Preliminary Result on Finding Treatments for Patients with Comorbidity

Yuanlin Zhang[1] and Zhizheng Zhang[2]

[1] Texas Tech University, Lubbock, USA y.zhang@ttu.edu
[2] Southeast University, Nanjing, China seu_zzz@seu.edu.cn

Abstract. According to some research, comorbidity is reported in 35 to 80% of all ill people [1]. Multiple guidelines are needed for patients with comorbid diseases. However, it is still a challenging problem to automate the application of multiple guidelines to patients because of redundancy, contraindicated, potentially discordant recommendations. In this paper, we propose a mathematical model for the problem. It formalizes and generalizes a recent approach proposed by Wilk and colleagues. We also demonstrate that our model can be encoded, in a straightforward and simple manner, in Answer Set Programming (ASP) – a class of Knowledge Representation languages. Our preliminary experiment also shows our ASP based implementation is efficient enough to process the examples used in the literature.

Keywords: Answer Set Programming, Clinical Practice Guidelines, Knowledge Representation, Comorbidity

1 Introduction

Clinical practice guidelines (CPGs)[2], created by experts and supported by medical evidences, are documents guiding the decisions in specific areas/conditions of healthcare. It is generally agreed that the use of guidelines can greatly improve the outcome of clinical medical care. To promote the use of CPGs and increase their accessibility, an important effort is to build systems that can automatically execute the guidelines given patients' information. Most of the early systems are based on the representation languages such as Asbru, GLIF, GUIDE, EON, PROforma [3,4,5]. CPGs are usually developed to target a single disease [3], and thus these systems and languages were focusing on single diseases too.

It has been noted that the majority of elderly patients have multiple co-morbidities and medications that must be addressed by their patient care team [6]. When applying multiple CPGs to comorbid patients, as pointed out by Sittig et al. [7], "the challenge is to create mechanisms to identify and eliminate redundant, contraindicated, potentially discordant, or mutually exclusive guideline based recommendations for patients presenting with comorbid conditions or multiple medications."

As an example, consider an ulcer patient with transient stroke. According to the ulcer CPG, *stop aspirin* is a necessary activity in the treatment of ulcer while *start aspirin*,

according to the transient stroke CPG, is also a necessary activity under certain situations. In this case, it is desirable for a CPG based system to identify the inconsistency of the *stop aspirin* and *start aspirin* activities resulted from two distinct CPGs, and then remove the inconsistency by replacing the activity of *start aspirin* by *start clopidogrel*.

Recently several attempts [8,9,10] have been made to attack the problem (or a part of it). We are particularly interested in the approach proposed by Wilk and colleagues [11,12,10] which consists of two steps. The first step is to identify the adverse and contradictory activities, i.e., inconsistencies, that are obtained by applying the CPG to each of the several diseases a patient has. The second step is to mitigate the inconsistencies. We call the problem introduced by Wilk at al. *guideline reconciling problem*.

Wilk and colleagues offered one of the first few automated solutions for the concurrent application of CPGs to two diseases. However, they only gave a solution for the problem, based on Constraint Logic Programming program and pseudo code algorithms, but did not give a mathematical definition of the problem. As a result of the lack of problem definition, it is not easy to evaluate how closely this problem models the real problems on the application of two or more CPGs, and the solution to the reconciling problem may be unnecessarily restricted (e.g., to the approach used in their work) too.

In the research reported here, we make the following contributions to improve Wilk and colleagues' work.

We separate the reconciling problem from its solution(s). We find that graph theory provides a handy tool for us to develop an explicit mathematical definition for the reconciling problem. Compared with Wilk et al.'s work, our definition does not depend on an programming languages or algorithms. We expect the definition to be more accessible to researchers in the medical area (maybe with help of computer scientists) and thus makes it easier for them to evaluate its capacity of modeling the real situation. On the other hand, once the problem is (mathematically) defined, computer scientists can focus on finding better ways to solve the problem, without worrying too much about the required medical background.

In our definition, we also generalize the problem implied by Wilk et al's algorithms by allowing OR decision nodes.

Once the problem is defined, there are many immediate ways to solve the problem under both declarative and imperative programming paradigms. As an example, we present a solution based on Answer Set Programming (ASP), a declarative programming paradigm. ASP is a well developed non-monotonic logic programming paradigm in the knowledge representation community. The logic rules of ASP are natural because a good amount of knowledge in guidelines are in the form of rules. More important, rules in CPGs usually involve exceptions, which can be addressed very well by the non-monotonicity property of ASP. Thanks to its declarativeness and non-monotonicity, ASP is elaboration tolerant [13,14] (which means that it is convenient to modify a set of facts expressed in ASP to take into account new phenomena or changed circumstances in the domain of concern), which is particularly amenable to the constant revision of guidelines driven by the growth of our knowledge on all aspects of diseases. Equally important to the expressiveness of ASP, several efficient ASP inference engines or solvers such as DLV [15] and CLASP [16] have been developed and maintained in the last decade. They enable the development of efficient ASP based solutions to application

problems. An important note in our decision of using ASP is there are many aspects of the problem (as a mathematical model for reconciling CPGs) that need to be improved to address the real life problems. Those improvements can very well make the problem NP-hard. So, we are not interested in ad hoc algorithm(s) specifically designed for the problem defined in its current form.

The encoding of the problem using ASP is natural and simple, almost a straightforward translation of the problem definition. Our preliminary experiment also shows that the examples mentioned in [10] can be solved efficiently. As far as we are aware of, there is no system available on applying multiple CPGs to several diseases. Our ASP program is available publicly and downloadable at `http://redwood.cs.ttu.edu/~yzhang/temp/KR-14/code-coMorbidity-dlv.lp`.

The rest of this paper is organized as follows. We first review the activity graph representation of clinical practice guidelines and answer set programming in Section 2. The formal definition of the problem of concurrent application of CPGs to a patient's comorbid diseases is given in Section 3. In Section 4, an answer set programming based solution is presented. We then present the implementation of the ASP approach and the preliminary evaluation of the program in Section 5. Finally, conclusion is made in the last section of the paper.

2 Preliminary

The work reported in [12,11] focuses on the activity graphs that are used to represent a major portion of a CPG (e.g., in SAGE [17]). We will recall activity graphs in the first subsection, and introduce some background knowledge about Answer Set Programming in the second subsection.

2.1 Activity Graph for CPGs

An activity graph (AG) is a directed graph that consists of context, action and decision nodes. A context node is the root node of the AG and it defines a clinical context where the CPG is applied to. As an example, "patient diagnosed with TIA" is the root node of the AG for transient ischemic attack (TIA) (Fig 1 right). An action is a clinical action to be performed according to the guideline. An example is "take aspirin." A decision step represents a decision point in a guideline. For example, a decision step in the guideline for TIA is whether hypoglycaemia is present. Decision nodes can be further divided into OR or XOR nodes. The former indicates more than one alternative can be resulted from a decision node while the later means that only one alternative can be resulted from the decision node. Fig 1 left shows an example of the AGs for duodenal ulcer (DU) and TIA. For every disease (clinical context) we assume there is one and only one clinical practice guideline for it and one and only one activity graph for it.

2.2 Answer Set Programming

We now give a brief introduction of answer set programming and refer the interested reader to the book [14] for more details.

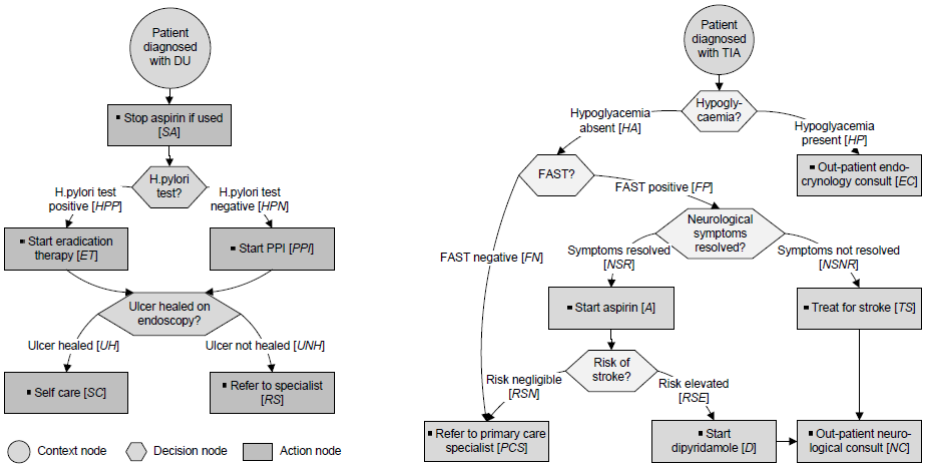

Fig. 1. Activity Graphs for DU (left) and TIA (right) (from [12])

Answer set programming originates from non-monotonic logic and logic programming. It is a logic programming paradigm based on the answer set semantics [18,14], which particularly offers an elegant declarative semantics to the negation as failure operator in Prolog. An ASP program consists of *rules* in the form:

$$l_0|...|l_k \ :- \ l_{k+1}, ..., l_m, not \ l_{m+1}, ..., not \ l_n.$$

where each l_i for $i \in [0..n]$ is a literal of some signature, i.e.,an expression of the form $p(t)$ or $\neg p(t)$ where p is a predicate and t is a term, and *not* is called *negation as failure* or *default negation* and $|$ epistemic disjunction. A rule without body is called a *fact*, and a rule without head is called a *denial*. The rule is read as: if one believes l_{k+1}, ..., and l_m and there is no reason to believe l_{m+1}, ..., and l_n, one must believe l_0, l_1, ..., or l_k. The answer set semantics of a program P assigns to P a collection of answer sets, i.e., interpretations of the signature of P corresponding to possible sets of beliefs (i.e., literals). These beliefs can be built by a rational reasoner by following the principles that the rules of P must be satisfied and that one shall not believe anything unless one is forced to believe.

There have been several research groups developing and maintaining high quality efficient ASP solvers. Examples include DLV [15], and Clasp [16]. These solvers have been employed to successfully solve problems ranging from monitoring elderly people in nursing homes [19] to decision support systems for the space shuttle controllers [20].

3 Definition of the Reconciling Problem

In this section, based on graphs, we define the reconciling problem in the concurrent application of CPGs to a patient with two diseases. We first define activity graph obtainable from the CPG for a disease, candidate treatment for a disease using the activity

graph, and valid treatment. We then define point of contention, i.e., conflicts, among candidate treatments and the mitigation operation to remove the conflicts. Finally, we define the reconciling problem.

3.1 Candidate Treatment

Definition 1 (Activity Graph). *An* activity graph *of a CPG of a disease is a directed graph with labels on some edges* $(CN \cup AN \cup DN_o \cup DN_{xor}, E, l : E \to L)$ *where*

- $CN, AN, DN_o, DN_{xor}, L$ *are disjoint sets,*
- $CN = \{x\}$ *and* x *is called the* context node*,*
- *Elements of* AN, DN_o, DN_{xor}, L *are called* action nodes, or-decision nodes, xor-decision nodes*, and* labels *respectively,*
- $E \subseteq V \times V$*, where* $V = CN \cup AN \cup DN_o \cup DN_{xor}$ *and an element* (x, y) *of* E *is called* an edge*, an* incoming edge *of* y *and an* outgoing edge *of* x*, such that there is no incoming edge for the context node and for any non decision node there is at most one outgoing edge,*
- l*, called a* labeling function*, is a partial function from edges to labels such that* $l((x, y))$ *is always defined if* x *is a decision node.*

The activity graph shown in the left of Fig 1 is as follows: $CN = \{du\}$ where du is the shorthand for "Patient diagnosed with DU" for convenience, $AN = \{sa, et, ppi, sc, rs\}$ (note that these names in the figure are in capital letters), $DN_o = \emptyset$, $DN_{xor} = \{htest, uhe\}$ where $htest$ is for "H. pylori test?" and uhe for "Ulcer healed on endoscopy?"

$E = \{(du, sa), (sa, htest), (htest, et), (htest, ppi),$
$(et, uhe), (ppi, uhe), (uhe, sc), (uhe, rs)\},$

$L = \{hpp, hpn, uh, unh\}$, and the labeling function l labels outgoing edges of decision nodes as follows: $l((htest, et)) = hpp$, $l((htest, ppi)) = hpn$, $l((uhe, sc)) = uh$, $l((uhe, rs)) = unh$.
For treatment oriented CPGs, an important task is to follow them to find a treatment with the given patient information. We now define the candidate treatment with respect to a CPG of a disease.

Definition 2 (Candidate Treatment). *A* candidate treatment *is the collection of actions of a subgraph* H *of* G *such that*

- *the context node of* G *belongs to* H*,*
- *every node of* H *is reachable from the context node wrt* H*,*
- *for every node* x *of* H*, there is a node* y *of* H *that is a leaf node of* G *such that* y *is reachable from* x *in* H*, and*
- *for every xor-decision-node* x *of* H*, its outgoing degree wrt* H *is one.*

H *is called an* underlying graph *of the candidate treatment.*

It is worth noting that or-decision may cause two or more decision options to be satisfied. Therefore, in our definition of candidate treatment we allow parallel outgoing edges from an or-decision node. We also note an interpretation of or-decision is that external preference information is usually used to help choose one of the several outgoing edges. With preference information given, the underlying graph of a candidate treatment is reduced to a path in the activity graph.

For simplicity, we assume that for any candidate treatment, there is a unique underlying graph without loss of generality. The majority of CPGs have xor-decision-node only. In this case, a candidate treatment is a path from the context node to a leaf node. In the left graph of Fig 1, there are totally four candidate treatments for patients diagnosed with DU (chronic condition):

- T_1^{du}: $\{sa, et, sc\}$ with the underlying path $du \to sa \to hpt \to uhd \to et \to sc$.
- T_2^{du}: $\{sa, ppi, sc\}$ with the underlying path $du \to sa \to hpt \to uhd \to ppi \to sc$.
- T_3^{du}: $\{sa, et, rs\}$ with the underlying path $du \to sa \to hpt \to uhd \to et \to rs$.
- T_4^{du}: $\{sa, ppi, rs\}$ with the underlying path $du \to sa \to hpt \to uhd \to ppi \to rs$.

The right graph of Fig 1 presents five candidate treatments for patients with TIA where tia is for "Patient diagnosed with TIA," hc is for "Hypoglycaemia?" $nsrtest$ for "Neurological symptoms resolved?" and $rstest$ for "Risk of stroke?"

- T_1^{tia}: $\{ec\}$ with the underlying path $tia \to ec$.
- T_2^{tia}: $\{pcs\}$ with the underlying path $tia \to hc \to fast \to pcs$.
- T_3^{tia}: $\{a, pcs\}$ with the underlying path $tia \to hc \to fast \to nsrtest \to a \to rstest \to pcs$.
- T_4^{tia}: $\{a, d, nc\}$ with the underlying path $tia \to hc \to fast \to nsrtest \to a \to rstest \to d \to nc$.
- T_5^{tia}: $\{ts, nc\}$ with the underlying path $tia \to hc \to fast \to nsrtest \to ts \to nc$.

3.2 Valid Treatment

For a patient with comorbid diseases, inconsistencies are often introduced in the possible treatments because of the amalgamation of multiple guidelines. We give the definition of the related concepts as follows.

Given two candidate treatments T_1, T_2 wrt two activity graphs G_1 and G_2 respectively, and a collection I of sets, called *incompatible sets*[3], of actions, a set AS of actions is a *point of contention* (POC for short) between T_1 and T_2 if every action of AS is an action of T_1 or T_2 and $AS \in I$.

For example, start aspirin a and stop aspirin sa form an incompatible set[4] $\{sa, a\}$. Suppose there is an ulcer patient diagnosed with transient ischemic attack. A point of contention between T_1^{du} and T_3^{tia} is $\{sa, a\}$.

When there are point of contentions between candidate treatments for two diseases, one can find ways to mitigate the point of contention. Mitigation operator is defined by Wilk et al. [12] as follows.

A *mitigation operator* (MO) for disease d_1 and d_2 is a tuple *(d_1, d_2, contentions, LHS, RHS, toBeRemoved)* where

[3] Many incompatible sets are known facts are directly from the medical field.

[4] They are logically inconsistent.

- d_1 is called a *base disease* and d_2 a *target disease*,
- *contentions* is a set of actions from the activity graphs for d_1 and d_2,
- *LHS* and *RHS* are a set of elements, called *action literals* of the form *pos(A)* or *neg(A)* where A is a medical action that may or may not be an action of activity graphs of d_1 or d_2,
- toBeRemoved is a set of actions of activity graph of d_2.

For example, the following MO1 and MO2 are MOs for TIA and UD addressing the point of contention $\{a, sa\}$.

- MO1: (*tia, du,* {*sa, a*}, {*pos(a), neg(d)*},{*neg(a), pos(cl)*}, {*sa*}).
- MO2: (*tia, du,* {*sa, a*}, {*pos(a), pos(d)*},{*pos(a), pos(d), pos(ppi)*}, {*sa*}).

Given candidate treatment T_1 and T_2 wrt activity graph G_1 of disease d_1 and activity graph G_2 of disease d_2, an MO α=(d_1, d_2, *contentions*, LHS, RHS, *toBeRemoved*) for d_1 and d_2 is *relevant* to T_1 and T_2 if *contentions* is a subset of $T_1 \cup T_2$. An MO is *applicable to* T_1 and T_2 if it is relevant and for every $pos(A) \in LHS$, action $A \in T_1$, and for every $neg(A) \in LHS$, action $A \notin T_1$. Suppose α is applicable to T_1 and T_2, the *modified treatment* by applying α to T_1 and T_2 is T_1' and T_2' where

- $T_1' = \{A : pos(A) \in RHS$ or ($A \in T_1$ but A occurs neither in LHS nor RHS) $\}$, and
- T_2' = $T_2 - toBeRemoved$.

Continue the example above. For T_1^{du} and T_3^{tia}, there is a point of contention $\{a, sa\}$ between them. MO1 is applicable and can be used to modify T_1^{du} and T_3^{tia}. By definition, the modified treatments of applying MO1 to T_1^{du} and T_3^{tia} are as follows: $T_3^{tia\prime}$: $\{cl, pcs\}$ and $T_1^{du\prime}$ = $\{et, sc\}$.
We next define treatments targeting a specific patient's situation.

Definition 3 (Patient Information). *We define* Patient Information *(PI) as a set of pairs (*decision, value*) where* decision *is a decision node and* value *is a label of an outgoing edge of the node* decision *in G. A candidate treatment T* agrees with *patient information I, if for every decision node x and every edge (x, y) of the underlying graph of T, $(x, val) \in I$ where val is the label of (x, y).*

For example, both T_3^{tia} and T_4^{tia} agree with PI $\{(hc, ha), (fast, fp), (nsrtest, nsr)\}$ (for ha, fp, see the right graph of Fig 1).

Definition 4 (Valid Treatment). *Given PI I of a patient, with two diseases, and candidate treatments T_1 and T_2 for these diseases respectively, $T_1 \cup T_2$ is* a valid treatment *with respect to I if T_1 and T_2 agree with I, and there is no point of contention between T_1 and T_2.*

Definition 5 (Reconciling Problem). *Given PI I of a patient with diseases D_1 and D_2 and a set of MOs for D_1 and D_2, the* reconciling problem *is to find a valid treatment with respect to I if there exists one, and otherwise find if there are candidate treatment T_1 and T_2 such that their modified treatment T_1' and T_2' by applying some of the MOs are valid.*

For example, let the PI of an ulcer patient diagnosed with transient stroke be $\{(hc, ha), (fast, fp), (nsrtest, nsr)\}$. Clearly, the candidate treatments for ulcer and transient stroke that agree with PI are T_3^{tia}, T_4^{tia} and $T_1^{du}, T_2^{du}, T_3^{du}, T_4^{du}$ respectively. There is no valid treatment for the patient because of a point of contention $\{sa, a\}$ between each pair of the candidate treatments. By applying MO1 to T_3^{tia} and T_1^{du}, we get a valid modified treatment $T_3^{tia'} \cup T_1^{du'}$ where $T_3^{tia'} = \{cl, pcs\}$ and $T_1^{du'} = \{et, sc\}$ as illustrated in the earlier example.

4 ASP Based Solution

In this section, we present an ASP based solution of finding a valid (modified) treatment for patients with comorbid diseases according to the CPGs for these diseases and mitigation operators.

Representation of an activity graph. We first introduce the predicates needed to represent the activity graph g for a disease d: `cNode(g,ct)` – ct is the context node of g, `aNode(g,Action)` – $Action$ is an action node of g, `oNode(g,N)` – N is an or decision node, `xNode(g,N)` – N is an xor decision node, `edge(g, X, Y)` – (X, Y) is an edge of g, `label(g, X, Y, L)` – the label on the edge (X, Y) is L. A given activity graph will be represented as facts using the predicates above.

Define candidate treatments. To specify a valid (modified) treatment, we first need to define a candidate treatment. In turn we need to construct a subgraph H of an activity graph G by Definition 2. By `candidateEdge(`G`, X, Y)`, we mean the edge `(X, Y)` is in H. The following rule is to define H:

```
candidateEdge(G, X, Y) | ¬candidateEdge(G, X, Y)
      :- node(G,X), not decisionNode(G, X).
```

which reads that any edge (X, Y) of G can be an edge of H. We will next present ASP rules to make sure H is the underlying graph of a candidate treatment by Definition 2.

First, the context node of an activity graph G must be a node of H. Note that we know only edges of H but not the nodes of H. Now we need a predicate `nodeInH(G, N)` to denote N is a node of H. It can be defined as:

```
nodeInH(G, X) :- candidateEdge(G, X, Y).
nodeInH(G, Y) :- candidateEdge(G, X, Y).
```

which can be read as any end of an edge (X, Y) of H is a node of H.

Now the rule `:- cNode(G, CN), not nodeInH(G, CN).` says that if CN is a context node, it must be in H.

Second, every node of `H` is reachable from the context node in `H`. We first define the reachability (`reachable(G, X, Y)` denotes that node Y is reachable from X in H) in a standard way:

```
reachable(G, X, X):- cNode(G,X).
reachable(G, X, Y):- candidateEdge(G,X,Y).
reachable(G, X, Y):- reachable(G,X,Z), candidateEdge(G,Z,Y).
```

The rule below restricts that for every node X of H, X must be reachable from the context node Cn:

```
:- nodeInH(G, X), cNode(G, Cn), not reachable(H, Cn, X).
```

Thirdly, every node of H reaches a leaf node. It is not hard to define a leaf node and use reachable to express this constraints. Rules are omitted here due to lack of space.

Finally, for every *xor* node of H, its outgoing degree must be one. We first define the existence of an outgoing edge for a node X:

```
existsOutgoingEdge(G, X) :- candidateEdge(G, X, Y).
```

Since there is at most one outgoing edge from an action node in any activity graph, the rule

```
:- nodeInH(G, X), xNode(G, X),
   not existsOutgoingEdge(H, X).
```

is sufficient to restrict that for any node X of H, it has one outgoing edge.

Now we are in a position to define a candidate treatment using H. It is not hard to write a rule to define actionInH(G, X) which holds if X is an action node of H. We omit the rule here.

Define Valid Treatments Given a patient informatin I, we present the ASP rules that encode patient information, the agreement of a candidate treatment to patient information, and the points of contention and finally a valid treatment, in terms of the corresponding definitions given in the previous section.

Patient Information. We use bDisease(d_1) to denote that d_1 is the base disease, tDdisease(d_2) to denote d_2 is the target disease, and patientInfo(x,l) to denote the value of the decision node x is l. The patient information is represented as facts using the predicates above.

Agreement of a Candidate Treatment to the Patient Information. For every decision node, it should agree with the patient information on this node, i.e., for any or-decision node X and any patientInfo(X, L) with (X, Y) labeled by L, (X, Y) must be a candidate edge:

```
:- decisionNode(G, X), nodeInH(G, X), patientInfo(X,L),
   label(AG, X, Y, L), not candidateEdge(G, X, Y).
```

Note here for any or-decision node, its outgoing edges in H correspond to a superset of the values on this node given by the patient information.

Incompatible Sets. For every incompatible set, we assign an id (index) for it and include in the program the fact incompSet(index). For every action a in the incompatible set with id index, we have the fact ncompSetAction(index, a).

POC of Two Treatments. We need a notion of active action here. An action is *active* if it is in a candidate treatment wrt a disease:

```
active(X):- actionInH (H, X).
```

An incompatble set is *active* if all its actions are active. Clearly, an active incompatible set is a POC. We first define a non active incompatible set `nonActiveIncompSet` which is then used to define POC using default negation.

```
nonActiveIncompSet(Index) :-
     not active(X),
     incompSetAction(Index, X).

isPOC(Index) :-
     incompSet(Index),
     not nonActiveIncompSet(Index).
```

We use `existsPOC` to denote the occurrence of a POC between two candidate treatments:

```
existsPOC :- isPOC(Index).
```

Valid Treatment. The last condition for candidate treatments to be valid is that they are POC free:

```
:∼ existsPOC.
```

Here we use a new ASP construct called *weak constraints* first introduced in DLV. This weak constraint means that there should not be `existsPOC` in any answer set if it is possible at all. However, `existsPOC` is allowed to be in an answer set if there is no other choice.

Valid Modified Treatments In this part, we present the ASP rules that apply mitigation operators to eliminate the points of contention between two candidate treatments.

Represent an MO. For every mitigation operator of the form (bD, tD, POC={a1,...,ak}, LHS={aL1,...,aLn}, RHS={aR1,...,aRm}, toBeRemoved={a11, ..., a1i}), we associate a unique identifier `id` for it, which is represented by the fact: `moId(id)`. The base disease and target disease in the MO with `id` are represented by: `moBD(id, bD)` and `moTD(id, tD)` respectively. For every action in the POC of the MO with `id`, we have `moPOC(id, a)`. For every action literal `aL` of LHS and `aR` of RHS of the MO with `id`, we have `moLHS(id, aL)` and `moRHS(id, aR)`. For every action `a` in toBeRemoved of the MO with `id`, we have `moToBeRemoved(id, a)`.

A Relevant MO. We use a method similar to that for active POC to define relevant MO's. The rules are omitted here. We use `relevant(I, ID)` to denote that the active POC with id `I` is relevant to the MO with id `ID`.

Applicability of an MO. Atom `applicable(I, ID)` denotes that an MO with id `ID` is applicable to a POC with id of `I`. The method for defining active POC can be used to define the applicability too. Rules again are not included here due to lack of space.

Generate MO's to Address POC. Let atom `applyMO(I, ID)` denote that the MO with id `ID` will be applied to mitigate the POC with id `I`. It is defined by

```
1{applyMO(I, ID): applicable(I, ID)}1 :- isPOC(I).
```

The new ASP construct `1{applyMO(I, ID): applicable(I, ID)}1` means that for a POC I, we may choose to apply any applicable MO to the POC with id I.

Apply an MO to the Candidate Treatments. Since we do not know the POC beforehand, our generator will "guess" an MO to apply to the POC if there is any. Atom `applyMO(I, ID)` denotes that the MO with id `ID` will be applied to mitigate the POC with id `I`. We use `modifiedTreatment(D, A)` to denote that `A` is an action for the disease `D` after applying the MOs. By the definition of modified treatment, we have the rule for the modified treatment for the target disease (rules for base disease are omitted):

```
modifiedTreatment(TD, Action) :-
      actionInH (TD, Action),
      applyMO(Index, MOID),
      moTD(MOID, TD),
      not moToBeRemoved(MOID, Action).
```

Valid Modified Treatments. Similarly to the definition of the POCs of candidate treatments, we can write similar rules to define POCs between the modified treatments. Rules are omitted here. Let `existsPOC_M` denote the existence of POC between modified treatments. To have free POCs between modified treatments, we need rule:

```
:- existsPOC_M.
```

Proposition 3. Given patient information I of a patient with diseases d_1 and d_2, let Π be the program obtained from the discussion above. Assume there is no valid treatment for d_1 and d_2. T_1' and T_2', without any POC between them, are the modified treatment resulted from the application of some MOs to some candidate treatments T_1 and T_2 which agree with I, if and only if there is an answer set S of Π such that $T_1' = \{a : modifiedTreatment(d_1, a)\} \in S$, and $T_2' = \{a : modifiedTreatment(d_2, a) \in S\}$.

5 Evaluation

We have implemented the proposed ASP approach using DLV. A major reason to use DLV, instead of other ASP solvers, is its capacity to represent weak constraints which are convenient for this application. However, since DLV does not support choice rules yet, in our implementation, we have translated the choice rules using epistemic disjunctions. The translation technique is well known in the ASP community [21].

The program consists of two parts. The first part is the general knowledge, shared by all CPGs, to generate candidate treatments, identify POC's, and find relevant and applicable MO's, if there is a POC, and apply them to obtain a valid modified treatment.

The second part consists of the representation of activity graphs of the CPGs, MO's, and patient information.

The DLV implementation of the ASP solution is straightforward. In contrast, we are not aware of experimental results in the existing work.

To evaluate the program, we consider the CPGs for duodenal ulcer (DU) and transient ischemic attack (TIA) that are used by Wilk et al in [12]. These CPGs (Fig 1) include only the crucial actions and decision nodes of the guidelines published by the National Institute for Health and Clinical Excellence, UK (NICE) [10].

As for mitigated operators for patients with both DU and TIA, we use MO1 and MO2, in the section of the definition of the reconciling problem, in our implementation.

We assume a patient has both conditions of DU and TIA. We consider two scenarios based on the report in [10]. The first is that the patient has a positive result for the H. Pylori test, negative result for hypoglycemia test, and negtive result for the FAST test. In this scenario, there is no POC. Our program output one valid treatment (by guessing a result for decision nodes whose result is unknown) for DU: {`sa (stop aspirin if used), et (start eradication therapy), sc (self care)`}, and one valid treatment for TIA:

{`pcs (refer to primary care specialist)`}.

In the second scenario, some adverse interaction is present. The patient has a negative result for the H. pylori test, negative result for the hypoglycemia test, positive result for the FAST test, and has had neurological symptoms resolved. In this scenario, there is an adverse interaction between the actions of *stopping aspirin* and *starting aspirin*. Some relevant mitigation operator has to be employed to find a new valid treatment. A valid treatment found by our program for DU is {`ppi (start PPI), sc (self care)`}, and that for TIA is {`cl (clopidogrel), pcs (refer to primary care specialist)`} where aspirin is replaced by clopidogrel.

We run the program on a Sony Vaio laptop with Intel i5 CPU at 2.53GHz, 4GB memory and Windows 7. The DLV solver we used is the version of *build BEN/Dec 21 2011*. The real time to run the above two scenarios is 0 second (i.e., not detectable by DLV solver).

The program together with the two scenarios is downloadable at `http://redwood.cs.ttu.edu/~yzhang/temp/KR-14/code-coMorbidity-dlv.lp`.

6 Related Work and Conclusion

We note some recent work on the study of treatment of comorbid patients. The first one by Riano and Collado [22] focuses on using rules to represent the MO's and acquiring these rules for Hypertension, Diabetes Mellitus and Heart Failure. In both Wilk et al's and our work, we assume the MO's are given. The second one is by Lopez-Vallverdu et al. [23]. They propose a model combining treatments based on the seriousness, evolution and acuteness of the patients' condition and examine a specific case for Hypetension and Heart Failure. However, they did not cover POC's and their mitigation.

The main ideas underlying the reported work here are from Wilk et al.'s work in identifying the point of contention between two treatments and employing MO's to mitigate the contention [10,12].

The major difference between our work and Wilk et al.'s lies in the separation of the definition of the problem from programming languages and algorithms. Specifically, we present a mathematical definition of the problem of mitigating the point of contention that may occur in treatments for two diseases when two CPGs for these diseases are used. We then offer a purely declarative ASP based solution which naturally models the original problem. The major advantages of our proposal is as follows. First, it is more accessible for the medical researchers to evaluate how closely the defined problem models the real problems involved in comorbidities. Second, the formal definition of the problem allows the discussion of the correctness of the proposed solutions. Our ASP based solution facilitates the proof of its correctness. Thirdly, our ASP based solution will benefit from the the well developed and maintained efficient ASP solvers. Fourthly, some key issues, such as dealing with more parallel paths, raised by Wilk et al. [10] can be addressed in a natural way by our approach. For example, the parallel paths problem has been addressed in our current problem definition and solution.

The proposed ASP based solution is easy to implement and efficient to address some scenarios reported in the literature. Clearly the current definition of the reconciling problem does not include the temporal information and how to balance the treatment to maxmaximize the patients' outcome. In the next step, we will work with medical professionals to refine the problem definition to better reflect the real practice in solving the problems related to comorbidity issues. We also plan to write ASP program for a complete CPG, which will help us further understand the limitations of the ASP approach and investigate how to address those challenges.

Acknowledgment

We would like to thank Michael Gelfond and Samson Tu for discussions on this subject. Yuanlin Zhang's work is partially supported by the NSF grants IIS-1018031 and CNS-1359359. Zhizheng Zhang's work is partially supported by Project 60803061 and 61272378 sponsored by National Natural Science Foundation of China, and Project BK2008293 by Natural Science Foundation of Jiangsu.

References

1. Jakovljević, M., Ostojić, L.: Comorbidity and multimorbidity in medicine today: challenges and opportunities for bringing separated branches of medicine closer to each other. Psychiatr Danub **1** (2013) 18–28
2. Field, M.J., Lohr, K.N., et al.: Guidelines for Clinical Practice:: From Development to Use. National Academies Press (1992)
3. Peleg, M., Tu, S., Bury, J., Ciccarese, P., Fox, J., Greenes, R.A., Hall, R., Johnson, P.D., Jones, N., Kumar, A., et al.: Comparing computer-interpretable guideline models: a case-study approach. Journal of the American Medical Informatics Association **10**(1) (2003) 52–68
4. de Clercq, P.A., Blom, J.A., Korsten, H.H., Hasman, A.: Approaches for creating computer-interpretable guidelines that facilitate decision support. Artificial intelligence in medicine **31**(1) (2004) 1–27

5. Isern, D., Moreno, A.: Computer-based execution of clinical guidelines: a review. International journal of medical informatics **77**(12) (2008) 787–808
6. Boyd, C.M., Darer, J., Boult, C., Fried, L.P., Boult, L., Wu, A.W.: Clinical practice guidelines and quality of care for older patients with multiple comorbid diseases. JAMA: the journal of the American Medical Association **294**(6) (2005) 716–724
7. Sittig, D.F., Wright, A., Osheroff, J.A., Middleton, B., Teich, J.M., Ash, J.S., Campbell, E., Bates, D.W.: Grand challenges in clinical decision support. Journal of biomedical informatics **41**(2) (2008) 387–392
8. Real, F., Riano, D.: An autonomous algorithm for generating and merging clinical algorithms. In: Knowledge Management for Health Care Procedures. Springer (2009) 13–24
9. Abidi, S.R., Abidi, S.S.R.: Towards the merging of multiple clinical protocols and guidelines via ontology-driven modeling. In: Artificial Intelligence in Medicine. Springer (2009) 81–85
10. Wilk, S., Michalowski, W., Michalowski, M., Farion, K., Hing, M.M., Mohapatra, S., et al.: Mitigation of adverse interactions in pairs of clinical practice guidelines using constraint logic programming. Journal of biomedical informatics **46**(2) (2013) 341–353
11. Michalowski, M., Hing, M.M., Wilk, S., Michalowski, W., Farion, K.: A constraint logic programming approach to identifying inconsistencies in clinical practice guidelines for patients with comorbidity. In: Artificial Intelligence in Medicine. Springer (2011) 296–301
12. Wilk, S., Michalowski, M., Michalowski, W., Hing, M.M., Farion, K.: Reconciling pairs of concurrently used clinical practice guidelines using constraint logic programming. In: AMIA Annual Symposium Proceedings. Volume 2011., American Medical Informatics Association (2011) 944
13. McCarthy, J.: Elaboration tolerance. In: Common Sense. Volume 98., Citeseer (1998)
14. Gelfond, M., Kahl, Y.: Knowledge Representation, Reasoning, and the Design of Intelligent Agents. Manuscript (2013)
15. Faber, W., Pfeifer, G., Leone, N., Dell'armi, T., Ielpa, G.: Design and implementation of aggregate functions in the dlv system. Theory Pract. Log. Program. **8**(5-6) (November 2008) 545–580
16. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artif. Intell. **187-188** (August 2012) 52–89
17. Tu, S.W., Campbell, J.R., Glasgow, J., Nyman, M.A., McClure, R., McClay, J., Parker, C., Hrabak, K.M., Berg, D., Weida, T., et al.: The sage guideline model: achievements and overview. Journal of the American Medical Informatics Association **14**(5) (2007) 589–598
18. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proceedings of ICLP-88. (1988) 1070–1080
19. Mileo, A., Merico, D., Pinardi, S., Bisiani, R.: A logical approach to home healthcare with intelligent sensor-network support. The Computer Journal **53**(8) (2010) 1257–1276
20. Nogueira, M., Balduccini, M., Gelfond, M., Watson, R., Barry, M.: An A-Prolog decision support system for the Space Shuttle. In Provetti, A., Son, T.C., eds.: Answer Set Programming: Towards Efficient and Scalable Knowledge Representation and Reasoning. AAAI 2001 Spring Symposium Series (Mar 2001)
21. Baral, C.: Knowledge Representation, Reasoning, and Declarative Problem Solving. Cambridge University Press (Jan 2003)
22. Riaño, D., Collado, A.: Model-based combination of treatments for the management of chronic comorbid patients. In: Artificial Intelligence in Medicine. Springer (2013) 11–16
23. López-Vallverdú, J.A., Riaño, D., Collado, A.: Rule-based combination of comorbid treatments for chronic diseases applied to hypertension, diabetes mellitus and heart failure. In: Process Support and Knowledge Representation in Health Care. Springer (2013) 30–41