

Goal-Based Decisions for Dynamic Planning

E. Black¹, D.W. Glasspool², M.A. Grando², V. Patkar³, and J. Fox¹

¹ Department of Engineering Science, University of Oxford

² School of Informatics, University of Edinburgh

³ Medical School, University College London

Abstract. The need for clinical guidelines to be implemented at different sites, to adapt to rapidly changing environments, and to be carried out by distributed clinical teams, implies a degree of flexibility beyond that of current guideline languages. We propose an extension to the *PROforma* language allowing hierarchical goal-based plans. Sub-plans to achieve goals are proposed at runtime so that changing circumstances may be flexibly accommodated without redefining the workflow.

1 Introduction

The effective deployment and maintenance of computer-interpretable guidelines (CIGs [5]) requires a great deal more than simply translating a paper guideline to machine-readable form [7,10]. Firstly, if a clinical guideline is to be rolled out over a number of sites, adapting it to work in each location can require significant effort [10]. Secondly, healthcare is a fundamentally distributed activity, a CIG must provide an execution model that allows the tasks to be delegated between individuals and to be carried out by different members of the clinical team. Thirdly, healthcare environments are highly dynamic, CIGs need to flexibly adapt to changing circumstances. Fourthly, healthcare environments include a great deal of uncertainty, it is typically difficult to predict exactly what the results of a procedure will be before it is carried out.

In this paper we propose a way to implement flexible goal-based plan specialisation within the task ontology of our group's own CIG language, *PROforma* [3], taking advantage of its argumentation-based decision model to separate decision-relevant knowledge from plan specifications. Our approach allows *a)* flexible local implementation of guidelines taking into account local resources and preferences, *b)* tailoring clinical management based on the patient's response to treatment and *c)* monitoring and manipulating significant clinical goals that are normally implicit in clinical guidelines rather than simple procedural execution of guideline plans which always have a danger of failing. In section 2 we outline the concept, using a concrete example from a complex medical domain (breast cancer treatment) to illustrate the approach. We discuss related approaches and future work in section 3.

Our starting point is the executable process modelling language *PROforma*, which has a declarative syntax and a well-defined operational semantics [8]. *PROforma* bases its process model on a minimal ontology of task classes that

can be composed into networks representing arbitrarily complex plans or procedures. There are four main task classes derived from the root class in the task ontology (called a “keystone”). *Actions* represent procedures to be executed on the external environment (e.g. administer a drug, update a database). *Enquiries* are carried out to acquire information from some person or external system. *Decisions* define choices about what to believe or what to do. *Plans* group tasks (including other plans) and connect them by simple scheduling constraints, where a task can only be performed if another task has been completed.

The four task types inherit a small set of attributes which control task enactment. *Precondition* is a boolean condition that must be satisfied for the task to start execution; *Postcondition* is a boolean condition asserted as true if the task completes successfully; *Goal* is an informal statement of the goal of the task; and the task *State* underlies the execution semantics for the language. *State* may take one of four values [8]: *Dormant*—task has not yet been started; *In progress*—task has been started and is being executed; *Completed*—task has been completely executed; and *Discarded*—task’s scheduling constraints have been satisfied but its preconditions are not true. When a PROforma plan is started all component tasks are in the dormant state. The execution engine repeatedly examines the properties of the tasks in order to determine what state changes and other actions should occur.

2 Goal-Based Hierarchical Planning Based on PROforma

Based on the PROforma task hierarchy, execution semantics and decision model, our proposal adds a new class and new attributes to the task ontology, revises the execution model and defines new scheduling constraints.

In current PROforma, a task may list as “antecedents” a set of tasks that must all be *completed* before it can become *in progress*. In [4], we found this too limited to express several typical workflow patterns and therefore we introduce standard Petri Net scheduling constraints (Begin, End, Join Xor/And, Split Xor/And). We add the new attributes *roles* and *actor* to the root *keystone* class of the task ontology: *roles* is fixed at design time and restricts the possible set of *actors* (which is fixed at runtime and corresponds to the actual executor of the task) allowing delegation of responsibility for tasks. Our proposal also adds a new type of task, *goal*, which inherits from the root task *keystone*. Plans may contain (sub)plans, goals, actions, decisions and enquiries.

Each plan has the new attributes *goalsToAchieve* (the set of goals that the plan could potentially achieve) and *expectedEffects*, which replaces the notion of *postcondition* from PROforma and corresponds to a set of states, not necessarily desirable, that may potentially be satisfied after execution of the plan. Our approach assumes a centralised plan library that might be authored by the relevant governing organisation; typically, each plan in this library will be a collection of goals connected with scheduling constraints, abstracting away from the details of *how* these goals must be achieved. At a local level (say at a particular health authority or hospital) we would expect another plan library with more concrete

plans, defining specific actions, decisions and enquiries to provide locally relevant methods for achieving more abstract goals.

We also assume a repository of clinical knowledge which can be used during plan execution to guide selection of appropriate methods to achieve goals. Again this would be centrally maintained and populated by clinical evidence such as synthesised trial data. This clinical knowledge is applied to the plan execution process via the medium of logical argumentation [2]. Facts and evidence in the repository are composed at run-time into arguments for and against using different plans to achieve outcomes. Arguments have the form:

$\langle PlanName, Sign, Conditions, Outcome, Level \rangle$

where *PlanName* is the identifier of the plan, *Sign* gives the polarity of the argument (*for* or *against*), *Conditions* is a set of conditions that must be satisfied in order to instantiate the argument, *Outcome* gives the particular outcome state that the argument refers to, and *Level* gives a level of support (where this is appropriate to the outcome in question). For example: ¹

“If a patient has early stage breast cancer, then there is an argument for carrying out a mastectomy with axillary clearance as this leads to improved breast cancer specific survival with a likelihood of 97%”, denoted as

$\langle mastAxClearance, for, \{earlyBrCa\}, imprBCSS, 97\% \rangle$

(For simplicity we assume that outcome, plan and goal identifiers are standardised at a national level.) These arguments may be synthesised using generic argument schemas [2] or may be provided ready-made in the repository.

During plan execution, when a goal *g* becomes *in-progress*, all available plan libraries are checked to find the set of candidate plans *P* that each include *g* in their *goalsToAchieve* and whose preconditions are met. For each plan in *P*, the clinical knowledge repository is consulted to construct or find all arguments whose *Outcome* is in the set of *expectedEffects* for the plan and whose set of *Conditions* can be satisfied to instantiate the argument.

For example, consider we have the active goal to improve Breast Cancer Specific Survival: *Achieve(imprBCSS)*. Checking the plan libraries to see which candidate plans include this goal as part of their *goalsToAchieve* attribute and whose preconditions are met gives us four candidate plans, however the *role* attributes of two of these plans require special surgical skills that are not available in this particular hospital. Hence, we are left with two candidate plans: carry out a lumpectomy axillary clearance *LumpAxClearPlan*, or carry out a mastectomy axillary clearance *MastAxClearPlan*. We consult the clinical knowledge repository to find the arguments whose *Outcome* is in the set of *expectedEffects* for these two plans and whose set of conditions are satisfied. This gives us the following set of arguments that are displayed to the user(s).

$\langle MastAxClearPlan, for, \{earlyBrCa\}, imprBCSS, 97\% \rangle$
 $\langle MastAxClearPlan, against, \{earlyBrCa\}, lossShoulderMob, 20\% \rangle$
 $\langle MastAxClearPlan, against, \{earlyBrCa\}, lossSelfImage, likely \rangle$
 $\langle LumpAxClearPlan, for, \{earlyBrCa\}, imprBCSS, 95\% \rangle$

¹ We make no claims about the clinical accuracy of any of the examples in this paper.

$\langle \text{LumpAxClearPlan}, \text{against}, \{\text{earlyBrCa}\}, \text{lossShoulderMob}, 20\% \rangle$
 $\langle \text{LumpAxClearPlan}, \text{against}, \{\text{earlyBrCa}\}, \text{extraRad6weeks}, 95\% \rangle$

A decision mechanism using argumentation-based reasoning (e.g. [2]) can now be applied in order to automatically recommended one or more candidate plans. However, here we allow the user to choose a candidate based on their valuation of the different outcomes. They select the less extensive surgical procedure *LumpAxClearPlan* because the patient highly values preserving her self image.

Goals have the attributes *success_condition* and *abort_condition*. When a goal first becomes *in progress*, the *success_condition* is checked; If it evaluates as true, then the state of the goal moves straight to *completed*. The *success_condition* is also checked when a goal is *in progress* and a candidate plan that has been selected to achieve it has either become *completed* or *discarded*; If the condition is true, then the goal state changes to *completed*; If the condition is false, then the goal remains *in progress* and another candidate plan must be selected to achieve the goal. In this manner, we provide some flexibility to deal automatically with unexpected failure of methods. If the *abort_condition* of an *in progress* goal becomes true at any time, then the goal becomes *discarded* and any *in progress* plan that is being used to try to achieve the goal also becomes *discarded*.

Continuing the running example, the sub plan *LumpAxClearPlan* is completed and the tumour is completely removed. However, histo-pathology report suggests the cancer has spread to the lymph nodes and is ER negative. Therefore the success condition of the goal *Achieve(imprBCSS)* is not yet satisfied and so this goal is still *in progress*. The plan repository is searched again and two alternative plans *AdjChemo1Plan* and *AdjChemo2Plan* (adjuvant chemotherapy plans 1 and 2) are generated as candidate plans (as preconditions on both these candidate plans are true and both plans have as part of their *goalsToAchieve* the goal *Achieve(imprBCSS)*). As before, arguments for and against these candidate plans are constructed from the clinical knowledge repository, allowing the users to select *AdjChemo1Plan*. However, whilst this plan is *in progress*, the patient develops distant cancer metastasis and the disease stage is no longer early Breast Cancer (*earlyBrCa*). Because *not(earlyBrCa)* is an *abort_condition* on our goal *Achieve(imprBCSS)* the goal is now *discarded* and so the remaining part of *AdjChemo1Plan* is also discarded.

3 Discussion

The Asbru language [6] has adopted a similar general approach to hierarchical refinement of plans during execution by selection of appropriate sub-plans from a library based on goals (“intentions” in Asbru) and pre- and post-conditions. Generic workflows that can be specialised at run time are also presented in [9] based on a state (referred to as a “scenario”).

We believe that three aspects of the present work are particularly distinctive: Firstly, in our approach goals are first-class members of the task hierarchy, not simply attributes of tasks. This allows complete separation between the goal and possible procedures that could achieve it, improving the flexibility and clarity of

the approach in our view. Secondly, the use of *argumentation logic* to abstract decision-relevant knowledge away from plan specifications allows clinical knowledge to be maintained entirely separately from the plan library. The approach also allows the information presented to the user to be highly focussed and appropriate. This has proved beneficial in many clinical decision systems [2]. Thirdly, the *PROforma* task hierarchy is simple and provides a sound basis for this extension. The execution semantics are essentially defined for the *keystone* class and need little adjustment to support the new *goal* class. Finally the support for roles and actors provides a basis for a move towards a more distributed setting, where multiple members of the clinical team work on multiple care plans in parallel.

Future work will take two directions. Firstly we propose to extend the argumentation model to support patients' and clinicians' *values* (see [1]), allowing finer-grained personalisation of the detailed decomposition of care plans. Secondly, the specification of *roles* provides a means of delegating responsibility for achieving a goal to another computer system or another clinician, the goal-based abstraction allowing the delegating system to ignore the details of what actions the delegate will take.

Acknowledgements. This work was supported by EPSRC grant EP/F057326/1 and by a programme grant from Cancer Research UK to J. Fox and D. Glasspool.

References

1. Atkinson, K., Bench-Capon, T.J.M.: Practical reasoning as presumptive argumentation using action based alternating transition systems. *Artificial Intelligence* 171(10-15), 855–874 (2007)
2. Fox, J., Glasspool, D.W., Grecu, D., Modgil, S., South, M.: Argumentation-based inference and decision-making—a medical perspective. *IEEE Intelligent Systems* 22, 34–41 (2007)
3. Fox, J., Johns, N., Rahmanzadeh, A.: Disseminating medical knowledge—the *PROforma* approach. *Artificial Intelligence in Medicine* 14, 157–181 (1998)
4. Grando, M.A., Glasspool, D.W., Fox, J.: Petri nets as a formalism for comparing expressiveness of workflow-based clinical guideline languages. In: 2nd Int. Workshop on Process-Oriented Information Systems in Healthcare (2008)
5. Peleg, M., Tu, S., Bury, J., Ciccarese, P., Fox, J., Greenes, R.A., Hall, R., Johnson, P.D., Jones, N., Kumar, A., Miksch, S., Quaglini, S., Seyfang, A., Shortliffe, E.H., Stefanelli, M.: Comparing computer-interpretable guideline models: a case-study approach. *J. American Medical Informatics Assoc.* 10(1), 52–68 (2003)
6. 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, 29–51 (1998)
7. Shiffman, R., Greenes, R.: Improving clinical guidelines with logic and decision-table techniques. *Medical Decision Making* 14(3), 247–343 (1994)
8. Sutton, D.R., Fox, J.: The syntax and semantics of *PROforma* guideline modelling language. *J. Am. Med. Inform. Assoc.* 10(5), 433–443 (2003)
9. Tu, S., Musen, M.A.: A flexible approach to guideline modeling. In: *Proc. AMIA Symposium*, pp. 420–424 (1999)
10. Waitman, L.R., Miller, R.A.: Pragmatics of implementing guidelines on the front lines. *J. American Medical Informatics Assoc.* 11(5), 436–438 (2004)