Using Constraint Logic Programming to Implement Iterative Actions and Numerical Measures During Mitigation of Concurrently Applied Clinical Practice Guidelines

Martin Michalowski[1], Szymon Wilk[2], Wojtek Michalowski[3], Di Lin[4], Ken Farion[5], and Subhra Mohapatra[3]

[1] Adventium Labs, Minneapolis MN, USA,martin.michalowski@adventiumlabs.com
[2] Poznan University of Technology, Poznan, Poland
[3] University of Ottawa, Ottawa ON, Canada
[4] McGill University, Montreal, PQ, Canada
[5] Children's Hospital of Eastern Ontario, Ottawa ON, Canada

Abstract. There is a pressing need in clinical practice to mitigate (identify and address) adverse interactions that occur when a comorbid patient is managed according to multiple concurrently applied disease-specific clinical practice guidelines (CPGs). In our previous work we described an automatic algorithm for mitigating pairs of CPGs. The algorithm constructs logical models of processed CPGs and employs constraint logic programming to solve them. However, the original algorithm was unable to handle two important issues frequently occurring in CPGs – iterative actions forming a cycle and numerical measurements. Dealing with these two issues in practice relies on a physician's knowledge and the manual analysis of CPGs. Yet for guidelines to be considered stand-alone and an easy to use clinical decision support tool this process needs to be automated. In this paper we take an additional step towards building such a tool by extending the original mitigation algorithm to handle cycles and numerical measurements present in CPGs.

Keywords: Clinical Decision Support Systems, Computerized Clinical Practice Guidelines, Constraint Logic Programming, Comorbidity

1 Introduction

Developing CPGs that explicitly address all potential comorbid diseases is not only difficult, but also impractical, and there is a need for formal methods that would allow combining several disease-specific CPGs in order to customize them to a patient [1]. Studies show that the lack of such methods is one of the obstacles in the adoption of CPGs in clinical practice and the development of these methods has been identified as one of the "grand challenges" for clinical decision support [2]. Our previous research [3] responded to the above challenge by proposing an automatic mitigation algorithm that verifies if pairs of CPGs can be applied simultaneously to a patient with comorbid diseases. We described the

novel use of a *constraint logic programming* (CLP) model to represent guidelines. We also explained how to codify domain knowledge associated with using these guidelines, and presented an algorithm that manipulates and solves the CLP model to propose a combined therapy for comorbid patient. The mitigation algorithm was originally developed under a number of assumptions, two of them being no iterative actions that may produce a cycle and the use of only binary action variables (i.e., variables associated with actions prescribed by a CPG). This paper revises the mitigation algorithm to relax these two assumptions.

2 Motivating Clinical Scenario

Consider an elderly male patient complaining of palpitations, shortness of breath and a syncopal episode. At the time of presentation he was not in any distress but had a rapid irregular pulse. A 12-lead ECG was done and revealed an irregular, wide complex tachycardia consistent with atrial fibrillation (AF) in the setting of Wolff-Parkinsons-White (WPW) syndrome. The attending physician concluded that the patient has a chronic condition (WPW) and an acute disease (AF).

When treating a patient for both AF and WPW there is an interplay between medications, specifically oral flecainide prescribed for WPW, and IV flecainide or amiodarone prescribed for AF. Overdosing the patient leads to a level of the drug in the blood that results in toxicity. Therefore dosages of flecainide need to be expressed in terms of exact, patient-specific (numerical) values that are revised during the management process. Additionally, the CPG for WPW contains a treatment cycle that adjusts flecainide dosages. Thus, handling both iterative actions manifesting as a cycle and the dosage of flecainide is essential to developing a combined therapy for this patient. While AF and WPW serve as the motivating clinical example, the described issues are common across many other pairs of diseases.

3 Extended Mitigation Algorithm

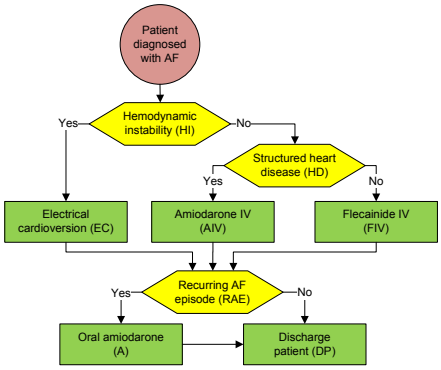

Fig. 1. AG_{AF}

Our mitigation algorithm (presented in detail in [3]) checks for possible adverse interactions between CPGs, addresses identified interactions and finally finds a combined therapy consisting of individual therapies derived from both CPGs. It accepts as input two CPGs given as actionable graphs and available patient information represented as a set of variable-value pairs. An actionable graph is a directed acyclic graph with context, action and decision nodes corresponding to context, decision and action

steps that appear in most formal CPG representations [4] and it can be obtained automatically from these representation [5]. An AG that represents CPG for AF (AG_{AF}) is given in Fig. 1.

The mitigation algorithm consists of three main phases: preparation, mitigating direct interactions, and mitigating indirect interactions. The extensions to handle iterative actions and numerical values, described below, affect the preparation phase of the algorithm where logical representations of actionable graphs are constructed.

Handling cycles requires a new step in the preparation phase, where a cyclic actionable graph is transformed into an acyclic one by expanding the cycle. The two key issues with cycles are: (1) they pose a problem when transforming an actionable graph into a logical model because a unique variable is needed for each action (and actions in cycles occur multiple times), and (2) often there is no clear definition of the number of iterations or a stopping condition. Based on consultations with medical experts we developed a two-step *expand* procedure (Figure 2) to remove a cycle by estimating the number of iterations (extracted directly from a CPG in the easy case or approximated using expert knowledge in the complex case) and expanding the identified cycle.

Require: $AG_i, Stop_i$
  $Cycle_i \leftarrow identify\_cycle(AG_i)$
  $MaxIter_i \leftarrow check\_conditions(Cycle_i, Stop_i)$
  $ForwardPath_i \leftarrow create\_path(Cycle_i)$
  $AG_i^{exp} \leftarrow replace\_cycle(AG_i, ForwardPath_i, MaxIter_i)$
  return AG_i^{exp}

Fig. 2. The *expand* procedure

The procedure begins by identifying the cycle in the actionable graph (AG_i) using a path-based strong component algorithm [6]. Note that the identification step assumes a single cycle only (handling multiple cycles could be done by using a recursive invocation of the *expand* procedure with AG_i). The procedure then uses the identified cycle and a stopping condition ($Stop_i$) to determine the maximum number of iterations ($MaxIter_i$). We establish the stopping condition according to expert's opinion, evidence, or patient information.

Next, we create a forward path ($ForwardPath_i$) to resolve the cycle. A forward path starts with the first node identified in the original cycle and includes the remaining cycle nodes. Finally, we create a revised actionable graph where the cycle is replaced by a set of connected forward paths, the number of these paths is equal to the maximum number of iterations. The updated actionable graph (AG_i^{exp}) is then used to create a logical model. We note that even in simple cases where the number of iterations is known, invoking the *expand* procedure is still necessary to introduce new variables needed to expand the cycle.

Numerical variables need special attention when creating logical models and they affect how action and decision variables are used. In the case of action variables, numerical variables add support for actions that are more complex than "go/no go" flags. For example, where previously a variable rep-

resenting the administration of flecainide was represented by a binary variable $F := true/false$, introducing a numerical variable $DF := [0 \ldots 500]$ enables considering administration of a medication and an associated dosage (medication is administered if its dosage is greater than 0). Numerical variables also allow for algebraic expressions that define conditions and they enable the computation of values for these variables. Whereas, previously considered logical models only allowed expressions such as $\neg(A \wedge F)$, after introducing this extension these models can include expressions such as $\neg(A \wedge DF \geq DF0) \wedge (DF = DF1 + DF2 + DF3 + \ldots)$.

4 Evaluation

To evaluate the extended mitigation algorithm, we use an instance of the clinical scenario described in Section 2. We apply interaction operators to represent external domain knowledge that describes indirect interactions between the maximum safe oral dosage of flecainide ($DF_{safe}(P) = 200mg$, coming from the guideline for WPW, where P indicates the level of plasma in blood) and oral dosage of amiodarone or IV flecainide. A revision operator is included that reduces the dosage of flecainide from the maximum safe level to 75% of this safety threshold (a revision developed after consulting with medical experts). We remove iterative actions from AG_{WPW} by invoking the *expand* procedure, resulting in AG^{exp}_{WPW} given in Fig. 3 (please note only AG_{AF} is given in Fig. 1 due to space limitations).

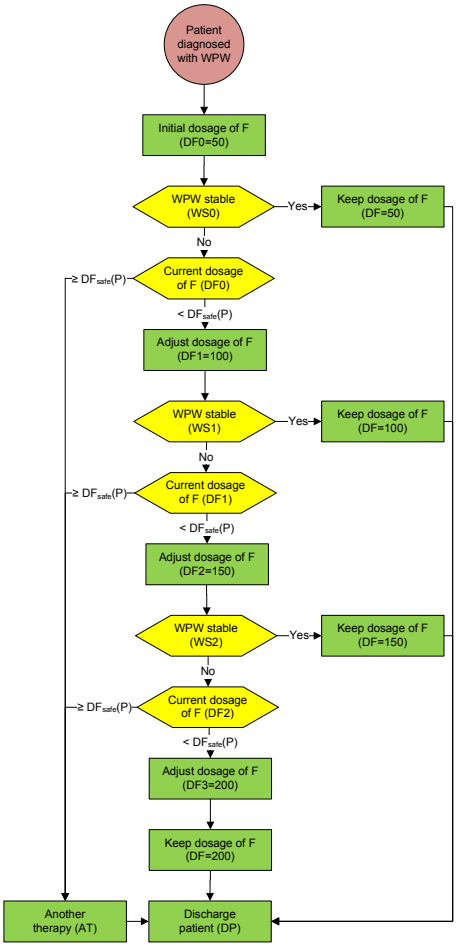

Fig. 3. AG^{exp}_{WPW}

Consider a patient, who has not been stabilized within the first three rounds of flecainide therapy (variables $WS0 = WS1 = WS2 = n$), has hemodynamic instability ($HI{=}y$), and who has a reoccurrence of AF ($RAE = y$). Given the logical model and an initial assignment of values to variables as defined by the above patient information, the extended mitigation algorithm invokes a CLP solver (we use the ECL^iPS^e system [7]) . No direct adverse interactions are detected so the model is augmented with the interaction operators

described above and the solver is again invoked. This time, an adverse indirect interaction is detected, corresponding to interplay between the dosage of flecainide and oral amiodarone. The extended mitigation algorithm proceeds to resolve this interaction by invoking the revision operator that lowers the dosage of flecainide from $DF_{safe}(P)$ by 25% (given P = 0.7, it translates to lowering from 200mg to 150mg). The revised model is passed to ECL^iPS^e and the solver produces the following solution: [*WS0 = n, WS1 = n, WS2 = n, DF0 = 50, DF1 = 100, DF2 = 150, DF3 = 200, DF= 150, HI = y, EC = true, RAE = y, PD = true*] that represents a combined therapy. In layman's terms this solution indicates that the dosage of flecainide is set to 150mg ($DF = 150$) as indicated by the revision operator, the patient undergoes electrical cardioversion ($EC = true$) and is discharged ($PD = true$). For instances with no found solution, we return the possible source of infeasibility to the physician who can use this information to determine the correct next steps given the patient information.

5 Discussion

In this paper we presented extensions to our mitigation algorithm. These extensions allow for numerical variables and add support for algebraic expressions and conditions. We introduced the *expand* procedure to identify and expand cycles, thus enabling the mitigation algorithm to operate on guidelines with complex relationships. This is an important step towards our goal of creating a comprehensive alerting system for physicians that will support the concurrent application of multiple CPGs. The extended mitigation algorithm is currently being incorporated into a mobile clinical decision support system to create a proof-of-concept application for evaluation in a clinical scenario. Further, we are working on a formal theory for representing and mitigating CPGs that will enable us to operationalize the application across a broad range of clinical scenarios.

References

[1] Fox, J., Glasspool, D., Patkar, V., Austin, M., Black, L., South, M., Robertson, D., Vincent, C.: Delivering clinical decision support services: there is nothing as practical as a good theory. J. Biomed. Inform. 43, 831–43 (2010)

[2] Sittig, D., Wright, A., Osheroff, J., Middleton, B., Teich, J., Ash, J., Campbell, E., Bates, D.: Grand challenges in clinical decision support. J. Biomed. Inform. 41, 387–92 (2008)

[3] Wilk, S., Michalowski, W., Michalowski, M., Farion, K., Hing, M., Mohapatra, S.: Mitigation of adverse interactions in pairs of clinical practice guidelines using constraint logic programming. Journal of Biomedical Informatics (201

[4] Isern, D., Moreno, A.: Computer-based execution of clinical guidelines: a review. Int. J. Med. Inform. 77, 787–808 (2008)

[5] Hing, M., Michalowski, M., Wilk, S., Michalowski, W., Farion, K.: Identifying inconsistencies in multiple clinical practice guidelines for a patient with co-morbidity. In: Proceedings of KEDDH-10. pp. 447–452 (2010)

[6] Tarjan, R.: Depth-first search and linear graph algorithms. SIAM J. Comput. 1, 146–160 (1972) 3)

[7] The ECLiPSe Constraint Programming System. Available from: http://www.eclipseclp.org