a research project in progress. The overall framework of plan management within Asgaard is completely defined, however, we have not employed all problem-solving methods in detail yet. In this article we will describe only those methods which are used to author clinical protocols. See [13] for a description of the execution modules, [14] for data abstraction, [3] for verification, and [20] for critiquing.

1.2 Plan Representation: Asbru

To represent the clinical protocols, we have developed a time-oriented, intention-based, skeletal plan-specification language, called **Asbru** [15, 19]. Clinical protocols are represented as time-oriented, skeletal plans. Skeletal plans are plan schemata at various levels of detail, which capture the essence of the procedure, but leave enough room for execution-time flexibility in the achievement of particular goals [5]. Thus, they are usually reusable in various contexts. In Asbru, we have enriched the idea of skeletal plans by adding knowledge roles, a rich set of ordering of actions and plans, and temporal dimension of states, actions, and plans. Therefore, Asbru enables, on the one hand, the designer to represent both the prescribed actions of a clinical protocol and the knowledge roles required by the various problem-solving methods performing the intertwined supporting sub-tasks, and, on the other hand, the executing agent (e.g., the physician) to monitor the patients and the applicability of clinical protocols in parallel leading to a more flexible dialog and to a better acceptance of systems by the medical staff.

Time-oriented, skeletal plans are uniformly represented and organized in the *plan-specification library*. Atomic plans—called *actions* or *operators* in the planning literature—are either user performed or implemented by external software objects. Each non-atomic plan in the plan-specification library is hierarchically composed of a set of plans with arguments and time annotations (representing the temporal scope of a plan). A plan is identified by its name and consists of five knowledge roles: *preferences*, *intentions*, *conditions*, *effects*, and a *plan body (layout)* which describes the steps to be taken. The major features of Asbru are that

- prescribed actions and states can be continuous;
- intentions, conditions, and world states are temporal patterns;
- uncertainty in both temporal scopes and parameters can flexibly be expressed by bounding intervals;
- particular conditions and operators are defined to monitor the plans' execution; and
- explicit intentions and preferences can be stated for each plan separately.

The basic syntactic construct is the temporal pattern. 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 parameter name, a value description, a context description and a time annotation. The time annotation allows the representation of uncertainty in starting time, ending time, and duration. The time annotation supports multiple time lines (e.g., different zero-time points and time units) by providing arbitrary *reference annotations*. Temporal shifts from the reference annotation are used to define the uncertainty in starting time, ending time, and duration. To allow temporal repetitions, sets of cyclical time points and cyclical time annotations are defined.

Asbru was jointly developed by Peter Johnson, Silvia Miksch, and Yuval Shahar at Stanford in 1996. The current version of Asbru is 7.2.

1.3 Authoring Protocols with Asbru

1.3.1 Plan Visualization: AsbruView

Applying (semi-)formal methods to represent and acquire clinical protocols implies that such protocols have to be written in powerful languages. However, such notations are very difficult to

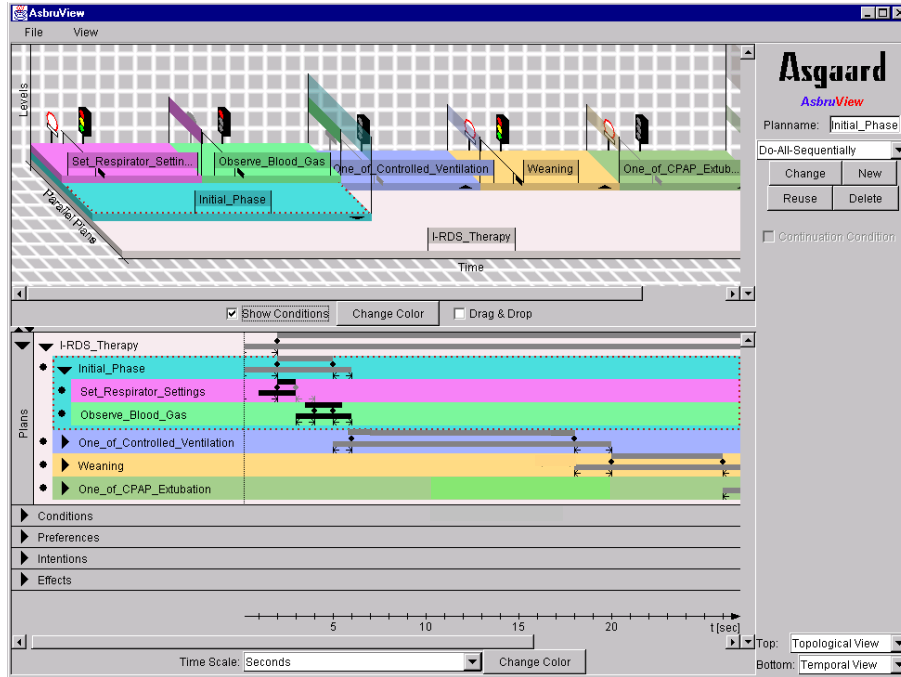


Figure 1: Authoring the clinical treatment protocols for Infant's Respiratory Distress Syndrome (I-RDS) with AsbruView.

read for the medical domain experts. Powerful methods are useless, if they cannot be used by the people they are intended for. Therefore, an appropriate user interface to the plan specification is needed which is easy to handle and which communicates the underlying concepts and issues to the domain users. Graphical techniques, like flow charts, graphical animation languages, visual (programming) languages, and process modeling techniques have the drawbacks that the visualization of temporal dimensions of observations, goals, and plans as well as the uncertainty of occurrences are limited. Complementarily, LifeLines [16] provide an excellent way to represent data and actions over time. Therefore, we developed a plan visualization, which employs metaphor graphics and extends the idea of LifeLines to be able to handle the above mentioned temporal dimensions [10]. Our plan visualization, called AsbruView, consists of two views (compare Figure 1).

- a topological view, which utilizes the metaphor graphics of "running tracks" and "traffic" and
- temporal view, which captures the temporal uncertainty and explains the plans, their sub-plans, and their components in more detail enhancing the idea of LifeLines.

Alternative metaphors of our problem domain could be "road maps" or "golf courses". However, we have chosen metaphors of "running tracks" and "traffic", which seemed easier to comprehend and more appropriate to our domain experts.

1.3.2 Standard Plan Editing: XML-Editors

Asbru originally had a LISP-like syntax specified in Backus-Naur form (BNF), which offered little support by available tools. Therefore, we refined and improved Asbru rewriting it in an XML¹-based language, called Harmless Specification Language (HSL). The syntax description of Asbru specified in HSL is input to Pontifex [9], which automatically creates (a) a Document

¹eXtensible Markup Language

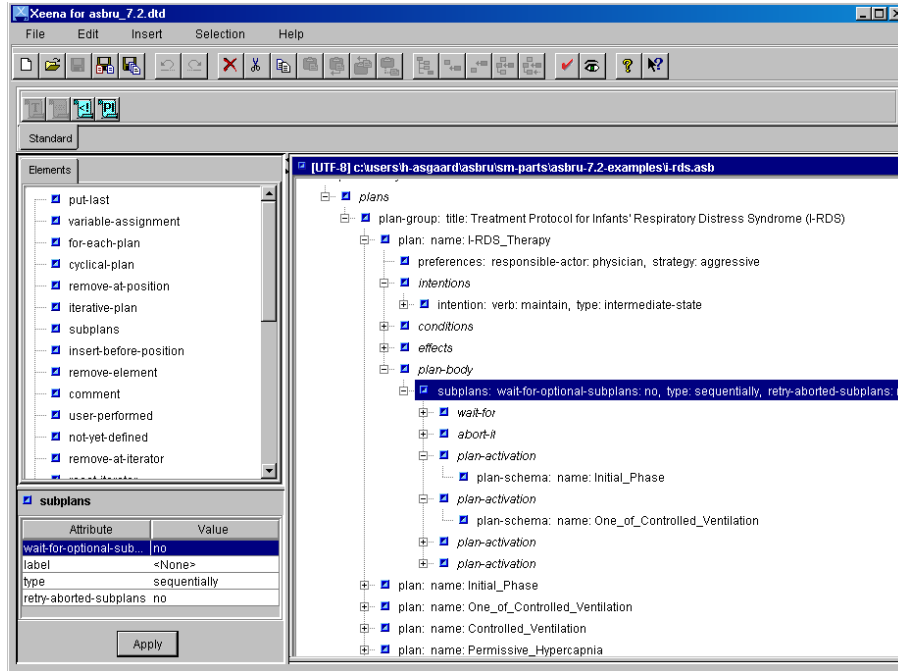


Figure 2: Authoring the clinical treatment protocols for Infant’s Respiratory Distress Syndrome (I-RDS) using Asbru’s DTD and the IBM’s Xena Editor.

Type Description (DTD), (b) classes and listeners related to the defined elements in the DTD, (c) a SAX-based parser to transform Asbru-code into these classes, (d) an HTML file containing documentation of the language, and (e) additional documentation material [7]. One benefit of the DTD-based syntax specification is that we can simply use any publicly available XML-Editor to write Asbru. Figure 2 shows the same part of clinical protocol of Figure 4 using the IBM’s Xena Editor. Additionally, the generated parser examines the pure syntactical correctness of clinical protocols written in Asbru and the created classes and listeners are a prerequisite of the other Asgaard’s problem solving methods providing a uniform representation in an object-oriented fashion.

1.3.3 Advanced Plan Edition: PIXEE

General purpose XML-editors do not cover the whole range of functionality desired to support the protocol modeling process. We are therefore developing an editor which utilizes additional information about the language given in HSL to provide flexible ways to focus on different aspects of plans for particular purposes. The focus on a particular aspect of the plan at a time is crucial given the complex nature of clinical guidelines and protocols and thus Asbru plans. The editor is currently under development.

1.4 Plan Verification - Verifying Clinical Protocols

Safety and transparency aspects are very critical issues not only in the medical domain [6]. 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. These features are understandably not part of the verification methods, described in [1, 17, 18].

We therefore proposed a partial and heuristic AI verification approach to identify particular fault classes, called anomalies. We apply the general verification methods (e.g. [17, 18]) for the non-domain-specific parts of clinical protocols, and design additional verification methods for the domain-specific parts. Our verification approach examines three levels of a plan: the plan itself, all its knowledge roles and all its subplans. The outcomes of our verification process are legal or meaningful plans, instead of insisting on complete or totally correct plans [3]. Our plan verification complements the clinical studies the medical team already performs to derive the clinical protocols.

2 Samson’s Dimensions in Asbru

2.1 Representation of Patient Data

In Asgaard, there is a principal distinction between data which is time-stamped and data which is not. Time-stamped data are called *parameters* and extensive reasoning over past states of a parameter are implemented. Timeless data is stored in variables which behave like variables in a programming language—past values cannot be retrieved.

2.1.1 Time-stamped Data: Parameters

Parameters are input channels of the system shared by all plans. Raw data is feed into the data abstraction unit from various input sources: monitoring devices, user input, and files containing recorded data. The data abstraction unit produces various types of abstracted parameters based on the raw data according to the *domain definitions* in the plan library.

Parameters are referred to by the *parameter propositions* in the conditions controlling the state transitions of plans. These parameter propositions describe both the (desired) value of the parameter and the temporal extend of the interval during which the parameter has that value. In addition, the parameter proposition contains the *context* of the abstraction. The context is formed by one or more context variables (or dimensions). A parameter proposition is not fulfilled unless the parameter has the described value under the given context during the specified time.

One of the most fundamental data abstractions is the context-dependent transformation of quantitative values into qualitative ones. This is crucial to arrive at high-level reasoning such as performed by physicians. They do not talk about fever exceeding 37.0 degrees Celsius or blood pressure exceeding 140 mm HG but about *increased* temperature and *high* blood pressure. As described in the end of the proposed study, "high" depends on the context, i.e., whether the patient has Diabetes Mellitus, proteinuria, etc.

The time mentioned above is the *valid time*, not the transaction time. To ensure that decisions need not be revoked, a maximum delay between valid time and transaction time is defined for each parameter.

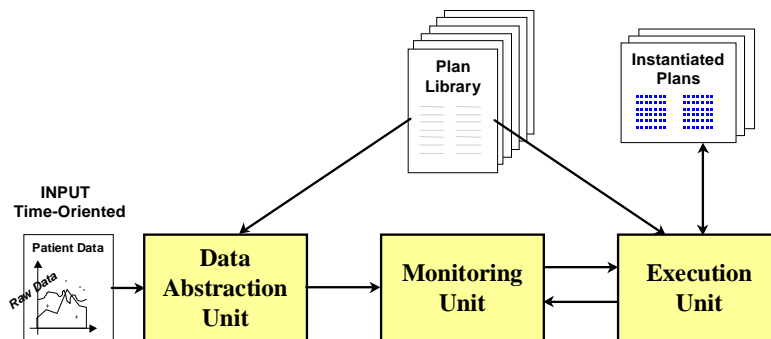


Figure 3: Runtime Modules (Units) of Asgaard.

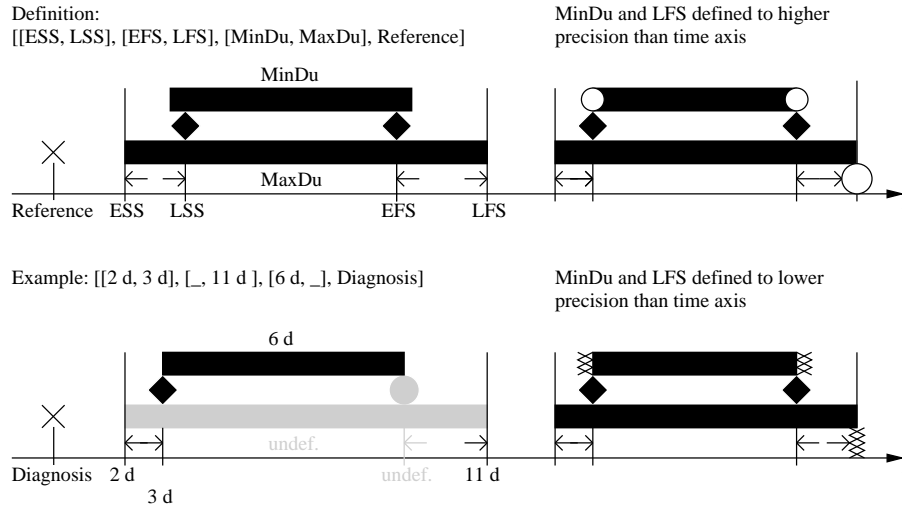


Figure 4: Glyphs representing time annotations. The time annotations are given in Asbru 6.5 syntax.

Figure 3 shows those parts of the Asgaard framework which deal with the execution of plans and gives an intuition of their interaction.

- The *data abstraction unit* receives the data from the input sources and processes it through a set of abstraction methods as listed in Section refsection:interpretation.
- 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.
- 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.

2.1.2 Time-Less Data: Variables

Variables in Asbru behave like variables in a programming language. They serve for subordinate purposes only, such as keeping status information, passing values to subplans and back, etc.

That part of the patient data, which is not expected to change, like name or date of birth, is modeled in the patient data record. It can either be entered by the user on initiation of the top-level plan (i.e., start of the treatment) or the data are obtained from the patient data management system of the hospital running the system using an exchangeable implementation module.

2.2 Criterion Language

Each plan (i.e., action) has an optional filter, **setup**, **suspend**, **reactivate**, **complete**, and **abort** condition (see table 1). Each of these conditions is made up of temporal patterns, containing either references to plan states or parameter propositions. A parameter proposition describes a state of a parameter and its temporal extension (earliest and latest starting and finishing shifts based on an arbitrary reference point, minimum and maximum allowed durations). Figure 4 illustrates the time annotations.

This means that we can exactly describe the conditions under which a plan is considered, ready, activated, suspended, reactivated, aborted, or completed. These states are described in the following section.

filter-preconditions	the preconditions which need to hold initially if the plan is applicable, but can not be achieved and are necessary for a plan to become possible
setup-preconditions	the preconditions which need to be achieved to enable a plan to start and allow a transition from a possible plan to a ready plan
activate-mode	a token which determines if the plan should be started manually or automatically
suspend-conditions	the conditions which determine when an activated plan has to be suspended
abort-conditions	the conditions which determine when an activated , suspended , or reactivated plan has to be aborted
complete-conditions	the conditions which determine when an activated or reactivated plan can be completed successfully
reactivate-conditions	the conditions which determine when a suspended plan has to be reactivated

Table 1: Conditions in Asbru. Compare Figure 5.

2.3 Decision Model

The decision process is modeled by the data abstraction process and by the plan-state model. The first was described before, the latter is described here. A set of mutually exclusive plan states describes the actual status of the plan during the plan selection and the plan execution. Particular state-transition criteria specify transition between neighboring plan states. Figure 5 illustrates the different plan states and their corresponding transition criteria shown on the arrows. The meaning of the state-transition criteria is explained in Table 1. We distinguish plan states during the plan-selection phase (left-hand side of Figure 5) and plan states during the execution phase (right-hand side of Figure 5). For example, if a plan has been **activated**, it can only be **completed**, **suspended**, or **aborted** depending on the corresponding criteria.

2.4 Specification of Action Sequences

Any actions are represented as plans. A plan can be a *user-performed plan*, a *primitive plan* or it is composed by subplans. In the first case, a message is displayed to the user who carries out the actions and enters their outcome and duration. In the second case, a module implemented in Java or another programming language controls the execution of the associated action, e.g., interfacing a hardware interface to actuators. In the third case, the subplans are either cyclical, sequential, parallel, unordered, or any-order-plans. While unordered subplans are completely free from any constraint, only one of a set of any-order-plans may be performed at a time, but their ordering is not fixed.

For each type of subplans (including unordered) none of the children are started before the parent's start and all of them are terminated if the parent terminates. The success or failure of child plans are propagated to the parent under various schemes (defined by the *propagation specification*). The parent can have its own complete and/or abort condition and it can wait for the completion of its children or of a subset of them (as defined in the *continuation specification*). Only if both the complete condition and the continuation specification are satisfied the parent terminates.

Both a plan itself and the activation of a plan can contain a time annotation limiting its temporal extend in the same way as it is done for parameters in the parameter proposition.

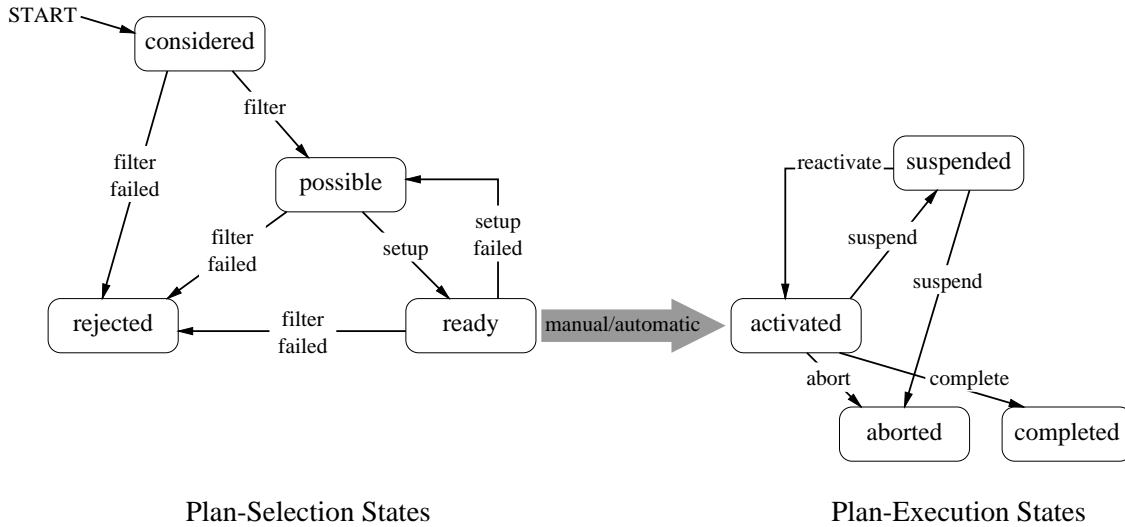


Figure 5: Plan States in Asgaard.

2.5 Specification of goals/intentions

Each plan has declared intentions, which are defined as parameter propositions to be achieved, maintained or avoided. In addition, each plan can have effects defined, which describe the changes to the environment as parameter propositions linked to the invocation of the plan itself or via qualitative relations to one of its arguments. Effects have a likelihood expressed as a factor.

2.6 Model of Actions

Each plan can be refined by having subplans as described above, or it is a *primitive plan* carried out by some domain specific method such as operating some controller, or it is a *user performed plan*. In the latter case, a message describing what to do is shown and the user acknowledges or refuses the task. She also informs the system about the end and outcome (complete or abort) of the plan/action. User performed and primitive plans may also have abort and complete (and any other type of) conditions and time annotation like any other plan which also control their operation.

2.7 Interpretation of Data

All input is time-stamped. The domain definition of the plan library specifies a set of parameters. These are either raw-data parameters representing the data as it enters the system, or abstractions based on other parameters. Abstractions include

- Arithmetic operations,
- Averaging and derivative over a certain period of time,
- Linear regression over a certain time window, and based on this average, trend, etc. [14],
- Context-specific qualitative abstractions,
- Abstractions based on logical expressions,
- Boolean abstractions based on a logical expression (which is a parameter proposition), and
- Logical combinations of Boolean parameters.

2.8 Specification of Supporting Materials and Strength of Evidence/Recommendation

Any language element in Asbru can have one or more of the following annotations: bibliographic reference to printed material, URL of related online information, and a free-text comment. Since a URL can address a certain place in a document (represented in HTML), we can exactly link language elements (such as the constant 5 ml) to the place in the protocol justifying it. There will be a graphical tool facilitating this association since we identified the process of transforming the free text of protocols and the comments on it by various domain experts into the formal representation of Asbru as one of the bottlenecks of the protocol authoring process.

We do not explicitly model the strength of a recommendation, but each plan can be started either manually or automatic, and in the first case a message (defined with this plan) is shown to the user explaining the pros and cons of taking this action. While this seems inferior to other approaches, we made this decision to arrive at a lean representation needed for high-frequency domains such as online monitoring.

When there are several plans which could be performed and which are started manually, they are displayed to the user according to their suitability, which is derived from their intentions, effects, estimated duration, and several other properties in a user defined way.

There was a *utility factor* in Asbru version 6.5 which was skipped for the mentioned reasons and which may come back in a future version depending on the development and utilization of the execution unit.

2.9 Representation and Use of Medical Knowledge and Medical Terminology

Currently, there is nothing like a domain ontology in Asbru. The only data types are strings, numbers, Booleans, and user-defined qualitative values. Typically a question is posed to the user and the answer is represented as a Boolean, string, or number together with the valid time of this value, since the input is modeled by time-stamped parameters. An upcoming version of Asbru will utilize some XML-based representation of ontologies such as OIL [4], and HL7 [2].

2.10 Relationship to Validation and Implementation Issues

The validation of Asbru plans is carried out on a series of levels:

- Syntactic correctness is proven by the XML-editor based on the Asbru-DTD.
- Semantic issues such as the mapping of references in different name spaces are covered by a compiler generated by Pontifex [9]. Pontifex takes a language description in HSL (Harmless Specification Language) and produces a parser, classes representing the elements, a DTD, and documentation for it.
- The intelligent editor PIXEE currently under development will also perform checks for semantic issues like matching definition and use of variables, parameters, etc. Its main aim is to improve the overview of the plan author over the various aspects of a plan library providing intelligent folding of language elements.
- A set of anomalies hinting at possibly conflicting definitions in the plan library can be detected by a special verification module [3]. Currently this is done by hand. An implementation is in progress.
- There will be a simulation mode to observe the plans' reactions to directly entered or pre-recorded input allowing to test a plan library "in the sandbox".
- AsbruView is used to structure the overall topology of the plan library.

The domain-dependent part of the plan library is separated from the independent part and several domains can coexist in a plan library. This allows for different definitions of parameters, patient data records, and primitive plans for different domains. Of course, these definitions must be compatible on the top level accessed by the plans.

Workflow issues such as resource management are not the focus of Asbru, although some aspects like specification of necessary resources are covered.

Events concerning the execution of plans are modeled as changes of parameter values.

3 Guideline 1 - Chronic Cough

3.1 Modeling Chronic Cough

Chronic cough is cough that lasts for at least 3 weeks.

There are two settings for monitoring cough, depending on whether the system or the user performs the abstraction involved.

3.1.1 System Performed Abstraction

In this scenario the user repeatedly enters whether the patient is currently coughing and the data abstraction unit performs the abstraction of repeated single instances of coughs to the decision whether cough lasts for 3 weeks.

This is modeled as an abstracted Boolean parameter based on a directly entered Boolean parameter `is-coughing`. The circumstances under which the patient is considered to be coughing are subject to the judgment of the physician. The question "Is the patient coughing?" is answered repeatedly by the physician (at her discretion) and as soon as a period of 3 weeks is over, at the beginning of which `is-coughing` was true (physician clicked *yes*) and during which there was no instance at which *no* was entered, the parameter proposition defining `has-chronical-cough` is fulfilled and the value of that parameter changes from *false* to *true*.

```
<parameter-def name="is-coughing" type="Boolean">
  <raw-data-def mode="manual" user-text="Is the patient coughing today?">
    <trust-period>
      <numerical-constant unit="week" value="2"/>
    </trust-period>
  </raw-data-def>
</parameter-def>
<parameter-def name="has-chronical-cough" type="Boolean">
  <boolean-def>
    <parameter-proposition parameter-name="is-coughing">
      <value-description type="equal">
        <qualitative-constant value="yes"/>
      </value-description>
      <context>
        <any/>
      </context>
      <time-annotation>
        <time-range>
          <duration>
            <minimum>
              <numerical-constant unit="week" value="3"/>
            </minimum>
          </duration>
        </time-range>
      </time-annotation>
    </parameter-proposition>
  </boolean-def>
</parameter-def>
```

```

        <now/>
    </time-annotation>
</parameter-proposition>
</boolean-def>
</parameter-def>

```

We added a definition of the time period, during which the entered value for the first parameter is considered valid. If someone is coughing today, she need not be coughing next week. On the other hand you do not want to ask her every day during the 3 weeks. So we set the `trust-period` to 2 weeks which makes sure, that entering *yes* once and forgetting about the patient does not automatically set `has-chronic-cough` three weeks later. Instead the value of `is-coughing` is set to *invalid* after two weeks. If the patient would show up again coughing, the first incident is not considered and a new period of three weeks is started. Therefore, the `trust-period` should not be too short either.

3.1.2 User Performed Abstraction

In the low-frequency domain, many people consider it more suitable to have the user of the system perform the abstraction on her own without interaction of the system. In our case, this means that the user simply answers the question "Is the patient coughing for at least 3 weeks?" at her own discretion. This eases the work for data entry but raises the full range of problems generally associated with users performed tasks—there is no way to ensure that the physician is not biased by the intensity of the acute coughing or that the patient is wrong in remembering when the cough started.

On the implementation side, there is not much to do in such a case—there is one manually entered Boolean parameter stating whether or not the patient is coughing for at least 3 weeks.

For the purpose of this study, we decided to take the first option.

3.2 Modeling the ordering of chest radiograph and treatment

Chronic cough is cough that lasts for at least 3 weeks. Chest radiographs should be ordered before any treatment is prescribed in nearly all patients with chronic cough (Grade II-2). Chest radiographs do not have to be routinely obtained before beginning treatment, for

- presumed PNDS (post nasal drip syndrome) in young nonsmokers,
- in pregnant women, or
- before observing the result of discontinuation of an ACE-I (ACE Inhibitor) for 4 weeks for patients who developed cough shortly after beginning to take an ACE-I.

This means that if one of the conditions is fulfilled, chest radiograph need not be taken before the treatment, but still it is ok to take it. Otherwise, it should be taken before the treatment.

This means that the logical condition for ordering chest radiographs is the following:

chronic-cough and not
 ((presumed-PNDS *and* young-nonsmoker) *or* pregnant *or* discontinuing-ACE-I)

Modeling chronic cough was discussed above. `Presumed-PNDS`, `young-nonsmoker`, and `pregnant` are Boolean parameters directly entered by the physician. `discontinuing-ACE-I` is a placeholder for a more complex expression in Asbru, describing the discontinuation of ACE-I administration.

As mentioned in Section 2.8, the only way to model the grade of evidence in Asbru is to display a suitable message to the user which is implemented by the `explanation`-element in plan `chest-radiograph`.

Figure 6 illustrates the different orderings by a screenshot taken from AsbruView.

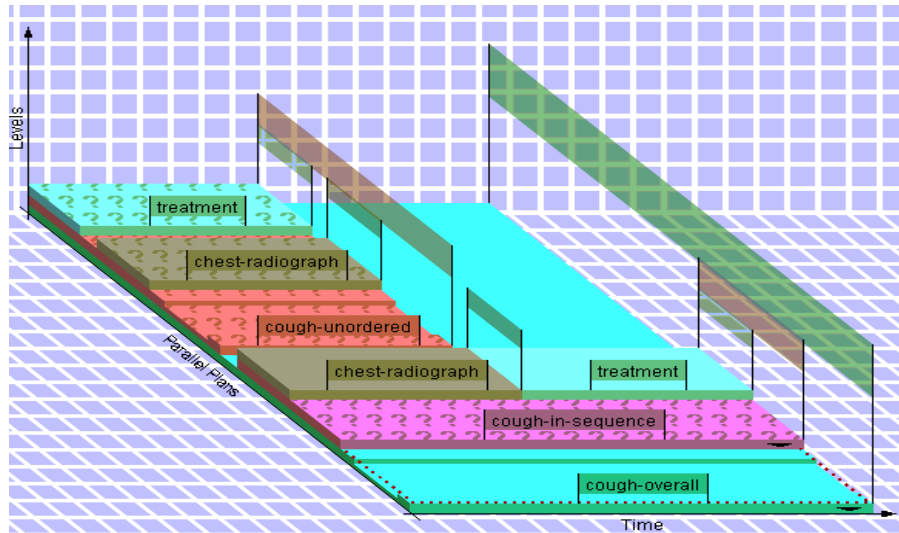


Figure 6: The temporal relations of taking a chest-radiograph and treatment: either unordered or sequential. Depending on the filter conditions, either plan cough-in-sequence or cough-unordered are executed.

```

<plan name="administer-ACE-I"
      title="Administer an Angiotensin-Converting
            Enzyme Inhibitor">
  <plan-body>
    <user-performed/>
  </plan-body>
</plan>
<plan name="cough-overall">
  <effects>
    <argument-dependency argument-name="chronic-cough"
                        likelihood="0.33"
                        parameter="PNDS"
                        relationship="positive-mon">
      <time-annotation>
        <any/>
      </time-annotation>
      <context>
        <context-ref name="chest-X-ray-normal"/>
      </context>
    </argument-dependency>
    <argument-dependency argument-name="chronic-cough"
                        likelihood="0.33"
                        parameter="Asthma"
                        relationship="positive-mon">
      <time-annotation>
        <any/>
      </time-annotation>
      <context>
        <context-ref name="chest-X-ray-normal"/>
      </context>
    </argument-dependency>
  </effects>
</plan>

```

```

<argument-dependency argument-name="chronic-cough"
                      likelihood="0.33"
                      parameter="GERD"
                      relationship="positive-mon">
  <time-annotation>
    <any/>
  </time-annotation>
  <context>
    <context-ref name="chest-X-ray-normal"/>
  </context>
</argument-dependency>
<argument-dependency argument-name="PNDS"
                      likelihood="0.30"
                      parameter="caused-by-sinusitis"
                      relationship="positive-mon">
  <time-annotation>
    <any/>
  </time-annotation>
  <context>
    <one-of name="cough-is-productive">
      <value-ref name="yes"/>
    </one-of>
  </context>
</argument-dependency>
<argument-dependency argument-name="PNDS"
                      likelihood="0.60"
                      parameter="caused-by-sinusitis"
                      relationship="positive-mon">
  <time-annotation>
    <any/>
  </time-annotation>
  <context>
    <one-of name="cough-is-productive">
      <value-ref name="no"/>
    </one-of>
  </context>
</argument-dependency>
</effects>
<plan-body>
  <subplans type="unordered">
    <wait-for>
      <one/>
    </wait-for>
    <plan-activation>
      <plan-schema name="cough-in-sequence"/>
    </plan-activation>
    <plan-activation>
      <plan-schema name="cough-unordered"/>
    </plan-activation>
  </subplans>
</plan-body>
</plan>
<plan name="cough-in-sequence">
  <conditions>

```

```

<filter-precondition>
  <constraint-combination label="filter" type="and">
    <parameter-proposition parameter-name="chronic-cough">
      <value-description type="equal">
        <qualitative-constant value="yes"/>
      </value-description>
      <context>
        <any/>
      </context>
      <time-annotation>
        <now/>
      </time-annotation>
    </parameter-proposition>
    <constraint-not>
      <constraint-combination type="or">
        <constraint-combination type="and">
          <parameter-proposition
            parameter-name="presumed-PNDS">
            <value-description type="equal">
              <qualitative-constant value="yes"/>
            </value-description>
            <context>
              <any/>
            </context>
            <time-annotation>
              <now/>
            </time-annotation>
          </parameter-proposition>
          <parameter-proposition
            parameter-name="young-nonsmoker">
            <value-description type="equal">
              <qualitative-constant value="yes"/>
            </value-description>
            <context>
              <any/>
            </context>
            <time-annotation>
              <now/>
            </time-annotation>
          </parameter-proposition>
        </constraint-combination>
      </constraint-combination>
      <parameter-proposition parameter-name="pregnant">
        <value-description type="equal">
          <qualitative-constant value="yes"/>
        </value-description>
        <context>
          <any/>
        </context>
        <time-annotation>
          <now/>
        </time-annotation>
      </parameter-proposition>
    </plan-state-constraint state="activated">
      <plan-pointer>

```

```

        <static-plan-pointer
            plan-name="administer-ACE-I"/>
    </plan-pointer>
    <time-annotation>
        <time-range>
            <finishing-shift>
                <earliest>
                    <numerical-constant
                        unit="week"
                        value="-4"/>
                </earliest>
                <latest>
                    <now/>
                </latest>
            </finishing-shift>
        </time-range>
        <reference-point>
            <now/>
        </reference-point>
    </time-annotation>
    </plan-state-constraint>
</constraint-combination>
</constraint-not>
</constraint-combination>
</filter-precondition>
</conditions>
<plan-body>
    <subplans type="sequentially">
        <wait-for>
            <all/>
        </wait-for>
        <plan-activation>
            <plan-schema name="chest-radiograph"/>
        </plan-activation>
        <plan-activation>
            <plan-schema name="treatment"/>
        </plan-activation>
    </subplans>
</plan-body>
</plan>
<plan name="cough-unordered">
    <conditions>
        <filter-precondition>
            <constraint-not>
                <refer-to label="filter"
                    plan-name="cough-in-sequence"/>
            </constraint-not>
        </filter-precondition>
    </conditions>
    <plan-body>
        <subplans type="unordered">
            <wait-for>
                <all/>
            </wait-for>

```

```

    <plan-activation>
      <plan-schema name="chest-radiograph"/>
    </plan-activation>
    <plan-activation>
      <plan-schema name="treatment"/>
    </plan-activation>
  </subplans>
</plan-body>
</plan>
<plan name="chest-radiograph">
  <explanation text="Evidence for ordering
    a chest radiograph before
    treatment is of Grade II-2."/>

  <plan-body>
    <user-performed/>
  </plan-body>
</plan>
<plan name="treatment">
  <plan-body>
    <user-performed/>
  </plan-body>
</plan>

```

3.3 Modeling causes of cough

When the chest X-ray is normal PNDS, Asthma, and GERD are the likely causes of chronic cough. In PNDS, sinusitis may be the cause up to approximately 30% of the time when cough is nonproductive, and up to approximately 60% of the time when cough is productive.

Causal dependencies like the above are modeled by *effects* in Asbru. Effects belong to a particular plan and describe expected behavior occurring during its execution. In our case, an overall (top-level) plan contains (has) the above effects. There are two varieties of effects in Asbru: *argument dependency* describes the qualitative relation between an argument of the plan and a parameter, and *plan effect* describes the influence of the plan's execution on the value of a parameter. Both varieties have a context, a time annotation, and a likelihood.

In our example, *chest X-ray is normal* is the context under which PNDS, Asthma, and GERD are likely causes of chronic cough. In Asbru, the likelihood is required. We therefore assumed a likelihood of 0.33 for each of the causes. In practice, these numbers must be acquired from domain experts.

Since the direction of chaining dependencies is forward we model the cause for chronic cough in passive form, i.e. as a *is caused by* relation between chronic cough and its reasons. The reason for this is the fact that chronic cough is known and its reasons are not known and we can only deduct new finding from know facts.

For the causes of PNDS the probability is known (as far as sinusitis is concerned). The type of cough forms the context of the relation between sinusitis and PNDS.

```

<effects>
  <argument-dependency argument-name="chronic-cough"
    likelihood="0.33"
    parameter="PNDS"
    relationship="positive-mon">
    <time-annotation>
    <any/>

```

```

    </time-annotation>
    <context>
      <context-ref name="chest-X-ray-normal"/>
    </context>
  </argument-dependency>
  <argument-dependency argument-name="chronic-cough"
    likelihood="0.33"
    parameter="Asthma"
    relationship="positive-mon">
    <time-annotation>
      <any/>
    </time-annotation>
    <context>
      <context-ref name="chest-X-ray-normal"/>
    </context>
  </argument-dependency>
  <argument-dependency argument-name="chronic-cough"
    likelihood="0.33"
    parameter="GERD"
    relationship="positive-mon">
    <time-annotation>
      <any/>
    </time-annotation>
    <context>
      <context-ref name="chest-X-ray-normal"/>
    </context>
  </argument-dependency>
  <argument-dependency argument-name="PNDS"
    likelihood="0.30"
    parameter="caused-by-sinusitis"
    relationship="positive-mon">
    <time-annotation>
      <any/>
    </time-annotation>
    <context>
      <one-of name="cough-is-productive">
        <value-ref name="yes"/>
      </one-of>
    </context>
  </argument-dependency>
  <argument-dependency argument-name="PNDS"
    likelihood="0.60"
    parameter="caused-by-sinusitis"
    relationship="positive-mon">
    <time-annotation>
      <any/>
    </time-annotation>
    <context>
      <one-of name="cough-is-productive">
        <value-ref name="no"/>
      </one-of>
    </context>
  </argument-dependency>
</effects>

```

3.4 Modeling Negative Recommendations

Sinus CT scans are not routinely recommended to evaluate for sinusitis. Four-view sinus radiographs should be ordered instead.

Negative recommendations can be modeled in intentions in Asbru. In our example the type of the intention is `intermediate-action` and the verb is `achieve` for four-view sinus radiographs and `avoid` for sinus CT scan. Intentions are properties of plans, in our example they belong to the plan `evaluating-sinusitis`.

There is no detailed information in the guideline how the plan activity/plan of evaluating sinusitis is invoked by other plans in the guideline, so we do not model its activation but add it to the plan library.

```
<plan name="evaluating-sinusitis">
  <intentions>
    <intention type="intermediate-action" verb="achieve">
      <plan-state-constraint state="activated">
        <plan-pointer>
          <static-plan-pointer
            plan-name="four-view-sinus-radiographs"/>
        </plan-pointer>
        <time-annotation>
          <any/>
        </time-annotation>
      </plan-state-constraint>
    </intention>
    <intention type="intermediate-action" verb="avoid">
      <plan-state-constraint state="activated">
        <plan-pointer>
          <static-plan-pointer plan-name="sinus-CT"/>
        </plan-pointer>
        <time-annotation>
          <any/>
        </time-annotation>
      </plan-state-constraint>
    </intention>
  </intentions>
</plan>
<plan name="four-view-sinus-radiographs">
  <plan-body>
    <user-performed/>
  </plan-body>
</plan>
<plan name="sinus-CT">
  <plan-body>
    <user-performed/>
  </plan-body>
</plan>
```

3.5 Modeling GERD

While 24 hour esophageal pH monitoring is the most diagnostically useful test for assessing for GERD as the cause of cough, conventional indices used by gastroenterologists to assess for esophagitis may be misleadingly normal. Therefore, until future studies provide better guidelines, the test should be read as normal when conventional indices are within the normal range and no suspicious reflux-induced coughs appear during the monitoring session (Grade II-2).

We model both the conventional indices and the suspicious reflux-induced coughs as qualitative parameters. The criterion indicating that the test results are interpreted as normal is the conjunction of normal indices and suspicious coughs being absent during 24 hour esophageal pH monitoring. The latter is represented by a user perform plan with a typical duration of that extend. There is no mentioning, that such a monitoring last for exactly such a duration, therefore we do not define minimum or maximum duration, but only the estimated typical duration. `suspicious-coughs` are considered, if they end or start during the 24 hours following the start of the monitoring plan.

```
<parameter-def name="conventional-indices" type="Boolean">
  <raw-data-def mode="manual"
    user-text="Are conventional indices normal?"/>
</parameter-def>
<parameter-def name="suspicious-coughs" type="Boolean">
  <raw-data-def mode="manual"
    user-text="Are there suspicious coughs?"/>
</parameter-def>
<parameter-def name="suppose-GERD" type="Boolean">
  <logical-combination-def operator="and">
    <boolean-def>
      <parameter-proposition
        parameter-name="conventional-indices">
        <value-description type="equal">
          <qualitative-constant value="yes"/>
        </value-description>
        <context>
          <any/>
        </context>
        <time-annotation>
          <now/>
        </time-annotation>
      </parameter-proposition>
    </boolean-def>
    <boolean-def>
      <parameter-proposition
        parameter-name="suspicious-coughs">
        <value-description type="equal">
          <qualitative-constant value="no"/>
        </value-description>
        <context>
          <any/>
        </context>
        <time-annotation>
          <time-range>
            <starting-shift>
```

```

        <latest>
            <numerical-constant unit="h" value="0"/>
        </latest>
    </starting-shift>
</finishing-shift>
    <earliest>
        <numerical-constant unit="h" value="24"/>
    </earliest>
</finishing-shift>
</time-range>
<plan-state-transition instance-type="last"
                        state="activated">
    <plan-pointer>
        <static-plan-pointer
            plan-name="esophageal-ph-monitoring"/>
    </plan-pointer>
</plan-state-transition>
</time-annotation>
</parameter-proposition>
</boolean-def>
</logical-combination-def>
</parameter-def>
</parameter-group>
</domain>
</domain-defs>
<plans>
    <plan-group>
        <plan name="esophageal-ph-monitoring">
            <defaults>
                <typical-duration>
                    <numerical-constant unit="h" value="24"/>
                </typical-duration>
            </defaults>
            <plan-body>
                <user-performed/>
            </plan-body>
        </plan>
    </plan-group>
</plans>
</plan-library>

```

4 Guideline 2 - Hypertension

This is to follow on Tuesday, 13th of February.

References

- [1] Adrion, W. R., Branstad, M. A. & Cherniavsky, J. C.: Validation, Verification, and Testing of Computer Software. Computing Review, 14(2), pp. 159-192, 1982.
- [2] Dolin, R.H. et al.: HL7 Document Patient Record Architecture: An XML Document Architecture Based on a Shared Information Model. In Lorenzi N. (ed.) Proceedings of the 1999 AMIA Symposium. Hanley & Belfus, Inc. 1999.

- [3] Duftschmid, G. & Miksch, S.: Knowledge-Based Verification of Clinical Guidelines by Detection of Anomalies. *Artificial Intelligence in Medicine*, to appear, 2001.
- [4] Fensel, D., Horrocks, I., van Harmelen, F., Decker, S., Erdmann, M. & Klein, M.: OIL in a Nutshell, In Dieng, R. (ed.) *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW'00)*, Springer, Lecture Notes in Artificial Intelligence, 2000/1999 AMIA Symposium.
- [5] Friedland, P. E. & Iwasaki, Y.: The Concept and Implementaion of Skeletal Plans. *Journal of Automated Reasoning*, 1(2), pp. 161-208, 1985.
- [6] Hammond, P., Sergot, M. J. & Wyatt, J. C.: Formalisation of Safety Reasoning in Protocols and Hazard Regulations, In Gardner, R. M. (ed.) *Proceedings of the Annual Symposium on Computer Applications in Medical Care (SCAMC-95)*, New Orleans, Louisiana, Hanley & Belfus, 1995.
- [7] Kosara, R., Hammermüller, K. & Miksch, S.: Co-Designing XML-based Languages and Classes with Pontifex, Vienna University of Technology, Institute of Software Technology, Asgaard-TR-2000-1, 2000.
- [8] Kosara, R. & Miksch, S.: Visualization Techniques for Time-Oriented, Skeletal Plans in Medical Therapy Planning, In Horn, W., Shahar, Y., et al. (eds.), *Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making (AIMDM'99)*, Aalborg, Denmark, Springer, Berlin, pp. 291-300, 1999.
- [9] Kosara, R.: Pontifex V1.0pre8 Program Documentation, Vienna University of Technology, Institute of Software Technology, Vienna, Technical Report, Asgaard-TR-2000-2, 2000.
- [10] Kosara, R. & Miksch, S.: Metaphors of Movement: A Visualization and User Interface for Time-Oriented, Skeletal Plans. *Artificial Intelligence in Medicine*, Special Issue, to appear, 2001.
- [11] Miksch, S.: Plan Management in the Medical Domain. *AI Communications*, 12(4), pp. 209-235, 1999.
- [12] Miksch, S., Kosara, R., Shahar, Y. & Johnson, P.: AsbruView: Visualization of Time-Oriented, Skeletal Plans, *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems 1998 (AIPS-98)*, June 7-10, 1998, Carnegie Mellon University, Pittsburgh Pennsylvania, USA., AAAI Press, pp. 11-18, 1998.
- [13] Miksch, S. & Seyfang, A.: Continual Planning with Time Oriented, Sekeletal Plans, In Horn, W. (ed.) *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000)*, Berlin, Germany, IOS Press, Amsterdam, pp. 511-515, 2000.
- [14] Miksch, S., Seyfang, A., Horn, W. & Popow, C.: Abstracting Steady Qualitative Descriptions over Time from Noisy, High-Frequency Data, In Horn, W., Shahar, Y., et al. (eds.), *Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making (AIMDM'99)*, Springer, Berlin, pp. 281-290, 1999.
- [15] Miksch, S., Shahar, Y. & Johnson, P.: Asbru: A Task-Specific, Intention-Based, and Time-Oriented Language for Representing Skeletal Plans, In Motta, E., van Harmelen, F., et al. (eds.), *Proceedings of the 7th Workshop on Knowledge Engineering: Methods & Languages (KEML-97)*, Milton Keynes, UK, January 22-24, The Open University, Milton Keynes, UK, pp. 9/1-9/20, 1997.
- [16] Plaisant, C., Mushlin, R., Snyder, A., Li, J., Heller, D. & Shneiderman, B.: LifeLines: Using Visualization to Enhance Navigation and Analysis of Patient Records, *Proceedings of the American Medical Informatic Association Annual Fall Symposium (AMIA 98)*, Hanley & Belfus, Inc., Medical Publishers, Philadelphia, pp. 76-80, 1998.

- [17] Preece, A., Batarekh, A. & Shinghal, R.: Verifying Rule-Based Systems. Knowledge Engineering Review, 7(2), pp. 115-141, 1992.
- [18] Preece, A. D. & Shinghal, R.: Foundation and Application of Knowledge Base Verification. International Journal of Intelligent Systems, 9(8), pp. 683-702, 1994.
- [19] Seyfang, A., Kosara, R. & Miksch, S.: Asbru 7.2 Reference Manual, Vienna University of Technology, Institut of Software Technology, Asgaard-TR-2000-3-1, 2000. available at http://www.ifs.tuwien.ac.at/asgaard/asbru/asbru_7.2_new/
- [20] Shahar, Y., Miksch, S. & Johnson, P.: The Asgaard Project: A Task-Specific Framework for the Application and Critiquing of Time-Oriented Clinical Guidelines. Artificial Intelligence in Medicine, 14,pp. 29-51, 1998.