# A Task-Specific Ontology for Design and Execution of Time-Oriented Skeletal Plans

Yuval Shahar[1)], Silvia Miksch[2)], Peter Johnson[1)]

[1)] Section on Medical Informatics, Medical School Office Building, x215
Stanford University, Stanford, CA 94305–5479, USA
Email: {shahar, pdj}@smi.stanford.edu

[2)] Vienna University of Technology
Department of Software Technology
Resselgasse 3/E188, A-10140 Vienna, Austria
Email: silvia@ifs.tuwien.ac.at

**Abstract:** Skeletal plans are a powerful way to reuse existing domain-specific procedural knowledge while leaving room for execution-time flexibility. Generic plan schemata can be instantiated and refined dynamically by the executing agent over significant periods of time and in highly dynamic environments. In the **Asgaard** project, we are investigating a set of tasks that support the execution of skeletal plans by a human executing agent, other than the original plan designing agent. We are developing task-specific problem-solving methods that perform these tasks in multiple clinical domains, given an instance of a clinical guideline plan and an electronic medical patient record. We point out the domain-specific knowledge roles required by each problem-solving method, and present a text-based, machine-readable language, called **Asbru**, to represent and to annotate execution plans. We represent explicitly the intentions underlying these plans, as temporal patterns to be achieved or avoided. We introduce an automated knowledge-acquisition tool for clinical guidelines, which we generated, using the PROTÉGÉ-II framework's suite of tools, from the shared (global) ontology of the methods.

## 1. INTRODUCTION: AUTOMATED SUPPORT OF PLAN EXECUTION

A common strategy for the representation and the reuse of domain-specific procedural knowledge is the representation of that knowledge as a library of skeletal plans. Skeletal plans are plan schemata at various levels of detail that capture the essence of the procedure, but leave room for execution-time flexibility in the achievement of particular goals (Friedland and Iwasaki, 1985). Thus, they are often highly reusable.

Execution of skeletal plans often involves an interpretation by one agent of plans that have been designed by another. We are interested in problems that occur while trying to provide several types of automated support to a human interpreting agent. Automated support includes tasks such as assessment of the applicability of the plan to a particular state of the world, guidance in proper execution of that plan, monitoring of the execution process, assessment of the results of the plan, critiquing the execution process and its results, and assistance in the modification of the original plan. We are focusing on domains that are time oriented with respect to both external states and plan actions, and that might require intermittent execution of plans over multiple (possibly disjoint) periods of time, an uncommon requirement in classical planning models.

We demonstrate our ideas in the context of the medical domain. However, as can readily be seen, the model presented is quite general.

### 1.1. Background: Clinical Guidelines and Protocols

**Clinical guidelines** are a set of schematic plans for management of patients who have a particular clinical condition (e.g., insulin-dependent diabetes). **Clinical protocols** are a more detailed version of clinical guidelines, used when the guidelines (possibly experimental) need to be applied by care providers in a similar fashion and to enable statistical analysis of outcomes for comparison among a set of guidelines (e.g., chemotherapy protocols for cancer therapy). The

application of clinical guidelines and protocols by human care providers involves collecting and interpreting considerable amounts of data over time, applying standard treatment plans in an episodic fashion, and revising those plans when necessary. The guidelines often involve implicit assumptions about the knowledge of the agent executing the plans, both in the data-interpretation and in the treatment-planning phases. Thus, clinical guidelines can be viewed as a shared library of highly reusable skeletal reactive plans, whose details need to be refined and executed by a reactive planner over significant periods of time when applied to a particular patient.

Clinical guidelines are often ambiguous or incomplete. A diabetes guideline might recommend a therapy target without any specific recommendations on ways to achieve it, or might suggest the use of a drug without a precise dose. Physicians often do not adhere to protocols, believing their actions to be closer to the intentions of the protocol designers (Hickam, et al. 1985). An automated assistant should recognize cases in which the care-provider's actions adhere to the overall intentions, and continue offering useful advice even if the guideline is not followed literally.

## 1.2. Related Approaches

During the past 15 years, there have been several efforts to create automated reactive planners to support the process of protocol-based care over significant periods of time. In the *prescriptive* approach, active interpretation of the guidelines is given; examples include ONCOCIN (Tu et al., 1989) in the oncology domain, T-HELPER (Musen et al., 1992) in the AIDS domain, and DILEMMA (Herbert et al., 1995), EON (Musen et al., 1996), and the European PRESTIGE Health-Telematics project, as general architectures. In the *critiquing* approach, the program critiques the physician's plan rather than recommending a complete one of its own. This approach concentrates on the user's needs and exploits the assumption that the user has considerable domain-specific knowledge (Miller 1986). A task-specific architecture implementing the critiquing process has been generalized in the HyperCritic system (Van der Lei and Musen 1991). Task-specific architectures assign well-defined problem-solving *roles* to domain knowledge and facilitate acquisition and maintenance of that knowledge.

Several approaches to the support of guideline-based care encode guidelines as elementary state-transition tables or as situation-action rules dependent on the electronic medical record (Sherman, et al. 1995), but do not include an intuitive representation of the guideline's clinical logic, and have no semantics for the different types of clinical knowledge represented. Other approaches permit hypertext browsing of guidelines via the World Wide Web (Barnes and Barnett 1995; Liem, et al. 1995), but do not use the patient's electronic medical record.

None of the current guideline-based-care systems have a sharable representation of guidelines that (1) has knowledge roles specific to the guideline-based-care task, (2) is machine and human readable, and (3) allows data stored in an electronic patient record to invoke an application that directly executes the guideline's logic and related tasks, such as critiquing. A sharable, human- and machine-readable representation of clinical guidelines, that has an expressive syntax and task-specific semantics, combined with the ability to interpret that representation in automated fashion, would facilitate guideline dissemination, real-time accessibility, and applicability. Such a representation also would support additional reasoning tasks, such as automated critiquing, quality assurance (Grimshaw and Russel 1993), and guideline evaluation. A task-specific, sharable representation would also facilitate authoring and modifying clinical guidelines.

A sharable skeletal-plan-execution language needs to be expressive with respect to temporal annotations and needs to have a rich set of parallel, sequential, and iterative operators. Thus, it should enable designers to express complex procedures in a manner similar to a real programming language (although typically on a higher level of abstraction). The language, however, also requires well-defined semantics for both the prescribed actions and the task-specific annotations, such as the plan's intentions and effects, and the preferences (e.g., implicit utility functions) underlying them. Thus, the executing agent's (e.g., the physician's) actions can be better

supported, leading to a more flexible dialog and, in the case of the clinical domains, to a better acceptance of automated systems for guideline-based care support. Clear semantics for the task-specific knowledge roles also facilitate acquisition and maintenance of these roles.

With these requirements in mind, we have developed a sharable, text-based, machine-readable language, called **Asbru**. The Asbru language is part of the **Asgaard**[*)] project, in which we are developing task-specific problem-solving methods that perform execution and critiquing tasks in medical domains. In the following we will introduce the underlying design-time and execution-time model, the required knowledge roles, the skeletal-plan-execution support tasks, and the architecture to perform these several tasks. In addition, the syntax and the semantics of the Asbru language will be explained using a medical example as illustration, namely a guideline for controlled observation and treatment of noninsulin-dependent gestational diabetes mellitus (GDM type II). Finally, we will present an object-oriented version of Asbru, which we used for generating an automated knowledge-acquisition tool by using the PROTÉGÉ-II framework's tools.

## 2. THE DESIGN-TIME VERSUS EXECUTION-TIME MODEL

During **design time** of a skeletal plan, an **author** (or a committee) designs a skeletal plan, such as a clinical guideline (Figure 1). The author prescribes (1) *actions* (e.g., administer a certain drug in the morning and in the evening), (2) an *intended plan*—the intended intermediate and overall pattern of actions, which might not be obvious from the description of the prescribed actions and is often more flexible (e.g., use some drug from a certain class of drugs twice a day), and (3) the *intended* intermediate and overall pattern of *world states* (e.g., patient states such as "morning blood glucose should stay within a certain range"). Intentions are *temporal patterns* of provider actions or patient states, to be achieved, maintained, or avoided.

During **execution time**, an executing agent, such as a **care provider** in the medical domain, performs *actions*, which are recorded, observed, and abstracted over time into an *abstracted plan* (see Figure 1). The *state* of the world (i.e., the patient in this case) also is recorded, observed, and abstracted over time. Finally, *the intentions of the executing agent* (in this case, the care provider) might be recorded too—inferred from her actions or explicitly stated by the provider.

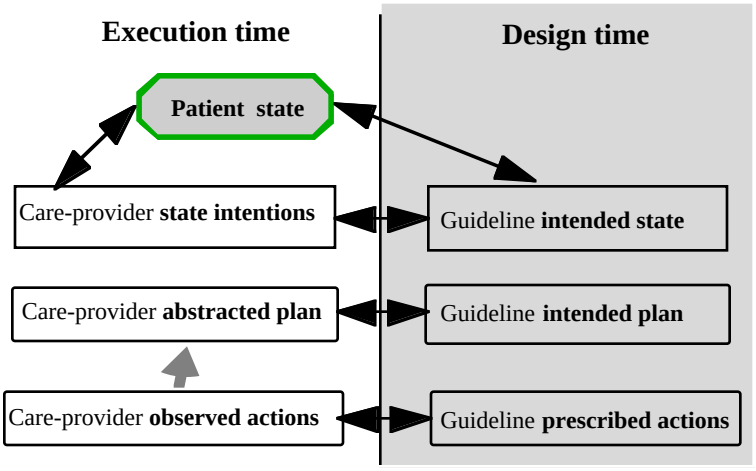


**Figure 1**. The design-time versus execution-time model in a clinical domain. Double-headed arrows denote a potential axis of comparison (e.g., for critiquing purposes) during runtime execution of the clinical guideline. Striped arrows denote an ABSTRACTED-INTO relationship.

*) In Norse mythology, Asgaard was the home and citadel of the gods, corresponding to Mount Olympus in Greek mythology. It was located in the heavens and was accessible only over the rainbow bridge, called Asbru (or Bifrost).

## 2.1. Support for skeletal-plan design and execution

Given the intention-based critiquing model for execution of skeletal plans, we can describe a set of tasks relevant to the design and execution of such plans, and analyze the knowledge requirements of problem-solving methods that perform these tasks (Table 1). The verification and validation tasks are relevant only during design time. The rest of the tasks are relevant during execution time.

Each task can be formulated as answering a specific set of questions (see Table 1). Furthermore, each task can be performed by a problem-solving method (Eriksson, et al. 1995) that has an **ontology**—a set of entities, relations, and domain-specific knowledge requirements assumed by the method. Since many of these knowledge requirements are common to different problem-solving methods relevant to the same set of tasks, we combine them into a *knowledge cluster*, akin to Pat Hayes's notion of an *axiom cluster* (Hayes 1978; Hayes 1985). Such a knowledge cluster can be viewed as either the *local ontology* of the overall method configured by the combination of the various problem-solving methods used to support skeletal-plan execution, or the *global ontology* of these problem-solving methods. Examples of knowledge roles include plan intentions, author preferences, and runtime required conditions. The semantics of the knowledge roles used in our skeletal-plan representation language are discussed in the Section 3. Given these knowledge roles, we can define what knowledge is required to solve each task (see Table 1).

**Table 1**: The skeletal-plan-execution support tasks and the knowledge roles required by problem-solving methods performing these tasks. Common roles can be viewed as shareable by the methods requiring them.

| Task | Questions to be answered | Required knowledge roles |
|---|---|---|
| Verification | Are the intended plans compatible with the prescribed actions? | Prescribed actions; intended overall action pattern |
| Validation | Are the intended states compatible with the prescribed actions and intended plans? | Prescribed actions, intended overall action pattern; intermediate and overall intended states; plan effects |
| Applicability of plans | What skeletal plans are applicable this time to this world? | Filter and setup preconditions; overall intended states; the world's state |
| Execution of plans | What should be done now according to the execution-plan's prescribed actions? | Prescribed actions; setup preconditions, suspension, restart, completion, and abort conditions; the world's state |
| Recognition of intentions | Why is the executing agent executing a particular set of actions, especially if those actions deviate from the skeletal plan's prescribed actions? | Executed actions and their abstraction to executed plans; intended actions and states; the world's state; plan effects; revision strategies; preferences |
| 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? | Executed actions and their abstraction to executed plans; intended intermediate and overall action pattern; intermediate and overall intended states; the world's state; plan effects; revision strategies; preferences |
| Evaluation of the plan | Is the plan working? | Intermediate and overall intended states; the world's state; intermediate and overall intended action pattern; executed actions |
| Modification of an executing plan | What alternative actions or plans are relevant at this time for achieving a given state intention? | Intermediate and overall intended states; the world's state; plan effects; filter and setup preconditions; revision strategies; preferences |

The following example demonstrates the tasks of plan-recognition and critiquing in the clinical domain of monitoring and therapy of patients who have insulin-dependent diabetes.

During therapy of a diabetes patient, hyperglycemia (a higher than normal level of blood glucose) is detected for the second time in the same week around bedtime. The diabetes-guideline's prescribed action might be to increase the dose of the insulin the patient typically injects before dinner. (Insulin reduces the level of blood glucose.) However, the provider recommends reduction of the patient's carbohydrate intake (e.g., bread) during dinner. This action seems to contradict the prescribed action. Nevertheless, the automated assistant notes that increasing the dose of insulin decreases the value of the blood-glucose level directly, while the provider's recommendation *decreases* the value of the same clinical parameter by *reducing* the magnitude of an action (i.e., ingestion of carbohydrates) that *increases* its value. The assistant also notes that the state intention of the guideline was "avoid more than two episodes of hyperglycemia per week." Therefore, the provider is still following the intention of the protocol. By recognizing this high-level intention and its achievement by a different strategy, the automated assistant can accept the provider's alternate set of actions, and even provide further support for these actions.

The *plan-recognition* ability demonstrated in the example can increase the usefulness of guideline-based decision-support systems to clinical practitioners, who often follow what they consider as the author's intentions rather than the prescribed actions (Hickam, et al. 1985). We assume knowledge about the *effects of interventions* on clinical parameters, and knowledge of domain-independent and domain-specific *guideline-revision strategies*; both effects and revision strategies can be represented formally (Shahar and Musen 1995).

The example also demonstrates a specific *execution-critiquing* model. The five comparison axes shown in Figure 1 imply a set of different **behaviors** of the execution. A care provider might not follow the precise *actions*, but still follow the intended *plan* and achieve the desired states. A provider might even not follow the overall plan, but still adhere to a higher-level *intention*. Alternatively, the provider might be executing the guideline correctly, but the patient's state might differ from the intended, perhaps indicating a complication that needs attention or a failure of the guideline. Several typical behaviors in the medical domain are presented in Table 2.

**Table 2**: Typical execution behaviors defined by the critiquing task, based on comparison of guideline intentions, physician actions and intentions, and patient states

| **Intended action vs. physician action** | **Intended plan vs. physician plan** | **Intended state vs. physician intention** | **Intended state vs. patient state** | **physician intention vs. patient state** | **Description of the behavior** |
|---|---|---|---|---|---|
| + | + | + | + | + | physician executes protocol as specified; protocol succeeds |
| + | + | + | – | – | physician follows guideline, has the same intentions, but guideline does not work |
| – | + | + | + | + | overall plan intention followed, albeit through different actions, and it works |
| – | – | + | + | + | physician follows neither actions nor overall plan; state intentions agree and both succeed |
| – | – | – | – | – | physician follows neither action nor plan; state intentions differ, and neither materializes |

In theory, there might be up to 32 ($2^5$) different behaviors, assuming binary measures of comparison along five axes. However, the use of consistency constraints (e.g., if the intended actions are followed by the physician, the pattern formed by the physician's actions [the physician plan] must match the intended plan, assuming no contradictions in the original intentions) and other reasonable assumptions prunes this number to about 10 major behaviors. These behaviors are used by the critiquing mechanism. (We also are investigating the use of continuous, rather than binary, measures of matching).

The meaning of *intentions* in general and for planning tasks in particular has been examined in philosophy (Bratman, 1987) and in artificial intelligence (Pollack, 1992). We view intentions as temporally extended goals at various abstraction levels (Bacchus and Kabanza, 1996).

A subtask implicit in several of the tasks in Table 1 is the abstraction of higher-level concepts from time-stamped data during the execution of the skeletal plan. Possible candidates for solving this subtask include the **RÉSUMÉ** system and the temporal data-abstraction component in the **VIE-VENT** system. The RÉSUMÉ system (Shahar and Musen 1993) is an implementation of a formal, domain-independent problem-solving method, the *knowledge-based temporal-abstraction method* (Shahar, in press) and has been evaluated in several clinical domains (Shahar and Musen 1996). VIE-VENT (Miksch, et al. 1993) is an open-loop knowledge-based monitoring and therapy planning system for artificially ventilated newborn infants, which includes context-sensitive and expectation-guided temporal data-abstraction methods (Miksch, et al. 1996b). These methods incorporate knowledge about data points, data intervals, and expected qualitative trend patterns to arrive at unified qualitative descriptions of parameters (temporal data abstraction) (Miksch, et al. 1996a).

In the Asgaard project, we are developing different task-specific reasoning modules that perform the skeletal-plan-execution tasks shown in Table 1, and we are applying these modules to clinical domains. Figure 2 presents the overall architecture. The task-specific reasoning modules require different types of knowledge, often outside of the scope of the execution-plan ontology. For instance, the knowledge-based temporal-abstraction method implemented by the RÉSUMÉ module requires knowledge about temporal-abstraction properties of measurable parameters, such as persistence of their values over time when these values are not recorded (Shahar, in press). These properties exist, however, in the domain's task-specific temporal-abstraction ontology (Shahar and Musen 1996).

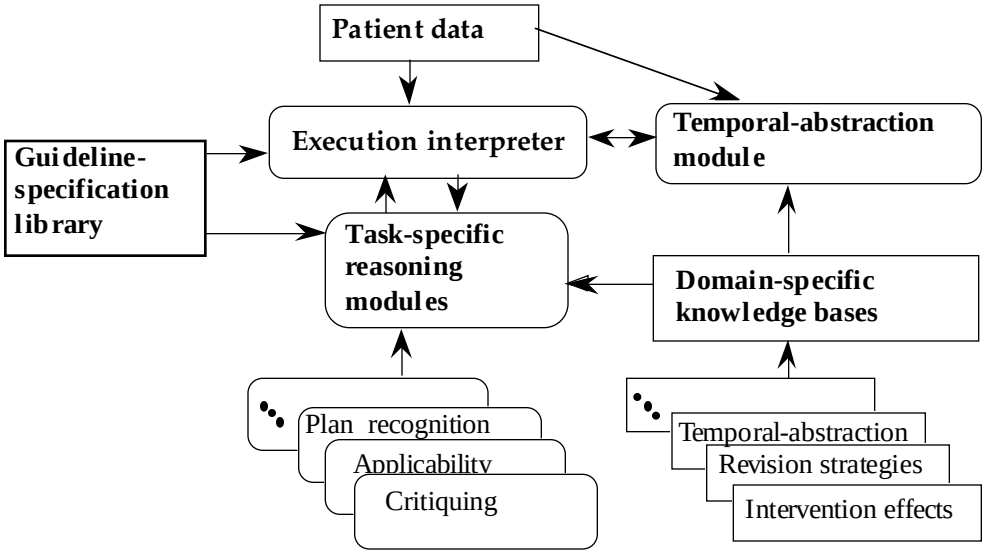


**Figure 2**. A skeletal-plan-execution architecture instantiated in the guideline-based care domain. Arrows denote data or knowledge flow.

Similarly, the plan-recognition and critiquing methods require generic and domain-specific plan-revision knowledge (Shahar and Musen 1995); much of that knowledge is not part of the ontology of executable plans, but is represented in a separate knowledge base. Plan effects can be represented as part of the plan, but can also be viewed as a separate knowledge base (Figure 2). The specifications of clinical guidelines and of their independent components (we refer to either of these entities as **plans** in this paper) are all represented uniformly and organized in a *guideline-specification library*. The execution plans are expressed in our task-specific language, Asbru.

## 3. ASBRU: THE EXECUTION-SUPPORT–METHODS GLOBAL ONTOLOGY

We have developed a language specific to the set of execution-support tasks and the problem-solving methods performing these tasks, which we call **Asbru**. Asbru enables a designer to represent both the prescribed actions of a skeletal plan and the knowledge roles required by the various problem-solving methods performing the several execution-support subtasks. The major features of Asbru are that prescribed actions can be continuous; plans might be executed in parallel, in sequence, in a particular order, or every time measure; temporal scopes and parameters of plans can be flexible, and explicit intentions and preferences can underlie the plan. These features are in contrast to many traditional plan-execution representations (e.g., (Fikes and Nilsson 1971; Tate, Drabble, and Kibry 1994; Kambhampati et al., 1995)), which assume instantaneous actions and effects. Actions often are continuous and might have delayed effects and temporally-extended goals (Bacchus and Kabanza 1996). The requirements of plan specifications in clinical domains (Tu, et al. 1989; Uckun 1994). are often a superset of the requirements in typical toy-problem domains used in planning research. We have defined a formal syntax for the language in Backus-Naur form, and an object-oriented version that defines a task-specific skeletal-plan-support ontology. We used this ontology for generating an automated knowledge-acquisition tool by employing the PROTÉGÉ-II framework's tools (Musen, et al. 1995). The Asbru language combines the flexibility and expressivity of standard procedural languages (e.g., the ARDEN syntax (Hripcsak, et al. 1994) in the clinical domain) with the semantic clarity of declaratively expressed knowledge roles in the task-specific ontology.

Asbru can be used to design specific plans as well as support the performance of different reasoning and executing tasks. Similar assumptions were made in the PROPEL language (Levinson 1995). During the design phase of plans, Asbru provides a powerful mechanism to express durative actions and plans caused by durative states of an observed agent (e.g., many actions and plans need to be executed in parallel or every particular time point). These plans are combined with intentions of the executing agent of plan. They are uniformly represented and organized in the *guideline-specification library*. During the execution phase an applicable plan is instantiated with distinctive arguments and state-transition criteria are added to execute and reason about different tasks. These tasks have been presented in Section 2. We are using a medical example to illustrate our language, namely a guideline for controlled observation and treatment of **gestational diabetes mellitus** (**GDM**) Type II. The entire example is listed in the Appendix.

### 3.1. Time Annotation

Intentions, world states, and prescribed actions are temporal patterns. A temporal pattern is either a *parameter preposition*—a parameter (or its abstraction), its value, a context, and a time annotation (e.g., the *state* abstraction of the blood-glucose parameter is HIGH, as defined in the context of therapy for GDM type II, during a certain time period)—or a combination of multiple parameter propositions (Shahar and Musen 1996). The time annotation we use allows a representation of uncertainty in starting time, ending time, and duration (Dechter, Meiri, and Pearl 1991; Rit 1986). The time annotation supports multiple time lines (e.g., different zero-time points and time units) by providing *reference annotations*. The reference annotation can be an absolute reference point, a reference point with uncertainty (defined by an uncertainty region), a function of a previously executed plan instance (e.g., start plan instance A1 20 minutes after having completed plan instance

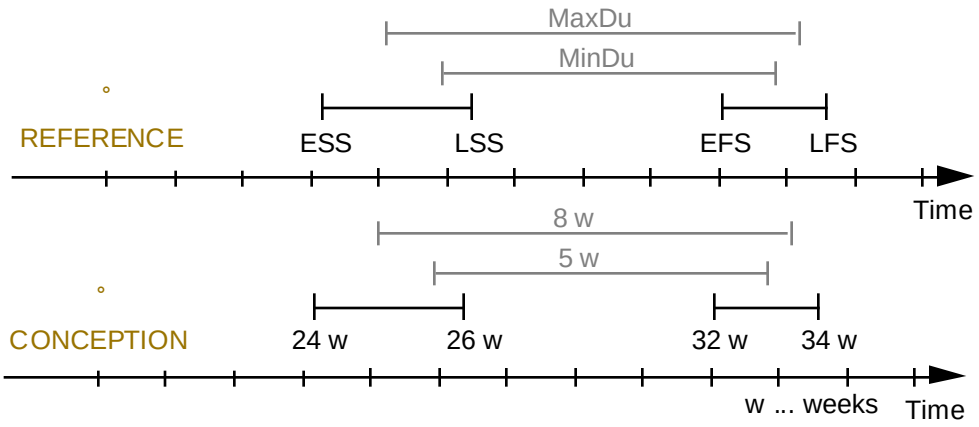


**Figure 3**. A schematic illustration of the Asbru time annotations. The upper part of the figure presents the generic annotation. The lower part shows a particular example representing the time annotation [[24 WEEKS, 26 WEEKS], [32 WEEKS, 34 WEEKS], [5 WEEKS, 8 WEEKS], CONCEPTION]), which means "starts 24 to 26 weeks after conception, ends 32 to 34 weeks after the conception, and lasts 5 to 8 weeks." REFERENCE = reference annotation, ESS = earliest starting shift, LSS = latest starting shift, EFS = earliest finishing shift, LFS = latest finishing shift, MinDu = minimal duration, MaxDu = maximal duration.

B1), or a domain-dependent time point variable (e.g., CONCEPTION). We define temporal shifts from the reference annotation to represent the uncertainty in starting time, ending time, and duration, namely earliest starting shift (ESS), latest starting shift (LSS), earliest finishing shift (EFS), latest finishing shift (LFS), minimal duration (MinDu), and maximal duration (MaxDu). The temporal shifts are associated with time units (e.g., minutes, days) or domain-dependent units (e.g., GESTATIONAL-WEEKS). Thus, our temporal annotation is written as ([ESS, LSS], [EFS, LFS], [MinDu, MaxDu], REFERENCE). Figure 3 illustrates our time annotation. ESS, LSS, EFS, LFS, MinDu, and MaxDu can be "unknown" or "undefined" to allow incomplete time annotation. Also, in cases such as ([T1 HOURS, T1 HOURS], [T2 HOURS, T2 HOURS], [_, _], REFERENCE), the time interval has exact starting and finishing times, T1 and T2, respectively. Therefore, the duration should not be specified, because (duration = T2 - T1) and maximal duration is equal to minimal duration.

To allow temporal repetitions, we define sets of cyclic time points (e.g., MIDNIGHTS, which represents a set of midnights, where each midnight is exactly at 0:00 am, every 24 hours) and cyclic time annotations (e.g., MORNINGS, which represents a set of mornings, where each morning starts at 8:00 am, ends at 11:00, and lasts at least 30 minutes). In addition, we allow short-cuts such as when a plan should start immediately at the current time, whatever that time is (using the symbol *NOW*), or when a condition should hold during the span of time over which the plan is executed, whatever that span is (using the symbol *).

For example,

```
MIDNIGHTS <- [0, 0 HOURS, 24 HOURS]
```

;;MIDNIGHTS is a set of cyclic time points

```
MORNINGS <- [[8 HOURS, 8 HOURS],[11 HOURS, 11 HOURS],[30 MINUTES, _],MIDNIGHTS]
```

;;MORNINGS is a set of cyclic time annotations or intervals with uncertainty concerning starting and ending that uses midnights as a reference annotation

All domain-dependent time annotations, units, and time abstractions have to be defined in advance to be applicable in all plans in the guideline-specification library. The definitions ensure that site-specific practice can be clarified and specified (e.g., DAYS start at 0:00 am or DAYS start at 7:00 am). We allow variable assignments of time units, domain-dependent time-points, time-intervals, and cyclic time abstractions.

In addition, a sampling-frequency argument specifies the frequency of sampling the external-wolrd's data, such as when verifying the applicability of a particular plan. Thus, we define a sampling frequency for examining the plan's state-transition criteria (see Section 3.3.3).

Our notation enables the expression of interval-based intentions, states, and prescribed actions with uncertainty regarding starting, finishing, duration, and the use of absolute, relative, and even cyclical (with a predetermined granularity) reference annotations.

### 3.2. The semantics of the Asbru task-specific knowledge roles

A (guideline) **plan** in the guideline-specification (plan) library is composed hierarchically, using the Asbru syntax, of a set of plans with arguments and time annotations. A decomposition of a plan into its subplans is always attempted by the execution interpreter, unless the plan is not found in the guideline-specification library, thus representing a nondecomposable plan (informally, an *action* in the classical planning literature). This can be viewed as a "semantic" halting condition. Such a plan is referred to the agent for execution, which may result in an interaction with a user or an external calling of a program. The library also includes a set of **primitive** (nondecomposable) **plans** to perform interaction with the user or external devices such as asking the user for advice or retrieving particular information from the medical patient record (e.g., OBSERVE, GET-PARAMETER, ASK-PARAMETER, DISPLAY, WAIT)). Plans have return values.

During the execution phase, an applicable plan is instantiated. A set of mutually exclusive **plan states** describes the actual status of the plan during execution. Particular **state-transition criteria** specify transition between neighboring plan states. For example, if a plan has been started, it can only be completed, suspended, or aborted depending on the corresponding criteria. Figure 4 illustrates the different plan states and their corresponding transition criteria mentioned on the arrows. The gray triangle includes the three basic states and associated transition criteria; these should always be defined. The `suspended` and `restarted` states are optional and are available for more complex plan types (the `restarted` state can be eliminated by combining it with the `started` state into an `active` state and using temporal queries to differentiate between the two types of `active` state). The state-transition conditions are explained below. Generic library plans (i.e., plan types) also have states, such as `considered`, `possible`, `rejected`, and `ready`, that determine if a plan type is applicable and whether a plan instance can be created.

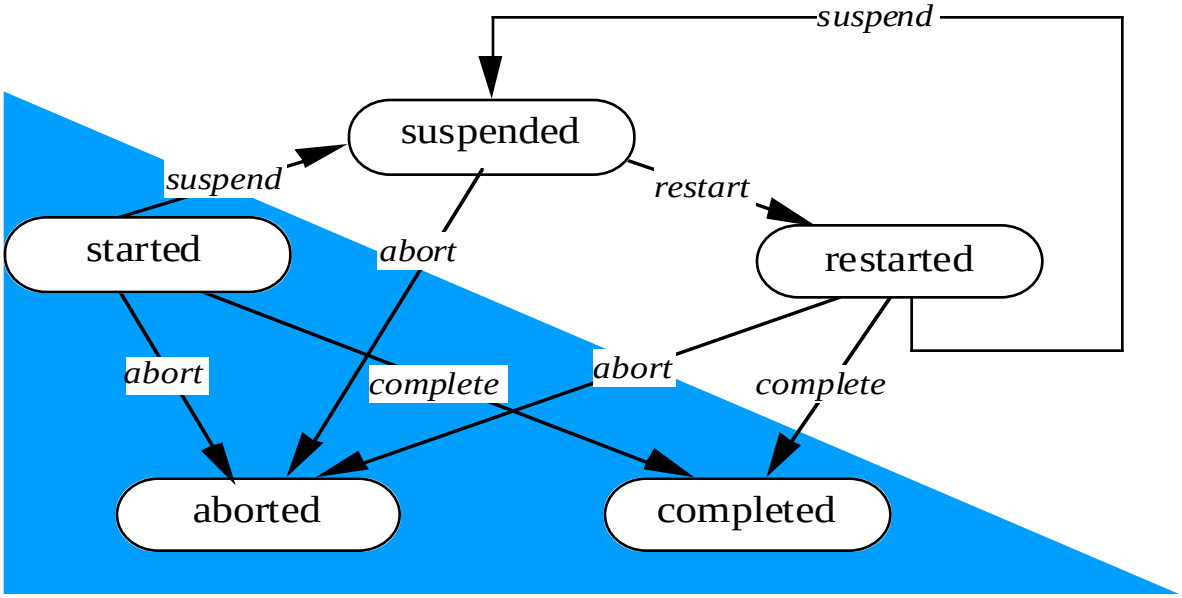


**Figure 4**. The various plan-instance states and associated state-transition criteria in Asbru.

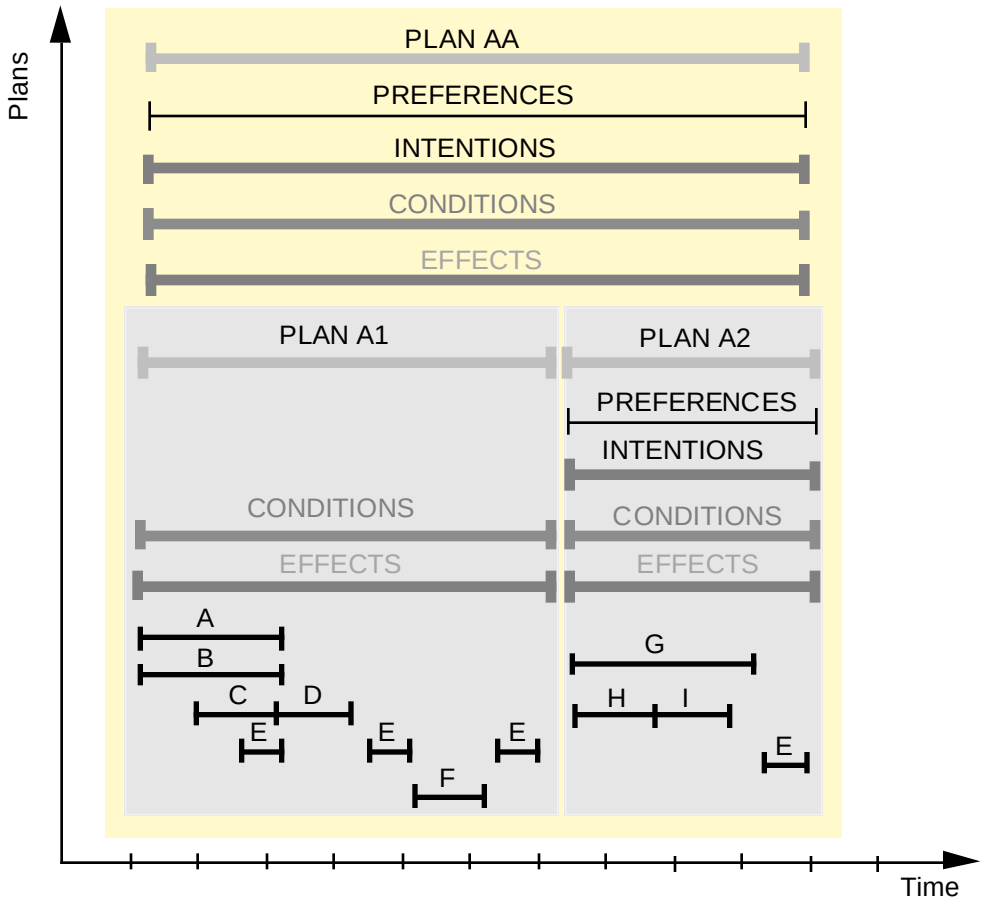


**Figure 5**. Graphical representation of a clinical-guideline specification represented in Asbru. Plan AA is of a aequential type and includes plans A1 and A2 in sequence; plan A1 is of a concurrent type and includes plans such as A, B, and C, and cyclical plan E.

A plan consists of a name, a set of arguments, including a time annotation (representing the temporal scope of the plan), and five components: **preferences**, **intentions**, **conditions**, **effects**, and a **plan body** which describes the actions to be executed. The general arguments, the time annotation, and all components are optional. A subplan has the same structure (Figure 5).

The relationship between the various shared task-specific knowledge roles mentioned in Table 1 and shown in Figure 5, and the Asbru language syntax is shown in Table 3.

We shall now examine in more detail each of the knowledge roles represented in Asbru.

**Preferences:** Preferences bias or constrain the selection of a plan to achieve a given goal and express a kind of behavior of the plan. We distinguish between:

(1) **Strategy**: a general strategy for dealing with the problem (e.g., aggressive, normal);
(2) **Utility**: a set of utility measures (e.g., minimize the cost or inconvenience);
(3) **Select-method**: a matching heuristic for the applicability of the whole plan (e.g., exact-fit);
(4) **Resources**: a specification of prohibited or obligatory resources (e.g., in certain cases of treatment of a pulmonary infection, surgery is prohibited and antibiotics must be used);
(5) **Start-conditions:** an indication whether transition from a `ready` generic plan to the `started` state of an actual plan instance is automatic (after applying the *filter* and *setup* preconditions—see below) or requires approval of the user.

**Table 3**: Relationship of Asbru syntactic elements to the task-specific knowledge roles

| Task-specific knowledge role | Syntactic element in Asbru |
|---|---|
| Preferences: constrain the selection of a plan | PREFERENCES (STRATEGY, UTILITY, LOOK-AHEAD, SELECT-METHOD, RESOURCES, START-CONDITION) |
| Intended intermediate state: pattern that is intended to hold during plan execution | INTENTION:INTERMEDIATE-STATE |
| Intended overall states: pattern that is intended to hold at the end of plan execution | INTENTION:OVERALL-STATE |
| Intended intermediate actions: action pattern that is intended hold during plan execution | INTENTION:INTERMEDIATE-ACTION |
| Intended overall actions: action pattern that is intended to hold at the end of plan execution | INTENTION:OVERALL-ACTION |
| Filter preconditions that need to be true for the plan to be applicable | FILTER-PRECONDITIONS |
| Setup preconditions that need to be achieved so that the plan can start | SETUP-PRECONDITIONS |
| Suspension conditions that cause the plan to be suspended | SUSPEND-CONDITIONS |
| Restarting conditions that restart a suspended plan | RESTART-CONDITION |
| Abort conditions that abort the plan | ABORT-CONDITIONS |
| Completion conditions that determine when the plan is completed | COMPLETE-CONDITIONS |
| Prescribed actions | PLAN-BODY |
| Effects of plans in relation to measurable parameters | ARGUMENT-DEPENDENCIES<br>PLAN-EFFECTS |

**Intentions:** Intentions are high-level goals at various levels of the plan, an annotation specified by the designer, which supports tasks such as critiquing and modification. Intentions are temporal patterns of executing-agent actions and external-world states that should be maintained, achieved, or avoided. We define four categories of intentions:

(1) **Intermediate state:** the state(s) that should be maintained, achieved, or avoided during the applicability of the plan (e.g., weight gain levels are slightly low to slightly high);
(2) **Intermediate action:** the action(s) that should take place during the execution of the plan (e.g., monitor blood glucose once a day);
(3) **Overall state pattern:** the overall pattern of states that should hold after finishing the plan (e.g., patient had less than one high glucose value per week);
(4) **Overall action pattern:** the overall pattern of actions that should hold after finishing the plan (e.g., patient had visited dietitian regularly for at least three months).

For example (see appendix),

```
(INTENTION:INTERMEDIATE-STATE
  (MAINTAIN STATE(mothers-body-weight-gain)
    (OR SLIGHTLY-LOW NORMAL SLIGHTLY-HIGH) GDM-Type-II
    [[24 WEEKS, 24 WEEKS], [DELIVERY, DELIVERY], [_,_], CONCEPTION]))
(INTENTION:OVERALL-ACTION
  (MAINTAIN visit-dietitian regularly GDM-Type-II
      [[24 WEEKS, 24 WEEKS], [DELIVERY, DELIVERY], [3 MONTHS,_], CONCEPTION])
```

**Conditions:** Conditions are temporal patterns, sampled at a specified frequency, that need to hold at particular plan steps to induce a particular state transition of the plan instance. We do not directly determine conditions that should hold during execution. We specify different conditions that enable transition from one plan state into another (see Figure 4). A plan is completed when the completed conditions become true, otherwise the plan's execution suspends or aborts. Aborting a plan's execution is often due to a failure of the plan or part of it. All conditions are optional.

We distinguish between:

(1) **Filter-preconditions**, which need to hold initially if the plan is applicable, but should not be achieved (e.g., patient is a pregnant female), and are necessary for a `possible` state;

(2) **Setup-preconditions**, which need to be achieved to enable a plan to start (e.g., patent had a glucose-tolerance test) and allow a transition from a `possible` plan to a `ready` plan;

(3) **Suspend-conditions**, which determine when a started plan has to be suspended (e.g., blood glucose has been high for at least four days); these implicitly what the planning literature calls *protection intervals* (Kambhampati et al. 1995), in which certain conditions need to hold;

(4) **Abort-conditions**, which determine when a started, suspended, or restarted plan has to be aborted (e.g., there is an insulin-indicator condition: the patient cannot be controlled by diet);

(5) **Complete-conditions**, which determine when a started or restarted plan has to be completed successfully (e.g., delivery has been performed);

(6) **Restart-conditions**, which determine when a suspended plan has to be restarted (e.g., blood glucose level is back to normal or is only slightly high); these can be seen as "automatic" or "internal" reactivation conditions: others might be imposed by the higher-level plan.

For example,

```
(SUSPEND-CONDITIONS (OR STARTED RESTARTED) ;; two possible transition-source states exist
  (STATE(blood-glucose) HIGH GDM-Type-II ;suspend if the glucose is HIGH during this period
    [[24 WEEKS,24 WEEKS], [DELIVERY, DELIVERY], [4 DAYS,_], CONCEPTION]
    (SAMPLING-FREQUENCY 24 HOURS))))
(ABORT-CONDITIONS (OR STARTED SUSPENDED RESTARTED)
  (insulin-indicator-conditions TRUE GDM-Type-II *
    (SAMPLING-FREQUENCY 24 HOURS))) ;abort simpler plan if there is an indication for need of insulin
```

**Effects:** Effects describe the functional relationship between the plan arguments and measurable parameters (e.g., the *dose* of insulin is inversely related to the level of blood glucose) or the overall effect of a plan on parameters (e.g., administration of insulin decreases the blood glucose). Effects have a likelihood annotation—a probability of occurrence.

For example, in the context of GDM, the dose argument of the insulin-administration plan has a negative-monotonic relationship to the blood-glucose level, for any reaction time. This effect relation (ignoring in this case the temporal span and likelihood) is written as

```
(ARG-DEPENDENCY (dose GDM glucose_level NEGATIVE-MON
                [[_, _], [_, _], [[_, _], *NOW*], _ ))
```

The overall effect of the plan in the context of GDM decreases the blood-glucose level, the reaction time is between 10 and 60 minutes, and the likelihood is 0.97. This overall effect is written as

```
(PLAN-EFFECTS (GDM glucose_level DEC
                [[_, _], [_, _], [10 MINUTES, 60 MINUTES], *NOW*] 0.97))
```

**Table 4:** Categorization of plan types according to continuation conditions and ordering constraints

| **Continuation condition --> Ordering Constraints** | All plans should be completed to continue | Some plans should be completed to continue |
|---|---|---|
| Start together | DO-ALL-TOGETHER (no continuation-condition; all plans must complete) | DO-SOME-TOGETHER (continuation-conditions specified as subset of plans) |
| Execute in any order | DO-ALL-ANY-ORDER (no continuation-condition; all plans must complete) | DO-SOME-ANY-ORDER (continuation-conditions specified as subset of plans) |
| Execute in total order (sequence) | DO-ALL-SEQUENTIALLY (no continuation-condition; all plans must complete) | ------ |

**Plan-Body:** The plan body is a set of plans to be executed in parallel, in sequence, in any order, or in some frequency. We distinguish among several types of plans: *sequential*, *concurrent*, and *cyclical*. Only one type of plan is allowed in a single plan body. A sequential plan specifies a set of plans that are executed in sequence; for continuation, all plans included have to be completed successfully. Concurrent plans can be executed in parallel or in any order. We distinguish two dimensions for classification of sequential or (potentially) concurrent plans: the number of plans that should be completed to enable continuation and the order of plan execution. Table 4 summarizes the dimensions of the two plan types. Using the two dimensions, we define the operators DO-ALL-TOGETHER, DO-SOME-TOGETHER, DO-ALL-ANY-ORDER, DO-SOME-ANY-ORDER, DO-ALL-SEQUENTIALLY. The continuation condition specifies the names of the plans that must be completed to proceed with the next steps in the plan. For instance:

```
(DO-ALL-TOGETHER ; a sequential plan type in which continuation depends on completion of the preceding plan
  (glucose-monitoring)
  (nutrition-management)
  (OBSERVE-insulin-indicators)); three subplans; the plan body of each can be of any type
```

the plans that must be completed to proceed with the next steps in the plan.

A cyclical plan (an EVERY clause) includes a plan that can be repeated, and optional temporal and continuation arguments that can specify its behavior. *Start* and *end* specify a starting and ending time point. *Time base* determines the time interval over which the plan is repeated and the start time, end time, and duration of the particular plan instance in each cycle (e.g., starting with the first Monday's morning, until next Tuesday's morning, perform plan A every morning for 10 minutes). The *times-completed* argument specifies how many times the plan has to be completed to succeed and the *times-attempted* argument specifies how many attempts are allowed. Obviously, the number of attempts must be greater or equal to the number of completions. A temporal pattern can be used as a stop condition of the cyclic plan. Finally, the plan itself is associated with its own particular arguments (e.g., dose). The start time, the time base, and the plan name are mandatory to the specification of a cyclic plan; the other arguments are optional.

For example, consider the following plan: "Administer 5 units of insulin every morning starting with the first morning following the innitiation of the plan." This plan could be written as:

```
(EVERY
  (START (FIRST(MIDNIGHT) after (STARTED *self*))
  (TIME-BASE [[8 HOURS, 8 HOURS], [11 HOURS, 11 HOURS], [_,_], MIDNIGHTS]
  (administer-insulin 5)
END-EVERY )
```

Note that *morning* is a cyclic time annotation and is represented in this case as the interval 8 a.m. to 11 a.m., expressed as a time shift from the cyclic set of time points MIDNIGHTS. The duration of administration of insulin in this case is not constrained, but it could be. Note also that no stop condition is defined in this case; the plan would continue indefinitely.

## 4. ACQUISITION AND MAINTENANCE OF SKELETAL PLANS

Domain experts, such as expert physicians, need not have familiarity with the syntax of the Asbru language to author skeletal plans, such as clinical guidelines. Graphic **knowledge-acquisition** (**KA**) tools can be generated automatically by systems such as **PROTÉGÉ-II** (Eriksson, et al. 1995; Musen, et al. 1995; Tu, et al. 1995). The KA tools can internally use the Asbru representation or its equivalent, but that representation need not necessarily be known to the user. In addition to creation of an internal (e.g., object-oriented) version of the plan, the KA tool should be able to generate a text-based Asbru version. The Asbru version can then be used as a sharable machine-readable version that does not depend on any particular platform and will also be useful for reading and editing by more knowledgeable designers. We have explored the option of generating an automated graphic KA tool for acquiring the set of shared knowledge roles, using the PROTÉGÉ-II suite of tools, with encouraging results.

### 4.1. Modeling a clinical-guideline ontology using the PROTÉGÉ-II methodology

PROTÉGÉ-II is a set of tools and a methodology to develop knowledge based systems. We used **PROTÉGÉ/Win** (the Windows version of PROTÉGÉ-II) to develop the ontology and to generate a KA tool automatically from the ontology. This ontology is shared by the task-specific cluster of problem-solving methods relevant to the support of skeletal-plan execution. In PROTÉGÉ-II terms, we have developed a *method ontology*, global to all our problem-solving methods. By this we mean that the ontology is in theory local (specific) to some hypothetical, all-encompassing method that performs the task cluster, but is in practice global to (shared by) all the methods that perform subtasks in that cluster. This is sensible in this case, as there is a great deal of overlap in the concepts needed for the different tasks, and the roles undertaken by these concepts are the same in all their uses by this set of tasks. The *domain ontology* in the case of clinical guidelines is also required, but not shown here. The domain ontology specifies concepts, such as drugs, diseases, patient findings, tests, clinic visit types.

Ontologies in PROTÉGÉ-II are represented as a hierarchy of classes. Each class is represented as a frame with slots. Slots may be constrained to basic data types, or to be instances of another class defined in the ontology, thus allowing the expression of relationships in the ontology. The PROTÉGÉ/Win OntologyEditor tool was used to capture the ontology of the cluster of methods supporting skeletal-plan execution. Figure 6 shows a portion of that ontology.

Once the ontology is defined, the PROTÉGÉ/Win LayoutEditor tool automatically generates a specification of a KA tool for this ontology. The specification of the KA tool is interpreted by the PROTÉGÉ/Win LayoutInterpreter. It is possible to change the layout of the user interface to some degree in the LayoutEditor. The resultant KA tool can then be used to acquire instances of the ontology, which in this case would be guidelines in the Asbru language.

The knowledge roles in the Asbru syntax can be viewed as a set of slots in a frame-based ontology of skeletal plans. The ontology that we have developed mirrors the BNF Syntax of the Asbru language. As this language is not object oriented, the ontology is fairly flat, in that inheritance is not used significantly. The concepts of inheritance and polymorphism can be usefully applied to this domain, indeed it seems more natural to express the ontology in this form. As an example, all plans in the current ontology share the same state transition criteria, which has been chosen as one which applies to most actions. However, in a hierarchical ontology it is easy to create special subclasses of the plan class which have variants of the state transition criteria. Such an ontology maps well as an object-oriented language.

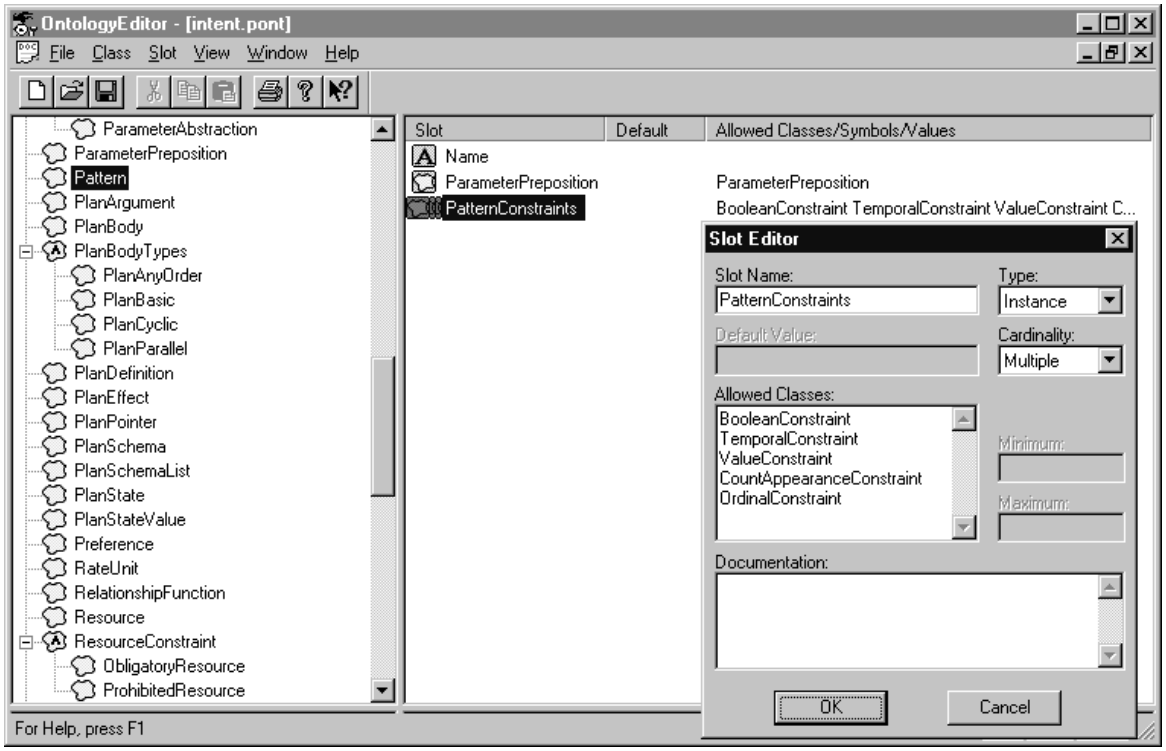


**Figure 6**. Part of the eexecution–support methods ontology, represented by the PROTÉGÉ/Win OntologyEditor

### 4.2. A knowledge-acquisition tool for support of clinical-guideline execution

We have generated an automated graphic KA tool for the object-oriented version of the Asbru language, using the PROTÉGÉ-II suite of tools, with encouraging results. Figure 7 shows an example of using the KA tool to acquire a part of the GDM guideline (see appendix).

If the user is conversant with the syntax of the Asbru language, it may be quicker to design guidelines by writing them in the language, using 'copy and paste' functions or editor macros. If the user, in particular a domain expert, is unfamiliar with the syntax, then it is easier to use the KA tool. The complexity of the ontology enforces the automatic generator of the KA tool to produce a user interface with many cascading, small dialogs. Thus, for providing an optimal dialog, more control of the layout of the automatically generated user interface was needed than was possible in early versions of PROTÉGÉ/Win. Once this additional feature became available, we managed to generate and customize significantly better KA tools, although more improvements can take place.

Another significant benefit of the KA tool approach is that it detects incorrect syntax while authoring a guideline. Thus, implicit syntactic support is provided at no cost.

## 5. SUMMARY AND DISCUSSION

Representing complex execution plans, such as clinical guidelines, and the intentions underlying them in a standard, sharable, acquirable, machine-readable, and machine-interpretable form is imperative for sharing execution plans and for useful, flexible automated assistance in the execution of these plans. In the multiple domains of clinical medicine, such a task-specific representation is crucial for dissemination of modern clinical knowledge, since the use of clinical guidelines can improve the quality of care (Grimshaw and Russel 1993). The representation we suggest supports several different *knowledge roles* that can be used by multiple reasoning modules, both for direct execution of a guideline and for related tasks such as recognition of the human executing agent's intentions and for critiquing constructively that agent's actions.

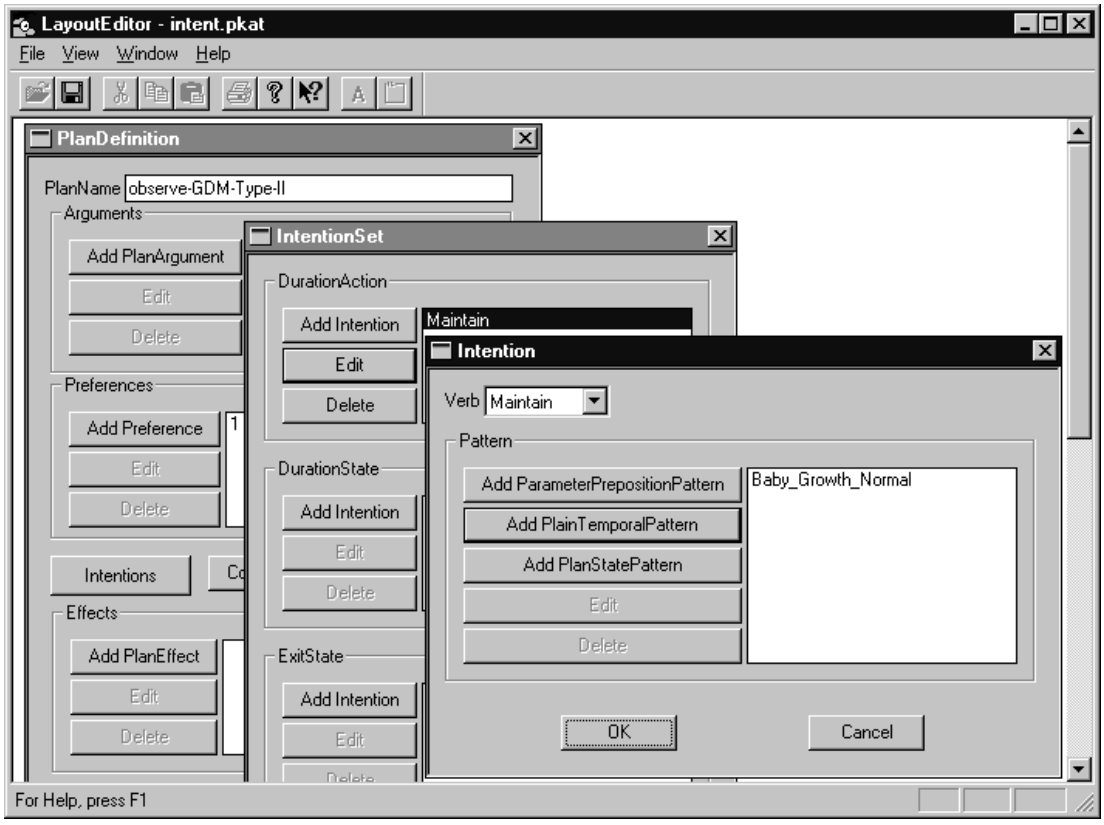


**Figure 7.** Screen shot of knowledge-acquisition tool, showing a part of the GDM type II guideline

Besides explicit knowledge roles specific to the set of tasks involved in supporting skeletal-plan execution, the Asbru language places a particular emphasis on an expressive representation for time-oriented actions and world states. Temporal reasoning is an important feature of dynamic execution domains such as those of clinical medicine and needs to be supported.

In the Asgaard project, we are currently focusing on the development of the execution, plan recognition, and critiquing problem-solving methods.

## Acknowledgments

This work has been supported by grants LM05708 and LM06245 from the National Library of Medicine, IRI-9528444 from the National Science Foundation, and DARPA Grant N66001-94-D-6055. Computing resources were provided by the Stanford CAMIS project, funded under Grant No. LM05305 from the National Library of Medicine. Silvia Miksch is supported by "Erwin Schrödinger Auslandstipendium, Fonds zur Förderung der wissenschaftliche Forschung", J01042-MAT.

## References

Bacchus, F. and Kabanza, F. (1996). Planning for Temporally Extended Goals. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, OR,* 1215–1222. AAAI Press/The MIT Press, Menlo Park, CA.

Barnes, M. and Barnett, G. O. (1995). An Architecture for a Distributed Guideline Server. In Gardner, R. M. (Ed.) *Proceedings of the Annual Symposium on Computer Applications in Medical Care (SCAMC-95)*, New Orleans, Louisiana, 233-7. Hanley & Belfus.

Bratman, M.E. (1987). *Intention, Plans and Practical Reason.* Cambridge, MA: Harvard University Press.

Dechter, R., Meiri, L., and Pearl, J. (1991). Temporal Constraint Networks. *Artificial Intelligence, Special Volume on Knowledge Representation,* 49(1-3):61–95.

Eriksson, H., Shahar, Y., Tu, S. W., Puerta, A. R., and Musen, M. A. (1995). Task Modeling with Reusable Problem-Solving Methods. *Artificial Intelligence,* 79(2):293-326.

Fikes, R. E. and Nilsson, N. J. (1971). A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence,* 2(3/4):189-208.

Friedland, P., and Iwasaki, Y. The Concept and Implementaion of Skeletal Plans. *Journal of Automated reasoning*, 1(2):161–208.

Grimshaw, J. M. and Russel, I. T. (1993). Effects of Clinical Guidelines an Medical Practice: A Systematic Review of Rigorous Evaluation. *Lancet,* 342:1317-22.

Hayes, P. (1978). The Naïve Physics Manifesto. In Michie, D. (Ed.) *Expert Systems in the Microelectronic Age*, Edinburgh: Edinburgh University Press.

Hayes, P. (1985). The Second Naïve Physics Manifesto. In Bobrow, D. G. (Ed.) *Qualitative Reasoning About Physical Systems*, Cambridge, MA: MIT Press.

Herbert, S. I., Gordon, C. J., Jackson-Smale, A., and Renaud Salis, J-L. (1995). Protocols for Clinical Care. *Computer Methods and Programs in Biomedicine,* 48:21-6.

Hickam, D. H., Shortliffe, E. H., Bischoff, M. B., Scott, A. C., and Jacobs, C. D. (1985). A Study of the Treatment Advice of a Computer-Based Cancer Chemotherapy Protocol Advisor. *Ann Intern Med.,* 103:928-36.

Hripcsak, G., Ludemann, P., Pryor, T. A., Wigertz, O. B., and Clayton, P. D. (1994). Rationale for the Arden Syntax. *Computers and Biomedical Research,* 27:291-324.

Kambampati, S., Knoblock, C., Yang, Q. (1995). Planning as Refinement Search: A Unified Framework for Evaluating Design Tradeoffs in Partial-Order Planning. *Artificial Intelligence, Special Volume on Planning and Scheduling,* 76(1-2):167–238.

Levinson, R. (1995). A General Programming Language for Unified Planning and Control. *Artificial Intelligence, Special Volume on Planning and Scheduling,* 76(1-2):319-375.

Liem, E. B., Obeid, J. S., Shareck, P. E., Sato, L., and Greens, R. A. (1995). Representation of Clinical Practice Guidelines Through an Interactive World-Wide-Wb Interface. In Gardner, R. M. (Ed.) *Proceedings of the Annual Symposium on Computer Applications in Medical Care (SCAMC-95),* New Orleans, Louisiana, 223-7. Hanley & Belfus.

Miksch, S., Horn, W., Popow, C., and Paky, F. (1993). VIE-VENT: Knowledge-Based Monitoring and Therapy Planning of the Artificial Ventilation of Newborn Infants. In Andreassen, S., et al. (Eds.), *Proceedings of the Artificial Intelligence in Medicine, 4th Conference on Artificial Intelligence in Medicine Europe (AIME-93)*, 218-29. IOS Press, Amsterdam.

Miksch, S., Horn, W., Popow, C., and Paky, F. (1996a). Context-Sensitive and Expectation-Guided Temporal Abstraction of High-Frequency Data. In Iwasaki, Y. and Farquhar, A. (Eds.), *Proceedings of the Tenth International Workshop for Qualitative Reasoning (QR-96),* Stanford Sierra Camp. Fallen Leaf Lake, CA, 154-63. AAAI Press, AAAI Technical Report WS-96-01, Melno Park.

Miksch, S., Horn, W., Popow, C., and Paky, F. (1996b). Utilizing Temporal Data Abstraction for Data Validation and Therapy Planning for Artificially Ventilated Newborn Infants. *Artificial Intelligence in Medicine,* forthcoming:

Miller, P. L. (1986).*Expert Critiquing System: Practice-Based Medical Consultation by Computer* New York, NY: Springer .

Musen, M. A., Carlson, C. W., Fagan, L. M., Deresinski, S. C., and Shortliffe, E. H. (1992). T-HELPER: Automated Support for Community-Based Clinical Research. In Frisse, M. E. (Ed.) *Proceedings of the Sixteenth Annual Symposium on Computer Applications in Medical Care (SCAMC-92)*, 719-23. McGraw Hill, New York, NY.

Musen, M. A., Gennari, J. H., Eriksson, H., Tu, S. W., and Puerta, A. R. (1995). PROTÉGÉ -II: A Computer Support for Development of Intelligent Systems from Libraries of Components. *Proceedings of the Eighth World Congress on Medical Informatics (MEDINFO-95),* Vancouver, BC, 766-70.

Musen, M.A., Tu, S.W., Das, A.K., and Shahar, Y. (1996). EON: A Component-Based Approach to Automation of Protocol-Directed Therapy. *Journal of the American Medical Information Association* **3** (6) (in press).

Pollack, M. (1992). The Use of Plans. *Artificial Intelligence,* 57(1):43-68.

Rit, J.-F. (1986). Propagating Temporal Constraints for Scheduling. *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86),*, 383-8. Morgan Kaufmann, Los Altos, CA.

Shahar, Y. (in press). A Framework for Knowledge-Based Temporal Abstraction. *Artificial Intelligence.*

Shahar, Y. and Musen, M. A. (1993). RÉSUMÉ: A Temporal-Abstration System for Patient Monitoring. *Computers and Biomedical Reearch*, 26(3):255-73.

Shahar, Y. and Musen, M. A. (1995). Plan Recognition and Revision in Support of Guideline-Based Care. *Proceedings of the Workshop Notes of the AAAI Spring Symposium on Representation Mental States and Mechanisms*, Stanford, CA, 118-26.

Shahar, Y. and Musen, M. A. (1996). Knowledge-Based Temporal Abstraction in Clinical Domains. *Artificial Intelligence in Medicine, Special Issue Temporal Reasoning in Medicine*, 8(3):267–298.

Sherman, E. H., Hripcsak, G., Starren, J., Jender, R. A., and Clayton, P. (1995). Using Intermediate States to Improve the Ability of the Arden Syntax to Implement Care Plans and Reuse Knowledge. In Gardner, R. M. (Ed.) *Proceedings of the Annual Symposium on Computer Applications in Medical Care (SCAMC-95)*, New Orleans, Louisiana, 238-42. Hanley & Belfus.

Tate, A., Drabble, B., and Kibry, R. (1994). O-Plan2: An Open Architecture for Command, Planning, and Controll. In Zweber, M., and Fox, M. S. (Eds.), *Intelligent Scheduling*, Vol. 1, pp. 213-39, San Francisco, CA: Morgan Kaufmann.

Tu, S. W., Eriksson, H., Gennari, J. H., Shahar, Y., and Musen, M. A. (1995). Ontology-Based Configuration of Problem-Solving Methods and Generation of Knowledge-Acquisition Tools: Application of PROTÉGÉ-II to Protocol-Based Decision Support. *Artificial Intelligence in Medicine*, 7(3):257-289.

Tu, S. W., Kahn, M. G., Musen, M. A., Ferguson, J. C., Shortliffe, E. H., and Fagan, L. M. (1989). Episodic Skeletal-Plan Refinement on Temporal Data. *Communications of ACM*, 32:1439-55.

Uckun, S. (1994). Instantiating and Monitoring Skeletal Treatment Plans, Knowledge Systems Laboratory, Stanford University, Stanford, CA 94305, USA, KSL 94-49.

Van der Lei, J. and Musen, M. A. (1991). A Model for Critiquing based on Automated Medical Records. *Computers and Biomedical Research*, 24(4):344-78.

## APPENDIX: Example: A gestational diabetes mellitus (GDM) guideline in Asbru

We represented a part of a guideline, used at the Stanford University Medical Center for controlled observation and treatment of noninsulin-dependent gestational diabetes mellitus (GDM type II), in the Asbru language. Implicit or unmentioned intentions and conditions in the guideline below have been acquired from domain experts and appear in the Asbru representation of the example.

```
Observation and Treatment of Gestational Diabetes Mellitus (GDM)
GLUCOSE MONITORING:
       (after GDM was dedected in third trimester pregnancy, tested by a
       glucose tolerance test (GTT) being between 140 and 200 mg/dl)
(1)  Patients will check glucose values four times/day (fasting and one hour
     postprandial glucose)
(2)  Preprandial, bedtime and 2 AM blood glucose will be added at the
     discretion of the physician
(3)  ... deleted ...
(4)  Treatment goals should be no higher than 130 mg/dl for 1-hour post meals,
     < 100 mg/dl fasting and preprandial
(5)  ... deleted ...

NUTRITION:
(1)  Patients should be taught a diet based on the patients weight, activity
     level and number of fetus (regular meals: 3 meals, 3 snacks).
... (omitted for lack of space) ...
INSULIN THERAPY:
... (omitted for lack of space) ...
```

The plan body consists of three plans that are executed in parallel. These plans are decomposable into other plans, which exist in the guideline-specification library. Nondecomposable plans are executed by the executing agent. Plan names are written in bold characters.

```
(PLAN observing-GDM-Type-II

;; the following time-annotations are local to the GDM example
(DOMAIN-DEPENDENT TIME-ASSIGNMENT
  (SHIFTS DELIVERY <– 38 WEEKS)
  ;; time shift from CONCEPTION
  (POINT CONCEPTION <– (ask (ARG "what is the conception-date?")))
(ABSTRACTION-ASSIGNMENT
  (CYCLIC
    MIDNIGHTS <– [0, 0 HOURS, 24 HOURS]
    BREAKFAST-START-TIME <– [0, 7 HOURS, 24 HOURS]))

  (PREFERENCES
    (SELECT-METHOD EXACT-FIT)
    (START-CONDITION AUTOMATIC))
    ;; The match in the filter conditions needs to be exact and the plan starts as soon as it is in a ready state

  (INTENTION:INTERMEDIATE-STATE
    (MAINTAIN blood-glucose-post-meal (<= 130) NIL ;; a raw data value (no context)
      [[24 WEEKS, 24 WEEKS], [DELIVERY, DELIVERY], [_,_], CONCEPTION])
    (MAINTAIN blood-glucose-fasting (<= 100) NIL   ;; a raw data value (no context)
      [[24 WEEKS, 24 WEEKS], [DELIVERY, DELIVERY], [_,_], CONCEPTION])
  (MAINTAIN STATE(mothers-body-weight-gain)
      (OR SLIGHTLY-LOW NORMAL SLIGHTLY-HIGH) GDM-Type-II ;; a context-specific value
      [[24 WEEKS, 24 WEEKS], [DELIVERY, DELIVERY], [_,_], CONCEPTION])
  )
  (INTENTION:INTERMEDIATE-ACTION
    (MAINTAIN diet regular-meals GDM-Type-II
      [[24 WEEKS, 24 WEEKS], [DELIVERY, DELIVERY], [_,_], CONCEPTION])
  )
  (INTENTION:OVERALL-STATE
    (AVOIDED STATE(blood-glucose) HIGH GDM-Type-II
      [[24 WEEKS, 24 WEEKS], [DELIVERY, DELIVERY], [7 DAYS,_], CONCEPTION])
    ;; avoid high blood-glucose level (as defined in the context of therapy for GDM type II) for more than 7 days
  ) ;; in the period from 24 conception weeks to delivery, using the estimated conception date as a reference point

(SETUP-PRECONDITIONS
  (PLAN-STATE one-hour-GTT COMPLETED
    [[24 WEEKS, 24 WEEKS], [26 WEEKS, 26 WEEKS], [_,_], CONCEPTION])
    ;; The patient must have had a glucose-tolerane test (another plan in the library) successfully completed
(FILTER-PRECONDITIONS
  (one-hour-GTT (140, 200) pregnancy
      [24 WEEKS, 24 WEEKS], [26 WEEKS, 26 WEEKS], [_,_], CONCEPTION])
(SUSPEND-CONDITIONS (OR STARTED RESTARTED)
  (STATE(blood-glucose) HIGH GDM-Type-II
    [[24 WEEKS,24 WEEKS], [DELIVERY, DELIVERY], [4 DAYS,_], CONCEPTION]
    (SAMPLING-FREQUENCY 24 HOURS)))
    ;; suspend if high blood-glucose level (in the context of GDM type II therapy) exists for at least 4 DAYS
(ABORT-CONDITIONS (OR STARTED SUSPENDED RESTARTED)
  (insulin-indicator-conditions TRUE GDM-Type-II *
    (SAMPLING-FREQUENCY 24 HOURS)))
(COMPLETE-CONDITIONS (OR STARTED RESTARTED)
  (delivery TRUE GDM-Type-II * (SAMPLING-FREQUENCY 30 MINUTES)))
```

```
(RESTART-CONDITIONS ;; restart from a suspended state
  (STATE(blood-glucose) (OR NORMAL SLIGHTLY-HIGH) GDM-Type-II
    [[24 WEEKS,24 WEEKS], [DELIVERY, DELIVERY], [_,_], CONCEPTION]
  (SAMPLING-FREQUENCY 24 HOURS)))

  ((PLAN-EFFECTS (GDM-Type-II glucose NORMAL
                ([_, _], [_, _], [10 MINUTES, 60 MINUTES], *NOW*) 0.97))

  (DO-ALL-TOGETHER
  (glucose-monitoring)
  (nutrition-management)
  (OBSERVE-insulin-indicators)
));; the plan body is a concurrent one and comprises three plans that start together, all of which need to complete

(PLAN glucose-monitoring
  (PREFERENCES
  (SELECT-METHOD EXACT-FIT)
  (START-CONDITION AUTOMATIC))

  (INTENTION:INTERMEDIATE-STATE
  (MAINTAIN STATE(blood-glucose)
      (OR NORMAL SLIGHTLY-HIGH) GDM-Type-II
      [[24 WEEKS, 24 WEEKS], [26 WEEKS, 26 WEEKS], [_,_], CONCEPTION])

  (SETUP-PRECONDITIONS
    (glycometer-equipment-available TRUE GDM-Type-II *))

  (FILTER-PRECONDITIONS
    (GDM-Type-II-diagnose TRUE pregnancy *))

  (DO-ALL-TOGETHER
  (monitor-fasting-glucose (ARG NORMAL glucometer))
  (monitor-one-hour-after-breakfast-glucose (ARG NORMAL glucometer))
  (monitor-one-hour-after-lunch-glucose (ARG NORMAL glucometer))
  (monitor-one-hour-after-dinner-glucose (ARG NORMAL glucometer))
  (IF (physician-decided-more-analyses TRUE GDM-Type-II *)
                THEN   (monitor-alternative-times
                       (ARG NORMAL glucometer))
  )))

(PLAN monitor-fasting-glucose (ARG glucose-value device)
  (PREFERENCES
  (START-CONDITION AUTOMATIC))
  (EVERY
  (START (FIRST(MIDNIGHT) after (STARTED *self*))
  ;; first midnight after started the current plan
  (TIME-BASE [[-1 HOURS, -1 HOURS], [-10 MINUTES, -10 MINUTES], [_,_],
                BREAKFAST-START-TIME])
  (UNTIL (COUNT-APPEARANCE 3
                (blood-glucose STATE(blood-glucose) HIGH GDM-Type-II
                 [[24 WEEKS,24 WEEKS], [DELIVERY, DELIVERY],
                 [3 DAYS,7 DAYS], CONCEPTION] ))
  ;;elevated-blood-glucose more than 3 times

  (observe blood-glucose device glucose-value)
END-EVERY )
```