Applying Speculative Computation to Guideline-based Decision Support Systems

Tiago Oliveira, José Neves and Paulo Novais
Department of Informatics/CCTC
University of Minho
Braga, Portugal
{toliveira,jneves,pjon}@di.uminho.pt

Ken Satoh
National Institute of Informatics
Sokendai University
Tokyo, Japan
ksatoh@nii.ac.jp

***Abstract*— Clinical Practice Guidelines, as evidence-based recommendations are the ideal support for Clinical Decision Support Systems. The intricacies of a guideline execution tool are related with the establishment of a careflow with an appropriate order between procedures and the modelling of decision points. One of such decision points is the choice between alternative tasks based on trigger conditions regarding a patient's state. It may be the case that, when there is the need to choose one of the alternative tasks, the system does not possess all the required information to do so, thus rendering impossible to reach an outcome. Speculative Computation and Abduction may increase the efficiency of this process by allowing the system to advance the computation of a solution, even while it is waiting for a response from the information sources. This work provides the basis for a Speculative Computation framework able to cope with decisions of clinical careflows. The methods developed herein were devised to support practitioners and to improve patient-centred medicine by providing maps of the most likely evolution of a patient, even when the information is incomplete.**

Keywords—Computer-Interpretable Guidelines, Clinical Decision Support, Speculative Com-putation, Abductive Reasoning.

I. INTRODUCTION

Currently, the use of Clinical Decision Support Systems (CDSSs) in daily health care delivery is becoming a reality [1]. There is evidence that the use of such systems can positively contribute to the improvement of health care services, namely in the prevention of medication errors, and the improvement of practitioner performance [2][3]. The main goal of these systems is to help health care professionals make decisions by dealing with clinical data and knowledge. The use of CDSSs that provide patient specific recommendations has been the object of great discussion in the last forty years, since the presentation of HELP (considered one of the first CDSSs), on whether they are truly helpful to practitioners. The introduction of evidence-based medicine as a support for these systems, in the form of Computer-Interpretable Guidelines (CIGs) [4], has brought a change in the way these systems are regarded. CIGs are machine readable and structured representations of Clinical Practice Guidelines which, in turn, are systematically developed statements that provide recommendations about appropriate care in specific circumstances.

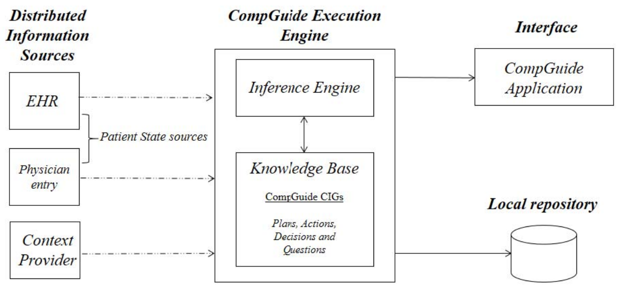

Fig. 1. Components that interact with the CompGuide execution engine. The *context provider* assesses the resources that a CDSS recommendation needs and determines if their execution is possible at the health care institution. The *local repository* stores the patient data retrieved from the information sources.

This work is focused on the use of Speculative Computation in the execution of CPGs represented in the *CompGuide* ontology, using the *CompGuide* execution engine for clinical decision support, to deal with uncertainty and incomplete information regarding the state of a patient. Incomplete information is a particular case of imperfect information. Other cases include uncertainty, inaccuracy and incoherence. As for incomplete information, it occurs when there is not available information regarding an attribute, a value is not specified. So far, the approaches to this type of problem include Bayesian belief networks, neural networks, case-based reasoning and other statistical methods [5][6]. However, given the rule-based structure of CIG systems, a method that can conform to them and structure the reasoning process is necessary.

Taking that into consideration, the current work uses the framework and semantics for Speculative Computation and Abduction provided by Satoh et al. [7] to describe the reasoning process in the automated execution of CIGs.

The article is organized as follows. Section two characterizes the domain and the case-study used do demonstrate the application of Speculative Computation. Section three provides the formal definitions for the framework and associated processes along with a proof procedure for the case-study. Finally, section four provides some conclusions and future work considerations.

II. Case-study

One of the components of the *CompGuide* project is an ontology for CPGs represented in Ontology Web Language 2 (OWL 2) more specifically in OWL Description Logics (OWL-DL). It represents CPGs as networks of tasks which can be *Plans*, *Actions*, *Questions* and *Decisions* [8]. The definition of a relative order between the tasks is done through a set of object properties. A *Plan* is linked to its first task by the *hasFirstTask* property, and the task that follows it is connected to the previous by *nextTask*. For special cases such as simultaneous tasks or a choice between tasks, one uses the *parallelTask* and the *alternativeTask* object properties, respectively. The *CompGuide* execution engine, represented in Fig. 1, handles this procedural logic. It fetches the information necessary to verify patient state related constraints placed on tasks either by querying remote *electronic health records* (*EHRs*) or by posing questions to the *physician* using the system. However, the line of communication with the *EHR* and the *context provider* may be broken or the *physician* may not have all the necessary information to enable the execution of a guideline, thus rendering the available information incomplete and casting uncertainty on the CIG being followed.

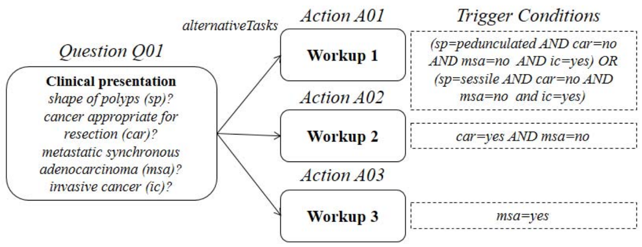

Fig. 2. Excerpt of the NCCN Guideline for Colon Cancer.

This is a problem when the system is presented with a situation such as the one depicted in Fig. 2 in which there is a choice between multiple alternative tasks. Each alternative task is associated with a set of trigger conditions that dictate their choice. When one wants to move from a given task to one of the alternative tasks, the execution engine verifies the trigger conditions of the alternatives, selecting the task whose trigger conditions hold true. The trigger conditions are related to the parameters of the patient state present in the EHR or provided by the physician. This case represents an excerpt of the National Comprehensive Cancer Network (NCCN) Guideline for Colon Cancer[1] [9] in which there is a *Question* task to get information about four clinical parameters: the shape of polyps, if the cancer is appropriate for resection, if there is a suspicion or proof of metastatic synchronous adenocarcinoma and if the cancer is invasive or not. Then, according to the trigger conditions in Fig. 2, the execution engine should propose the next task from the alternatives. If information is missing, how can the computation of the next task proceed? The work presented herein features a methodology to handle these cases, based on the following assumptions: (1) the guideline execution algorithm will recommend the next task in the clinical workflow, but, before that, it has to make a plan; (2) the execution instance can only move to another task if that task is linked to the current one by the *alternativeTask* property; (3) to move to one of the alternative tasks, the trigger conditions of such task must be met; (4) the transition is only possible if the resources to perform the proposed task are available at the healthcare institution as verified by the *context provider (cp)*; (5) the information necessary to verify the trigger conditions will be acquired from the *patient information sources* (*pis*) during the planning and may not be immediately available; (6) the system has a set of default values about the information of the patient's state and the context resulting from previous executions of the guideline; (7) the derivation of defaults occurs differently for parameters referring to the patient's state and for parameters referring to the practicability of tasks. In the first case the data of patients from previous executions of the guideline, stored in the *local repository*, is used to retrieve the values of the parameters based on the ones with the highest relative frequencies. In the second case, the defaults are assumed as being the last responses of the *cp* regarding the alternative tasks, stored in the local repository.

[1] Available at *http://www.nccn.org/professionals/physician_gls/f_guidelines.asp* .

III. Speculative Computation

The theory of Speculative Computation and Abduction, as presented by Satoh [7] and based on the seminal work of Kakas and Kowalski [10], may provide a solution to increase the efficiency of the alternative task computation process. This theory combines dynamically revisable computation and abductive reasoning to handle incomplete information. Given a computation problem, a tentative computation is made, based on default assumptions. The necessary information is asked to its providers and, as it arrives, the consistency of this information with the default assumptions is evaluated. The execution of a Speculative Computation framework occurs in two phases: *Process Reduction* and *Fact Arrival*. The framework for this methodology as well as its execution phases and a proof procedure for the case study are presented in the following subsections.

A. Framework and Preliminary Definitions

The Framework of Speculative Computation in Clinical Decision Support Systems (SF_{CDSS}) has the same components as those in [7]. It is defined in terms of the following tuple $\langle \Sigma, \varepsilon, \Delta, \mathcal{A}, \mathcal{P}, \mathcal{I} \rangle$ with slight differences at the system component identifiers in order to translate the architecture of Fig. 1. wherein the tuple:

- Σ is a finite set of constants. An element in Σ is a *system component identifier*;
- ε is a set of predicates called *external predicates*. When Q is a literal with an external predicate and S is the identifier of a remote information source, $Q@S$ is called an *askable literal*, $\sim Q@S$ is defined as $(\sim Q)@S$;
- Δ is called the *default answer* set and consists in a set of ground *askable literal*s satisfying the condition that Δ does not contain both $p(t_1,\ldots,t_n)@S$ and $\sim p(t_1,\ldots,t_n)@S$ at the same time;
- $\mathcal{A}$ is a set of predicates called *abducible* predicates and when Q is a literal with an *abducible* predicate, Q is called *abducible*;
- $\mathcal{P}$ is a set of rules of the form:

- $H \leftarrow B_1, B_2, \ldots, B_n$., where H is a positive ordinary literal and each of $B_1, B_2, \ldots, B_n$ is an ordinary literal or an *askable literal* or an *abducible*;
- H is called a head of rule R denoted as *head(R)*, always non-empty, and $B_1, B_2, \ldots, B_n$ is the body denoted as *body(R)*, with the possibility of $body(R)=\emptyset$;

- $\mathcal{I}$ is a set of integrity constraints of the form:

- $\bot \leftarrow B_1, B_2, \ldots, B_n$., where $\bot$ is a special symbol for contradiction and each $B_1, B_2, \ldots, B_n$ is an ordinary literal or an *askable literal* or an *abducible*, but at least one of them is an *askable literal* or an *abducible*.

An *askable literal* has two meanings:

1. An *askable literal* $Q@S$ in a rule in $\mathcal{P}$ represents a question the CDSS asks to a remote component of the system S; and
2. An *askable literal* in Δ represents the default truth value, if $p(t_1,\ldots,t_n)@S \in \Delta$, $p(t_1,\ldots,t_n)@S$ is normally true for a question to a system component S, and if $\sim p(t1,\ldots,tn)@S \in \Delta$, $p(t_1,\ldots,t_n)@S$ is normally false for a question to a system component S.

Now, the framework is applied to the case depicted in Fig. 2. In the formalization below *alt(a,b)* means that *b* is an alternative task linked to *a*. *nex(b)* means that the *cp* considers that task *b* cannot be executed. *tcv(b)* indicates that the trigger conditions for task *b* are validated. The default values for the clinical parameters and the practicability of tasks are defined in Δ. *gtt(a,b)* symbolizes the transition from task *a* to *b* and *nt(a,b)* means that *b* has been selected as the task following *a*. The example of the previous section, in which the guideline execution engine has to plan the next task of the careflow, is formalized under the Speculative Computation framework as follows:

- $\Sigma=\{pis,cp\}$
- $\varepsilon=\{sp,car,msa,ic,nex\}$
- $\Delta=\{sp(pedunculated)@pis, car(no)@pis, msa(no)@pis, ic(yes)@pis, \sim nex(a01)@cp, \sim nex(a02)@cp, nex(a03)@cp\}$
- $\mathcal{A}=\{gtt\}$
- $\mathcal{P}$ *is the following set of rules:*
  - *nt(X,F)←alt(X,F), tcv(F),gtt(X,F).*
  - *tcv(a01)←sp(pedunculated)@pis,car(no)@pis, msa(no)@pis, ic(yes)@pis.*
  - *tcv(a01)←sp(sessile)@pis,car(no)@pis, msa(no)@pis, ic(yes)@pis.*
  - *tcv(a02)←car(yes)@pis,msa(no)@pis.*
  - *tcv(a03)←msa(yes)@pis.*
  - *alt(q01,a01)←.*
  - *alt(q01,a02)←.*
  - *alt(q01,a03)←.*

- $\mathcal{I}$ *is the following set of integrity constraints:*
  - $\bot \leftarrow gtt(X,Y), nex(Y)@cp.$

In the representation it is assumed that the shape of the polyps presented by the patient is pedunculated, the cancer is not appropriate for resection, there is no evidence or proof of metastatic synchronous adenocarcinoma and the cancer is invasive.

When applied to CDSSs, Speculative Computation implies the generation of hypotheses based on default values. These hypotheses are possible paths for the execution engine. The *Process Reduction* phase is, in this context, the normal execution of a program, such as the one presented above, in which processes are created when choice points are encountered and an active process is reduced to a new one [7]. The *Fact Arrival* phase corresponds to an interruption when an answer arrives from an information source [7]. A process terminates successfully if all the computation is complete and the defaults have not been contradicted. To understand the notion of process, the preliminary definitions from previous work [7] are necessary, namely those of extended literal and process:

- **Definition 1** An extended literal is either a literal or an expression of the form $fail(\{L_1,\ldots,L_n\})$ where L_i is a literal. $fail(\{L_1,\ldots,L_n\})$ is used to demonstrate that there is no proof for L_i [11].
- **Definition 2** A process is the tuple $\langle GS,OD,IA,ANS\rangle$, in which: *GS* is a set of extended literals to be proved called a goal set and expresses the current status of an alternative computation, *OD* is a set of *askable literal*s called outside defaults and represents the assumed information about the outside world, *IA* is a set of negative literals or *abducibles* called inside assumptions and represents the values assumed during a process, and *ANS* is a set of instantiations of variables in the initial query.
- **Definition 3** A process set *PS* is a set of processes. A set of already asked questions *AAQ* is a set of *askable literal*s. A current belief state *CBS* is a set of *askable literal*s.

PS expresses all the alternative computations considered. *AAQ* is used to avoid asking redundant questions to information sources and the *CBS* contains the system's belief of the current status of the outside world. The definition of *active* process and *suspended* process, which is also necessary for the proof procedure, is as follows:

- **Definition 4** Let $\langle GS,OD,IA,ANS\rangle$ be a process and *CBS* be a current belief state. A process is active with respect to *CBS* if $OD \subseteq CBS$. A process is suspended with respect to *CBS* otherwise.

This definition emphasizes that, for a process to remain active, its outside defaults have to be consistent with the current belief state.

B. Process Reduction Phase

During the Process Reduction phase changes occur in the process set. In the following description of this phase, changed *PS*, *AAQ* and *CBS* are represented as *NewPS*, *NewAAQ* and *NewCBS*. The steps for Process Reduction are:

- **Initial Step:** Let *GS* be an initial goal set. $\langle GS,\emptyset,\emptyset,ANS\rangle$ is given to the proof procedure where *ANS* is a set of variables in *GS*. That is, $PS=\{\langle GS,\emptyset,\emptyset,ANS\rangle\}$. Let $AAQ=\emptyset$ and $CBS=\Delta$.
- **Iteration Step:** Do the following

- **Case 1** If there is an active process $\langle\emptyset,OD,IA,ANS\rangle$ with respect to *CBS* in *PS*, terminate the process by returning outside defaults *OD*, inside assumptions *IA*, and instantiation of variables *ANS*.

- **Case 2** If there is no active process, terminate the process by reporting a failure of the goal.
- **Case 3** Select an active process *⟨GS,OD,IA,ANS⟩* with respect to *CBS* from *PS* and select an extended literal *L* in *GS*. Let *PS'=PS-{⟨GS,OD,IA,ANS⟩}* and *GS''=GS-L*. For the selected extended literal *L*, do the following.
  - **Case 3.1** If *L* is a positive ordinary literal, $NewPS=PS' \cup \{\langle(\{body(R)\} \cup GS')\theta, OD, IA, ANS\theta\rangle \mid \exists R \in \mathcal{P}$ and $\exists$most general unifier (mgu) θ so that $head(R)\theta=L\theta\}$.
  - **Case 3.2** If *L* is a ground negative ordinary literal or a ground *abducible* then:
    - **Case 3.2.1** If $L \in IA$ then $NewPS=PS' \cup \{\langle GS',OD,IA,ANS\rangle\}$.
    - **Case 3.2.2** If $\overline{L} \in IA$ then *NewPS=PS'*.
    - **Case 3.2.3** If $L \notin IA$ then $NewPS=PS' \cup \{\langle NewGS,OD,IA \cup \{L\},ANS\rangle\}$ where $NewGS= \{fail(BS) \mid BS \in resolvent(L, \mathcal{P} \cup \mathcal{I})\} \cup GS'$ and *resolvent(L,T)* is defined as follows:
      - If *L* is a ground negative ordinary literal, $resolvent(L,T)=\{\{L_1\theta,...,L_k\theta\} \mid H \leftarrow L_1,...,L_k \in T$ so that $\overline{L}=H\theta$ by a ground substitution $\theta\}$.
      - If *L* is a ground *abducible*, $resolvent(L,T)=\{\{L_1\theta,...,L_{i-1}\theta,L_{i+1}\theta,...,L_k\theta\} \mid \bot \leftarrow L_1,...,L_k \in T$ so that $L=L_i\theta$ by a ground substitution $\theta\}$.
  - **Case 3.3** If *L* is *fail(BS)*, then
    - If BS=$\emptyset$, *NewPS=PS'*.
    - *If BS≠*$\emptyset$, then do the following:
      1. Select *B* from *BS* and let *BS'=BS-{B}*.
      2. **Case 3.3.1** If *B* is a positive ordinary literal, $NewPS=PS' \cup \{\langle NewGS \cup GS',OD,IA,ANS\rangle\}$ where $NewGS=\{ fail((\{body(R)\} \cup BS')\theta) \mid \exists R \in \mathcal{P}$ and $\exists$mgu θ so that $head(R)\theta=B\theta\}$.
         Case 3.3.2 If *B* is a ground negative ordinary literal or a ground *askable literal* or an *abducible*, *NewPS*=PS' $\cup \{\langle\{fail(BS')\} \cup GS',OD,IA,ANS\rangle\} \cup \{\langle\{\overline{B}\} \cup GS',OD,IA,ANS\rangle\}$.
  - **Case 3.4** If *L* is a ground *askable literal*, *Q@S*, then do the following:
    1. If $L \notin AAQ$ and $\overline{L} \notin AAQ$, then send the question *Q* to the slave agent *S* and $NewAAQ=AAQ \cup \{L\}$.
    2. If $\overline{L} \in OD$ then *NewPS=PS'* else $NewPS=PS' \cup \{\langle GS',OD \cup \{L\},IA, ANS\rangle\}$

C. Fact Arrival Phase

In the Fact Arrival phase the current belief state is revised in light of the information that arrives from the information sources. Supposing that answer *Q* is returned from an information source *S*. Let *L=Q@S*. By [7], After finishing one step of process reduction:

- If $\overline{L} \in$ CBS, then $NewCBS=CBS-\{\overline{L}\} \cup \{L\}$
- Else if $L \notin$ CBS, then $NewCBS=CBS \cup \{L\}$.

Some *askable literals* might not be included in the initial belief set. If this happens, processes that use these *askable literals* and their complements are suspended until the arrival of the answers

D. Proof Procedure and Execution Example

Stable model semantics is used to ensure the correctness of the proof procedure [11].

An execution example of the program in Section 3.1 is presented. The strategy followed for process reduction consists in, when a positive literal is reduced, creating new processes along with the rule order in the program, which are unifiable with the positive literal, and always selecting a newly created or newly resumed process and a left most literal. A selected literal in the selected active process is underlined. The following is the execution trace for *nt(q01,Y)*.

1. $PS=\{\langle\{\underline{nt(q01,Y)}\},\emptyset,\emptyset\rangle\}$[2],
   $AAQ=\emptyset$,
   CBS={sp(pedunculated)@pis, car(no)@pis, msa(no)@pis, ic(yes)@pis, ~nex(a01)@cp, ~nex(a02)@cp, nex(a03)@cp}.
2. By **Case 3.1**, $PS=\{\langle\{\underline{alt(q01,Y)},tcv(Y),gtt(q01,Y)\},\emptyset,\emptyset\rangle\}$
3. By **Case 3.1**, $PS=\{\langle\{\underline{tcv(a01)},gtt(q01,a01)\},\emptyset,\emptyset\rangle, \langle\{tcv(a02),gtt(q01,a02)\},\emptyset,\emptyset\rangle, \langle\{tcv(a03),gtt(q01,a03)\},\emptyset,\emptyset\rangle\}$
4. By **Case 3.1**, $PS=\{\langle\{\underline{sp(pedunculated)@pis},car(no)@pis, msa(no)@pis, ic(yes)@pis,gtt(q01,a01)\},\emptyset,\emptyset\rangle,\langle\{sp(sessile)@pis,car(no)@pis,msa(no)@pis, ic(yes)@pis,gtt(q01,a01)\}, \emptyset,\emptyset\rangle ,P_1,P_2$[3]$\}$
5. By **Case 3.4**, *sp(pedunculated)* is asked to the *pis*.
   $PS=\{\langle\{\underline{car(no)@pis},msa(no)@pis,ic(yes)@pis, gtt(q01,a01)\},\{sp(pedunculated)@pis\},\emptyset\rangle, P_1,P_2,P_3$[4]$\}$
   AQQ={sp(pedunculated)@pis}
6. By **Case 3.4**, *car(no)* is asked to the *pis*.
   $PS=\{\langle\{\underline{msa(no)@pis},ic(yes)@pis,gtt(q01,a01)\}, \{sp(pedunculated)@pis,car(no)@pis\} ,\emptyset\rangle, P_1,P_2,P_3\}$
   AQQ={sp(pedunculated)@pis,car(no)@pis}
7. *sp(pedunculated)* **is returned from** *pis*. Nothing changes.
8. By **Case 3.4**, *msa(no)* is asked to the *pis*.
   $PS=\{\langle\{ \underline{ic(yes)@pis},gtt(q01,a01)\},\{sp(pedunculated)@pis,car(no)@pis, msa(no)@pis\},\emptyset\rangle, P_1,P_2,P_3\}$
   AQQ={sp(pedunculated)@pis,car(no)@pis,msa(no)@pis}
9. *~car(no)* **is returned from** *pis*. By Fact Arrival Phase,
   CBS={sp(pedunculated)@pis,~car(no)@pis, msa(no)@pis, ic(yes)@pis, ~nex(a01)@cp, ~nex(a02)@cp,nex(a03)@cp}
   $PS=\{\langle\{\underline{tcv(a02)},gtt(q01,a02)\},\emptyset,\emptyset\rangle,P_2,P_3,P_4$[5]$\}$

[2] Representation of a process *⟨GS,OD,IA,ANS⟩*. The *ANS* part is omi ted in the process because there is no variable in the initial query.

[3] From now on P_1 and P_2 are abbreviations of $\langle\{tcv(a02)),gtt(q01,a02)\},\emptyset,\emptyset\rangle$ and $\langle\{tcv(a03),gtt(q01,a03)\},\emptyset,\emptyset\rangle$.

[4] From now on P_3 will be used as an abbreviation of $\langle\{sp(sessile)@pis,car(no)@pis, msa(no)@pis,ic(yes)@pis,gtt(q01,a01)\}, \emptyset,\emptyset\rangle$.

10. By **Case 3.1**,
    PS={⟨{$\underline{\textit{car(yes)@pis}}$*,msa(no)@pis,gtt(q01,a02)},Ø,Ø⟩,*P_2*,* P_3*,*P_4*}*
11. *By* **Case 3.4***, car(yes)* is asked to the *pis*.
    AQQ={sp(pedunculated)@pis,car(no)@pis,msa(no)@pis, car(yes)@pis}
    There is no default for *car(yes)* so the process is suspended.
    PS={⟨{$\underline{\textit{tcv(a03}}$*),gtt(q01,a03)},Ø,Ø⟩,* P_3*,*P_4*,*P_5[6]*}*
12. *msa(no)* **is returned from** *pis*!! Nothing changes.
13. By **Case 3.1**, *PS={⟨{*$\underline{\textit{msa(yes)@pis}}$*,gtt(q01,a03)},Ø,Ø⟩,* P_3*,*P_4*,*P_5*}*
14. By **Case 3.4***, msa(yes)* is asked to the *pis*.
    AQQ={sp(pedunculated)@pis,car(no)@pis,msa(no)@pis,c ar(yes)@pis, msa(yes)@pis}
    There is no default for *mas(yes)* so the process is suspended.
15. *car(yes)* **is returned from** *pis*!! By Fact Arrival Phase,
    CBS={sp(pedunculated)@pis,~car(no)@pis,car(yes),msa(no)@pis,ic(yes)@pis,~nex(a01)@cp,~nex(a02)@cp, nex(a03)@cp }
    PS={⟨{$\underline{\textit{msa(no)@pis}}$*,gtt(q01,a02)},{car(yes)@pis},Ø⟩,* P_3*,*P_4*,*P_6[7]*}*
16. By **Case 3.1**,
    PS={⟨{$\underline{\textit{gtt(q01,a02)}}$*},{car(yes)@pis,msa(no)@pis},Ø⟩,* P_3*,*P_4*,*P_6*}*
17. By **Case 3.2.3**,
    PS={⟨{$\underline{\textit{fail(\{nex(a02)@cp\})}}$*},{car(yes)@pis,msa(no)@pis} ,gtt(q01,a02)⟩,* P_3*,*P_4*,*P_6*}*
18. By **Case 3.3.2**,
    PS={⟨{$\underline{\textit{fail(Ø)}}$*},{car(yes)@pis,msa(no)@pis}, gtt(q01,a02)⟩,*
    ⟨{~nex(a02)@cp},{car(yes)@pis,msa(no)@pis},P_3*,*P_4*,*P_6*}*
19. *~msa(yes)* **is returned from** *pis*!! By Fact Arrival Phase,
    CBS={sp(pedunculated)@pis,~car(no)@pis,car(yes), msa(no)@pis, ~msa(yes)@pis,ic(yes)@pis, ~nex(a01)@cp, ~nex(a02)@cp, nex(a03)@cp}
20. By **Case 3.3**,
    PS={⟨{$\underline{\textit{\textasciitilde nex(a02)@cp}}$*},{car(yes)@pis,msa(no)@pis}, gtt(q01,a02)⟩},*P_3*,*P_4*,*P_6*}*
21. By **Case 3.4**, *~nex(a02)* is asked to the *cp*,
    PS={⟨Ø,{car(yes)@pis,msa(no)@pis,~nex(a02)@cp}, gtt(q01,a02)⟩,P_3*,*P_4*,*P_6*}*
    AQQ={sp(pedunculated)@pis,car(no)@pis,msa(no)@pis,c ar(yes)@pis, msa(yes)@pis,~nex(a02)@cp}
22. *{car(yes)@pis,msa(no)@pis}* is returned as outside defaults and *gtt(q01,a02)* is returned as inside assumptions.
23. The answer *(ANS)* is *{nt(q01,a02}*. The answer set would remain unchanged by the answer *~nex(a02)@cp*, which is still missing.

The procedure results in a recommendation that selects *a02* as the next clinical task, i.e., *Workup 2*. At steps 5, 6, 8 and 21 the active processes assume the values in the default set while the real truth values of the literals are asked to the information sources, demonstrating the effect of Speculative Computation. Without this method the respective processes would have to be suspended. At step 17, an ordinary abduction is performed, i.e, it is assumed that one goes from task *q01* to *a02* (*gtt(q01,a02)*) and any integrity constraints are checked in order not to lead to contradiction. The procedure was able to generate an answer without possessing all the information, i.e., with uncertainty associated to the clinical parameters, and even without receiving the information about the practicability of task *a02*. This allowed the construction of a possible, and also most likely, scenario.

If one considers the complexity of a guideline such as the NCCN Guideline for Colon Cancer (as represented in Fig. 3), with multiple data entry points that ultimately generate splitting points, it is possible to use Speculative Computation and Abduction on each one and, through it, present the most likely execution threads by summing the computation of these choices. Fig. 3 shows an example of how the SF_{CDSS} could be applied to a guideline algorithm represented in *CompGuide*. For every decision point in the algorithm there is an SF_{CDSS} that runs on top of the procedural knowledge provided by the ontology. Assuming that information is missing in each question task, the SF_{CDSS} formulates a probable choice for the next task at *q01*, *q02*, *q03* and *q04*. Then, by grouping the proposals of the SF_{CDSS}, it is possible to build a tentative execution path which in the case of Fig. 3 would be *q01-a02-q02-q03-a06*. The framework is responsible for handling incomplete information regarding the clinical parameters necessary for making a decision. This would be useful for a practitioner as it would provide him a map of the potential evolution of a patient, thus giving him time to devise countermeasures if it shows that the treatment is following an undesirable direction.

IV. Conclusions and Future Work

This work shows that the application of Speculative Computation and Abduction to CIG-based CDSSs is possible. Moreover, this approach provides a way to clearly structure the reasoning process of the execution engine and endows it with dynamic belief revision capabilities.

The original contribution of the article is the application of an already existing framework and semantics provided by Satoh [7] to the domain of medical decision, having the *CompGuide* model as a basis.

The inclusion of Speculative Computation is a differentiating factor from other CIG execution engines such as Arezzo, DeGeL, GLARE, GLEE and SAGE [12] which usually only execute their coded rules [4], without additional intelligent functionalities. In situations such as the one represented in the case-study, a battery of tests must be performed in order to determine the shape of polyps, the possibility of resection, if there are synchronous metastases and the invasiveness of the cancer. The results of these tests may take some time to be known or may turn out to be inconclusive. In the first case, the effect of Speculative Computation enables the construction of a possible scenario for the next procedure before knowing the results, and in the latter it provides the most likely values (the

[5] From now on P_4 will be used as an abbreviation of *⟨{ic(yes)@pis,gtt(q01,a01)}, {sp(pedunculated)@pis,car(no)@pis, msa(no)@pis} ,Ø⟩.*

[6] From now on P_5 will be used as an abbreviation of *{⟨{car(yes)@pis,msa(no)@pis, gtt(q01,a02)},Ø,Ø⟩.*

[7] From now on P_6 will be used as an abbreviation of *{⟨{msa(yes)@pis,gtt(q01,a03)},Ø,Ø⟩.*

defaults) for the missing parameters, thus computing the most appropriate clinical procedure taking into account the current state of the information. For instance, the ressectability of cancer is difficult to assess, given the different opinions regarding cancer how long should a clear surgical margin be. So it may be inconclusive and Speculative Computation may offer a default for this parameter to be considered in the reasoning process.

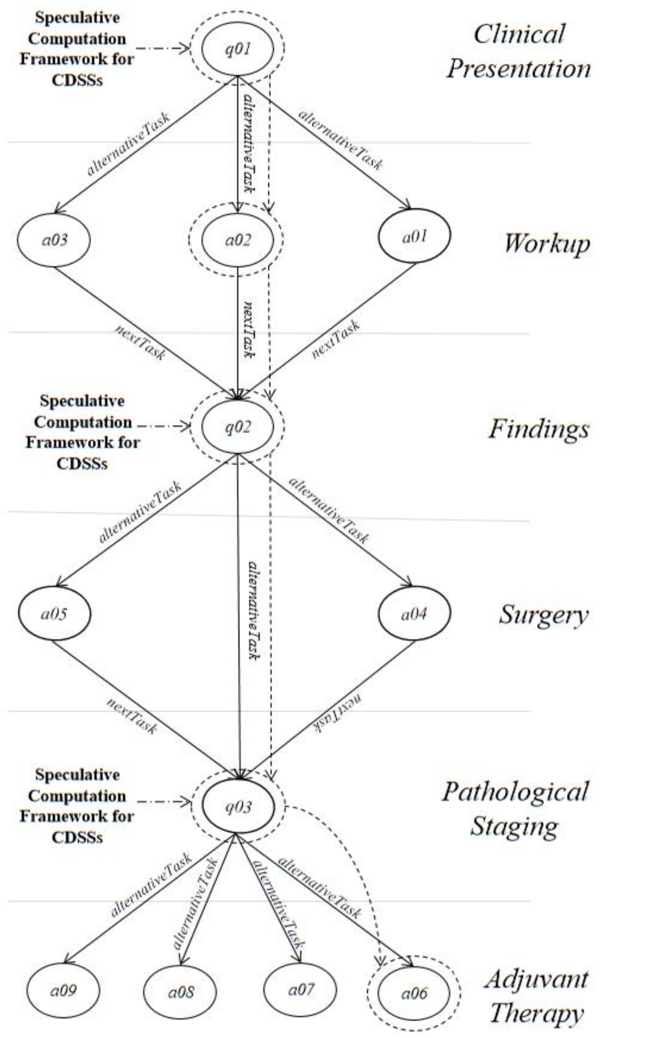

Fig. 3. An excerpt of the NCCN Guideline for Colon Cancer showing a symbolic representation of tasks as questions and actions through the different phases of the clinical process: *Clinical Presentation* (also represented in Fig. 2), *Workup*, *Findings, Surgery*, *Pathological Staging* and *Adjuvant Therapy*. The dashed trace shows the most likely execution path provided by the application of Speculative Computation..

The work presented herein has the potential of enhancing one of the most important elements in health care which is people's capacity to make decisions. Any procedure (e.g., surgery and drug prescription) can be harmful if the decision to perform it is made incorrectly. The existence of multiple options may increase decision difficulty, which may lead to practitioners choosing the most distinctive option or maintaining the status quo [13]. This cognitive bias is produced by intuitive thought processes that are error prone. Being aware of the likelihood of various options and of their effect further ahead is one way of casting aside this biased behavior. In further works one will provide more details about the generation of defaults, as it is an important aspect of the inference mechanism. Currently, the formalisms presented are only capable of dealing with nominal values for the parameters. However, clinical parameters may be expressed in numerical values, which implies endowing the mechanism with the capability of reasoning with intervals. Furthermore, the inclusion in the reasoning process of degrees of confidence that stem directly from the statistics of the default generation process.

Acknowledgement

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 – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-028980 (PTDC/EEI-SII/1386/2012). The work of Tiago Oliveira is supported by doctoral grant by FCT (SFRH/BD/85291/2012).

References

[1] M. A. Musen, Y. Shahar, and E. H. Shortliffe, "Clinical decision-support systems," *Biomed. Informatics*, pp. 698–736, 2006.

[2] R. Kaushal, K. G. Shojania, and D. W. Bates, "Effects of computerized physician order entry and clinical decision support systems on medication safety: a systematic review.," *Arch. Intern. Med.*, vol. 163, no. 12, pp. 1409–16, Jun. 2003.

[3] N. K. J. Adhikari, J. Beyene, J. Sam, and R. B. Haynes, "Effects of Computerized Clinical Decision Support Systems on Practitioner Performance," vol. 293, no. 10, pp. 1223–1238, 2005.

[4] M. Peleg, "Computer-interpretable clinical guidelines: A methodological review.," *J. Biomed. Inform.*, no. June, Jun. 2013.

[5] F. K. J. Sheridan, "A survey of techniques for inference under uncertainty," *Artif. Intell. Rev.*, vol. 5, no. 1, pp. 89–119, 1991.

[6] G. Kong, D. Xu, J. Yang, and M. Business, "Clinical Decision Support Systems: A Review on Knowledge Representation and Inference Under Uncertainties," *Int. J. Comput. Intell. Syst.*, vol. 1, no. 2, pp. 159–167, 2008.

[7] K. Satoh, "Speculative Computation and Abduction for an Autonomous Agent," *IEICE - Trans. Inf. Syst.*, vol. E88-D, no. 9, pp. 2031–2038, 2005.

[8] T. Oliveira, P. Novais, and J. Neves, "Representation of Clinical Practice Guideline Components in OWL," in *Trends in Practical Applications of Agents and Multiagent Systems SE - 10*, vol. 221, J. B. Pérez, R. Hermoso, M. N. Moreno, J. M. C. Rodríguez, B. Hirsch, P. Mathieu, A. Campbell, M. C. Suarez-Figueroa, A. Ortega, E. Adam, and E. Navarro, Eds. Springer International Publishing, 2013, pp. 77–85.

[9] A. Benson and E. Al., "NCCN Clinical Practice Guideline in Oncology Colon Cancer," 2013.

[10] A. C. Kakas, R. A. Kowalski, and F. Toni, "The Role of Abduction in Logic Programming," in *Handbook of Logic in Artificial Intelligence and Logic Programming 5*, 1998, pp. 235–324.

[11] A. C. Kakas and P. Mancarella, "On the Relation between Truth Maintenance and Abduction," in *Proccedings of PRICAI-90*, 1990, pp. 438–433.

[12] D. Isern and A. Moreno, "Computer-based execution of clinical guidelines: a review," *Int. J. Med. Inform.*, vol. 77, no. 12, pp. 787–808, 2008.

[13] D. Redelmeier and E. Shafir, "Medical Decision Making in Situations that offer multiple options," *JAMA J. Am. Med. Assoc.*, vol. 273, no. 4, pp. 302–305, 1995.