Mitigation of Adverse Interactions in Pairs of Clinical Practice Guidelines Using Constraint Logic Programming

Szymon Wilk[a*], Wojtek Michalowski[b], Martin Michalowski[c], Ken Farion[d], Marisela Mainegra Hing[b], Subhra Mohapatra[e]

[a] *Institute of Computing Science, Poznan University of Technology*
Piotrowo 2, 60-965 Poznan, Poland

[b] *Telfer School of Management, University of Ottawa*
55 Laurier Ave East, Ottawa, ON K1N 6N5, Canada

[c] *Adventium Labs, 111 Third Ave South, Suite 100*
Minneapolis, MN 55401 USA

[d] *Division of Emergency Medicine, Children's Hospital of Eastern Ontario*
401 Smyth Rd., Ottawa, ON, K1H 8L1 Canada

[e] *Faculty of Medicine, University of Ottawa*
451 Smyth Rd., Ottawa ON, K1H 8M5

Abstract

We propose a new method to mitigate (identify and address) adverse interactions (drug-drug or drug-disease) that occur when a patient with comorbid diseases is managed according to two concurrently applied clinical practice guidelines (CPGs). A lack of methods to facilitate the concurrent application of CPGs severely limits their use in clinical practice and the development of such methods is one of the grand challenges for clinical decision support. The proposed method responds to this challenge.

We introduce and formally define logical models of CPGs and other related concepts, and develop the mitigation algorithm that operates on these concepts. In the algorithm we combine domain knowledge encoded as interaction and revision operators using the constraint logic programming (CLP) paradigm. The operators characterize adverse interactions and describe revisions to logical models required to address them, while CLP allows us to efficiently solve these models – a solution represents a feasible therapy that may be safely applied to a patient.

The mitigation algorithm accepts two CPGs and available (likely incomplete) patient information. It reports whether mitigation has been successful or not, and on success it gives a feasible therapy and points at identified interactions (if any) together with the revisions that address them. Thus, we consider the mitigation algorithm as an alerting tool to support a physician in the concurrent application of CPGs that can be implemented as a component of a clinical decision support system. We illustrate our method in the context of two clinical scenarios involving a patient with duodenal ulcer who experiences an episode of transient ischemic attack.

Keywords: clinical practice guideline; comorbid diseases; constraint logic programming; domain knowledge; clinical decision support.

[*] Corresponding author; e-mail: szymon.wilk@cs.put.poznan.pl, tel.: +48 61 665 29 30, fax: +48 61 877 15 25

1. INTRODUCTION

A *clinical practice guideline* (CPG) [1] is a knowledge-based tool for disease-specific patient management. It is created by a panel of experts and is supported by medical evidence coming from specialized repositories [2] (such as The Cochrane Library [3]). A CPG encapsulates best practices in identifying relevant patient data, drawing conclusions from this data with regards to possible diagnoses, and prescribing the most appropriate treatment. It is generally agreed that the use of CPGs at the point of care has a positive impact on a patient's outcomes and providing guidelines to physicians in a computer executable format advances their adherence to standards of practice and improves quality of care [4]. While the ability to enact a CPG for available patient data plays an important role in guideline uptake by the medical profession, it does not address the issue of using guidelines on patients with comorbid diseases. Such a need is especially clear for elderly patients – several studies [2, 5, 6] have shown that about half of people 65 years or older have two or more comorbid conditions. The concurrent application of disease-specific CPGs for such patients often has undesired consequences due to the inconsistencies in the guidelines that lead to adverse drug-drug or drug-disease interactions [7].

Providing methods and tools that would enable the application of multiple CPGs to a patient with comorbid diseases has been listed as one of the "grand challenges" for clinical decision support [8]. The authors maintain that CPGs have been underutilized in practice because they have not accounted for co-morbidity issues. At the same time the authors call for new "combinatorial, logical, or semantic approaches" that would allow for merging several CPGs and cross-checking their recommendations. The call for formal theories for clinical decision support allowing for advanced analysis of CPGs (e.g., checking for inconsistencies and adverse interactions) has been repeated in [9].

The method proposed in this paper responds to these calls. It builds on our earlier research [10, 11] that applied constraint logic programming (CLP) [12] to identify adverse interactions resulting from using multiple guidelines on the same patient. However, current research significantly expands our past work by:

- Introducing logical models of analyzed CPGs that act as an intermediate layer between common CPG representations and CLP models,
- Introducing interaction and revision operators that codify domain knowledge indicating possible adverse interactions between logical models and describing necessary revisions of the models,
- Proposing a new algorithm (further referred to as the *mitigation algorithm*) that operates on the logical models and uses operators in order to identify and address adverse interactions that are demonstrated by the concept of a potential source of infeasibility.

From a clinician's perspective we see our method (the mitigation algorithm in particular) as an "alerting tool" to support the physician in the concurrent application of multiple CPGs by warning about possible adverse interactions that may occur and by suggesting a clinically appropriate resolution of these interactions.

In order to ground our approach, we make the following simplifying assumptions:

- The number of mitigated CPGs is limited to two due to computational and logical complexities associated with the identification and resolution of adverse interactions.
- A CPG does not include parallel actions that can be followed concurrently (however, they can be replaced by a single sequence of actions).
- Temporal aspects of CPG modeling and execution are not taken into account (we only consider situations where guidelines are applied during a single patient-physician encounter).
- There is full semantic interoperability between the CPGs.

- A CPG is revised so it does not contain sub-guidelines (if sub-guidelines are present, they are expanded and included in the main guideline).

In Section 5 we describe how some of the above assumptions might be relaxed. Additionally, we assume individual CPGs are logically coherent, thus it is always possible to derive a feasible therapy if a single guideline is applied to a patient. This assumption is consistent with the intended purpose of a CPG in practice (i.e., it is designed to always propose a therapy for a given patient).

The paper is organized as follows. We start with related work. This is followed by a description of methodologies, including a brief presentation of the CLP paradigm, and a detailed description of our method. Then we illustrate our proposal using a simplified clinical case study. We conclude with a discussion and present areas for future research.

2. RELATED WORK

Related research on advanced analysis and processing CPGs falls under the following categories: (1) verifying a CPG [13-15], (2) using a CPG to critique actual actions [16, 17], and (3) combing multiple CPGs for a patient with comorbid diseases [18-22]. The common theme shared by all categories is expressing processed CPGs using one of the computer-interpretable representations (e.g., GLIF3, SAGE, Asbru or PROforma, see [23-25] for a review) or ontological modeling.

The goal of CPG verification is early identification (before a guideline is applied to a patient) of issues such as semantic errors and inconsistencies in the definition of a guideline. Three types of methods are employed for this purpose: theorem proving [14], model checking [13] and knowledge-based checking [15]. Theorem proving implies representing a CPG as a theory that can be processed by an automatic theorem prover, and a failed proof indicates problems with the CPG. The use of theorem proving is described for example in [14], where the authors proposed a translation of Asbru constructs into a format acceptable by the KIV prover and apply it in order to verify two CPGs. Model checking methods assume checking a formal model of the CPG (usually obtained from its computer-interpretable representation) against a set of specifications describing expected properties of the guideline model. Specifications that cannot be verified indicate problems in the CPG. For example, in [13] the authors proposed a framework where a CPG represented as an UML state chart is being checked against a set of common types of the requirements in the form of temporal logical statements. Finally, knowledge-based checking methods employ methodologies developed for rule-based systems, where a CPG is represented as a set of rules that are being verified for possible anomalies (e.g., semantic errors). Definitions of anomalies can be either generic or domain-specific and are derived using expert knowledge. Such a knowledge-based checking method is described in [15], where the authors identify anomalies as unsatisfiable conditions, unreachable sequence of states or ambiguous state transitions.

Research on critiquing looks at comparing actual actions performed by a clinician to “ideal” actions suggested by a CPG in order to identify various types of non-compliance (e.g., conflicting actions). While conceptually different from CPG verification (critiquing is used when a CPG is being applied to a patient), it often uses similar methods, mostly model checking. In [16] the authors proposed an approach where a CPG represented as a state transition model is being checked against a set of temporal logic formulas describing actual actions. The proposed approach is able to identify two classes of non-compliance – related to actions (e.g., conflicting or missing mandatory actions) and related to clinical findings or patient data (e.g., missing relevant findings or wrong finding for a given action). A different approach to critiquing is presented in [17], where a CPG is translated into a set of “if conditions, then criticize” rules that are further used to verify treatments prescribed by physicians and to provide additional explanation why a particular treatment should not be recommended.

It is worth mentioning that there are approaches combining CPG verification with compliance assessment for more comprehensive decision support. For example, the approach described in [26] uses an ontological model of domain knowledge together with abductive reasoning to evaluate whether a CPG is complete and appropriate (in terms of suggested actions and their justification), and to assess the compliance of a physician's actions with a CPG. Moreover, it provides support when executing a CPG by summarizing and explaining observed patient's clinical states and by warning about the violation of so-called safety rules. These rules are part of the ontological model and characterize adverse drug-drug or disease-drug interactions; they may also explain how specific interactions can be prevented.

Despite its clinical importance, research on combining multiple CPGs is in its early stages – as stated in [18], combining multiple CPGs "is not a trivial task". Proposed solutions vary from manual interventions, where human experts combine CPGs using a specialized editing tool [19], through semi-automatic approaches, where experts resolve automatically discovered conflicts [18], to fully automatic approaches that employ codified expert knowledge and require no direct human involvement when an algorithm is being executed [20-22]. In [22] the authors describe an approach for combining multiple CPGs using the SDA* representation [27]. Their method translates each of the combined CPGs into a set of state-action pairs, expands them into state-action-prognosis triples using prognosis rules, and finally merges these triples into a single CPG. Merging is driven by restriction and substitution rules. The former indicate states that cannot be combined together, while the latter indicate drug interactions that may arise in combined actions and prescribe how to address them. Prognosis, restriction and substation rules need to be defined by experts or they can be discovered from data using data mining techniques [21]. Another approach to combining multiple CPGs that is based on ontological models is described in [20]. The models represent individual CPGs and they are merged along common actions, decisions, concepts and locations using mapping and alignment constructs associated with ontology.

The rationale behind the research described in this paper is to respond to the postulates presented in [8] and [9] by proposing a method that verifies if multiple CPGs can be applied concurrently to a patient with comorbid diseases, and introduces necessary modifications when such application is not feasible. The proposed method differs in several aspects from what has been presented so far. It allows for rich definitions of interactions that involve states and actions thus they go beyond drug-drug interactions. It requires no additional expert knowledge beyond the definition of interaction and revision operators (in other words there is no need for prognosis rules that may be difficult to define). Finally, the analysis and processing is driven by available patient data, thus it allows for personalization and customization of the resulting CPGs. Our method also shares some similarities with the approaches reviewed above. Similarly to CPG verification, it employs models of the CPGs (logical models in particular) for analysis and processing. Similarly to critiquing techniques, it uses available patient data as input information. In terms of checking for adverse interactions between the CPGs, it uses interaction operators that are conceptually similar to the safety rules proposed in [26] and restriction rules proposed in [22]. Moreover, to address adverse interactions our method uses revision operators that play similar role to substitution rules proposed in [22].

From a general perspective, our method can be placed in the broad category of meta-level control and reasoning methods in medicine [28, 29], and our approach to dealing with the adverse interactions in multiple CPGs shares similarities with conflict resolution methods in automated reasoning [30].

3. METHODOLOGY

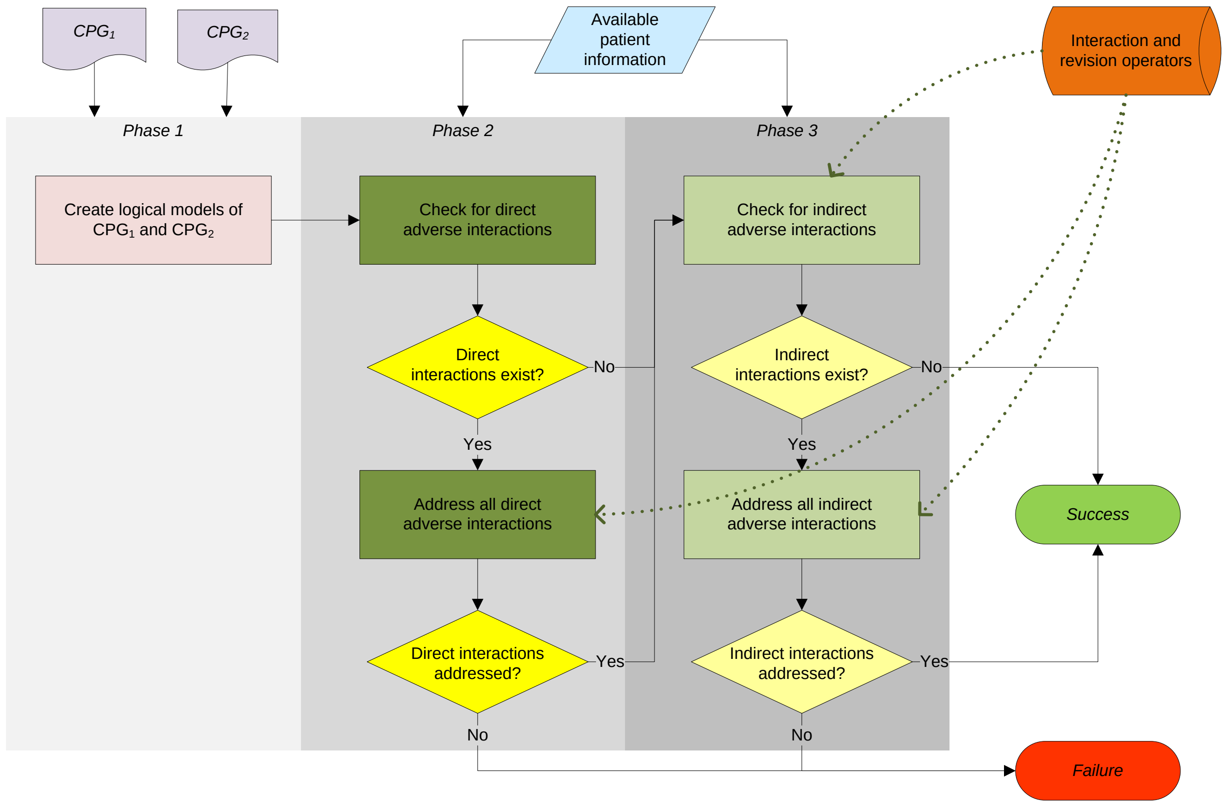
We consider a situation when two CPGs are applied to a patient with comorbid diseases in order to obtain a combined therapy. A typical situation would involve a chronic disease and an acute disease, although other cases (e.g., two acute conditions) are also admissible. A combined therapy consists of two individual therapies derived from disease-specific CPGs. Given a patient's state (characterized by available patient information) a feasible combined therapy does not exist if there are direct adverse interactions between individual actions that manifest as contradictory recommendations (e.g., "take aspirin" in one CPG and "don't take aspirin" in the other). Even, if there are no direct interactions, individual therapies need to be further checked for indirect adverse interactions that are caused by drug-drug or drug-disease interactions and in most cases are not explicitly represented in the guidelines.

If none of the adverse interaction (direct or indirect) has been encountered, then a feasible combined therapy can be formulated. On the other hand, if the combined therapy cannot be formulated, disease-specific CPGs and associated therapies need to be revised using clinical knowledge that goes beyond what is encoded in the guidelines and comes from domain experts, textbooks, or repositories of clinical evidence.

We call the process of checking for adverse interactions (direct or indirect) and addressing *mitigation*. In practice a physician mitigates guidelines and during this process she relies on clinical acumen (and evidence) when managing a comorbid patient. Our intention is to automate this mitigation process by:

1. Representing CPGs as "computable" logical models (discussed in Section 3.2.2),
2. Representing clinical acumen using two types of operators – *interaction operators* (discussed in Section 3.3.1) that characterize indirect adverse interactions, and *revision operators* (discussed in Section 3.3.2) that describe possible revisions to the logical models of CPGs,
3. Introducing a dedicated *mitigation algorithm* (discussed in Section 3.4) that processes logical models of considered CPGs, checks for possible adverse interactions and addresses them using operators.

Figure 1. Overview of the mitigation algorithm


CPG₁
CPG₂
Available patient information
Interaction and revision operators
Phase 1
Phase 2
Phase 3
Create logical models of CPG₁ and CPG₂
Check for direct adverse interactions
Check for indirect adverse interactions
Direct interactions exist?
Indirect interactions exist?
No
Yes
Address all direct adverse interactions
Address all indirect adverse interactions
Success
Direct interactions addressed?
Indirect interactions addressed?
Yes
No
Failure

The mitigation algorithm forms the core of our method as it brings together logical models and operators and applies the CLP paradigm (outlined in Section 3.1) to solve the models given available patient data. The existence of a solution implies the ability to formulate a feasible combined therapy, while the lack of a solution indicates the presence of adverse interactions. The source of these interactions, further called *potential source of infeasibility* (see Section 3.2.5), is identified with the help of interaction operators and addressed with the help of revision operators. The high level overview of the mitigation algorithm is given in Figure 1. The algorithm is composed of three phases: (1) creating logical models of CPGs, (2) mitigating direct adverse interactions and (3) mitigating indirect adverse interactions. The algorithm may terminate after the second phase if it has failed to mitigate direct interactions.

3.1. Constraint logic programming

CLP unifies logic programming (LP) and a constraint satisfaction problem (CSP) by using LP as a constraint programming language to solve a CSP [12]. A logic program is seen as logical theory composed of a set of rules called clauses. CLP extends it by including constraints in the body of clauses, and querying the program about the provability of a goal produces a solution to the CLP. The proof for the goal is composed of clauses whose bodies are satisfiable constraints.

A CLP model follows the CSP nomenclature. As such it is made up of a set of variables, a set of clauses with constraints and a goal to be satisfied, that define the logic program. The clauses in the model capture the relationships between value assignments for variables and they restrict the possible combinations of values assigned to variables. Solving a CLP model entails satisfying the goal given the set of constraints, where a value is assigned to each variable such that no constraints are violated (bodies of all clauses are satisfied). Variables related to available information are instantiated prior to solving the model and cannot be revised by the solving procedure.

3.2. Basic concepts

In order to illustrate selected formal concepts introduced in this section we use a clinical case involving the concurrent application of CPGs for deep vein thrombosis (DVT) and uncontrolled hypertension (UHTN). While this example has been simplified to better illustrate relevant properties of the introduced concepts, it is medically valid and it was prepared with help of a clinical expert.

3.2.1. Actionable graph

Our method focuses on the efficient processing of CPGs and to make it independent of any specific CPG representation, we have introduced the concept of an *actionable graph* as the way to express the guidelines. This concept has been inspired by the SDA* formalism [27] – one of the most recent CPG representations that builds on the well-established foundations (e.g., Asbru and PROforma) and addresses many of their shortcomings (e.g., complexity that makes them difficult to understand and use by healthcare professionals). An actionable graph is a simplified version of the SDA* representation and it can be easily obtained from any CPG representation that distinguishes between context, decision and action steps (according to [25] these steps are common for most CPG representations, including Asbru, PROforma or GLIF3).

An actionable graph (AG_i) for a specific CPG (denoted as CPG_i) is defined as a directed graph:

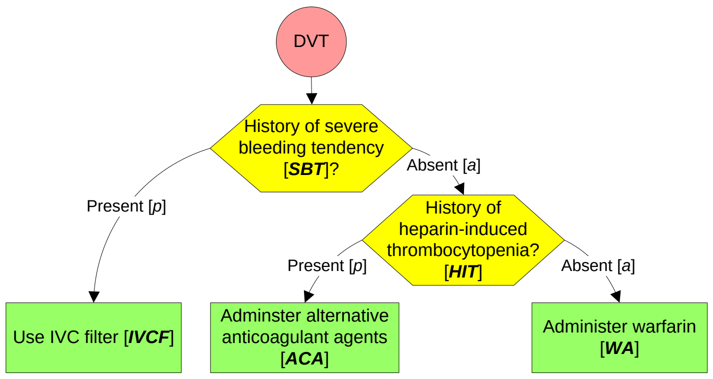
$$AG_i = \langle N_i, A_i \rangle,$$

where N_i is a set of context, action and decision nodes, and A_i is a set of arcs representing transitions between nodes. A context node provides a clinical context at a specific point in CPG_i. AG_i has a context node as its root (starting node) that defines an entry point and labels the disease managed by CPG_i. An action node corresponds to an action step from CPG_i and indicates a task that needs to be conducted. A decision node corresponds to a decision step and indicates selection of one from several alternative choices that are associated with the arcs emanating from a decision node.

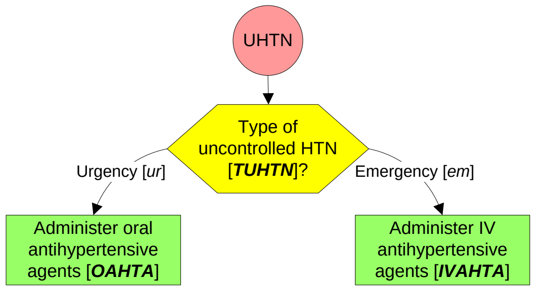
Examples of actionable graphs representing simplified CPGs for DVT and UHTN are given in Figure 2. In this figure the context node is indicated with an oval, decision nodes are indicated with diamonds, and action nodes with rectangles. Moreover, the figure labels variables associated with specific nodes and values corresponding to alternative choices – they are given in square brackets after node and choice descriptions. For example, the *SBT* variable is associated with the “history of severe bleeding tendency” decision node. There are two alternative choices at this decision node – present and absent. They are represented as *p* and *a* values respectively (these values define the domain of the *SBT* variable).

Figure 2. Sample actionable graphs

a) for DVT (AG_{DVT})


DVT
History of severe bleeding tendency [SBT]?
Present [p]
Absent [a]
History of heparin-induced thrombocytopenia? [HIT]
Present [p]
Absent [a]
Use IVC filter [IVCF]
Adminster alternative anticoagulant agents [ACA]
Administer warfarin [WA]

b) for UHTN (AG_{UHTN})


UHTN
Type of uncontrolled HTN [TUHTN]?
Urgency [ur]
Emergency [em]
Administer oral antihypertensive agents [OAHTA]
Administer IV antihypertensive agents [IVAHTA]

3.2.2. Logical model

A *logical model* (LM_i) provides a logical representation of an AG_i and is defined as:

$$LM_i = \langle d_i, V_i, PLE_i \rangle,$$

where d_i is the label of a disease retrieved from the context node of AG_i, V_i is a set of action and decision variables associated with actions and decision nodes in AG_i respectively, and PLE_i is a set of logical expressions representing paths in AG_i, each being a conjunction of variable-value pairs. Action variables are Boolean – *true* value assigned to an action variable means that an action should be carried out, while *false* indicates it should not. Decision variables have domains defined by possible alternative choices and assigning a specific value to a decision variable indicates making a specific choice.

Sample logical models created from actionable graphs in Figure 2 are given in Figure 3 (their creation is described in detail in Section 3.4). In all logical expressions in Figure 3 and in the following text we use a simplified notation for Boolean variables – i.e., $V = true$ is denoted simply as V, while $V = false$ as $\neg V$.

Figure 3. Sample logical models

a) for DVT (LM_{DVT})

LM_{DVT} = <DVT, {$SBT, HIT, IVCF, ACA, WA$},
{
    $(SBT = p) \wedge IVCF \wedge \neg ACA \wedge \neg WA$,
    $(SBT = a) \wedge (HIT = p) \wedge ACA \wedge \neg WA \wedge \neg IVCF$,
    $(SBT = a) \wedge (HIT = a) \wedge WA \wedge \neg ACA \wedge \neg IVCF$
}>

b) for UHTN (LM_{UHTN})

LMUHTN = <UHTN, {$TUHTN, OAHTA, IVAHTA$},
{
    $(TUHTN = ur) \wedge OAHTA \wedge \neg IVAHTA$,
    $(TUHTN = em) \wedge IVHTA \wedge \neg OAHTA$
}>

A *combined logical model* ($CLM_{i,j}$) brings together the pair of logical models defined above and information about possible adverse interactions between the underlying CPGs. It is defined as:

$$CLM_{i,j} = <LM_i, LM_j, ILE_{i,j}>,$$

where $ILE_{i,j}$ is a set of logical expressions that represent adverse interactions between CPG_i and CPG_j.

A combined logical model for DVT and HTN ($CLM_{DVT,UHTN}$) includes both logical models from Figure 3 and for the sake of brevity we do not repeat them here.

3.2.3. CLP-CPG model

A *CLP-CPG model* ($CCM_{i,j}$) constitutes a solvable representation of $CLM_{i,j}$. It is defined as:

$$CCM_{i,j} = <V_{i,j}, CL_{i,j}>,$$

where $V_{i,j}$ is a set of variables such that $V_{i,j} = V_i \cup V_j$ and $CL_{i,j}$ is a set of constraints established from PLE_i, PLE_j and $ILE_{i,j}$.

A CLP-CPG model for a single CPG_i is a specific instance of the $CCM_{i,j}$ where $V_{i,j} = V_i$ and $CL_{i,j}$ is created from PLE_i only.

3.2.4. Solution

A solution to $CCM_{i,j}$ (labeled as $SOL_{i,j}$) is defined as an assignment of values to all variables from $V_{i,j}$ such that all constraints from $CL_{i,j}$ are satisfied.

Note that $SOL_{i,j}$ provides truth evaluations of the logical expressions from PLE_i and PLE_j. Considering that actionable graphs do not allow for parallel paths to be traversed simultaneously, there is only one expression from PLE_i and one from PLE_j that is evaluated as *true* (each of these two expressions corresponds to a disease-specific therapy). Therefore, a solution of $CCM_{i,j}$ that is presented to a physician includes variable-value pairs that appear in these two expressions (as opposed to value assignments to all variables) – more specifically it includes variable-value pairs for all decision variables in these expressions and action variables assigned *true*. Thus, the filtered solution represents a combined therapy and it points out actions that have been done or should be done for a patient and indicates decisions (actual or inferred) under which these actions are valid.

3.2.5. Potential source of infeasibility

The lack of a solution to $CCM_{i,j}$ indicates the inability to find a set of variable-value pairs satisfying all constraints. Violated constraints provide valuable information about the source of infeasibility in the model and we use this information in defining a *potential source of infeasibility* ($PSI_{i,j}$). Specifically, $PSI_{i,j}$ is defined as a set of variables from $V_{i,j}$ that appear in violated constraints from the $CL_{i,j}$.

We assert our assumption that an individual CPG is logically consistent (there is always a sequence of action and decision steps that corresponds to a path in a related actionable graph), and thus the associated CLP-CPG model must always have a solution. Consequently, the lack of a solution can only occur when a CLP-CPG model is derived from two CPGs.

3.3. Operators

Operators are applied to $CLM_{i,j}$ and each is composed of activation and knowledge components. The activation component defines when an operator is applicable (i.e., when it can be activated), and the knowledge component establishes revisions to be introduced to $CLM_{i,j}$.

In order to illustrate both types of operators we use the same clinical case as in Section 3.2.

3.3.1. Interaction operator

An *interaction operator* (IO^k) is defined as a 3-tuple:

$$IO^k = <D^k, V^k, le^k>,$$

where D^k is a set of disease labels (it can include a wildcard value '*' that indicates any disease label), V^k is as set of variables and le^k is a logical expression. D^k and V^k form the activation component, and le^k forms the knowledge component codifying a single adverse interaction that is added to $ILE_{i,j}$.

Considering the illustrative clinical case, adverse interactions may occur when a patient with hypertensive urgency (defined as elevated blood pressure above 180/110) is treated with warfarin because of the danger of hypertensive crisis. Such indirect adverse interaction is codified by the interaction operator given in Figure 4. This operator contains the wildcard in its first component, thus it is activated for any combined logical model that includes variables *TUHTN* (type of uncontrolled hypertension) and *WA* (warfarin). Then, the expression given as the third component and describing the adverse interaction between hypertensive urgency ($TUHTN = ur$) and provision of warfarin (WA) is added to the $ILE_{i,j}$ component of the combined logical model.

Figure 4. A sample interaction operator

$$IO^1 = <\{*\}, \{TUHTN, WA\}, TUHTN = ur \wedge WA\}$$

3.3.2. Revision operator

A *revision operator* (RO^k) is defined as 4-tuple:

$$RO^k = <D^k, V^k, sle^k, tle^k>$$

where D^k is a set of disease labels (as previously, it can also include the '*' wildcard), V^k is a set of variables and sle^k and tle^k are logical expressions. Similarly to IO^k, D^k and V^k form the activation component, and sle^k and tle^k form the knowledge component. The knowledge component of RO^k states that whenever sle^k appears in logical expressions from ILE_i or ILE_j, it is replaced by tle^k.

A sample revision operator that addresses the interaction described by the interaction operator from Figure 4 is given in Figure 5. This operator is activated whenever the identified possible source of infeasibility contains variables *TUHTN* and *WA* (i.e., has been caused by the interaction presented in Section 3.3.1). It revises the combined logical model by identifying all the logical expressions describing paths in actionable graphs that imply treating a patient only with warfarin and discouraging the use of IVC (inferior vena cava) filter ($\neg IVCF$), and replacing them with an expression that discards warfarin ($\neg WA$) and replaces it with the IVC filter ($IVCF$).

Figure 5. A sample revision operator

$$RO^1 = <\{*\}, \{TUHTN, WA\}, WA \land \neg IVCF, \neg WA \land IVCF \}$$

3.4. Mitigation algorithm

The mitigation algorithm is composed of a number of procedures that are described in further detail below. The main procedure, called *mitigate*, implements the flow of the algorithm (see Figure 1) and invokes the auxiliary procedures. The algorithm assumes that all interaction and revision operators have been defined and stored in an accessible and easy to search repository. The creation of operators is a research topic in itself and for the sake of simplicity we assume the repository with operators is prepared manually with the help of medical experts. This assumption is in line with recent attempts to create a centralized and shared repository with standardized drug-drug interactions [31].

The pseudo-code for the *mitigate* procedure is given in Figure 6. The procedure accepts as input actionable graphs (AG_i and AG_j) and available patient information (PI) given as a set of variable-value pairs. It reports *success* if both CPGs can be concurrently applied to the patient and *failure* otherwise. It also provides additional and more detailed results as output parameters. They include a possible solution of the CLP-CPG model ($SOL_{i,j}$), potential source of infeasibility ($PSI_{i,j}$) and the revision operator that addresses it (RO^k). Possible combinations of the returned values and output parameters together with their interpretations are given in Table 1.

Figure 6: Pseudo-code for the *mitigate* procedure

```
procedure mitigate(in $AG_i$, in $AG_j$, in $PI$,
               out $SOL_{i,j}$, out $PSI_{i,j}$, out $RO^k$)
begin
    create_LM($AG_i$, $LM_i$)
    create_LM($AG_j$, $LM_j$)
    $CLM_{i,j}$ := <$LM_i$, $LM_j$, {}>
    if $V_i \cap V_j \neq$ {} then begin
         solve_CCM($CLM_{i,j}$, $PI$, $PSI_{i,j}$, $SOL_{i,j}$)
         if $SOL_{i,j}$ = null then begin
              $PSI_{i,j}$ := $V_i \cap V_j$
              address_PSI($CLM_{i,j}$, $PI$, $PSI_{i,j}$, $SOL_{i,j}$, $RO^k$)
              if $SOL_{i,j}$ = null then
                   return failure
         end
    end
    apply_IOs($CLM_{i,j}$)
    solve_CCM($CLM_{i,j}$, $PI$, $PSI_{i,j}$, $SOL_{i,j}$)
    if $SOL_{i,j} \neq$ null then
         $SOL_{i,j}$ := filter $SOL_{i,j}$ according to $PLE_i$ and $PLE_j$
         return success
    else begin
         address_PSI($CLM_{i,j}$, $PI$, $PSI_{i,j}$, $SOL_{i,j}$, $RO^k$)
         if $SOL_{i,j}$ = null then
              return failure
         else
              $SOL_{i,j}$ := filter $SOL_{i,j}$ according to $PLE_i$ and $PLE_j$
              return success
    end
end
```

Table 1. Returned values and output parameters of the *mitigate* procedure

Returned value	$SOL_{i,j}$	$PSI_{i,j}$	RO^k	Interpretation
success	not null	null	null	Both CPGs (represented as AG_i and AG_j) can be applied concurrently; there is no infeasibility given the available patient information (*PI*) that needs to be addressed. A solution ($SOL_{i,j}$) is returned.
success	not null	not null	not null	Both CPGs (represented as AG_i and AG_j) can be applied concurrently; a potential source of infeasibility ($PSI_{i,j}$) has been encountered, but it was successfully mitigated with the revision operator (RO^k). A solution ($SOL_{i,j}$) together with mitigation information are returned.
failure	null	not null	null	Both CPGs (represented as AG_i and AG_j) cannot be applied concurrently. The encountered source of infeasibility ($PSI_{i,j}$) cannot be addressed by the available revision operators and no solution exists.

The *mitigate* procedure starts by constructing logical models (lines 1-3). Models that represent actionable graphs are constructed by the *create_LM* procedure (see Figure 7) that takes an actionable graph (AG_i) as an input and reports a logical model (LM_i) on output. The *create_LM* procedure is relatively simple, only the conversion of a path from AG_i to a logical expression (line 4) requires some explanation. A path including nodes n_p^1, n_p^2, …, n_p^k is converted into a conjunction of terms according to the following rules:

- If n_p^m is a context node, then it is discarded,
- If n_p^m is an action node, then ($v_p^m = true$) is added to the conjunction, where v_p^m is the action variable associated with the node n_p^m,
- If n_p^m is a decision node, then ($v_p^m = a_p^m$) is added to the conjunction, where v_p^m is the decision variable associated with the node n_p^m and a_p^m is an alternative choice linked to the arc emanating from n_p^m and traversed in the considered path.
- Finally, for each action node n_p^m which is not included in the path, ($v_p^m = false$) is added to the conjunction, where v_p^m is the action node variable. This ensures that actions not listed in the path are not realized.

The logical models created by *create_LM* are put together into the combined logical model ($CLM_{i,j}$). The $ILE_{i,j}$ component of the $CLM_{i,j}$ model is initially empty – indirect adverse interactions are considered only in the third phase of the algorithm.

Figure 7. Pseudo-code for the *create_LM* procedure

```
procedure create_LM(in $AG_i$, out $LM_i$)
begin
1   $d_i$ := extract disease label from the context node in $AG_i$
2   $V_i$ := set of variables associated with action and decision nodes in $AG_i$
3   $P_i$ := enumerate all paths in $AG_i$ using depth-first search
4   $PLE_i$ := convert all paths from $P_i$ to logical expressions
5   $LM_i$ := <$d_i$, $V_i$, $PLE_i$>
end
```

After creating the logical models, the *mitigate* procedure proceeds to the second phase of the algorithm (lines 4-12) where it identifies and addresses direct adverse interactions. First, the *solve_CCM* procedure (see Figure 8) is invoked to create a CLP-CPG model ($CCM_{i,j}$) from the $CLM_{i,j}$ model and to solve $CCM_{i,j}$ given available patient information (*PI*). On *success* it reports a solution ($SOL_{i,j}$), and on *failure* (no solution) it reports the potential source of infeasibility ($PSI_{i,j}$). Creation of constraints in the $CCM_{i,j}$ model (line 2) requires additional explanation. The first two constraints in the $CL_{i,j}$ component of $CCM_{i,j}$ are disjunctions of all logical expressions in PLE_i and PLE_j – they ensure one path in each underlying actionable graph will be traversed. Next, each expression from $ILE_{i,j}$ is negated and becomes a separate constraint to ensure that indirect interactions are avoided. In fact, these constraints are created only in the last phase of the algorithm (when indirect adverse interactions are being mitigated), as in earlier phases $ILE_{i,j}$ is empty. Then, the $CCM_{i,j}$ model is solved using a CLP solver.

Figure 8. Pseudo-code for the *solve_CCM* procedure

```
procedure solve_CCM(in $CLM_{i,j}$, in $PI$, out $PSI_{i,j}$, out $SOL_{i,j}$)
begin
1    $V_{i,j}$ := $V_i \cup V_j$
2    $CL_{i,j}$ := {constraint based on $PLE_i$, constraint based on $PLE_j$}
             $\cup$ { constraints based on $ILE_{i,j}$ }
3    $CCM_{i,j}$ := <$V_{i,j}$, $CL_{i,j}$>
4    $SOL_{i,j}$ := solve $CCM_{i,j}$ given $PI$ using a CLP solver
4    if $SOL_{i,j} \neq$ null then begin
5        $PSI_{i,j}$ := null
6        return success
7    end
8    else begin
9        $PSI_{i,j}$ := variables from violated constraints in $CCM_{i,j}$
10       return failure
11   end
end
```

If no solution has been found by the *solve_CCM* procedure, indicating the presence of direct adverse interactions, the *mitigate* procedure invokes *address_PSI* (see Figure 9) in order to address the reported potential source of infeasibility ($PSI_{i,j}$). The *address_PSI* procedure applies revision operators to the $CLM_{i,j}$ model to ensure solvability of the derived CLP-CPG model given available patient information (*PI*). It first invokes *activate_ROs* (see Figure 10) to activate applicable restriction operators and to obtain their ordered list (*ARO*). The operators are ordered according to the scope of changes (operators that introduce smaller number of revisions are preferred), and in the case of a tie, they are ordered according to the number of variables from $PSI_{i,j}$ covered by the operator (operators that cover more variables are preferred). Then, *address_PSI* iterates over *ARO* and in each iteration it attempts to revise the $CLM_{i,j}$ model using the current operator (RO^k) and to solve the derived CLP-CPG model (line 4).

Figure 9: Pseudo-code for the *address_PSI* procedure

```
procedure address_PSI(inout $CLM_{i,j}$, in $PI$, in $PSI_{i,j}$, out $SOL_{i,j}$, out $RO^k$)
begin
1    activate_ROs($CLM_{i,j}$, $PSI_{i,j}$, $ARO$)
2    for each $RO^k \in ARO$ do begin
3        $CLM'_{i,j}$ := $CLM_{i,j}$
4        if revise_CLM($CLM'_{i,j}$, $RO^k$) $\wedge$ solve_CCM($CLM'_{i,j}$, $PI$, $PSI'_{i,j}$, $SOL_{i,j}$) then begin
5            $CLM_{A,B}$ := $CLM'_{A,B}$
6            return success
7    end
8    $RO^k$ := null
9    return failure
end
```

Figure 10: Pseuo-code for the *activate_ROs* procedure

```
procedure activate_ROs(in $CLM_{i,j}$, in $PSI_{i,j}$, out ARO)
begin
1    $ARO$ = {}
2    for each revision operator $RO^k \in$ repository do begin
3        if ($D^k$ = {*} $\vee$ $D^k \cap \{d_i, d_j\} \neq \{\}$) $\wedge$ $V^k \subseteq PSI_{i,j}$ then
4            $ARO := ARO \cup \{ RO^k \}$
5    end
6    order ARO according to the scope of changes (ascending) and to $|V^k \cap PSI_{i,j}|$ (descending)
end
```

Revision are introduced by the *revise_CLM* and *revise_LM* procedures (see Figure 11) that apply RO^k to logical models and report *success* or *failure* depending whether any revisions have been made or not. Revised $CLM_{i,j}$ may include new variables and if *revise_CLM* is called during the last phase of the algorithm, it invokes the *apply_IOs* procedure (see Figure 12) to update the $ILE_{i,j}$ component of the $CLM_{i,j}$ model with logical expressions describing all possible indirect interactions.

Figure 11: Pseudo-code for the *revise_CLM* nad *revise_LM* procedures

```
procedure revise_CLM(inout $CLM_{i,j}$, in $RO^k$)
begin
1    if revise_LM($LM_i$, $RO^k$) $\vee$ revise_LM($LM_j$, $RO^k$) then begin
2        if mitigating indirect adverse interactions
             and new variables have been added to $V_i$ or $V_j$ then
3            apply_IOs($CLM_{i,j}$)
4        return success
5    end
6    else
7        return failure
end

procedure revise_LM(inout $LM_i$, in $RO^k$)
begin
1    replace $sle^k$ by $tle^k$ in all logical expressions in $PLE_i$
2    if no replacements have been made then
3        return failure
4    if $tle^k$ contains variables not in $V_i$ then
5        add new variables to $V_i$
6    return success
end
```

Figure 12: Pseudo-code for the *apply_IOs* procedure

```
procedure apply_IOs(inout $CLM_{i,j}$)
begin
1    $ILE_{i,j}$ := {}
2    for each interaction operator $IO^k \in$ repository do begin
3        if ($D^k$ = {*}) $\vee$ $D^k \cap \{d_i, d_j\} \neq \{\}$) $\wedge$ $V^k \subseteq V_i \cup V_j$ then
4            $ILE_{i,j} := ILE_{i,j} \cup \{le^k\}$
5    end
end
```

The *address_PSI* procedure solves the model revised by *revise_CLM* ($CLM'_{i,j}$). If the revised model still does not have a solution, the procedure applies the next revision operator from the *ARO* list until a solution is found or until all activated revision operators have been considered. In the latter case the *mitigate* procedure terminates with *failure* (lines 9-10) to signal that direct interactions could not be addressed.

We would like to point at two elements that improve the operation of the part of the *mitigate* procedure that corresponds to the second phase of the algorithm (i.e., mitigating direct adverse interactions). First, the procedure checks for direct interactions only if there are variables shared between logical models (line 4). Second, the potential source of infeasibility is narrowed down to the shared variables (line 7), thus subsequent operations require a smaller number of attempts. It is possible to focus only on shared variables because of our assumption that individual guidelines (and their corresponding logical models) are consistent and always have a solution.

The last part of the *mitigate* procedure (lines 13-25) corresponds to the third phase of the algorithm – identifying and addressing indirect adverse interactions. First, *apply_IOs* is invoked (line 13) to expand the the $ILE_{i,j}$ component of the $CLM_{i,j}$ model with logical expressions characterizing possible indirect interactions. The remaining steps are similar to what takes place in the second phase of the algorithm. The expanded $CLM_{i,j}$ model is solved using *solve_CCM* (line 14). In this phase of the algorithm the $ILE_{i,j}$ component is not empty, thus the $CCM_{i,j}$ model now includes additional constraints. The existence of a solution indicates the lack of indirect interactions, thus the *mitigate* procedure filters the solution (see Section 3.2.4) and terminates with *success* (lines 15-17). Otherwise the procedure again invokes *address_PSI* to address the potential source of infeasibility (line 19). Depending on the result (solvability of the revised combined logical model) the *mitigate* procedure reports either *failure* (line 21) or *success* (line 24).

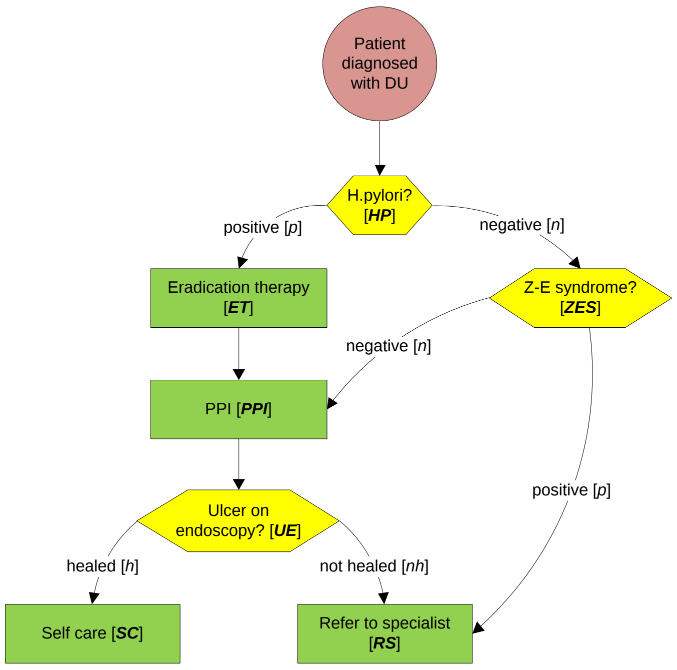
4. CASE STUDY: CONCURRENT MANAGEMENT OF DUODENAL ULCER AND TRANSIENT ISCHEMIC ATTACK

In this section we illustrate the use of our method in two clinical scenarios that involve a chronic disease and an acute condition. In both scenarios a patient that is treated for a duodenal ulcer (DU) experiences an episode of transient ischemic attack (TIA). CPGs used in this example are based on the guidelines published by the National Institute for Health and Clinical Excellence, UK (NICE) [32] and they have been simplified to include only the relevant action and decision nodes.

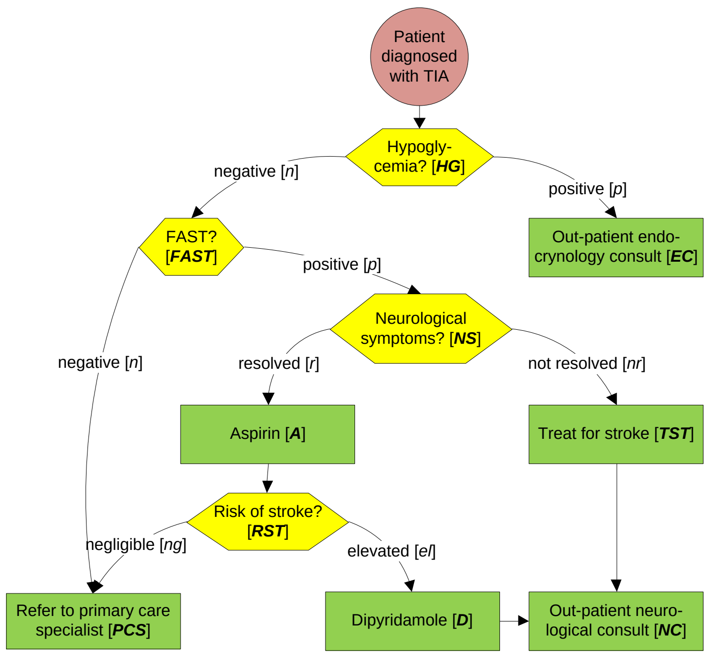
4.1. Actionable graphs

Figure 13. Actionable graphs representing the case study CPGs

a) for DU (AG_{DU})



b) for TIA (AG_{TIA})



Transforming a CPG from a formal representation into an actionable graph is relatively straightforward – we proposed an algorithm in [33]. Figure 13 presents actionable graphs constructed from CPGs for DU and TIA guidelines respectively. It employs the same notation as used in Figure 2, i.e., ovals for context nodes, diamonds for decision nodes and rectangles for action nodes. As previously, the figure labels variables associated with specific nodes and values corresponding to alternative choices.

4.2. Operators

Operators developed for this case study are given in Figure 14. As indicated earlier the operators have been created with help from the medical expert on our team. The interaction operator (IO^1) represents a drug-disease interaction (increased risk of bleeding) that occurs when a DU patient is prescribed aspirin (A) without a proton-pump inhibitor (PPI). The revision operator RO^1 modifies the therapy for a DU patient by replacing aspirin with clopidogrel (CL), if aspirin is prescribed without dipyridamole (D). Revision operator RO^2 modifies the therapy for a DU patient by adding a proton-pump inhibitor (PPI) when therapy includes aspirin and dipyridamole.

Figure 14. Interaction and revision operators for DU and TIA

$IO^1 := <\{DU, TIA\}, \{A, PPI\}, A \wedge \neg PPI>$
$RO^1 := <\{DU, TIA\}, \{A, PPI\}, A \wedge \neg D, \neg A \wedge \neg D \wedge CL>$
$RO^2 := <\{DU, TIA\}, \{A, PPI\}, A \wedge D, A \wedge D \wedge PPI>$

4.3. Clinical scenario 1: no adverse interactions

In this scenario we assume a patient who has tested positive for H.pylori ($HP := p$), is undergoing eradication therapy ($ET := true$), on presentation to the emergency department with TIA symptoms has tested negative for hypoglycemia ($HG := n$) and has failed the FAST test ($FAST := n$).

The mitigation algorithm invokes the *mitigate* procedure with the following input parameters: AG_{DU}, AG_{TIA} and available patient information $PI := \{HP := p, ET := true, HG := n, FAST := n\}$. It starts by calling the *create_LM* procedure to create logical models LM_{DU} and LM_{TIA} (see Figure 15). This is followed by the construction of the combined logical model $CLM_{DU,TIA}$ with $ILE_{DU,TIA}$ being empty.

Figure 15. Logical models

a) for DU (LM_{DU})

LM_{DU} = <
DU,
$\{HP, ZES, UE, ET, PPI, SC, RS\}$,
{
$(HP = n) \wedge (ZES= p) \wedge RS \wedge \neg ET \wedge \neg PPI \wedge \neg SC,$
$(HP = n) \wedge (ZES= n) \wedge PPI \wedge (UE =nh) \wedge RS \wedge \neg SC \wedge \neg ET,$
$(HP = n) \wedge (ZES= n) \wedge PPI \wedge (UE =h) \wedge SC \wedge \neg RS \wedge \neg ET,$
$(HP = p) \wedge ET \wedge PPI \wedge (\mathrm{UE} = nh) \wedge RS \wedge \neg SC,$
$(HP = p) \wedge ET \wedge PPI \wedge (UE = h) \wedge SC \wedge \neg RS$
}
>

b) for TIA (LM_{TIA})

LM_{TIA} = <
TIA,
$\{HG, FAST, NS, RST, EC, A, TST, PCS, D, NC\}$,
{
$(HG = p) \wedge EC \wedge \neg A \wedge \neg TST \wedge \neg PCS \wedge \neg D \wedge \neg NC,$
$(HG = n) \wedge (FAST = p) \wedge (NS = nr) \wedge TST \wedge NC \wedge \neg EC \wedge \neg A \wedge \neg PCS \wedge \neg D$,
$(HG = n) \wedge (FAST = p) \wedge (NS = r) \wedge A \wedge (RST = el) \wedge D \wedge NC \wedge \neg TST \wedge \neg EC \wedge \neg PCS,$
$(HG = n) \wedge (FAST = p) \wedge (NS = r) \wedge A \wedge (RST = ng) \wedge PCS \wedge \neg D \wedge \neg NC \wedge \neg TST \wedge \neg EC,$
$(HG = n) \wedge (FAST = n) \wedge PCS \wedge \neg TST \wedge \neg NC \wedge \neg EC \wedge \neg A \wedge \neg D$
}
>

Logical models LM_{DU} and LM_{TIA} do not share variables, thus direct adverse interactions are absent. The *mitigate* procedure checks for indirect adverse interactions. It first invokes the *apply_IOs* procedure to modify the $CLM_{DU,TIA}$ model by applying IO^1. The resulting model is shown in Figure 16 (logical models LM_{DU} and LM_{TIA} are given in full in Figure 15 and for the sake of brevity are not repeated).

Figure 16. Combined logical model for DU and TIA ($CLM_{DU, TIA}$)

$CLM_{DU,TIA} = <LM_{DU}, LM_{TIA}, \{A \wedge \neg PPI\}>$

Next, the *solve_CLM* procedure constructs the $CCM_{DU,TIA}$ model (given in Figure 17) from $CLM_{DU,TIA}$ and successfully solves it given patient information *PI*. A solution, filtered to return only relevant actions and decisions, is as follows: $SOL_{DU,TIA} := \{HP := p, ET := true, PPI := true, UE := h, \mathrm{SC} := true, HG := n, FAST := n, PCS := true\}$. It indicates that the patient in question should be given PPI ($PPI := true$). Also, since the result of the endoscopy (UE) is not given (not introduced to the model as part of *PI*), the CLP solver has selected a therapy corresponding to a healed ulcer ($UE := h$) and involving referral to self-care ($SC := true$).

Figure 17. CLP-CPG model for DU and TIA ($CCM_{DU,TIA}$)

$CCM_{DU,TIA} = <$
$\{HP, ZES, UE, ET, PPI, SC, RS, HG, FAST, NS, RST, EC, A, TST, PCS, D, NC\}$,
{
$((HP = n) \wedge (ZES = p) \wedge RS \wedge \neg ET \wedge \neg PPI \wedge \neg SC) \vee$
$(HP = n) \wedge (ZES = n) \wedge PPI \wedge (UE = nh) \wedge RS \wedge \neg SC \wedge \neg ET) \vee$
$(HP = n) \wedge (ZES = n) \wedge PPI \wedge (UE = h) \wedge SC \wedge \neg RS \wedge \neg ET) \vee$
$(HP = p) \wedge ET \wedge PPI \wedge (\mathrm{UE} = nh) \wedge RS \wedge \neg SC) \vee$
$(HP = p) \wedge ET \wedge PPI \wedge (UE = h) \wedge SC \wedge \neg RS)$,
$((HG = p) \wedge EC \wedge \neg A \wedge \neg TST \wedge \neg PCS \wedge \neg D \wedge \neg NC) \vee$
$(HG = n) \wedge (FAST = p) \wedge (NS = nr) \wedge TST \wedge NC \wedge \neg EC \wedge \neg A \wedge \neg PCS \wedge \neg D) \vee$
$(HG = n) \wedge (FAST = p) \wedge (NS = r) \wedge A \wedge (RST = el) \wedge D \wedge NC \wedge \neg TST \wedge \neg EC \wedge \neg PCS) \vee$
$(HG = n) \wedge (FAST = p) \wedge (NS = r) \wedge A \wedge (RST = ng) \wedge PCS \wedge \neg D \wedge \neg NC \wedge \neg TST \wedge \neg EC) \vee$
$(HG = n) \wedge (FAST = n) \wedge PCS \wedge \neg TST \wedge \neg NC \wedge \neg EC \wedge \neg A \wedge \neg D)$,
$\neg(\mathrm{A} \wedge \neg \mathrm{PPI})$
}
$>$

4.4. Clinical scenario 2: adverse interactions present

In this scenario we consider a patient who has tested negative for H.pylori ($HP := n$), positive for Zollinger-Ellison syndrome ($ZES := p$), and who on presentation to the emergency department with TIA symptoms has tested negative for hypoglycemia ($HG := n$), passed the FAST test ($FAST := p$) and has had neurological symptoms resolved ($NS := r$).

Initial operations of the *mitigate* procedure are the same as in Scenario 1 (the same logical models as given in Figure 15 and Figure 16 are constructed). Considering the available patient information the $CCM_{DU,TIA}$ model does not have a solution, because of the violated constraint ($\neg(A \wedge \neg PPI)$) that indicates the presence of indirect adverse interactions. The potential source of infeasibility is identified by the CLP solver as $PSI_{DU,TIA} := \{A, PPI\}$ and the algorithm invokes the *address_PSI* procedure.

The *activate_ROs* procedure is called to activate and order applicable revision operators. Activated revision operators are iteratively applied to $CLM_{DU,TIA}$. RO^2 is tried first (it introduces less changes than RO^1) – the *revise_CLM* and *revise_LM* procedures are called to create $CLM'_{DU,TIA}$ – a revised version of $CLM_{DU,TIA}$, where PLE_{TIA} has been modified by replacing $A \wedge D$ with $A \wedge D \wedge PPI$ where applicable. The *solve_CCM* procedure is invoked to solve a CLP-CPG model constructed from $CLM'_{DU,TIA}$. A solution still does not exist due to direct adverse interaction related to PPI (it should be prescribed according to the revised TIA guideline, while it shouldn't according to the DU guideline).

Since RO^2 has failed to address $PSI_{DU,TIA}$, RO^1 is now applied. Again, *revise_LM* and *revise_CLM* are called to make the revisions – PLE_{TIA} is modified by replacing $A \wedge \neg D$ with $\neg A \wedge \neg D \wedge CL$ when necessary, and another $CLM'_{DU,TIA}$ is constructed. Invocation of the *solve_CCM* procedure is successful and a solution with filtered and relevant actions and decisions is $SOL_{DU,TIA} := \{HP := n, ZES := p, RS := true, HG := n, FAST := p, NS := r, RST := ng, CL := true, PCS := true\}$. The mitigation algorithm returns *success* and reports the solution $SOL_{DU,TIA,}$ the potential source of infeasibility $PSI_{DU,TIA}$ and the successfully applied revision operator RO^1.

The solution $SOL_{DU,TIA}$ indicates a therapy where clopidogrel should be prescribed ($CL := true$) and the patient should be referred to primary care specialist ($PCS := true$). Considering that the risk of stroke (RST) is not given (and therefore value of corresponding variable has not introduced to the algorithm as part of the patient information PI), the CLP solver has selected a therapy corresponding to negligible risk ($RST := ng$) because it is the only one that is feasible.

5. DISCUSSION AND CONCLUSIONS

In this paper we presented a method for combining and checking pairs of CPGs that at its core has an algorithm for mitigating adverse interactions. The algorithm takes CPGs represented as actionable graphs and accepts incomplete patient data. Considering the logical complexity of mitigating adverse interactions, the algorithm does not support parallel paths in the guidelines (such paths need to be pre-processed and expanded). Its operation starts by creating logical models from actionable graphs. Logical models are further transformed into a CLP-CPG model that is solved given patient data – if a solution exists, it represents a feasible combined therapy. A lack of a solution (related to a potential source of infeasibility) indicates adverse interactions between the guidelines and an attempt to address them is automatically initiated. Mitigating adverse interactions requires clinical knowledge that goes beyond what is presented in the guidelines. This knowledge can come from medical experts (as in our example), textbooks, repositories of clinical evidence or centralized repositories of interactions [31]. We encode this knowledge as two types of operators – interaction operators that characterize indirect adverse interactions, and revision operators that describe possible revisions to logical models.

We note that logical models form the cornerstone of the mitigation algorithm. They can be easily obtained from actionable graphs, processed using interaction and revision operators and transformed into CLP-CPG models handled by effective CLP solvers. While it would be possible to modify the mitigation algorithm so it accepts CPGs in one of the well-known representations (e.g., Asbru) instead of actionable graphs and translates them directly into logical models, replacing the logical formalism entirely with another representation would require major changes to the algorithm and could affect its efficiency, especially if this new representation requires a more complex translation into CLP-CPG models.

Using illustrative clinical scenarios we have shown how our method can automatically process two CPGs, develop a computable model that can be checked for adverse interactions, and mitigate these interactions if possible. Providing such, even limited, support for the concurrent application of the guidelines is an important step in the process of creating a fully automated, point of care decision support system for the management of comorbid patients.

Building further on the research described in this paper, there are five key issues we are working on:

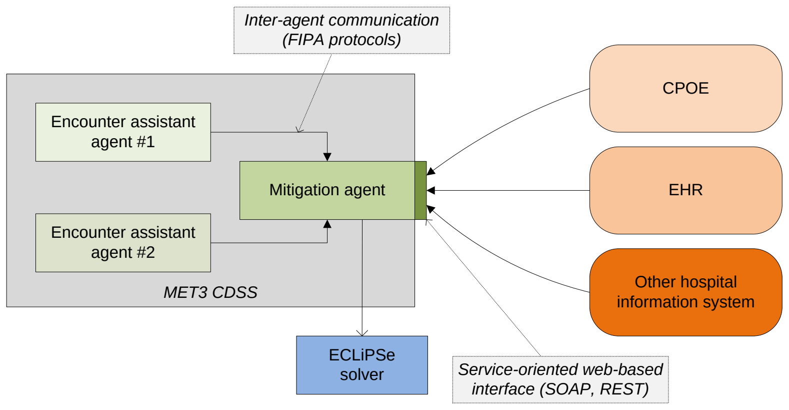
1. Expanding the methodology to accept more than two CPGs at a time. While the proposed concepts are general enough, the increased complexity associated with operating on multiple guidelines requires substantial changes to the mitigation algorithm.
2. Providing direct support for evaluating parallel paths in the guidelines. This requires revisiting the definitions of logical models and a new process of translating them into CLP-CPG models.
3. Providing direct support for representing and evaluating sub-guidelines. This asks for hierarchical logical models and appropriate changes in the mitigation algorithm.
4. Allowing for dosages of medications for a richer description of actions and adverse interactions. This requires revisiting definitions of logical models and operators.
5. Allowing for the automatic extraction of clinical knowledge from diversified sources for encoding as interaction and revision operators.

The ultimate goal of our research is to implement the mitigation algorithm as part of MET3 – our multi-agent clinical decision support system (CDSS) [34, 35]. To this end we have already developed a pilot version of the algorithm. The pilot is implemented as a Java application that invokes an external CLP solver to solve CLP-CPG models. Specifically, we have employed the ECLiPSe solver [36] because it provides a specialized library (called *repair*) to monitor constraints and to retrieve violated ones – this significantly facilitates the identification of a possible source of infeasibility in the *solve_CCM* procedure (see Figure 8). Moreover, the use of a specialized CLP solver ensures appropriate computational performance of the implemented algorithm. Currently available CLP solvers can efficiently solve problems with thousands of variables and constraints as demonstrated by the results of several constraint solver competitions (e.g., [37] or [38]). Given our experience with the CPGs considered so far, the complexity of CLP-CPG models was (at least) an order of a magnitude smaller – the number of variables did not exceed 100 and the number of constraints did not exceed 20. For CLP-CPG models of such size we were able to get immediate results (solving CLP-CPG models lasted less than 1 second when running the pilot on a Motion Computing C5 tablet PC, often used as a mobile computing platform in hospitals). Considering the scalability of CLP solvers demonstrated by their extensive benchmarks, we believe the efficiency of the implemented algorithm should be acceptable from a practical perspective even for extremely large and complex CPGs.

The next step is to integrate the pilot implementation of the mitigation algorithm with MET3 by embedding the pilot into a specialized *mitigation agent*. This agent will be available to other agents in the system, especially an *encounter assistant agent* that supports the physician throughout the entire patient management process. Moreover, following recent research on CDSS architectures, specifically on the service model [39], the mitigation agent will expose its services to other hospital systems such as a computerized physician order entry (CPOE) or an electronic health record (EHR) using a web-based interface. Such a service-oriented approach has been adopted in several CDSSs (e.g., SEBASTIAN [40], SANDS [41]) and has already demonstrated its capability for easy integration of advanced decision support with existing hospital information systems [42]. Moreover, such a solution has been mandated by the HL7 organization that proposed a standard for decision support services [43].

The envisioned integration of the implemented algorithm with the MET3 CDSS and other hospital information systems is presented in Figure 18. Communication between agents within the MET3 system follows standard FIPA protocols for multi-agent systems [35], while the service-oriented interface relies on standard web-based technologies and protocols (like SOAP and REST) [41].

Figure 18. Envisioned implementation of the mitigation algorithm


Inter-agent communication (FIPA protocols)
Encounter assistant agent #1
Mitigation agent
Encounter assistant agent #2
MET3 CDSS
CPOE
EHR
Other hospital information system
ECLiPSe solver
Service-oriented web-based interface (SOAP, REST)

The MET3 system expanded with the mitigation agent will fully support a complex patient management process aimed at developing the combined therapy. In addition to the already assisted steps of the process (data collection, diagnosis and therapy planning), the system will apply the mitigation algorithm to identify possible adverse interactions, to warn the physician about them and to present in advance possible ways of addressing them. According to Fieschi et al. [44] an effective CDSS should place a physician at the center of the decision making loop – this means that the system would need to provide the user with all possible combined therapies, and if adverse interactions between individual therapies exist, with all possible ways of addressing them, and let the physician make the ultimate decision. The mitigation algorithm in its current formulation is aimed at automatic execution. While defining operators requires significant involvement of clinical experts, once all relevant operators have been supplied, the algorithm runs without any interaction with the user, and it reports a single, arbitrary selected (i.e., first found) therapy and revisions addressing discovered interactions. However, it can be easily modified to act as an assistant as opposed to an oracle – most of the changes would be introduced in the *address_PSI* procedure (see Figure 9) in order to iterate over all applicable revision operators and to return multiple solutions. Such changes to the mitigation algorithm should result in better clinical acceptance of the CDSS implementing it.

6. ACKNOWLEDGMENT

The authors would like to thank the reviewers for their constructive comments.

The research described in this paper was supported by grants from the Natural Sciences and Engineering Research Council of Canada, University of Ottawa, and from the Polish National Science Centre.

References

[1] R.M. Rosenfeld, R.N. Shiffman, Clinical practice guideline development manual: a quality-driven approach for translating evidence into action, Otolaryngol. Head Neck Surg. 140 (2009) S1-43.
[2] R.S.A. Hayward, M.C. Wilson, S.R. Tunis, E.B. Bass, G. Guyatt, Users' guides to the medical literature. VIII. How to use clinical practice guidelines A. Are the recommendations valid?, JAMA 274 (1995) 570-574.
[3] The Cochrane Library. Available at http://www.thecochranelibrary.com. Accessed on August 26, 2012.
[4] R. Goud, M. van Engen-Verheul, N.F. de Keizer, R. Bal, A. Hasman, I.M. Hellemans, N. Peek, The effect of computerized decision support on barriers to guideline implementation: a qualitative study in outpatient cardiac rehabilitation, Int. J. Med. Inform. 79 (2010) 430-7.
[5] M. van den Akker, F. Buntinx, J.F. Metsemakers, S. Roos, J.A. Knottnerus, Multimorbidity in general practice: prevalence, incidence, and determinants of co-occurring chronic and recurrent diseases, J. Clin. Epidemiol. 51 (1998) 367-75.
[6] G. Anderson, J. Horvath, J. Knickman, D. Colby, S. Schear, M. Jung, Chronic Conditions: Making the Case for Ongoing Care, Partnership for Solutions, Johns Hopkins University, Baltimore, MD, 2002.
[7] C.M. Boyd, J. Darer, C. Boult, L.P. Fried, L. Boult, A.W. Wu, Clinical practice guidelines and quality of care for older patients with multiple comorbid diseases: implications for pay for performance, JAMA 294 (2005) 716-24.
[8] D.F. Sittig, A. Wright, J.A. Osheroff, B. Middleton, J.M. Teich, J.S. Ash, E. Campbell, D.W. Bates, Grand challenges in clinical decision support, J. Biomed. Inform. 41 (2008) 387-392.
[9] J. Fox, D. Glasspool, V. Patkar, M. Austin, L. Black, M. South, D. Robertson, C. Vincent, Delivering clinical decision support services: there is nothing as practical as a good theory, J. Biomed. Inform. 43 (2010) 831-843.
[10] M. Michalowski, M. Mainegra Hing, S. Wilk, W. Michalowski, K. Farion, A constraint logic programming approach to identifying inconsistencies in clinical practice guidelines for patients with comorbidity, in: M. Peleg, N. Lavrac, C. Combi (Eds.), Artificial Intelligence in Medicine, 13th Conference on Artificial Intelligence in Medicine, AIME 2011, Bled, Slovenia, July 2-6, 2011, Proceedings, Springer, Berlin/Heidelberg/New York, 2011, pp. 296-301.
[11] S. Wilk, M. Michalowski, W. Michalowski, M. Mainegra Hing, K. Farion, Reconciling pairs of concurrently used clinical practice guidelines using constraint logic programming, AMIA Annu. Symp. Proc. (2011) 944-953.
[12] R. Dechter, Constraint Processing, The MIT Press, 1989.
[13] B. Perez, I. Porres, Authoring and verification of clinical guidelines: a model driven approach, J. Biomed. Inform. 43 (2010) 520-536.
[14] A. ten Teije, M. Marcos, M. Balser, J. van Croonenborg, C. Duelli, F. van Harmelen, P. Lucas, S. Miksch, W. Reif, K. Rosenbrand, A. Seyfang, Improving medical protocols by formal methods, Artif. Intell. Med. 36 (2006) 193-209.
[15] G. Duftschmid, S. Miksch, Knowledge-based verification of clinical guidelines by detection of anomalies, Artif. Intell. Med. 22 (2001) 23-41.
[16] P. Groot, A. Hommersom, P.J. Lucas, R.J. Merk, A. ten Teije, F. van Harmelen, R. Serban, Using model checking for critiquing based on clinical guidelines, Artif. Intell. Med. 46 (2009) 19-36.
[17] J.B. Lamy, V. Ebrahiminia, C. Riou, B. Seroussi, J. Bouaud, C. Simon, S. Dubois, A. Butti, G. Simon, M. Favre, H. Falcoff, A. Venot, How to translate therapeutic recommendations in clinical practice guidelines into rules for critiquing physician prescriptions? Methods and application to five guidelines, BMC Med. Inform. Decis. Mak. 10 (2010) 31.

[18] D. Isern, A. Moreno, G. Pedone, D. Sánchez, L. Varga, Home care personalisation with individual intervention plans, in: D. Riaño (Ed.) Knowledge Management for Health Care Procedures: ECAI 2008 Workshop, K4HelP 2008, Patras, Greece, July 21, 2008, Revised Selected Papers, Springer, Berlin/Heidelberg, 2009, pp. 134-151.

[19] D. Riaño, F. Real, J.A. Lopez-Vallverdu, F. Campana, S. Ercolani, P. Mecocci, R. Annicchiarico, C. Caltagirone, An ontology-based personalization of health-care knowledge to support clinical decisions for chronically ill patients, J. Biomed. Inform. 45 (2012) 429-446.

[20] S.R. Abidi, S.S. Abidi, Towards the Merging of Multiple Clinical Protocols and Guidelines via Ontology-Driven Modeling, in: C. Combi, Y. Shahar, A. Abu-Hanna (Eds.), Artificial Intelligence in Medicine. 12th Conference on Artificial Intelligence in Medicine, AIME 2009, Verona, Italy, July 18-22, 2009. Proceedings, Springer, Berlin/Heidelberg/New York, 2009, pp. 81-85.

[21] F. Real, D. Riaño, An autonomous algorithm for generating and merging clinical algorithms, in: D. Riaño (Ed.) Knowledge Management for Health Care Procedures. ECAI 2008 Workshop, K4HelP 2008, Patras, Greece, July 21, 2008, Revised Selected Papers, Springer, Berlin/Heidelberg, 2009, pp. 13-24.

[22] F. Real, D. Riaño, Automatic combination of formal intervention plans using SDA* representation model, in: D. Riaño (Ed.) Knowledge Management for Health Care Procedures. From Knowledge to Global Care. AIME 207 Workshop K4CARE 2007. Revised Selected Papers, Springer, Berlin/Heidelberg/New York, 2008, pp. 75-86.

[23] P.A. de Clercq, J.A. Blom, H.H. Korsten, A. Hasman, Approaches for creating computer-interpretable guidelines that facilitate decision support, Artif. Intell. Med. 31 (2004) 1-27.

[24] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R.A. Greenes, R. Hall, P.D. Johnson, N. Jones, A. Kumar, S. Miksch, S. Quaglini, A. Seyfang, E.H. Shortliffe, M. Stefanelli, Comparing computer-interpretable guideline models: a case-study approach, J. Am. Med. Inform. Assoc. 10 (2003) 52-68.

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

[26] M. Peleg, S. Tu, G. Leonardi, S. Quaglini, P. Russo, G. Palladini, G. Merlini, Reasoning with effects of clinical guideline actions. Using OWL: AL amyloidosis as a case study, in: D. Riaño, ten Teije, A. , S. Miksch (Eds.), 3th International Workshop on Knowledge Representation for Health Care (KR4HC'11), Springer, Berlin/Heidelberg, 2011, pp. 65-79.

[27] D. Isern, A. Moreno, D. Sánchez, Á. Hajnal, G. Pedone, L.Z. Varga, Agent-based execution of personalised home care treatments, Appl. Intell. 34 (2009) 155-180.

[28] E. Horvitz, Automated reasoning for biology and medicine, in: R. Fortuner (Ed.) Advances in Computer Methods for Systematic Biology: Artificial Intelligence, Databases, and Computer Vision, John Hopkins University Press, Baltimore, MD, 1993, pp. 3-30.

[29] M.T. Cox, A. Raja, Metareasoning: Thinking about Thinking, MIT Press, 2010.

[30] C.J. Petrie, Revised dependency-directed backtracking for default reasoning, in: Proceedings AAAI-87 Seattle, WA, 1987, pp. 167-172.

[31] S. Phansalkar, A.A. Desai, D. Bell, E. Yoshida, J. Doole, M. Czochanski, B. Middleton, D.W. Bates, High-priority drug-drug interactions for use in electronic health records, J. Am. Med. Inform. Assoc. 19 (2012) 735-743.

[32] National Institute for Health and Clinical Excellence. Available at http://www.nice.org.uk/. Accessed on August 20, 2012.

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

[34] S. Wilk, W. Michalowski, D. O'Sullivan, K. Farion, J. Sayyad-Shirabad, C. Kuziemsky, B. Kukawka, A task-based support architecture for developing point-of-care clinical decision support systems for the emergency department, Methods Inf. Med. (2013) (in print).

[35] J. Sayyad Shirabad, S. Wilk, W. Michalowski, K. Farion, Implementing an Integrative Multi-agent Clinical Decision Support System with Open Source Software, Journal of Medical Systems 36 (2012) 123-137.

[36] The ECLiPSe Constraint Programming System. Available at http://www.eclipseclp.org. Accessed on August 23, 2012.

[37] 4th International CSP Solver Competition (CSC'2009). Available at http://www.cril.univ-artois.fr/CPAI09/. Accessed on December 27, 2012.

[38] MiniZinc Challenge 2012. Available at http://www.g12.cs.mu.oz.au/minizinc/challenge2012/. Accessed on December 27, 2012.

[39] A. Wright, D.F. Sittig, A four-phase model of the evolution of clinical decision support architectures, International Journal of Medical Informatics 77 (2008) 641-649.
[40] K. Kawamoto, D.F. Lobach, Design, implementation, use and preliminary evaluation of SEBASTIAN, a standards-based Web service for clinical decision support, AMIA Annu. Symp. Proc. (2005) 380-384.
[41] A. Wright, D.F. Sittig, SANDS: a service-oriented architecture for clinical decision support in a National Health Information Network, J. Biomed. Inform. 41 (2008) 962-81.
[42] D. Borbolla, C. Otero, D.F. Lobach, K. Kawamoto, A.M. Gomez Saldano, G. Staccia, G. Lopez, S. Figar, D. Luna, F.G. Bernaldo de Quiros, Implementation of a clinical decision support system using a service model: results of a feasibility study, Stud. Health Technol. Inform. 160 (2010) 816-20.
[43] HL7 Version 3 Standard: Decision Support Service. Available at http://www.hl7.org/implement/standards/product_brief.cfm?product_id=12. Accessed on December 19, 2012.
[44] M. Fieschi, J.C. Dufour, P. Staccini, J. Gouvernet, O. Bouhaddou, Medical decision support systems: old dilemmas and new paradigms?, Methods Inf. Med. 42 (2003) 190-8.