

SPECULATIVE COMPUTATION WITH CONSTRAINT PROCESSING FOR THE GENERATION OF CLINICAL SCENARIOS

Tiago Oliveira¹, Ken Satoh², Paulo Novais¹ and José Neves¹

¹*Department of Informatics/CCTC, University of Minho, Braga, Portugal*
{toliveira, pjon, jneves}@di.uminho.pt

²*National Institute of Informatics, Sokendai University, Tokyo, Japan*
ksatoh@nii.ac.jp

ABSTRACT

Clinical decision making often involves making decisions in situations of uncertainty. Clinical Decision Support Systems are tools devised to help in such moments, but the information may not be available during the decision process. Be it because of communication failure or errors in data input, the truth is that it would be beneficial to present the most likely clinical scenarios to a physician, given the incompleteness of the information. Speculative Computation offers a way to structure such a scenario generation process. This work presents a framework for clinical decision support with disjunctive constraint processing that acts as an interface with computer-interpretable versions of Clinical Practice Guidelines. Being a reasoning process based on defaults, it has to rely on a default generation process. For that we propose Bayesian Networks. The interaction between the different components of the system resulted in a process capable of generating clinical scenarios.

KEYWORDS

Computer-Interpretable Guidelines, Logic Programming, Speculative Computation, Clinical Decision.

1. INTRODUCTION

In clinical decision making, there may be situations in which uncertainty is present because of missing key patient information such as demographics, episodic and clinical diagnosis detail. In Clinical Decision Support Systems (CDSSs), the above-mentioned concerns can translate into a poor data input into the Electronic Health Record (EHR) system or a communication failure between the EHR system and the CDSS, making impossible the timely retrieval of the information for the decision (Sittig et al. 2008). The inability of CDSSs to deal with uncertainty calls forth the need to create new clinical decision support functionalities such as reasoning mechanisms with predictive capabilities. The objective of these prediction operations is to produce scenarios for a physician. Moreover, such mechanisms would enable the physician to take pre-emptive measures, in the event that the default reasoning suggests the clinical process is following an undesirable direction. The work presented herein describes the application of a framework for Speculative Computation (Satoh et al. 2003; Ceberio et al. 2006) to the generation of clinical scenarios, using default constraints. It acts as an interface to machine-readable versions of Clinical Practice Guidelines (CPGs). The foundations for Speculative Computation are provided by Logic Programming, and constraint processing is used in order to handle different types of rules and answers manipulated in the procedures. The paper is organized as follows. Section 2 provides a brief description of related work. Section 3 contains a summary of the CompGuide model for CPGs and a description of the case study used to demonstrate the application of Speculative Computation. The definition of the framework is provided in section 4, along with an execution example. Finally, section 5 contains some conclusions about the work so far and future directions.

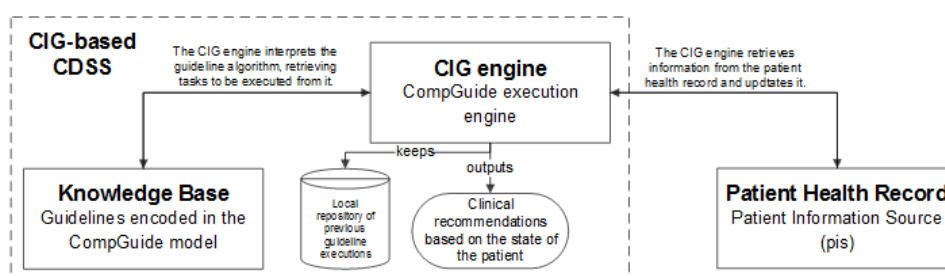
2. RELATED WORK

Computer-Interpretable guidelines (CIGs) are machine-readable versions of CPGs for CDSSs. The representation of CPGs in the form of networks of tasks is, arguably, the most followed paradigm in CIG design. Good examples of this are the models Asbru, PROforma, GLIF3, and SAGE (Peleg 2013). Each one of these models has an associated execution engine that interprets guideline algorithms and runs them against patient information in EHRs. The systems in which these execution engines are included typically follow the architecture presented in Fig. 1. Most of the existing examples provide straightforward reasoning only when all the necessary information for the verification of conditions is gathered. There is no treatment of incomplete information for cases in which they are unable to retrieve the information from the patient health record. There are different techniques for handling uncertainty stemming from classical logic and probability theory. The most notable variations of classical logic developed to address this problem include Default Reasoning, Multivalued Logic and Autoepistemic Logic (Sheridan 1991). Despite their theoretical soundness, they have not been applied much to the medical field. On the other hand, numerical techniques such as Bayesian Probabilities, Certainty Factors, Dempster-Shafer Theory, and Fuzzy Logic (Sheridan 1991) have examples, though not very expressive, of applications in the field (Straszecka 2006). As stated, the type of uncertainty addressed in this work is related with missing information at guideline execution by the CIG engine. Here, guidelines provide solid rules to make inference and guide the process.

3. COMPUTER-INTERPRETABLE GUIDELINES: STAGING OF COLON CANCER

The CompGuide project for CIGs (please consult (Oliveira et al. 2014) for more details) includes a task network model represented in Ontology Web Language (OWL). In the model, the procedures of guidelines are encoded as instances of the following four primitive classes: *Plans*, *Actions*, *Questions*, and *Decisions*. To show how Speculative Computation is used, we will resort to a simple example. It was extracted from the Clinical Practice Guideline in Oncology for Colon Cancer from the National Comprehensive Cancer Network (Benson et al. 2013). The treatment of colon cancer was chosen as the domain, and staging will be the basis for the example. Colon cancer staging is an assessment of the cancer's pathological stage. This information is expressed in terms of three parameters of the TNM classification system (Benson et al. 2013). These parameters are: primary tumor (T), which indicates the morphology and degree of tumor invasion; regional lymph nodes (N), a parameter used for expressing the number of metastases in lymph nodes; and distant metastasis (M), which are the metastases found in other organs or sites. Staging is the step that determines which kind of adjuvant therapy the patient needs, which may include different types of chemotherapy regimens. Table 1 shows the process of choosing the most adequate adjuvant therapy in the form of different tasks according to the CompGuide ontology. There may be situations in which the CompGuide engine is unable to retrieve the information from the patient's EHR, or the information is not in the EHR yet. As such, the case-study is based on the following assumptions: (1) the execution engine will recommend the next task in the clinical workflow; (2) the transition from one task to another is only possible if the first is connected to the latter through the *hasAlternativeTask* property; (3) to move to one of the alternative tasks, the trigger conditions of such task must be met; (4) the information necessary to verify the trigger conditions will be acquired from an external information source, the EHR, here referred to as *pis*, but may not be available; and (6) the system has a set of default constraints about the information of the patient's state.

Figure 1. The conceptual structure of the CompGuide CDSS and the interaction between its basic components: knowledge base, patient information source, and CIG engine.



4. SPECULATIVE COMPUTATION WITH DISJUNCTIVE CONSTRAINT PROCESSING

Speculative Computation (Sato et al. 2003; Ceberio et al. 2006) is a dynamically revisable computation. Using a set of default constraints for the missing information, a tentative computation is made. This computation produces the most likely clinical scenarios for a physician. Clinical parameters either take the form of nominal variables, with multiple categories, or numerical variables. This means that it is necessary to express and process conditions in terms of set constraints, and equality and inequality constraints.

4.1 Generation of Default Constraints

The process for extracting defaults has to be data-driven in order to generate default constraints capable of producing the most likely scenarios through Speculative Computation. In this way, the objective is to use data from past executions of guidelines stored in the CDSS. Following the example provided in section 3.2, we used a data set containing information about the TNM parameters of 518 patients who underwent colon cancer treatment at the hospital of Braga and followed the above-mentioned guideline. The generation of defaults also has to account for the existence of dependences between the parameters whose values we want to know. Bayesian Networks (BNs) model the statistical dependences and independences between variables and provide a probability distribution for them (in this case $P(T,N,M)$). As such, they fit the requirements demanded by the problem at hand. The process works as follows. The execution engine analyses the task selection moments in a guideline represented in CompGuide and for each one it builds two BNs based on a reference data set for that particular situation. The BNs are constructed with two different algorithms standing for two different forms of score-based network learning. The Hill-Climbing algorithm is used for score-based learning; and the Max-Min Hill-Climbing algorithm is used as a hybrid algorithm (including score-based and constraint-based learning) (Scutari 2010). The Akaike Information Criterion (AIC) is used in the process to measure the relative quality of the learned networks. The results of the process can be seen in Table 2. Based on the AIC, the best network is chosen. In this case the choice falls upon network 1. The defaults are obtained through a Maximum a Posteriori Probability (MAP) (Korb & Nicholson 2003) query which provides the most likely setting for the variables in the network, based on their conditional dependences. The MAP query is set with Hill-Climbing as the search method and the the Maximum Likelihood (ML) estimation is defined for the initialization. This whole process is automated and integrated in the execution engine. The code in Java runs an R instance through the Java/R Interface (JRI) API (available at <http://rforge.net/JRI/>) in order to use the learning capabilities of the *bnlearn* package for R. The networks and respective scores are passed on to the Java code which, in turn, handles the choice of the network and the MAP query using the inference library API of the SamIam project (available at <http://reasoning.cs.ucla.edu/samiam/>). The values for the variables are then stored for when they are needed at guideline execution. Network 1 has a structure with serial arcs which shows that T and M are conditionally independent given N. This means that instantiating N will block the flow of probabilities from T to M, thus making these two clinical parameters independent. The result of the MAP query on network 1 reveals that the most likely setting is: $T=t3$, $N=n2$, and $M=m1$. The corresponding probability is $P(MAP/e) \approx 0.4399$.

4.2 Framework and Definitions

During guideline execution, the execution engine runs a Prolog instance through the JPL API of SWI and maps the trigger conditions and the information of task alternatives to predicates, along with the default values obtained from the default generation process. Based on the work of (Sato et al. 2003), a Framework of Speculative Computation in Clinical Decision Support Systems with disjunctive constraint processing SF_{CDSS} is defined in terms of the following tuple $(\Sigma, \mathcal{E}, \Delta, P)$ where: Σ is a finite set of constants, an element in Σ is a system component identifier; \mathcal{E} is a set of predicates called external predicates, when Q is an atom

with an external predicate and S is the identifier of a remote information source, $Q@S$ is called an askable atom; Δ is the default answer set and consists in a set of default rules called default rules w.r.t. (w.r.t. is used as an abbreviation of with respect to) $Q@S$, of the following form: $Q@S \leftarrow C ||$, where $Q@S$ is an askable atom and C is a set of constraints called default constraints for $Q@S$, a default rule w.r.t. $Q@S$ is denoted as $\sigma(Q@S)$; and P is a constraint logic program of the form: $H \leftarrow C || B_1, B_2, \dots, B_n$, where H is a positive ordinary literal called a head of rule R , denoted as $head(R)$, C is a set of constraints called body constraints of R , denoted as $const(R)$, with the possibility of $const(R) = \emptyset$, and each B_1, B_2, \dots, B_n is an ordinary literal, or an askable literal referred to as body of R , denoted as $body(R)$, with the possibility of $body(R) = \emptyset$.

Table 1. The clinical parameters of the TNM classification in the form of trigger conditions along with the representation of the assessment procedure in the CompGuide ontology.

Trigger Conditions	T (t)	N (n)	M (m)	Task
A	tis or t0	--	--	Action 1
B	t1 or t2	n0	m0	Action 1
C	t3	n0	m0	Action 2
D	t4	n0	m0	Action 3
E	t1 or t2 or t3 or t4	n1 or n2	m0	Action 4
F	t1 or t2 or t3 or t4	n0 or n1 or n2	m1	Action 5

Graphical Representation

The representation of the case-study in the framework is provided below. The predicate $nt(a, b)$ means that b is the task that follows a and is used in the initial query. $alt(a, b)$ indicates that b is an alternative task linked to a . $tcv(b)$ means that the trigger conditions for task b are validated. The default constraints for the clinical parameters in Δ . The remaining abbreviations are in accordance with the description of the case-study:

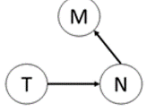
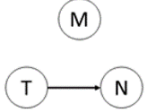
- $\Sigma = \{pis\}$;
- $\mathcal{E} = \{t, n, m\}$;
- Δ is the following set of rules:
 $t(T)@pis \leftarrow T \in \{t3\} ||$.
 $n(N)@pis \leftarrow N \in \{n2\}$.
 $m(M)@pis \leftarrow M \in \{m1\} ||$.
- P is the following set of rules:
 $nt(X, F) \leftarrow || alt(X, F), tcv(F)$.
 $tcv(F) \leftarrow F \in \{action1\}, T \in \{tis, t0\} || t(T)@pis$.
 $tcv(F) \leftarrow F \in \{action1\}, T \in \{t1, t2\}, N \in \{n0\}, M \in \{m0\} || t(T)@pis, n(N)@pis, m(M)@pis$.
 $tcv(F) \leftarrow F \in \{action2\}, T \in \{t3\}, N \in \{n0\}, M \in \{m0\} || t(T)@pis, n(N)@pis, m(M)@pis$.
 $tcv(F) \leftarrow F \in \{action3\}, T \in \{t4\}, N \in \{n0\}, M \in \{m0\} || t(T)@pis, n(N)@pis, m(M)@pis$.
 $tcv(F) \leftarrow F \in \{action4\}, T \in \{t1, t2, t3, t4\}, N \in \{n1, n2\} \in \{m0\} || t(T)@pis, n(N)@pis, m(M)@pis$.
 $tcv(F) \leftarrow F \in \{action5\}, T \in \{t1, t2, t3, t4\}, N \in \{n0, n1, n2\}, M \in \{m1\} || t(T)@pis, n(N)@pis, m(M)@pis$.
 $alt(question1, F) \leftarrow F \in \{action1\} ||$.
 $alt(question1, F) \leftarrow F \in \{action2\} ||$.
 $alt(question1, F) \leftarrow F \in \{action3\} ||$.
 $alt(question1, F) \leftarrow F \in \{action4\} ||$.
 $alt(question1, F) \leftarrow F \in \{action5\} ||$.

Regarding the execution of the framework, a non askable atom in a goal is reduced into subgoals according to the rules above. The execution engine sends a query to the information sources and waits for the answers. Upon the return of the answer constraints, they are added to the current constraints and the execution is resumed, ending only when the empty goal is found. The following definitions provide a more detailed characterization of the framework and its inner workings. A goal has the form of $\leftarrow C || B_1, \dots, B_n$ where: C is a set of constraints and each of B_1, \dots, B_n is either an atom or an askable atom. A reply set for \mathcal{E} is a set of rules of the form $Q@S \leftarrow C ||$, where $Q@S$ is an askable atom, each argument of Q is a variable and C is a set of constraints over those variables. A reduction of a goal $\leftarrow C || B_1, \dots, B_n$ w.r.t. a constraint logic

program P , a reply set R and a subgoal B_i is a goal $\leftarrow C \parallel B'$ such that: there is a rule R in $P \cup R$ so that $C \wedge \{B_i = \text{head}(R)\} \wedge \text{const}(R)$ is consistent and $\{B_i = \text{head}(R)\}$ is a conjunction of constraints equaling the arguments of B_i and $\text{head}(R)$; $C' = C \wedge \{B_i = \text{head}(R)\} \wedge \text{const}(R)$; and $B' = \{B_1, \dots, B_{(i-1)}, B_{(i+1)}, \dots, B_n\} \cup \text{body}(R)$. A derivation of a goal $\leftarrow C \parallel B_1, \dots, B_n$ w.r.t. to a speculative computation constraint framework $\langle \Sigma, \mathcal{E}, \Delta, P \rangle$ and a reply set R is a chain of reductions $\leftarrow C \parallel B_1, \dots, B_n, \dots, \leftarrow C' \parallel \emptyset$ w.r.t. P and R where \emptyset denotes an empty goal. C' is called an answer constraint w.r.t. the goal, the framework and the reply set.

The execution of the framework happens in two phases: *process reduction phase* and *fact arrival phase*. The first is the normal execution of a program by the Speculative Computation module in the execution engine, whereas the latter is an interruption phase when an answer arrives from an information source. Whenever a choice point in the computation is reached, a new process is generated. Hence, each process represents an alternative way of computation, an alternative scenario. In *process reduction*, an active process set is reduced into a new process set. The computation starts with default constraints until the real answers arrive. If there is a constraint inconsistent with the default, the respective process is suspended. If there is consistency, the process remains active. When an answer arrives, if it entails the default, the process is further reduced. Otherwise, it is suspended. In the same way, suspended processes may be resumed if their constraints are entailed by the answers. The following definitions provide more detail about the characteristics of processes. A process is a tuple $\langle \leftarrow C \parallel GS, OD \rangle$ in which: $\leftarrow C \parallel$ is a set of constraints, GS is a set of literals to be proved, called a goal set, and expresses the current status of an alternative computation; OD is a set of askable atoms, called outside defaults, and represents the assumed information about the outside world. An active process is a tuple $\langle \leftarrow C \parallel GS, OD \rangle$. A suspended process is a tuple $\langle SG, \leftarrow C \parallel GS, OD \rangle$, where SG is an askable atom called a suspended atom. A current belief state CBS is a set of rules of the form $Q @ S \leftarrow C \parallel$. It contains the system's belief of the current status of the outside world. At the beginning of the computation, the CBS assumes the constraints in the default answer set Δ . Let $\langle \leftarrow C \parallel GS, OD \rangle$ be a process and CBS be a current belief state. A process is active with respect to CBS if $C \subseteq CBS$. A process is suspended with respect to CBS otherwise. APS is a set of active processes. SPS is a set of suspended processes. Already asked questions AAQ is a set of askable atoms. AAQ is used to avoid asking redundant questions to information sources. Finally, returned facts RF are a set of rules of the form: $Q @ S \leftarrow C \parallel$ where $Q @ S$ is an askable atom and C is a set of constraints. It represents the answers from the information sources.

Table 2. Results of the BN learning process for the extraction of defaults.

	Network	Algorithm	AIC score	P(T,N,M)
1		Hill-Climbing	-962.6823	P(T).P(N T).P(M N)
2		Max-Min Hill-Climbing	-964.9433	P(T).P(N T).P(M)

4.3 Process Reduction Phase

During the *process reduction phase*, changes occur in the process set. In the following description of this phase, changed APS , SPS , AAQ , CBS and RF are represented as $NewAPS$, $NewSPS$, $NewAAQ$, $NewCBS$ and $NewRF$. The steps for process reduction are:

- **Initial Step:** Let GS be an initial goal set. $\langle \leftarrow C \parallel GS, \emptyset \rangle$ is given to the proof procedure. That is, $APS = \{ \langle \leftarrow C \parallel GS, \emptyset \rangle \}$. Let $SPS = AAQ = RF = \emptyset$ and $CBS = \Delta$.
- **Iteration Step:** Do the following:
 - Case 1:** If there is an active process $\langle \leftarrow C \parallel \emptyset, OD \rangle$ with respect to CBS in APS , then output constraints C and return outside defaults OD .
 - Case 2:** Select an active process $\langle \leftarrow C \parallel GS, OD \rangle$ from APS with respect to CBS and select an atom L in GS . Let $APS' = APS - \{ \langle \leftarrow C \parallel GS, OD \rangle \}$ and $GS' = GS - \{L\}$. For the selected atom L , do the following:
 - Case 2.1:** If L is a non-askable atom $NewPS = APS' \cup \{ \langle \leftarrow (C \wedge \{B_i = \text{head}(R)\}) \wedge \text{const}(R) \parallel (\text{body}(R) \cup GS'), OD \rangle \mid C \wedge \{B_i = \text{head}(R)\} \wedge \text{const}(R) \text{ is consistent} \}$.

Case 2.2: If L is an askable atom, $Q@S$, then

- if $L \notin AAQ$, then send the question Q to the system component S and $NewAAQ = AAQ \cup \{L\}$.
- if $L \in OD$, then $NewAPS = APS' \cup \{\leftarrow C \parallel GS', OD\}$.
- else if $(L \leftarrow C_r \parallel) \in RF$, then if $C \wedge C_r$ is consistent then $NewAPS = APS' \cup \{C \wedge C_r \parallel GS', OD\}$, else $NewAPS = APS'$.
- else if a default constraint C_d exists then,
 - if $C \wedge C_d$ is consistent then $NewAPS = APS' \cup \{\leftarrow C \wedge C_d \parallel GS', OD \cup \{L\}\}$, else $NewAPS = APS'$.
 - if $C \wedge \neg C_d$ is consistent then $NewSPS = SPS \cup \{L, \leftarrow \alpha \parallel GS', OD\}$ where $C \wedge \neg C_d \neq \alpha$.

4.4 Fact Arrival Phase

Supposing a constraint is returned from an information source S for a question $Q@S$. The returned constraint is passed on to Prolog. The returned constraint is denoted as $Q@S \leftarrow C \parallel$. Then, after a step of process reduction is finished, the following procedures should be followed:

- $NewRF = RF \cup \{Q@S \leftarrow C_r \parallel\}$
- If a default constraint C_d for $Q@S$ exists then:
 - If C_r entails C_d :
 - $NewAPS = APS - DeletedAPS \cup AddedAPS$ where $DeletedAPS = \{\leftarrow C \parallel GS, OD\} \in APS \mid Q@S \in OD\}$ and $AddedAPS = \{\leftarrow (C \wedge C_r \parallel GS, OD) \mid \leftarrow C \parallel GS, OD\} \in DeletedAPS$ and $C \wedge C_r$ is consistent.
 - $NewSPS = SPS - DeletedSPS \cup AddedSPS$ where $DeletedSPS = \{\leftarrow SG, \leftarrow (C \parallel GS, OD) \in SPS \mid SG = Q@S \text{ or } Q@S \in OD\}$ and $AddedSPS = \{\leftarrow SG, \leftarrow (C \wedge C_r \parallel GS, OD) \mid \leftarrow SG, \leftarrow C \parallel GS, OD\} \in DeletedSPS$ and $Q@S \in OD$ and $C \wedge C_r$ is consistent.
 - If C_r contradict C_d :
 - $NewAPS = APS - DeletedAPS \cup ResumedSPS$ where $DeletedAPS = \{\leftarrow C \parallel GS, OD\} \in APS \mid Q@S \in OD\}$ and $ResumedSPS = \{\leftarrow (C \wedge C_r) \parallel GS, OD\} \mid \{Q@S, \leftarrow C \parallel GS, OD\} \in SPS$ and $C \wedge C_r$ is consistent.
 - $NewSPS = SPS - DeletedSPS$ where $DeletedSPS = \{\leftarrow SG, \leftarrow C \parallel GS, OD\} \in SPS \mid SG = Q@S \text{ or } Q@S \in OD\}$.
 - If C_r does not entail C_d nor contradicts C_d :
 - $NewAPS = APS - DeletedAPS \cup ResumedSPS$ where $DeletedAPS = \{\leftarrow C \parallel GS, OD\} \in APS \mid Q@S \in OD\}$ and $AddedAPS = \{\leftarrow (C \wedge C_r) \parallel GS, OD\} \mid \leftarrow C \parallel GS, OD\} \in DeletedAPS$ and $C \wedge C_r$ is consistent} and $ResumedSPS = \{\leftarrow (C \wedge C_r) \parallel GS, OD\} \mid \{Q@S, \leftarrow C \parallel GS, OD\} \in SPS$ and $C \wedge C_r$ is consistent}.
 - $NewSPS = SPS - DeletedSPS \cup AddedSPS$ where $DeletedSPS = \{\leftarrow SG, \leftarrow (C \wedge C_r) \parallel GS, OD, IA\} \in SPS \mid SG = Q@S \text{ or } Q@S \in OD\}$ and $AddedSPS = \{\leftarrow SG, \leftarrow (C \wedge C_r) \parallel GS, OD, IA\} \mid \leftarrow SG, \leftarrow C \parallel GS, OD, IA\} \in DeletedSPS$ and $Q@S \in OD$ and $C \wedge C_r$ is consistent.

4.5 Execution Example

When an atom is reduced, new processes are created by following the rule order unifiable with the atom. The selection of the atoms for reduction is done by a newly created or a newly resumed process, and a left-most atom. The following are the steps of an execution trace for $nt(question1, F)$, a query to obtain the task following $Question1$ in the clinical workflow:

1. **Initial Step:**
 $APS = \{\{\leftarrow \parallel nt(question1, F)\}, \emptyset\}$
 $SPS = \emptyset$
 $AAQ = \emptyset$
 $RF = \emptyset$
 $CBS = \{t(T)@pis \leftarrow T \in \{t3\} \parallel, n(N)@pis \leftarrow N \in \{n2\}, m(M)@pis \leftarrow M \in \{m1\}\}$
2. **By Case 2.1:**
 $APS = \{\{\leftarrow \parallel alt(question1, F), tcv(F)\}, \emptyset\}$
3. **By Case 2.1:**
 $APS = \{\{\leftarrow F \in \{action1\} \parallel \parallel tcv(F)\}, \emptyset\},$
 $\{\{\leftarrow F \in \{action2\} \parallel \parallel tcv(F)\}, \emptyset\},$
 $\{\{\leftarrow F \in \{action3\} \parallel \parallel tcv(F)\}, \emptyset\},$
 $\{\{\leftarrow F \in \{action4\} \parallel \parallel tcv(F)\}, \emptyset\},$
 $\{\{\leftarrow F \in \{action5\} \parallel \parallel tcv(F)\}, \emptyset\}$
4. **By Case 2.1:**
 $APS = \{\{\leftarrow F \in \{action1\}, T \in \{tis, t0\} \parallel \parallel t(T)@pis\}, \emptyset\},$

- $\{\leftarrow F \in \{action1\}, T \in \{t1, t2\}, N \in \{n0\}, M \in \{m0\} \mid \underline{t(T)@pis}, n(N)@pis, m(M)@pis\}, \emptyset$
 $\{\leftarrow F \in \{action2\}, T \in \{t3\}, N \in \{n0\}, M \in \{m0\} \mid \underline{t(T)@pis}, n(N)@pis, m(M)@pis\}, \emptyset$,
 $\{\leftarrow F \in \{action3\}, T \in \{t4\}, N \in \{n0\}, M \in \{m0\} \mid \underline{t(T)@pis}, n(N)@pis, m(M)@pis\}, \emptyset$,
 $\{\leftarrow F \in \{action4\}, T \in \{t1, t2, t3, t4\}, N \in \{n1, n2\} \in \{m0\} \mid \underline{t(T)@pis}, n(N)@pis, m(M)@pis\}, \emptyset$,
 $\{\leftarrow F \in \{action5\}, T \in \{t1, t2, t3, t4\}, N \in \{n0, n1, n2\}, M \in \{m1\} \mid \underline{t(T)@pis}, n(N)@pis, m(M)@pis\}, \emptyset$
5. **By Case 2.2:**
 $t(T)@pis$ is asked to pis and since $((t(T)@pis \leftarrow T \in \{t3\}) \in \Delta)$:
 $APS = \{\{\leftarrow F \in \{action2\}, T \in \{t3\}, N \in \{n0\}, M \in \{m0\} \mid \underline{n(N)@pis}, m(M)@pis\}, \{t(T)@pis\}\}$,
 $\{\leftarrow F \in \{action4\}, T \in \{t3\}, N \in \{n1, n2\} \in \{m0\} \mid \underline{n(N)@pis}, m(M)@pis\}, \{t(T)@pis\}\}$,
 $\{\leftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n0, n1, n2\}, M \in \{m1\} \mid \underline{n(N)@pis}, m(M)@pis\}, \{t(T)@pis\}\}$
 $SPS = \{\{\underline{t(T)@pis}, \leftarrow F \in \{action1\}, T \in \{t1, t2\}\} \mid \emptyset, \emptyset\}^{*(P1)}$,
 $\{\underline{t(T)@pis}, \leftarrow F \in \{action1\}, T \in \{t1, t2\}, N \in \{n0\}, M \in \{m0\} \mid n(N)@pis, m(M)@pis\}, \emptyset\}^{*(P2)}$,
 $\{\underline{t(T)@pis}, \leftarrow F \in \{action3\}, T \in \{t4\}, N \in \{n0\}, M \in \{m0\} \mid n(N)@pis, m(M)@pis\}, \emptyset\}^{*(P3)}$,
 $\{\underline{t(T)@pis}, \leftarrow F \in \{action4\}, T \in \{t1, t2, t4\}, N \in \{n1, n2\}, M \in \{m0\} \mid n(N)@pis, m(M)@pis\}, \emptyset\}^{*(P4)}$,
 $\{\underline{t(T)@pis}, \leftarrow F \in \{action5\}, T \in \{t1, t2, t4\}, N \in \{n0, n1, n2\}, M \in \{m1\} \mid n(N)@pis, m(M)@pis\}, \emptyset\}^{*(P5)}$ }
 $AAQ = \{t(T)@pis\}$
6. **By Case 2.2:**
 $n(N)@pis$ is asked to pis and since $((n(N)@pis \leftarrow N \in \{n2\}) \in \Delta)$:
 $APS = \{\{\leftarrow F \in \{action4\}, T \in \{t3\}, N \in \{n2\}, M \in \{m0\} \mid \underline{m(M)@pis}\}, \{t(T)@pis, n(N)@pis\}\}$,
 $\{\leftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n2\}, M \in \{m1\} \mid \underline{m(M)@pis}\}, \{t(T)@pis, n(N)@pis\}\}$
 $SPS = \{\{\underline{n(N)@pis}, \leftarrow F \in \{action2\}, T \in \{t3\}, N \in \{n0\}, M \in \{m0\} \mid m(M)@pis\}, \{t(T)@pis\}\}^{*(P6)}$,
 $\{\underline{n(N)@pis}, \leftarrow F \in \{action4\}, T \in \{t3\}, N \in \{n1\}, M \in \{m0\} \mid m(M)@pis\}, \{t(T)@pis\}\}^{*(P7)}$,
 $\{\underline{n(N)@pis}, \leftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n0, n1\}, M \in \{m1\} \mid m(M)@pis\}, \{t(T)@pis\}\}^{*(P8)}$,
 P_1, P_2, P_3, P_4, P_5
 $AAQ = \{t(T)@pis, n(N)@pis\}$
7. **By Case 2.2:**
 $m(M)@pis$ is asked to pis and since $((m(M)@pis \leftarrow M \in \{m1\}) \in \Delta)$:
 $APS = \{\{\leftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n2\}, M \in \{m1\} \mid \emptyset\}, \{t(T)@pis, n(N)@pis, m(M)@pis\}\}\}$
 $SPS = \{\{\underline{m(M)@pis}, \leftarrow F \in \{action4\}, T \in \{t3\}, N \in \{n2\}, M \in \{m0\} \mid \emptyset\}, \{t(T)@pis, n(N)@pis\}\}^{*(P9)}$,
 $P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8$
 $AAQ = \{t(T)@pis, n(N)@pis, m(M)@pis\}$
8. $(t(T)@pis \leftarrow T \in \{t3\} \mid \mid)$ is returned from pis and by **Fact Arrival Phase**:
 CBS remains unchanged.
 $t(T)@pis$ is removed from OD in the active processes because it is no longer a default.
 $APS = \{\{\leftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n2\}, M \in \{m1\} \mid \emptyset\}, \{t(T)@pis, n(N)@pis\}\}$
 $SPS = \{P_7, P_8, P_9\}$
 P_1, P_2, P_3, P_4, P_5 are terminated.
 $RF = \{(T)@pis \leftarrow T \in \{t3\} \mid \mid\}$
9. $(n(N)@pis \leftarrow n \in \{n2\} \mid \mid)$ is returned from pis and by **Fact Arrival Phase**:
 CBS remains unchanged.
 $n(N)@pis$ is removed from OD in the active processes because it is no longer a default.
 $APS = \{\{\leftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n2\}, M \in \{m1\} \mid \emptyset\}, \{t(T)@pis\}\}$
 $SPS = \{P_9\}$
 P_7, P_8 are terminated.
 $RF = \{(T)@pis \leftarrow T \in \{t3\} \mid \mid, n(N)@pis \leftarrow n \in \{n2\} \mid \mid\}$
10. $(m(M)@pis \leftarrow m \in \{m0\} \mid \mid)$ is returned from pis and by **Fact Arrival Phase**:
 $CBS = \{(T)@pis \leftarrow T \in \{t3\} \mid \mid, n(N)@pis \leftarrow n \in \{n2\}, m(M)@pis \leftarrow m \in \{m0\} \mid \mid\}$
The process $\{\leftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n2\}, M \in \{m1\} \mid \emptyset\}, \{t(T)@pis\}$ is terminated.
Process P_9 is resumed.
 $SPS = \emptyset$
 $APS = \{\{\leftarrow F \in \{action4\}, T \in \{t3\}, N \in \{n2\}, M \in \{m0\} \mid \emptyset, \emptyset\}\}$
End of execution.

From steps 1 to 4, the goals in GS are non-askable atoms and successively unify with the head of some rule. The effects of Speculative Computation are seen throughout steps 5 to 7. For instance, in step 5, the process for *action 4* is split into two processes regarding the $t(T)@pis$ literal; one active process using the default constraint $(t(T)@pis \leftarrow T \in \{t3\} \mid \mid)$ and one suspended process using the negation of the default constraint. The same happens for the process of *action 5* in the same step. At step 7, it is possible to arrive at an active process with an empty goal set by *process reduction*. By outputting C and OD one gets an alternative computation that represents a scenario for a situation of incomplete information. In this case, the scenario suggests that the next clinical task should be *action 5* for a reasoning exclusively based on defaults. Based on this scenario, the physician may prepare for the direction the treatment process will most likely

follow. As the answers from the information source arrive, the constraints of the active processes and suspended processes are revised. If the rule structure were different, there could be more than one process with an empty *GS*. If that is the case, the execution engine presents all the active scenarios. *Fact arrival* may cause active processes to be terminated, or suspended processes to be resumed. The answers from the information sources are regarded as definitive. At steps 8 and 9, the active processes contain both returned and default constraints. The returned facts may affect the conditional probability distributions of the remaining variables, thus requiring a default revision process. Such default revision process is possible with the help of the network obtained in the default generation process, by performing a MAP query with the returned facts has evidence. The improvement of the Speculative Computation framework with default revision is one of the objectives for the work ahead.

5. CONCLUSIONS AND FUTURE WORK

The value of this proposal of Speculative Computation is in structuring a scenario generation process that can deal with incomplete information, which, as previously mentioned, may have different causes. Such a framework provides a physician with a complete layout of the most likely unfolding of the clinical process. Moreover, the use of BNs provides advantages that surpass the simple generation of defaults. They enable the extraction of a likelihood measure for the setting, which can potentially improve the role of Speculative Computation has a reasoning interface to CIGs. Simulation is a growing field in medicine. This may also be an area of application for this system. Future work includes the improvement of the framework with the possibility of revising defaults. The default generation process also needs improvement because there are different dimensions that were not considered, with time being one of them, despite their importance in clinical decision making.

ACKNOWLEDGEMENTS

This work is part-funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-028980 (PTDC/EEI-SII/1386/2012) and project PEst-OE/EEI/UI0752/2014. The work of Tiago Oliveira is supported by a doctoral grant by FCT (SFRH/BD/85291/2012).

REFERENCES

- Benson, A. et al., 2013. *NCCN Clinical Practice Guideline in Oncology Colon Cancer*, Available at: http://www.nccn.org/professionals/physician_gls/f_guidelines.asp.
- Ceberio, M., Hosobe, H. & Satoh, K., 2006. Speculative Constraint Processing with Iterative Revision for Disjunctive Answers. In T. Francesca & T. Paolo, eds. *Computational Logic in Multi-Agent Systems*. Springer, pp. 340–357.
- Korb, K.B. & Nicholson, A.E., 2003. *Bayesian artificial intelligence*, London: CRC Press.
- Oliveira, T. et al., 2014. Webifying the Computerized Execution of Clinical Practice Guidelines. In J. Bajo Perez et al., eds. *Trends in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection SE - 18*. Advances in Intelligent Systems and Computing, Springer International Publishing, pp. 149–156.
- Peleg, M., 2013. Computer-interpretable clinical guidelines: A methodological review. *Journal of biomedical informatics*, 46(4), pp.744–63.
- Satoh, K., Codognot, P. & Hosobe, H., 2003. Speculative Constraint Processing in Multi-agent Systems. In *Intelligent Agents and Multi-Agent Systems*. Springer, pp. 133–144.
- Scutari, M., 2010. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3), pp.1–22.
- Sheridan, F.K.J., 1991. A survey of techniques for inference under uncertainty. *Artificial Intelligence Review*, 5(1-2), pp.89–119. Available at: <http://www.springerlink.com/index/10.1007/BF00129537>.
- Sittig, D.F. et al., 2008. Grand challenges in clinical decision support. *Journal of biomedical informatics*, 41(2), pp.387–92.
- Straszeczka, E., 2006. Combining uncertainty and imprecision in models of medical diagnosis. *Information Sciences*, 176(20), pp.3026–3059.