Noname manuscript No.
(will be inserted by the editor)

A Dynamic Default Revision Mechanism for Speculative Computation

Tiago Oliveira · Ken Satoh · Paulo Novais · José Neves · Hiroshi Hosobe

Received: date / Accepted: date

Abstract In this work a default revision mechanism is introduced into Speculative Computation to manage incomplete information. The default revision is supported by a method for the generation of default constraints based on Bayesian Networks. The method enables the generation of an initial set of defaults which is used to produce the most likely scenarios during the computation, represented by active processes. As facts arrive, the Bayesian Network is used to derive new defaults. The objective with such a new dynamic mechanism is to keep the active processes coherent with arrived facts. This is achieved by changing the initial set of default constraints during the reasoning process in Speculative Computation. A practical example in clinical decision support is described.

Keywords Default Revision, Incomplete Information, Speculative Computation, Bayesian Networks.

1 Introduction

In order to tackle situations of problem solving and decision making in which cases of incomplete information may occur, a framework of Speculative Com-

T. Oliveira · P. Novais · J. Neves
Algoritmi Research Centre/Department of Informatics, University of Minho
Braga, Portugal
E-mail: {toliveira,pjon,jneves}@di.uminho.pt

K. Satoh
National Institute of Informatics, Sokendai University
Tokyo, Japan
E-mail: ksatoh@nii.ac.jp

H. Hosobe
Department of Digital Media, Hosei University
Tokyo, Japan
E-mail: hosobe@hosei.ac.jp

putation was proposed, first for yes/no questions [23,24] and then for general questions using constraints [22,9]. The term Speculative Computation is used to describe a series of procedures by which an answer for a problem is searched and computed using default beliefs. A default belief is defined here as something thought to be the case in replacement of the true information, if the last is not available. Speculative Computation is formalized in terms of a framework and process semantics featuring mechanisms for the reduction and revision of processes. Such a framework allows an agent to reason with default beliefs while waiting for the other agents to reply. In this way, the computation of answers is brought forward and, if the arriving information matches the default beliefs, it is possible to save time in the search for an answer. This kind of framework is particularly useful in multi-agent systems since it is often difficult to guarantee efficient and reliable communications between agents. A multi-agent system may be deployed in an unreliable network or need human intervention to promote interactions between agents, which may cause situations that delay or even prevent communications. An agent which is part of such a system and uses information provided by the other agents may see its inference process blocked or delayed. The use of the term information here refers to the elements that serve as premises in the inference process and which will later enable the agent to derive logical conclusions, in order to solve a problem or make a decision.

In [23,24,22,9], fixed default beliefs are used in Speculative Computation. However, when applied to a real setting, the default beliefs are highly dependent on the context. The same is to say they depend on the set of circumstances and facts that surround a problem and change over time. A default belief may change according to the information obtained by later replies from the information sources and may become incoherent with the returned facts, which would result in tentative answers that do not represent the most likely scenarios. A scenario is a process representing an answer obtained through reductions based on default beliefs, meaning that, in the context of a problem, it represents an outline of a possible event.

To account for this drawback, the present work proposes a dynamic revision of default beliefs. It is important to state that this revision does not refer to the replacement of default beliefs with true information, but to their replacement with new default beliefs. It is performed on-line since it occurs as answers are searched and facts arrive. In a real setting, it is important to anticipate changes of state and therefore, in the present work, default beliefs are revised into new default beliefs as information arrives, which allows the prompt adjustment of scenarios. These default beliefs assume the form of default constraints. The same is to say that the information used to reduce a process involving a variable for which no real value is known (i.e., the default) is represented as a constraint in a Constraint Logic Program (CLP). The behaviour of an agent is also specified in a CLP, where derivations are handled as alternative answers. The manipulation of such alternative answers is done with the processing of disjunctive constraints. In order to distinguish

this framework from previous ones, the term Speculative Computation with Default Revision will be used. Its main contribution is twofold:

- The first contribution is a default generation method for Speculative Computation based on Bayesian Networks (BNs). A BN featuring the variables used in the CLP is constructed with data of previous attempts of an agent to solve a problem. The probability distribution provided by the BN enables the derivation of default constraints before the beginning of the computation, without any information, and during the computation, with partial information. This represents a new use for BNs, given that, in this context, they assume a supportive role, feeding the CLP with default beliefs, and are not the main component in solving the problem.
- The second, and most important, is a mechanism for the dynamic revision of default constraints. The Speculative Computation framework is augmented with a default revision phase in which, upon the arrival of information and change in default beliefs, existing processes are revised in order to become consistent with the new defaults. A gradual convergence of scenarios towards the real answer of a problem is achieved with this revision mechanism.

The paper is organized in six sections. Section 2 provides related work about Speculative Computation and the use of default beliefs in concurrent constraint programming, belief revision, and defeasible logic. In Section 3, the Speculative Computation with Default Revision framework, with its elements and procedures, is explained. Section 3 also presents the formalization of an example using the framework and features a claim that shows the correctness of the procedures in the phases of Speculative Computation with Default Revision, thus demonstrating their effectiveness. The procedures for the generation of default constraints are explained in Section 4. An execution trace of an example regarding a clinical decision support system is discussed in Section 5. The situations recreated in the example allow the observation of the convergence of scenarios towards the real answer of a problem. In Section 6, conclusions are presented along with future work considerations.

2 Related Work

Speculative Computation was first presented as a search algorithm that performs a speculative evaluation of expressions, i.e., if the value of an expression is not known, a computation may proceed until the value for that expression is needed [2]. The algorithm implements a form of parallelism in the evaluation of function arguments. The proposal was based on the principle that problems could be solved more quickly on parallel machines if some work could be started before it is known to be necessary. The underlying principle was later adopted by [23] to develop a decision making and problem solving framework using default beliefs for multi-agent systems in the presence of incomplete information. The framework was subsequently extended with iterative belief revision [24],

default constraints and constraint processing [22,9], and abductive reasoning [21]. All these extensions targeted more dynamic and interactive environments, where there is a potential for failure in communication and for the creation of states of incomplete information. The strength of Speculative Computation fits systems which have a well-defined procedural logic and are based on rules. The framework performs the role of an interface that manages the state of the information and the tentative computations. In previously developed work [23, 24,22,21,9], the set of default beliefs used is fixed, which may be a drawback because newly arrived information may have an impact on the remaining variables for which default beliefs are being used. As a result, the scenarios from the tentative computations might not represent the most likely outcomes in terms of answers. The Speculative Computation with Default Revision is the latest of a series of developments arising from the previously mentioned work.

The work developed in the field of concurrent constraint programming bears resemblances with the present work, namely the focus on distributed computing and computation with partial information. Some similarities with the work presented herein may be found in constraint programming languages such as the Andorra Kernel Language (AKL) [5] and the Oz programming language [26]. AKL is a concurrent programming language that uses guards. A guard consists of determinate goals, which only require the execution of one clause. When a goal is selected in the guard, it is executed across all the clauses, and those that succeed originate local speculative variable bindings. Afterwards, each successful variable binding is tested against the goals belonging to the body of the clause in a depth-first search. The use of the term speculative here refers to the uncertainty as to whether the variable bindings in the guard will later succeed or not, which represents a guess about what might happen or be true. As for Oz, it is based on the Oz Programming Model for concurrent programming. The procedures in the model allow the control of multiple computation spaces assigned to various agents. The synchronization of the agents is achieved through a centralized update of a constraint store. The speculative component of the model lies in the local computations performed by the agents, which take place with partial information. The speculative elements used in both AKL and Oz are meant for parallel computing, which means that all the possible computation pathways are simultaneously explored until they fail. On the other hand, the Speculative Computation with Default Revision framework channels the computation resources to the most plausible computation pathways, which, from the point of view of this work, are the ones worth exploring. This is achieved by extracting default constraints using a probabilistic model and using them to fill in incomplete information.

Concurrent constraint programming covers a wide variety of systems. A common denominator to all of them is that they may have to deal with partial information. In the timed default concurrent constraint programming system presented in [20], there is a timed default mechanism by which a default value is assigned to a variable if any other value, different from the default, has not been added to the constraint store at a given time. The distinctive feature of this constraint system is that there is a time limit for constraints to be

added after which the processes are reduced using default beliefs. The default belief is also fixed and there is no mechanism by which it can be changed. Treating default beliefs in an isolated way automatically discards dependence relationships between variables.

Another approach sharing a few similarities with Speculative Computation with Default Revision is probabilistic concurrent constraint programming [7]. In this model, a probability mass function is assigned to a variable, which allows a constructor to choose a value for the variable according to its probability distribution. However, probabilistic constraint programming is used to model non-deterministic choices in systems that require components to exhibit different behaviours on different runs. Again, there is no notion of default belief, but there is a set of random variables with a dynamic behaviour, without fixed values. We aim to imprint this dynamic behaviour on default beliefs. From our perspective, they should not be fixed and should change according to newly arrived facts from the agent information sources. This is possible using a probabilistic model to derive new default constraints for the variables that are not covered by the newly arrived facts.

The LIFF [19] model devises an updating mechanism for logic programs based on tags added during the computation of explanations. It allows the addition of if-then rules by updating the existing solutions through the existing tags, thus removing the need to perform backtracking and computing a new answer from scratch. In Speculative Computation with Default Revision the same underlying idea of updating parts of the answers rather than computing new solutions is followed as well, with the difference that the updates are in the form of constraints, and default constraints can be updated to definitive constraints representing true information or to other default constraints.

Default revision has not been widely explored in non-monotonic logic. However, there are procedures described in the literature that are close to it. In [17], a semantics and a proof theory of a system for defeasible argumentation are presented. The prevalence of some arguments over others is determined by priorities placed on the rules that support them. The defaults in this case are the hierarchical relationships between rules. However, the priorities between arguments are not fixed and, instead, are defeasibly derived within the logic program. The proposal is based on the idea that fixed priorities between arguments in a real world problem are impractical. Similarly, in [6] the revision of rule priorities is considered in the legal domain. But, in it, defaults are revised to accommodate the preferred answers of participants. Although the relationship between these works and the work herein is only at a conceptual level, the underlying principle that fixed default information is unrealistic is the same. In Speculative Computation with Default Revision, default constraints are dynamically revised according to newly known information during reasoning, thus one may say that on-line default revision is performed, as opposed to off-line default revision in previous works. In these last two approaches, priorities are revised after facts are known and conclusions are drawn. The mechanism operates independently and is disconnected from the outside world. Conversely, herein a form of on-line default revision is

proposed, which means that the mechanism interacts with the outside world during the reasoning process, while answers are computed, and changes its beliefs according to said interaction.

In [12] a Distributed Defeasible Speculative Reasoning (DDSR) framework is presented. It combines Speculative Computation with defeasible logic in order to model a multi-context system with autonomous logic-based agents. Internally, each agent has a set of default hypotheses about the beliefs of its peers and a local defeasible theory in which the default hypotheses are used to draw conclusions. The conclusions depend on the hierarchical relationship between the arguments that build them and the score of the beliefs. Such scores are derived from a reputation table featuring each agent in the environment. Like previous works in Speculative Computation, it features a process reduction phase, in which tentative conclusions are produced, and a fact arrival phase, in which beliefs change and processes are revised according to the replies from other agents. The defeasible logic component is used to sort out conflicts within the local theory of an agent. In this sense, this work covers an aspect that is not present in our proposal, which is conflicting information. However, Speculative Computation with Default Revision is focused solely on the mechanisms by which Speculative Computation takes place and it tackles different problems. In [12] there is no indication of how to produce the default hypotheses. Furthermore, these default hypotheses are fixed and can only be changed into their real value according to the replies from other agents. These aspects are addressed in Speculative Computation with Default Revision with the introduction of a mechanism for the generation of defaults and a new phase for the revision of default beliefs.

3 A Framework for Speculative Computation with Default Revision

Speculative Computation with Default Revision has two components. The first is a Framework for Speculative Computation with Default Revision (SF_{DR}), which will be described in this section. It is where all the necessary elements to solve a problem and the different procedures to manage information are encoded. The other component is the Generation of Default Constraints, described in Section 4, which contains a BN that generates default constraints, both before and during the execution of procedures in the framework, based on the replies received by the speculative agent. The term speculative agent will be used to refer to the agent performing Speculative Computation. Within the setting of the multi-agent system defined for this work, this agent is responsible for providing an answer to a problem based on its internal logic theory and the information he receives from the other agents. The speculative agent hosts the SF_{DR}.

A description of the elements and procedures in the SF_{DR} is presented in the sections below. A summary of the abbreviations and symbols used throughout this section is provided in Table 1 of Annex A.

3.1 Elements of the Framework

An SF_{DR} featuring disjunctive constraint processing is defined in terms of the tuple $\langle \Sigma, \mathcal{E}, \Delta, \mathcal{P} \rangle$. This formulation is based on the work presented in [9]. The framework is hosted by a speculative agent in a multi-agent system. It is used to structure the inference process and to manage the information the agent needs from other agents in the system, in order to solve a problem. The subsequent elements of the tuple have the following meaning:

- Σ is a finite set of constants. An element in Σ is the identifier of an agent in a multi-agent system and represents an information source from which the speculative agent obtains information;
- $\mathcal{E}$ 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 agent information source belonging to Σ, $Q@S$ is called an askable atom;
- Δ is the default answer set and consists of a set of default constraint rules called default rules w.r.t. $Q@S$, of the following form:
  $$\text{“}Q@S \hookleftarrow C \parallel\text{.”}$$
  where
  - $Q@S$ is an askable atom;
  - C is a set of constraints called default constraints for $Q@S$;
  - A default rule w.r.t. $Q@S$ is denoted as $\delta(Q@S)$.
- $\mathcal{P}$ is a constraint logic program of the following form:
  $$\text{“}H \hookleftarrow C \parallel B_1, B_2, \ldots, B_n\text{.”}$$
  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 rule R, denoted as $const(R)$, where $const(R)$ may be empty;
  - Each of “$B_1, B_2, \ldots, B_n$” is an ordinary literal or an askable literal. “$B_1, B_2, \ldots, B_n$” is referred to as the body of R, denoted as $body(R)$, where $body(R)$ may be empty.

In order to show how a problem can be formalized in the SF_{DR} and Speculative Computation with Default Revision can be applied to specific domains and systems, an example featuring a clinical decision support system is provided. Clinical decision support is a domain where cases of incomplete information frequently occur. The outline of the system is shown in Fig. 1, and its main purpose is to provide advice to health care professionals in the form of clinical tasks, based on machine-interpretable versions of clinical guidelines [27]. The structure of this system mirrors similar examples that can be found in the literature [10, 16, 14]. The elements of its architecture include:

- A *guideline engine*: responsible for interpreting clinical guideline instructions represented in the knowledge base against information about the state of a patient;

- A *knowledge base*: containing machine-interpretable versions of clinical guidelines represented in an ontology;
- A *local repository*: with information for other patients in previous executions of a clinical guideline;
- A *speculative module*: an addition to the system and a module hosted by the *guideline engine* which implements an SF_{DR} with a Generation of Defaults mechanism.

The *guideline engine* obtains information about the state of a patient from distributed agent information sources, as described in Fig. 1, one of which is the oncology information system (*ois*). As such, the *guideline engine* is the speculative agent hosting the SF_{DR} and, together with the agent information sources, constitutes a multi-agent system.

The problem which the *guideline engine* has to solve is the choice of the clinical task with the appropriate treatment following colon cancer surgery. This decision is typically made based on the TNM Classification of Malignant Tumors, a classification in which T describes the degree of tumor invasion into the wall of the colon, N is the number of metastases in regional lymph nodes, and M represents the detection of distant metastases in other organs, such as the liver or the lungs. In cancer assessment, there are situations in which physicians are uncertain of the value of T, because tumor invasion may be difficult to assess solely from imagiology techniques. N and M may also be unknown and are dependent on the completion of laboratory exams whose results may take some time to be known. This setting was extracted from the Clinical Practice Guideline in Oncology for Colon Cancer [1], and the medical content is greatly simplified for the sake of explaining the dynamics of Speculative Computation with Default Revision.

Fig.1

In Fig. 2, the problem is represented according to an ontology for clinical guidelines [15] in which every step is displayed as a task. In the graph, first there is a question task, *question1*, to obtain the values of the T, N and M variables. Then, there are five action tasks, linked to the previous through the *hasAlternativeTask* property. A task is selected based on the fulfilment of trigger conditions, also depicted in Fig. 2. For instance, *action2*, which recommends the participation in a clinical trial, a period of observation, or chemotherapy with capecitabine or 5-FU/leucovorin, will only be proposed by the *guideline engine* if the value of T is $t3$ and the value of M is $m0$.

Fig.2

Based on this situation the example for the SF_{DR} can be completely stated as follows.

Example The assumptions are that (1) the *guideline engine* will recommend the next task for a patient; (2) the transition from one task to another is only possible if the first is connected to the second through the *hasAlternativeTask*

property; (3) to move to one of the alternative tasks, the trigger conditions of that task must be met; (4) the information necessary to verify the trigger conditions (namely the values for the T, N, and M parameters) will be acquired from an external agent information source, in this case, the oncology information system (*ois*) ; and (6) the *guideline engine* has an SF_{DR} which uses default constraints to continue the execution in the event that there is no answer from *ois*.

The example can be now represented according to the tuple $\langle \Sigma, \mathcal{E}, \Delta, \mathcal{P} \rangle$. In the representation below, a few considerations were made regarding predicate names. The predicate $nt(a, b)$ indicates that b is the task that follows a. The predicate $alt(a, b)$ indicates that b is an alternative task linked to a and reflects the meaning conveyed by the *hasAlternativeTask* property. $tcv(b)$ denotes that the trigger conditions for task b are validated. Predicate names t, n, and m represent their homonymous elements in the TNM classification. In the representation below, every capital letter symbolizes a variable. If the variable is not bounded by a constraint, it is a general purpose variable. For instance, the use of X in the first rule of $\mathcal{P}$ is a way to denote a generic task. The complete representation is as follows:

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

Since, for the current example, the *ois* is the only agent information source, it is the only element in Σ. The *guideline engine*, as the speculative agent, sends questions to the *ois* to know the values for the variables of askable atoms t, n, and m, which are the elements of $\mathcal{E}$. The first rule in $\mathcal{P}$ reflects the primary condition in the example; for a task to be recommended as the next task,

it has to be connected to the previous one as an alternative and its trigger conditions have to be validated. The rules that follow represent the trigger conditions for task selection, and the last five rules represent the alternative relationship between *question1* and the other action tasks. However, there is an element missing in the framework, that is the default answer set Δ, which is generated through the procedures for the Generation of Default Constraints, described in Section 4. The execution of the logic program $\mathcal{P}$ starts with a query. In the example, the *guideline engine* would try to answer the query $nt(question1, F)$, in order to determine which task should follow *question1*. An execution trace for this example is provided in Section 5.2.

3.2 Preliminary Definitions

In order to understand the execution of a constraint logic program $\mathcal{P}$ in the framework, it is important to introduce the notions of goal, reply set, reduction, and derivation. The definitions for these concepts are as follows.

- **Definition 1.** A goal has the form of "$\leftarrow C \parallel B_1, \ldots, B_n$" where:
  - C is a set of constraints;
  - each of $B_1, \ldots, B_n$ is either an atom or an askable atom.

The initial query becomes the first goal in the execution. Following the example described in the previous section, it would be "$\leftarrow \parallel nt(question1, F)$". At this point there would be no constraints C for the goal.

- **Definition 2.** A reply set $\mathcal{R}$ for $\mathcal{E}$ is a set of rules of the form "$Q@S \leftarrow C \parallel$.", where:
  - $Q@S$ is an askable atom;
  - each argument of Q is a variable;
  - C is a set of constraints over those variables.
- **Definition 3.** A reduction of a goal "$\leftarrow C \parallel B_1, \ldots, B_n$" w.r.t. a constraint logic program $\mathcal{P}$, a reply set $\mathcal{R}$ and a subgoal B_i is a goal "$\leftarrow C' \parallel B'$" such that:
  - there is a rule R in $\mathcal{P} \cup \mathcal{R}$ so that $C \wedge \{B_i = head(R)\} \wedge const(R)$ is consistent and $\{B_i = head(R)\}$ is a conjunction of constraints equal to the arguments of B_i and $head(R)$;
  - $C' = C \wedge \{B_i = head(R)\} \wedge const(R)$;
  - $B' = B_1, \ldots, B_{i-1}, B_{i+1}, \ldots, B_n \cup body(R)$.
- **Definition 4.** A derivation of a goal "$\leftarrow C \parallel B_1, \ldots, B_n$" w.r.t. to a speculative computation constraint framework with default revision $\langle \Sigma, \mathcal{E}, \Delta, \mathcal{P} \rangle$ and a reply set $\mathcal{R}$ is a chain of reductions "$\leftarrow C \parallel B_1, \ldots, B_n, \ldots, \leftarrow C' \parallel \emptyset$" w.r.t. $\mathcal{P}$ and $\mathcal{R}$, where $\emptyset$ denotes an empty goal. C' is called an answer constraint w.r.t. the goal, the framework and the reply set.

Regarding the execution of a program, a non-askable atom in a goal is reduced into subgoals according to the rules in $\mathcal{P}$. For an askable atom $Q@S$ in a goal, the speculative agent sends a question asking its real value to an agent S in Σ and waits for the answer. Meanwhile, the goal is reduced according to the current reply set $\mathcal{R}$. This set holds the constraints for askable atoms at a given point in the execution. It may contain default constraints, or fact constraints if the speculative agent has already received a reply from S. If no reply is returned, the default constraint is used as a tentative answer. An answer for $Q@S$ is returned in the form of constraints for the variables in the query. A goal is continuously reduced until it becomes empty.

A central notion in the SF_{DR} and its operational model is that of process. It is used to express an alternative computation and a possible path for the resolution of a problem. This concept and others related to it are defined as follows.

- **Definition 5.** An active process is a tuple $\langle \hookleftarrow C \parallel GS, UD \rangle$ in which:
  - "$\hookleftarrow C \parallel$" 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;
  - UD is a set of askable atoms called used defaults, and represents the assumed information about the outside world, i.e., it contains the askable atoms for which default constraints were used in order to reduce a goal in the process.
- **Definition 6.** A suspended process is a tuple $\langle SAS, \hookleftarrow C \parallel GS, UD \rangle$ in which:
  - SAS is called a suspended atom set and contains askable atoms $Q@S$. The askable atoms in this set are those whose constraints are responsible for suspending the process;
  - GS is a set of literals to be proved, called a goal set, and expresses the current status of an alternative computation;
  - UD is a set of askable atoms called used defaults, and represents the assumed information about the outside world, i.e., it contains the askable atoms for which default constraints were used in order to reduce a goal in the process.
- **Definition 7.** A current belief state CBS is a set of rules of the form "$Q@S \hookleftarrow C \parallel$". It contains the beliefs of the speculative agent about the values of askable atoms.
- **Definition 8.** Let $\langle \hookleftarrow C \parallel GS, UD \rangle$ be a process and CBS be a current belief state. A process is active w.r.t. CBS if $C \subseteq CBS$. A process is suspended w.r.t. to CBS otherwise.
- **Definition 9.** APS is the set of active processes.
- **Definition 10.** SPS is the set of suspended processes.
- **Definition 11.** AAQ is the set of already asked questions for askable atoms. AAQ is used to avoid asking redundant questions to agent information sources.

- **Definition 12.** The set of returned facts RF contains rules of the form: "$Q@S \hookleftarrow C \parallel$" where $Q@S$ is an askable atom and C is a set of constraints. These rules stand for replies from the agent information sources.
- **Definition 13.** A scenario is an active process $\langle \hookleftarrow C \parallel GS, UD \rangle$ in which $UD \neq \emptyset$.

Processes represent alternative ways of computation created when choice points are reached, either by case splitting or default handling. An active process succeeds if, when all the computation is done, there is no reply from the information sources that contradicts the constraints used to reduce askable atoms. A process fails whenever a newly arrived fact constraint contradicts a used default constraint. The mechanisms for the creation, alteration and removal of processes are included in the three phases of Speculative Computation with Default Revision: the *process reduction* phase, the *fact arrival* phase, and the *default revision* phase. The last is a recent addition and the object of study in this work. *Process reduction* is the normal execution of a program, the computation begins with the default rules in Δ and whenever a choice point in the computation is reached, a new process is generated. *Fact arrival phase* occurs when a reply from an agent arrives, it is an interruption phase. After a round of *process reduction* there is the *default revision* phase, in which changes to default constraint rules are assessed and the processes are revised accordingly.

3.3 Process Reduction Phase

The possible steps of the *process reduction* phase are represented in Figs. 3 and 4. The former corresponds to the very first step of the execution of the framework, while the latter describes the procedures for an iteration step.

The initial step of Fig. 3 coincides with the deployment of the framework, in which a query is submitted and becomes the initial goal set GS. There is only one active process, constructed based on the initial goal set. Since no outside information is known, the current belief state assumes the constraints in the default set. At this point, there are no suspended processes, already asked questions, or returned facts, and, as such, the sets corresponding to these elements are initiated as empty sets. Referring back to the example presented in Section 3.1 and the query $nt(question1, F)$, by the iteration step, the first active process would become $\langle \{\hookleftarrow \parallel nt(question1, F)\}, \emptyset \rangle$, with $\{\hookleftarrow \parallel nt(question1, F)\}$ as the initial goal set. The complete procedures for the initial step can be consulted in Algorithm 1 of Annex B.

Fig.3

The iteration step of Fig. 4 reflects the speculative nature of the framework. Case 1 corresponds to a point in the execution where process reduction, through the derivation of goals (as specified in Definition 4), produced an active process with an empty goal set. When such a process is obtained, its

constraints C and used defaults UD should be returned. As specified in Definition 14, if UD is not an empty set, the process is a scenario.

The procedures under Case 2 correspond to the reduction of a goal (as specified in Definition 3) in the goal set of an active process. There are two cases here. If the goal is a non-askable atom (Case 2.1), then the goal is unified with the head of a rule, a new active process is created replacing the previous in the set of active processes, the constraints in the rule are added to the constraints of the process, and the atoms in the body of the rule are added to the goal set of the process. However, if the goal is an askable atom (Case 2.2), it is necessary to check whether it is in the already asked questions, and thus has already been asked to the agent information sources, and whether it is already a used default, i.e., if it is in the used default set of the process. If this last condition is not fulfilled, the goal is reduced either with a fact constraint in the set of returned facts or with a default constraint. If there is a default constraint available, the constraint is assumed. Upon reduction using default constraints, there are usually two resulting processes, an active process using the default constraint and a suspended process as an alternative that does not use the default. The alternative process is suspended since it is considered to have a lower probability of success.

The complete operational semantics of the iteration step in *process reduction* is disclosed in Algorithm 2 of Annex B with an exact correspondence of case numbers to those of Fig. 4.

Fig.4

3.4 Fact Arrival Phase

Suppose a constraint is returned from an agent denoting an information source S for a question $Q@S$. This triggers a *fact arrival* phase, whose procedures are outlined in Fig. 5. The returned constraint is denoted as "$Q@S \hookleftarrow C_r \parallel$".

In *fact arrival*, the set of returned facts RF, which keeps all the replies from the agent information sources, is updated with the fact constraint. The CBS is also updated with the fact returned from S, which means the old default constraint is replaced with the fact constraint. After that, three cases are considered.

In Case 1 of Fig. 5, the reply entails the default constraint, and, as such, the processes using the default in the set of active processes APS are replaced with updated versions. These updated versions take into account the new fact constraint and carry out the execution trace in their origin. The set of suspended processes SPS is revised as well. Suspended processes which use the default constraint are also replaced with updated versions. Additionally, the processes suspended because of the default constraint, identified as the ones having the corresponding predicate in the suspended atom set SAS, are removed from the computation.

In Case 2, the reply contradicts the default constraint. The active processes using the default are removed and the suspended processes which are consistent

with the new fact constraint are resumed. Having a *default revision* phase has repercussions on *fact arrival*. As a product of *default revision*, there can be suspended processes with multiple suspended askable atoms, and thus it is necessary to check whether, after *fact arrival*, the SAS of a suspended process becomes an empty set or not. If it does, the process can be resumed, otherwise it remains suspended.

Finally, in Case 3, the reply does not entail or contradict the default constraint, but is consistent with it. Only the active processes which are consistent with the new fact constraint are kept as updated versions in APS. Similarly, only the suspended processes which are consistent with the new fact constraint are activated and added as updated versions to APS. All the others are removed from the computation.

The complete operations for fact arrival are described in Algorithm 3 in Annex B.

Fig.5

3.5 Default Revision Phase

When a fact arrives, the Generation of Defaults (presented in Section 4) produces a set containing only the changed default constraint rules. This set is referred to as $New\Delta$ and each of its members is a new default denoted as "$Q_d@S \hookleftarrow C_{newd} \parallel$.". *Default revision* takes place after *fact arrival* and a round of *process reduction*, based on $New\Delta$. This phase denotes a change in the other default constraints as a result of *fact arrival*. For this to occur, $New\Delta$ cannot be an empty set.

The default revision phase consists in changing all the processes according to the new default constraints provided by the method for the generation of defaults. Active and suspended processes are revised in order to determine if their constraints are consistent with the new default constraint. According to the position of the atom attached to the new default, i.e., if it is a suspended atom or a used default, the existing processes may generate new active and suspended processes, but their execution trace is never removed, unlike what happens in the *fact arrival* phase. This keeps the scenarios coherent with the newly arrived fact constraints. It is an on-line dynamic mechanism because it responds to the replies from the outside agent information sources, updates the default constraints used in the computation, and keeps the processes consistent with those updates.

The procedures for *default revision* are shown in Fig. 6. Its first operation is to update the CBS so that future *process reduction* phases take the changed defaults into account.

In Case 1 of Fig. 6, the new default constraint entails the old default constraint, and, as such, the processes using the old default in the set of active processes APS are replaced with updated versions. These updated versions take into account the new default constraint and carry out the execution trace

in their origin. The set of suspended processes SPS is revised as well. Suspended processes which use the old default constraint are also replaced with updated versions. However, it is important to note that the portions of active and suspended processes that are consistent with the negation of the new default constraint (the same is to say inconsistent with the new default constraint) are still kept in the computation as suspended processes.

In Case 2 of Fig. 6, the new default constraint contradicts the old default constraint. The active processes using the old default become suspended and the suspended processes which are consistent with the new default constraint are resumed. However, suspended processes or parts of suspended processes which are consistent with the negation of the new default constraint are still kept in the computation as updated suspended versions.

Finally, in Case 3 of Fig. 6, the new default constraint does not entail or contradict the old, but is consistent with it. In this case both active and suspended processes are revised, and the parts of these processes which are consistent with the new default constraint continue or become active in the form of updated versions, while the parts that are consistent with the negation of the new default constraint are suspended.

A curious aspect of *default revision*, and a major difference from previous works [24,22,21,9], is that processes may be suspended due to default constraints about more than one askable atom. This happens when there is a revision of the SPS for a new default constraint that contradicts the old one. A process may be in a suspended state due to constraints about a variable unrelated to the new default, and the old default constraint may already be part of the process, which means that the corresponding askable atom is in the set of used defaults UD. In such a situation, it is necessary to remove that askable atom and add it to the suspended atom set, which means that the process is now suspended due to an additional constraint. As a result, it became necessary to perform additional operations for the management of askable atoms in *fact arrival*, since a process can only be resumed if it has an empty suspended atom set.

The complete operational semantics for the *default revision* phase is shown in the Algorithms 4 and 5 in Annex B. There is an exact correspondence between the cases in Fig. 6 and the cases featured in the algorithms.

Fig.6

3.6 Correctness of the Operational Model

The following claim shows the correctness of the operational model presented in Sections 3.3, 3.4, and 3.5:

Claim 1: Let SF_{DR} be a Speculative Computation Framework with Default Revision. Let P be an ordinary process, GS_{init} be an initial goal set, UD a set of askable atoms used as defaults, C an answer constraint obtained from the

operational model, and RF a set of rules returned from the other information sources when a reply arrives. Then, there exists an answer constraint C' w.r.t. the constraint framework $\langle \Sigma, \mathcal{P} \rangle$ and the reply set $RF \cup \{\delta(Q@S) | Q@S \in UD\}$ s.t. $\pi_V(C)$ entails $\pi_V(C')$ where V is the set of variables that occur in GS_{init}, and π_V is the projection of constraints onto V.

A sketch of proof for Claim 1 can be consulted in Annex C.

4 Generation of Default Constraints

The objective of this component of Speculative Computation with Default Revision is to produce the set of default constraint rules Δ and the set of changed default constraint rules $New\Delta$.

The Generation of Default Constraints encompasses two different types of procedures, both described in Fig. 7. The first is the Learning of the Default Model and it consists of steps to produce a BN model from which it is later possible to extract the default constraints. This extraction happens in the second procedure, the Collection of Defaults, which has two different sets of steps, depending on the moment of speculative computation the execution is at. Although a part of Speculative Computation with Default Revision, the Generation of Default Constraints is exterior to the operational model of the SF_{DR} described in Section **??** and assumes a supportive role.

The following sections describe our reasoning for choosing BNs as the underlying default model and the different steps in the Generation of Defaults.

4.1 Choosing the Underlying Default Model

In order to extract a meaningful set of default constraints that can fill in information gaps, the default generation method has to be data-driven. Additional requirements of the method are that it should be capable of finding existing dependence relationships between the variables of askable atoms in $\mathcal{P}$ so as to produce the most likely set of default constraints, and be transparent in the sense that it conveys those relationships in a clear way to both humans and machines. Within the scope of predictive modelling, the problem at hand deviates from a classification or regression problem because there is no target class/variable. Instead, the goal is to find the most probable collective state of information, i.e., the most likely values for variables whose real values are unknown. These requirements excluded predictive models such as Neural Networks or Support Vector Machines [32].

Taking these aspects into account, the attention turned to graphical probabilistic models, and more specifically to BNs, which are able to represent this information. With BNs it is possible to update the calculations according to evidences, which fits the kind of dynamic revision mechanism required for Speculative Computation with Default Revision. Furthermore, it is possible

to establish relationships of cause and effect as opposed to undirected models such as pure Markov Networks. The works in [30] and [29] reflect the main use of BNs in medicine, namely in the identification of disease dynamics for diagnosis, prognosis or sudden changes in the state of a patient. Although the medical field is a prominent domain of application for BNs, they have also been used in many other domains to model dependability and perform risk analysis and maintenance. For a comprehensive review please consult [31]. However, in the work proposed here, BNs are used as a support for speculative computation in order to generate default constraints, which is a new application of this knowledge representation model.

4.2 Learning of the Default Model

The Learning of the Default Model in Fig. 7 comprises four sequential steps. The first is the analysis of askable atoms, in which the speculative agent analyses the logic program component $\mathcal{P}$ of the SF_{DR} and identifies the relevant variables for speculative computation. In the example provided in Section 3.1, the *guideline engine* would identify t, n, and m as askable atoms and T, N, and M as relevant variables.

These variables are then used in the second step, the retrieval of relevant data from the database. The speculative agent retrieves data of previous cases about the identified variables from a local information repository. Referring again to the example in Section 3.1, the *guideline engine* would use the data about previous executions of the clinical guideline in the *local repository*.

The following step is to perform cross-validation with different BN learning algorithms on the retrieved data set in order to determine which one will generate the model with the best predictions on future data. BN learning algorithms can be divided into two groups according to their search strategy [25]: score-based learning and constraint-based learning. The former assign a score to each candidate BN and try to maximize it with an heuristic search, while the latter learn the network structure by analysing the probabilistic relations entailed by the Markov property of BNs with conditional independence tests. As this is not the main topic of the paper, additional details about structure learning are provided in [25]. The set of learning algorithms used in this step includes: two score-based learning algorithms, the Hill-Climbing (*hc*) and the Tabu-Search (*tabu*); three constraint-based search algorithms, the Grow-Shrink (*gs*), the Incremental Association (*iamb*), and the Chow-Liu (*chowliu*); and one hybrid algorithm, the Max-Min Hill-Climbing (*mmhc*). The loss function used in cross-validation is the *Log Likelihood Loss* (*logl*), typically applied to this kind of problem and defined as in Equation 1 [8]:

$$logl = -log[P(D|G)] \tag{1}$$

where D is the data used to learn the network structure and G is the graph structure produced by the algorithm. The *logl* provides a measure of the entropy that a model exports in order to keep its own entropy low. The

target is to minimize this value and choose the model with the lowest *logl* [8]. The *logl* is a common way to compare how well a distribution for a model fits the data. However, using the *logl* as a scoring function in learning BNs has the disadvantage of producing networks that are too complex and slightly overfitted. But the *logl* is at the core of other functions, such as the Akaike Information Criterion, the Bayesian Information Criterion, and so forth, which have penalties to compensate for that and produce more sparse structures [13]. The use of *logl* herein is intended as a demonstration of the role played by the scoring function in the Generation of Default Constraints and, as such, the simplest measure was used.

With the results from cross-validation, the best algorithm is selected and with it the BN is constructed in the last step of the Learning of the Default Model. The whole procedure takes place at a point in time before the execution of a program in the SF_{DR} and outputs a BN ready for the Collection of Defaults. Going back to the example of Section 3.1, the output would be a BN that establishes a relationship between the T, N, and M variables.

4.3 Collection of Defaults

The Collection of Defaults is a procedure by which the BN produced by the Learning of the Default Model is conditioned in order to obtain default constraints. It is based on the *maximum a posteriori* estimation (MAP), a form of posterior distribution in Bayesian statistics that allows one to obtain a point estimation of unobserved variables based on collected evidence. It is defined as in Equation 2 [11]:

$$\theta_{MAP} = \max_{\theta} P(\theta|e) \tag{2}$$

where θ represents the goal variables for which the estimation is calculated and e represents the available evidence. A MAP calculates the values of the unobserved variables that maximize the probability distribution, providing the most likely setting for the variables in the network.

The first set of steps of the Collection of Defaults described in Fig. 7 occurs before the execution of a program in the SF_{DR} by the speculative agent. It corresponds to the initial MAP estimation, in which there is no reply from the agent information sources, and thus no evidence to submit. In the MAP estimation the goal variables correspond to all the variables in askable atoms. The result is a set of values for each variable that, once retrieved, are put into the form of constraint rules like "$Q@S \hookleftarrow C \parallel$." where Q is an askable atom, S is an agent information source and C is a constraint. The constraints are used to build the default answer set Δ which is then delivered to the SF_{DR}.

The second set of steps in Fig. 7 occurs during the execution of a program in the SF_{DR}. The speculative agent keeps track of all the replies from the agent information sources and, when a fact constraint arrives with true information about a variable in an askable atom, it performs a MAP estimation using the

new fact and all the previous ones as evidence in order to obtain an estimation for the yet unobserved variables. If there are no previous replies, the newly arrived fact is the only evidence in the MAP. The resulting values are then converted to constraints and compared with the existing default constraints for the respective askable atoms. Those that are different are put into a new default answer set $New\Delta$ and then delivered to the SF_{DR}, triggering a *default revision* phase. This procedure takes place every time a fact arrives.

The Collection of Defaults and the *default revision* phase provide agents with a form of on-line revision capability. They ensure that the scenarios represented by active processes get progressively closer to the actual state of the real world. This is an advantage for agents which have to deal with problems that demand an outcome, even in the presence of incomplete information.

Fig.7

5 Example: Clinical Decision Support

The principles of Speculative Computation with Default Revision can be applied to various domains and tasks. To demonstrate the formalization of a specific situation, an example in clinical decision support was provided in Section 3.1. Now, the same example will be used to demonstrate how the default set Δ can be obtained in order to complete the SF_{DR}, based on the steps outlined in Section 4. Afterwards there is a step by step execution of the example in the framework. Incomplete information in the example may be due to the difficulty in determining the true value for a variable or problems in the communication between the *guideline engine*, which assumes the role of the speculative agent, and *ois*, with the role of agent information source. Another possibility is that the *ois* does not possess the information required by the *guideline engine*.

5.1 Default Model and Initial Set of Default Constraints

The first procedure to be applied from the Generation of Defaults, before the execution of the program in the SF_{DR} and any attempt from the *guideline engine* to solve the problem in the program, is the Learning of the Default Model. As stated above, the objective is to obtain a BN with the variables that are featured in the problem the speculative agent has to solve. From the analysis of askable atoms t, n, and m in the SF_{DR} of Section 3.1, it is necessary to extract data from the *local repository* about variables T, N, and M. Following the example provided above, data about TNM staging of 515 patients were used for the Learning of the Default Model and to assess the process. These patients had surgery to the colon and underwent colon cancer treatment following the above-mentioned guideline [1] at the Hospital of Braga, in Portugal.

In order to estimate the performance of the predictive model, 5-fold cross-validation was used, and its results, in terms of *logl* are shown in Fig. 8. The algorithms with the worst performance were the score-based, while the constraint-based and the hybrid algorithms performed fairly better. Given that the *iamb* is the one with the lowest *logl* ($logl = 1.815$), it was selected to construct the BN, yielding the network represented in Fig. 9. As can be seen, the algorithm was able to establish a dependence relationship between T and N, but not with M, which means that M remains unaffected by changes in the other two variables. This is particularly useful for it indicates to the speculative agent that tumor invasion and local lymph node metastases might be correlated, but distant metastases are not dependent on the other two. This type of conclusion is something the BN is capable of providing and has an impact in the generation of default constraints. For instance, it is possible to know that if a value for T is known to be true, it might affect the default constraint of N, but it will not affect the default constraint of M.

With the BN of Fig. 9, it is possible to proceed to the Collection of Defaults in order to generate the initial set of default constraints, before the execution of the program in the SF_{DR}. The goal variables are T, N, and M. After performing the MAP estimation with no evidence on the network, the values obtained were respectively $T = t3, N = n2$, and $M = m1$, with $TNM_{MAP} = \max_{T,N,M} P(T, N, M \mid \emptyset) \approx 0.4395$. These will be the values used as initial default constraints in speculative computation. The default answer set is thus defined as follows:

– Δ is the following set of rules:
  $t(T)@ois \hookleftarrow T \in \{t3\} \parallel$.
  $n(N)@ois \hookleftarrow N \in \{n2\} \parallel$.
  $m(M)@ois \hookleftarrow M \in \{m1\} \parallel$.

The Δ set is added to the SF_{DR} defined for the example in Section 3.1 and now all the elements of the framework are gathered. The network is later used during the execution phases in order to derive new default constraints according to newly arrived facts.

All the resources for the learning and evaluation of BNs mentioned above are available in the *bnlearn* library for R [25], and the resources for the MAP estimation are available in the Java *inflib* library from the SamIam project [4].

Fig.8

Fig.9

5.2 Execution of the Example and Discussion

Now, based on the formalization provided in Section 3.1 and predictive model provided in Section 5.1, the execution of the program in the example is shown. The starting point is the already specified $SF_{DR} = \langle \Sigma, \mathcal{E}, \Delta, \mathcal{P} \rangle$. The operations in Sections 3.3, 3.4, and 3.5 are applied to the program in $\mathcal{P}$.

A goal in the goal set GS at a leftmost position from a newly created or newly resumed process is always selected for reduction. In the representation, the selected goal is underlined, and, if a set remains unchanged from one step to another, it is omitted. The following steps represent the execution trace for the query $nt(question1, F)$, which aims to determine which task should be recommended after *question1* in the management of a patient with colon cancer:

1. **Process Reduction, Initial Step (Fig. 3):**
   - The query originates the first active process and takes the place of its first goal for reduction;
   - There are no suspended processes, already asked questions, or returned facts, and, as such, the corresponding sets are empty;
   - The current belief state assumes the constraint rules in the default answer set.

   $APS = \{\langle\{\hookleftarrow\|\underline{nt(question1, F)}\}, \emptyset\rangle\}$

   $SPS = \emptyset$

   $AAQ = \emptyset$

   $RF = \emptyset$

   $CBS = \{$
   $t(T)@ois \hookleftarrow T \in \{t3\}\ \|,$
   $n(N)@ois \hookleftarrow N \in \{n2\}\ \|,$
   $m(M)@ois \hookleftarrow M \in \{m1\}\ \|$
   $\}$

2. **Process Reduction, by Case 2.1 (Fig. 4):**
   - $nt(question1, F)$ is not an askable atom and, as such, it unifies with the head of a rule in $\mathcal{P}$;
   - The body of the rule replaces $nt(question1, F)$ in the goal set of the updated version of the active process;
   - Since $alt(question1, F)$ is the only element in the goal set, it is selected for the next reduction step.

   $APS = \{\langle\{\hookleftarrow\|\underline{alt(question1, F)}, tcv(F)\}, \emptyset\rangle\}$

3. **Process Reduction, by Case 2.1 (Fig. 4):**
   - $alt(question1, F)$ is not an askable atom and, as such, it unifies with the head of five rules in $\mathcal{P}$, splitting the active process into five new active processes;
   - The constraints of the rules are added to the constraints of the processes they originated;

– Since $tcv(F)$ is at a leftmost position in the goal set, it it selected for the next reduction step in the active processes.

$APS = \{$
$\langle \{\hookleftarrow F \in \{action1\} \parallel \underline{tcv(F)}\}, \emptyset \rangle,$
$\langle \{\hookleftarrow F \in \{action2\} \parallel \underline{tcv(F)}\}, \emptyset \rangle,$
$\langle \{\hookleftarrow F \in \{action3\} \parallel \underline{tcv(F)}\}, \emptyset \rangle,$
$\langle \{\hookleftarrow F \in \{action4\} \parallel \underline{tcv(F)}\}, \emptyset \rangle,$
$\langle \{\hookleftarrow F \in \{action5\} \parallel \underline{tcv(F)}\}, \emptyset \rangle$
$\}$

4. **Process Reduction, by Case 2.1 (Fig. 4):**
   – $tcv(F)$ is not an askable atom and, as such, it unifies with the head of rules in $\mathcal{P}$, splitting the active processes into new active processes according to the consistency of the constraints in the process and the constraints in the rules;
   – The body of the rules replaces $tcv(F)$ in the goal set of the updated versions of the active processes and the constraints of the rules are added to the processes;
   – Since $t(T)@ois$ is at a leftmost position in the goal set of active processes, it it selected for the next reduction step.

$APS = \{$
$\langle\{\hookleftarrow F \in \{action1\}, T \in \{tis, t0\} \parallel \underline{t(T)@ois}\}, \emptyset\rangle,$
$\langle\{\hookleftarrow F \in \{action1\}, T \in \{t1, t2\}, N \in \{n0\}, M \in \{m0\} \parallel \underline{t(T)@ois}, n(N)@ois, m(M)@ois\}, \emptyset\rangle,$
$\langle\{\hookleftarrow F \in \{action2\}, T \in \{t3\}, M \in \{m0\} \parallel \underline{t(T)@ois}, m(M)@ois\}, \emptyset\rangle,$
$\langle\{\hookleftarrow F \in \{action3\}, T \in \{t4\}, N \in \{n0\}, M \in \{m0\} \parallel \underline{t(T)@ois}, n(N)@ois, m(M)@ois\}, \emptyset\rangle,$
$\langle\{\hookleftarrow F \in \{action4\}, T \in \{t1, t2, t3, t4\}, N \in \{n1, n2\}, M \in \{m0\} \parallel t(T)@ois, n(N)@ois, m(M)@ois\}, \emptyset\rangle,$
$\langle\{\hookleftarrow F \in \{action5\}, T \in \{t1, t2, t3, t4\}, N \in \{n0, n1, n2\}, M \in \{m1\} \parallel \underline{t(T)@ois}, n(N)@ois, m(M)@ois\}, \emptyset\rangle$
$\}$

5. **Process Reduction, by Case 2.2 (Fig. 4):**

$t(T)$ is asked to ois and since $(t(T)@ois \hookleftarrow T \in \{t3\} \parallel) \in \Delta$):

- $t(T)@ois$ is added to AAQ denoting that a question for this askable atom has already been sent;
- Since the CBS holds the default constraint rule $(t(T)@ois \hookleftarrow T \in \{t3\} \parallel)$, the active processes are reduced with it;
- The processes or parts of processes which are consistent with the default constraint remain active as updated versions with $t(T)@ois$ in the used default set;
- The processes or parts of processes which are inconsistent with the default constraint become suspended as updated versions with $t(T)@ois$ in the suspended atom set;
- Since $n(N)@ois$ is at a leftmost position in the goal set of active processes, it it selected for the next reduction step.

$APS = \{$
$\langle\{\hookleftarrow F \in \{action2\}, T \in \{t3\}, M \in \{m0\} \parallel m(M)@ois\}, \{t(T)@ois\}\rangle,$
$\langle\{\hookleftarrow F \in \{action4\}, T \in \{t3\}, N \in \{n1, n2\}, M \in \{m0\} \parallel \underline{n(N)@ois}, m(M)@ois\}, \{t(T)@ois\}\rangle,$
$\langle\{\hookleftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n0, n1, n2\}, M \in \{m1\} \parallel \underline{n(N)@ois}, m(M)@ois\}, \{t(T)@ois\}\rangle$
$\}$

$SPS = \{$
$\langle\{\{t(T)@ois\}, \hookleftarrow F \in \{action1\}, T \in \{tis, t0\} \parallel \emptyset\}, \emptyset\rangle^{P_1},$
$\langle\{\{t(T)@ois\}, \hookleftarrow F \in \{action1\}, T \in \{t1, t2\}, N \in \{n0\}, M \in \{m0\} \parallel n(N)@ois, m(M)@ois\}, \emptyset\rangle^{P_2},$
$\langle\{\{t(T)@ois\}, \hookleftarrow F \in \{action3\}, T \in \{t4\}, N \in \{n0\}, M \in \{m0\} \parallel n(N)@ois, m(M)@ois\}, \emptyset\rangle^{P_3},$
$\langle\{\{t(T)@ois\}, \hookleftarrow F \in \{action4\}, T \in \{t1, t2, t4\}, N \in \{n1, n2\}, M \in \{m0\} \parallel n(N)@ois, m(M)@ois\}, \emptyset\rangle^{P_4},$

$\langle\{\underline{\{t(T)@ois\}}, \hookleftarrow F \in \{action5\}, T \in \{t1, t2, t4\}, N \in \{n0, n1, n2\}, M \in \{m1\} \parallel n(N)@ois, m(M)@ois\}, \emptyset\rangle^{P_5}$
}

$AAQ = \{t(T)@ois\}$

6. **Process Reduction, by Case 2.2 (Fig. 4):**

$n(N)$ is asked to *ois* and since $(n(N)@ois \hookleftarrow N \in \{n2\} \parallel) \in \Delta$):
- $n(N)@ois$ is added to AAQ denoting that a question for this askable atom has already been sent;
- Since the CBS holds the default constraint rule $(n(N)@ois \hookleftarrow N \in \{n2\} \parallel)$, the active processes are reduced with it;
- The processes or parts of processes which are consistent with the default constraint remain active as updated versions with $n(N)@ois$ in the used default set;
- The processes or parts of processes which are inconsistent with the default constraint become suspended as updated versions with $n(N)@ois$ in the suspended atom set;
- Since $m(M)@ois$ is the only element in the goal set of active processes, it it selected for the next reduction step.

$APS = \{$
$\langle\{\hookleftarrow F \in \{action2\}, T \in \{t3\}, M \in \{m0\} \parallel \underline{m(M)@ois}\}, \{t(T)@ois\}\rangle,$
$\langle\{\hookleftarrow F \in \{action4\}, T \in \{t3\}, N \in \{n2\}, M \in \{m0\} \parallel \underline{m(M)@ois}\}, \{t(T)@ois, (N)@ois\}\rangle,$
$\langle\{\hookleftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n2\}, M \in \{m1\} \parallel \underline{m(M)@ois}\}, \{t(T)@ois, n(N)@ois\}\rangle$
}

$SPS = \{$
$\langle\{\underline{\{n(N)@ois\}}, \hookleftarrow F \in \{action4\}, T \in \{t3\}, N \in \{n1\}, M \in \{m0\} \parallel m(M)@ois\}, \{t(T)@ois\}\rangle^{P_6},$
$\langle\{\underline{\{n(N)@ois\}}, \hookleftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n0, n1\}, M \in \{m1\} \parallel m(M)@ois\}, \{t(T)@ois\}\rangle^{P_7},$
P_1, P_2, P_3, P_4, P_5
}

$AAQ = \{t(T)@ois, n(N)@ois\}$

7. **Process Reduction, by Case 2.2 (Fig. 4):**

$m(M)$ is asked to *ois* and since $((m(M)@ois \hookleftarrow M \in \{m1\} \parallel) \in \Delta$):
- $m(M)@ois$ is added to AAQ denoting that a question for this askable atom has already been sent;

- Since the CBS holds the default constraint rules $(m(M)@ois \hookleftarrow M \in \{m1\} \parallel)$, the active processes are reduced with it;
- The processes or parts of processes which are consistent with the default constraint remain active as updated versions with $m(M)@ois$ in the used default set;
- The processes or parts of processes which are inconsistent with the default constraint become suspended as updated versions with $m(M)@ois$ in the suspended atom set;

$APS = \{$
$\langle\{\hookleftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n2\}, M \in \{m1\} \parallel \emptyset\}, \{t(T)@ois, n(N)@ois, m(M)@ois\}\rangle^{P_8}$
$\}$

$SPS = \{$
$\langle\{\underline{\{m(M)@ois\}}, \hookleftarrow F \in \{action2\}, T \in \{t3\}, M \in \{m0\} \parallel \emptyset\}, \{t(T)@ois\}\rangle^{P_9}$,
$\langle\{\underline{\{m(M)@ois\}}, \hookleftarrow F \in \{action4\}, T \in \{t3\}, N \in \{n2\}, M \in \{m0\} \parallel \emptyset\}, \{t(T)@ois, n(N)@ois\}\rangle^{P_{10}}$,
$P_1, P_2, P_3, P_4, P_5, P_6, P_7$
$\}$

$AAQ = \{t(T)@ois, n(N)@ois, m(M)@ois\}$

8. **Process Reduction, by Case 1 (Fig. 4):**

$APS = \{$
$\langle\{\hookleftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n2\}, M \in \{m1\} \parallel \emptyset\}, \{t(T)@ois, n(N)@ois, m(M)@ois\}\rangle^{P_8}$
$\}$

$C = \{F \in \{action5\}, T \in \{t3\}, N \in \{n2\}, M \in \{m1\}\}$

$UD = \{t(T)@ois, n(N)@ois, m(M)@ois\}$

Up until step 4 *process reduction* occurs with the unification of the goals in the GS with the head of rules in $\mathcal{P}$. The initial CBS, at step 1, assumes the values in Δ. Initial SPS, AAQ, and RF are empty since there are no suspended processes, no questions were asked to the agent information sources, and no replies were received. At step 1, the first active process is created with the query as its initial goal in GS. As a consequence of several steps of *process reduction*, the process is split into new active processes with new goal sets. By step 4 the active processes represent the different alternative tasks.

At step 5, the agent needs to reduce processes with $t(T)@ois$ as the selected goal and, since there is only a default constraint for that goal in the CBS, that constraint is used in *process reduction*, yielding active processes which are

consistent with the default and suspended processes which are not. $t(T)@ois$ is an askable atom and, as such, a question is sent to *ois* so as to know the real value of $t(T)@ois$. The procedure is similar whenever an askable atom is found. It is possible to observe that the active processes now have $t(T)@ois$ in UD and this askable atom was removed from the GS. At the same time, the suspended processes resulting from the reduction have $t(T)@ois$ as a suspended atom in the SAS. The results of disjunctive constraint processing are also observable; an example is the splitting of the active process concerning *action4* in step 4. At step 5 this process originates an active process and a suspended process.

By continuing *process reduction* it is possible to arrive at step 7, where there is a process with an empty goal set. This was achieved purely by relying on default constraints and, as such, the process is a scenario. By outputting C and UD at step 8, the speculative agent informs that the most likely task to follow *question1* is *action5*, which recommends a series of workup exams such as a colonoscopy, an abdominal/pelvic CT, and so forth. Based on this, the health care professional may start the preparations for executing this task. The effect of Speculative Computation with Default Revision is observed here; with no information whatsoever, it is possible to arrive at the most likely scenario in terms of probabilities, represented as an answer to the initial query. Depending on the rules in $\mathcal{P}$, there might be cases in which more than one active process with an empty goal set is produced. In such situations all the processes are presented as scenarios.

Assuming the information arrives from the *ois* at some point, regardless of the order, one may have:

9. $(n(N)@ois \hookleftarrow N \in \{n0\} \parallel)$ is returned from *ois* and by **Fact Arrival Phase (Fig. 5)**:
   - The returned fact replaces the default constraint in the CBS;
   - The returned fact is added to RF;

$CBS = \{$
$t(T)@ois \hookleftarrow T \in \{t3\} \parallel,$
$\underline{n(N)@ois \hookleftarrow N \in \{n0\} \parallel},$
$m(M)@ois \hookleftarrow M \in \{m1\} \parallel$
$\}$

$RF = \{\underline{n(N)@ois \hookleftarrow N \in \{n0\} \parallel}\}$

   - Process P_8 in the APS, along with processes P_6 and P_{10} from SPS are removed because they are inconsistent with the returned fact;
   - P_7 is resumed and originates P_{11};
   - The newly activated process P_{11} still has goals in the goal set and thus $m(M)@ois$ is selected for process reduction.

$APS = \{$

$\langle\{\hookleftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n0\}, M \in \{m1\} \parallel \underline{m(M)@ois}\}, \{t(T)@ois\}\rangle^{P_{11}}$
}

$SPS = \{P_1, P_2, P_3, P_4, P_5, P_9\}$

10. **Process Reduction, by Case 2.2 (Fig. 4):**

Since $m(M)@ois \in AAQ$ and $((m(M)@ois \hookleftarrow M \in \{m1\} \parallel) \in \Delta)$:

$APS = \{$
$\langle\{\hookleftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n0\}, M \in \{m1\} \parallel \emptyset\}, \{t(T)@ois, m(M)@ois\}\rangle^{P_{11}}$
}

$SPS = \{P_1, P_2, P_3, P_4, P_5, P_9\}$

11. **By Collection of Defaults:**

Since there is now a returned fact for $n(N)@ois$, the MAP estimation is $TM_{MAP} = \max_{T,M} P(T, M \mid N = n0) \approx 0.9126$ with $T = t1$ and $M = m1$, thus producing $New\Delta = \{t(T)@ois \hookleftarrow T \in \{t1\} \parallel\}$.

As such, by **Default Revision Phase (Fig. 6):**
- The new default constraint replaces the old default constraint in the CBS;

$CBS = \{$
$\underline{t(T)@ois \hookleftarrow T \in \{t1\} \parallel},$
$\overline{n(N)@ois \hookleftarrow N \in \{n0\} \parallel},$
$m(M)@ois \hookleftarrow M \in \{m1\} \parallel$
}
- Process P_{11} becomes suspended on $t(T)@ois$;
- Processes P_2, P_4 and P_5 are resumed, forming processes P_{12} and P_{13}, P_{14} and P_{15}, and P_{16} and P_{17}, respectively;
- Process P_9 originates process P_{18} because a new atom is added to the suspended atom set;
- The newly activated processes still have elements in the goal set, so they are further reduced according to the CBS.

$APS = \{$
$\langle\{\hookleftarrow F \in \{action1\}, T \in \{t1\}, N \in \{n0\}, M \in \{m0\} \parallel \underline{n(N)@ois}, m(M)@ois\}, \{t(T)@ois\}\rangle^{P_{12}},$
$\langle\{\hookleftarrow F \in \{action4\}, T \in \{t1\}, N \in \{n1, n2\}, M \in \{m0\} \parallel \underline{n(N)@ois}, m(M)@ois\}, \{t(T)@ois\}\rangle^{P_{14}},$

$\langle \hookleftarrow F \in \{action5\}, T \in \{t1\}, N \in \{n0, n1, n2\}, M \in \{m1\} \parallel \underline{n(N)@ois}, m(M)@ois\}, \{t(T)@ois\}\rangle^{P_{16}}$
}

$SPS = \{$
$\langle \{\underline{\{t(T)@ois\}}, \hookleftarrow F \in \{action1\}, T \in \{t2\}, N \in \{n0\}, M \in \{m0\} \parallel n(N)@ois, m(M)@ois\}, \emptyset\rangle^{P_{13}}$,
$\langle \{\underline{\{t(T)@ois\}}, \hookleftarrow F \in \{action4\}, T \in \{t2, t4\}, N \in \{n1, n2\}, M \in \{m0\} \parallel n(N)@ois, m(M)@ois\}, \emptyset\rangle^{P_{15}}$,
$\langle \{\underline{\{t(T)@ois\}}, \hookleftarrow F \in \{action5\}, T \in \{t2, t4\}, N \in \{n0, n1, n2\}, M \in \{m1\} \parallel n(N)@ois, m(M)@ois\}, \emptyset\rangle^{P_{17}}$,
$\langle \{\{m(M)@ois, \underline{t(T)@ois}\}, \hookleftarrow F \in \{action2\}, T \in \{t3\}, M \in \{m0\} \parallel \emptyset\}, \emptyset\rangle^{P_{18}}$,
P_1, P_3, P_{11}
}

12. After steps of **Process Reduction, by Case 2.2 (Fig. 4):**
    - Process P_{14} is deleted because it is inconsistent with the returned fact for $n(N)@ois$;
    - Process P_{12} is suspended.

$APS = \{$
$\langle \{\hookleftarrow F \in \{action5\}, T \in \{t1\}, N \in \{n0\}, M \in \{m1\} \parallel \emptyset\}, \{t(T)@ois, m(M)@ois\}\rangle^{P_{16}}$
}

$SPS = \{$
$\langle \{\underline{\{m(M)@ois\}}, \hookleftarrow F \in \{action1\}, T \in \{t1\}, N \in \{n0\}, M \in \{m0\} \parallel \emptyset\}, \{t(T)@ois\}\rangle^{P_{12}}$,
$P_1, P_3, P_{11}, P_{13}, P_{15}, P_{17}, P_{18}$
}

13. $(t(T)@ois \hookleftarrow T \in \{t1\} \parallel)$ is returned from *ois* and by **Fact Arrival Phase (Fig. 5):**
    - The returned fact coincides with the default constraint;
    - The returned fact is added to RF;

CBS remains unchanged.

$RF = \{\underline{t(T)@ois \hookleftarrow T \in \{t1\} \parallel}, n(N)@ois \hookleftarrow N \in \{n0\} \parallel\}$

- $t(T)@ois$ is removed from UD in the active processes because it is no longer a default.
- Processes $P_1, P_3, P_{11}, P_{13}, P_{15}, P_{17}$ and P_{18} are deleted.

$APS = \{$
$\langle\{\hookleftarrow F \in \{action5\}, T \in \{t1\}, N \in \{n0\}, M \in \{m1\} \parallel \emptyset\},$
$\{m(M)@ois\}\rangle^{P_{16}}$
$\}$

$SPS = \{P_{12}\}$

14. **By Collection of Defaults:**

    Since there is another returned fact, in this case for $(t(T)@ois$, the MAP estimation is $M_{MAP} = \max_{M} P(M \mid T = t1, N = n0) \approx 0.9126$ with $M = m1$, thus making it unnecessary to produce a $New\Delta$ and perform default revision.

15. $(m(M)@ois \hookleftarrow M \in \{m1\} \parallel)$ is returned from *ois* and by **Fact Arrival Phase (Fig. 5)**:
    - The returned fact coincides with the default constraint;
    - The returned fact is added to RF;

    CBS remains unchanged.

    $RF = \{$
    $t(T)@ois \hookleftarrow T \in \{t1\} \parallel,$
    $n(N)@ois \hookleftarrow N \in \{n0\} \parallel,$
    $\underline{m(M)@ois \hookleftarrow M \in \{m1\} \parallel}$
    $\}$

    - $m(M)@ois$ is removed from UD in the active processes because it is no longer a default;
    - Process P_{12} is deleted.

    $APS = \{\langle\{\hookleftarrow F \in \{action5\}, T \in \{t1\}, N \in \{n0\}, M \in \{m1\} \parallel \emptyset\}, \emptyset\rangle^{P_{16}}\}$

    $SPS = \emptyset$

16. **Process Reduction, by Case 1 (Fig. 4):**

    $APS = \{$
    $APS = \{\langle\{\hookleftarrow F \in \{action5\}, T \in \{t1\}, N \in \{n0\}, M \in \{m1\} \parallel \emptyset\}, \emptyset\rangle^{P_{16}}\}$
    $\}$

    $C = \{F \in \{action5\}, T \in \{t1\}, N \in \{n0\}, M \in \{m1\}\}$

    $UD = \emptyset$

    End of the execution.

In *fact arrival*, active processes may be deleted and suspended processes may be resumed as the answers from the *ois* are regarded as definitive. At step 9 there are two new active processes resulting from the arrival of a fact for $n(N)@ois$. The CBS is updated with the fact "$n(N)@ois \hookleftarrow N \in \{n0\} \parallel$", replacing the default constraint. All the processes (both active and suspended) that use the default constraint are removed from the execution, resulting in the elimination of processes P_8, P_6, and P_{10}. On the other hand, process P_7 is resumed and, after processing the constraints for $n(N)@ois$, it originates process P_{11}. In this process $n(N)@ois$ does not appear in UD since the askable atom is not used as a default. Furthermore, upon resuming process P_7, no alternative suspended process is generated because there cannot be another value for $n(N)@ois$ besides the newly arrived fact. Upon the arrival of the fact "$n(N)@ois \hookleftarrow N \in \{n0\} \parallel$" from the *ois*, the system does not add a brand new suspended process "$\langle\{\hookleftarrow F \in \{action5\}, T \in \{t3\}, N \in \{n1\}, M \in \{m1\} \parallel m(M)@ois\}, \{t(T)@ois\}\rangle$", generated from P_7, because it would not represent a feasible scenario. Further *process reduction* results in the processes of step 10.

After applying the Collection of Defaults using the newly arrived fact as evidence for the MAP estimation, a $New\Delta$ set is produced with a new default constraint for $t(T)@ois$. As such, *default revision* is performed at step 11. $t(T)@ois \hookleftarrow T \in \{t1\} \parallel$ replaces the old default constraint in the CBS and process P_{11} is suspended because it becomes inconsistent with the new default constraint. Previously suspended processes are resumed, yielding new active processes consistent with the new default constraint and new suspended processes consistent with the negation of the default constraint. That is what happens with processes P_2, P_4, and P_5, each one originating two new processes. Another effect of Speculative Computation with Default Revision is observed in the change operated in P_9. This process used the previous default for $t(T)@ois$ and, when revised with the new default constraint, it becomes P_{18} which is suspended on the account of two askable atoms. If this newly suspended process were to be activated it would have to be revised for both $m(M)@ois$ and $t(T)@ois$, and SAS would have to become an empty set. This situation can only be created with default revision. At the same time, the probability value in the MAP estimation of this step is higher than the initial MAP estimation, which indicates that the collective state of the information and, thus, the scenarios resulting from the *default revision* phase are more likely to be closer to the real outcome.

At the end of step 12, after *process reduction*, there is an active process representing the most likely scenario given the configuration of the information at the time, achieved through facts and default constraints. This is another observable effect of Speculative Computation with Default Revision; the default constraints adjusted themselves to the arrived fact, which, in turn, triggered the adjustment of processes, resulting in a scenario that is closer to reality. It is an example of how the on-line *default revision* takes place. This would inform a health care professional executing the guideline that, although the prediction changed, the most plausible answer remains the same, that is $action5$. At step

13, there is another reply from *ois*, which triggers a *fact arrival* phase, but, as the fact matches the default constraint, the change in the active process is only the removal of $t(T)@ois$ from UD. All the suspended processes that are not consistent with the newly arrived fact are removed. The MAP estimation, from the Collection of Defaults at step 14, produces the same value as in step 11 due to M being independent from the other two variables, which means that a $New\Delta$ is not produced. The remaining execution represents the arrival of the information that is still missing, until a process composed solely of facts is obtained. This happens at step 16, by outputting C and UD. The answer remains the recommendation of $action5$, achieved through a different path. The process that originated the solution, P_{16}, suffered changes, going from a process using facts and defaults to a process using only facts. This demonstrates how a speculative computation can be used in order to advance the computation of an answer and then be utilized to achieve a definitive answer.

Demonstrating all the possible cases in Speculative Computation with Default Revision with an example would be impractical. The example above only shows its main effects. However, with the claim of Section 3.6 it is possible to show that the procedures in the different phases of the SF_{DR} yield correct results.

It is possible to argue that simply choosing the most probable set of default constraints and deriving the most likely situation as a basis for action, with no regard for the impact that action could have, may be unrealistic or, at the very least, incomplete in terms of decision criteria.

Indeed, the framework presented herein lacks a more comprehensive and well-defined strategy for selecting not only the most plausible scenario, but also the most useful from the point of view of its costs and potential benefits. The use of probabilities, and more specifically BNs, is motivated by the notion that the knowledge we have about the world is imperfect and that, through a Bayesian approach, it is possible to get a degree of belief that something may be the case. As a complement to this degree of belief, an assessment of the strengths and weaknesses of the different alternative actions that satisfy a scenario could, or rather should, be combined with the probability resulting from the MAP assignments. This would ensure that the costs of taking a certain action do not outweigh (or far outweigh, when compared with other actions) its benefits. Cost-effectiveness analysis is used across multiple domains to appraise the desirability of an alternative, given a certain decision making problem [3]. It does so by comparing the relative costs and outcomes (effects) of two or more courses of action. Cost-effectiveness analysis is different from cost-benefit analysis in that it does not assign a monetary value to the measure of effect. It is widely used in health care for allocation decisions and to highlight interventions that are relatively inexpensive, but have the potential to reduce the disease burden substantially [18]. Taking this domain as an illustrative example, in the available literature for this kind of analysis, the costs are expressed in monetary units and the effectiveness of interventions are expressed in units of measure that are characteristic of health care such as years of life gained with an intervention or the disability-adjusted life

year (which measures not only the years gained but also the improved health). By doing this, cost-effectiveness analysis, unlike cost-benefit analysis, draws attention exclusively to health benefits, which are not monetized. Another element commonly taken into account in cost-effectiveness analysis is time, more specifically the changes in costs and effects of interventions with the passage of time, which enhances the utility of an evaluation and makes it more realistic [28]. Clinical guidelines often provide references to cost-effectiveness studies regarding their recommendations. Although the guideline from which this example was retrieved does not go as far as to quantify in any way the costs and effects of the alternatives featured in the example, it features numerous references of this type [1]. The main point to retain is that the combination of MAP assignments with this kind of analysis would capture a given setting in a more realistic way and, therefore, would result in more helpful scenarios.

That being said, cost-effectiveness analysis has inherent challenges and involves a sequence of steps until a utility is achieved. Every analysis requires a host of assumptions, complex calculations, and the careful judgement of analysts. Furthermore, it is necessary to quantify the exact costs and effects of alternatives, which is not always clear and available in the knowledge about the domain. Thus, setting the framework for this kind of analysis would only be possible for very specific domains where this information is available.

6 Conclusions and Future Work

In this work, BNs assume a supportive role, which is slightly different from their common application. Here, the BN model itself is not the target, but an auxiliary tool. With arrived facts MAP estimation values may change, yielding new estimations for the information that is still unobserved. By dynamically revising default constraints it becomes possible to ensure that the information has the most likely configuration at all times and, in this way, produce the most likely scenarios. Speculative Computation with Default Revision is a way to manage the state of the information, fulfilling the role of an interface to rule-guided problems. The mechanism for default revision is richer and more encompassing than those presented in previous works. At the same time, it is an on-line process that reflects the dynamic nature of real environments. It takes into account the arrival of new facts to change default constraints, which is a more realistic perspective about real-world problems.

As far as we know this is the first on-line framework that uses machine learning techniques to adapt default rules. It provides a combination of logic programming-based, declarative user interface and an effective, adaptive operational behaviour in the background.

Future work includes further developing the model in order to compute utility values for the scenarios and make full use of the probability provided in the MAP estimation. So far, this probability is only used in the background, but it would be useful to include it in the different phases of Speculative Computation as a likelihood measure. As a complement to the MAP assign-

ments and in order to build a utility function for speculative computation, cost-effectiveness can be explored as a strategy to depict the usefulness of alternatives more accurately and realistically. Furthermore, this kind of analysis can be made in such a way that tackles the importance of time in these decision problems. In certain cases, it may be necessary to decide before the state of a system deteriorates and changes irreversibly, which undoubtedly would affect the utility of the scenarios generated for it. This is even more evident in the domain chosen as an example for this work. In clinical decision support, health care professionals may have to make hard decisions before the state of a patient gets worse. As such, we are also working towards the inclusion of time as a relevant dimension in Speculative Computation.

A small prototype was developed for this example. The Generation of Defaults was implemented using the resources provided by the *bnlearn* library for R [25] and the Java *inflib* library from the SamIam project [4]. The Speculative Computation with Default Revision operational model was implemented in Prolog.

References

1. Benson, A., Bekaii-Saab, T., Chan, E., Chen, Y.J., Choti, M., Cooper, H., Engstrom, P.: NCCN clinical practice guideline in oncology colon cancer. Tech. rep., National Comprehensive Cancer Network (2013). URL `http://www.nccn.org/professionals/physician\_gls/f\_guidelines.asp`
2. Burton, F.W.: Speculative computation, parallelism and functional programming. IEEE Transactions on Computers **34**(12), 1190–1193 (1985)
3. Cellini, S.R., Kee, J.E.: Cost Effectiveness and Cost-Benefit Analysis. In: Handbook of Practical Program Evaluation, pp. 493–530. Wiley Online Library (2010)
4. Darwiche, A.: Samiam - sensitivity analysis, modelling, inference and more. url-http://reasoning.cs.ucla.edu/samiam/ (2010). Accessed: 2015-03-15
5. Franzén, T., Haridi, S., Janson, S.: An overview of the andorra kernel language. In: L.H. Eriksson, L. HallnÃđs, P. Schroeder-Heister (eds.) Extensions of Logic Programming, *Lecture Notes in Computer Science*, vol. 596, pp. 163–179. Springer Berlin Heidelberg (1992)
6. Governatori, G., Olivieri, F., Scannapieco, S., Cristani, M.: Superiority based revision of defeasible theories. In: M. Dean, D.J. Hall, A. Rotolo, S. Tabet (eds.) Semantic Web Rules, *Lecture Notes in Computer Science*, vol. 6403, pp. 104–118. Springer Berlin Heidelberg (2010)
7. Gupta, V., Jagadeesan, R., Saraswat, V.: Probabilistic concurrent constraint programming. In: A. Mazurkiewicz, J. Winkowski (eds.) CONCUR'97: Concurrency Theory, *Lecture Notes in Computer Science*, vol. 1243, pp. 243–257. Springer Berlin Heidelberg (1997)
8. Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Friedman, J., Tibshirani, R.: The elements of statistical learning: data mining, inference, and prediction, 2 edn. Springer-Verlag, New York (2009)
9. Hosobe, H., Satoh, K., Codognet, P.: Agent-based speculative constraint processing. IEICE Transactions on Information and Systems **E90-D**(9), 1354–1362 (2007)
10. Isern, D., Moreno, A.: Computer-based execution of clinical guidelines: a review. International Journal of Medical Informatics **77**(12), 787–808 (2008)
11. Korb, K., Nicholson, A.: Bayesian artificial intelligence, 2 edn. CRC Press, London (2011)
12. Lam, H.P., Governatori, G., Satoh, K., Hosobe, H.: Distributed defeasible speculative reasoning in ambient environment. In: M. Fisher, L. Torre, M. Dastani, G. Governatori

(eds.) Proceedings of the Computational Logic in Multi-Agent Systems: 13th International Workshop, CLIMA XIII, Lecture Notes in Computer Science, pp. 43–60. Springer Berlin Heidelberg (2012)

13. Liu, Z., Malone, B., Yuan, C.: Empirical evaluation of scoring functions for bayesian network model selection. BMC Bioinformatics **13 Suppl 1**(Suppl 15), S14 (2012)
14. Oliveira, T., Neves, J., Novais, P., Satoh, K.: Applying speculative computation to guideline-based decision support systems. In: 2014 IEEE 27th International Symposium on Computer-Based Medical Systems (CBMS), pp. 42–47. IEEE (2014)
15. Oliveira, T., Novais, P., Neves, J.: Representation of clinical practice guideline components in owl. In: J.B. Pérez, J.M.C. Rodríguez, J. Fëhndrich, P. Mathieu, A. Campbell, M.C. Suarez-Figueroa, A. Ortega, E. Adam, E. Navarro, R. Hermoso, M.N. Moreno (eds.) Trends in Practical Applications of Agents and Multiagent Systems, *Advances in Intelligent Systems and Computing*, vol. 221, pp. 77–85. Springer International Publishing (2013)
16. Peleg, M.: Computer-interpretable clinical guidelines: a methodological review. Journal of Biomedical Informatics **46**(4), 744–63 (2013)
17. Prakken, H., Sartor, G.: Argument-based extended logic programming with defeasible priorities. Journal of Applied Non-Classical Logics **7**(1-2), 25–75 (1997)
18. Russell, L., Gold, M., Siegel, J., Daniels, N., Weinstein, M.: The role of cost-effectiveness analysis in health and medicine. JAMA **276**(14), 1172–1177 (1996)
19. Sadri, F., Toni, F.: Interleaving belief updating and reasoning in abductive logic programming. Frontiers in Artificial Intelligence and Applications **141**, 442–446 (2006)
20. Saraswat, V., Jagadeesan, R., Gupta, V.: Timed default concurrent constraint programming. Journal of Symbolic Computation **22**(5), 475–520 (1996)
21. Satoh, K.: Speculative computation and abduction for an autonomous agent. IEICE - Transactions on Information and Systems **E88-D**(9), 2031–2038 (2005)
22. Satoh, K., Codognet, P., Hosobe, H.: Speculative constraint processing in multi-agent systems. In: Intelligent Agents and Multi-Agent Systems, vol. 2891, pp. 133–144. Springer (2003)
23. Satoh, K., Inoue, K., Iwanuma, K., Sakama, C.: Speculative computation by abduction under incomplete communication environments. In: Proceedings of the Fourth International Conference on Multi-Agent Systems, vol. 12, pp. 263–270. IEEE (2000)
24. Satoh, K., Yamamoto, K.: Speculative computation with multi-agent belief revision. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems: Part 2, AAMAS '02, pp. 897–904. ACM (2002)
25. Scutari, M.: Learning bayesian networks with the bnlearn R package. Journal of Statistical Software **35**(3), 1–22 (2010)
26. Smolka, G.: The oz programming model. In: M. Broy, B. Schieder (eds.) Mathematical Methods in Program Development, *NATO ASI Series*, vol. 158, pp. 409–432. Springer Berlin Heidelberg (1997)
27. Ten Teije, A., Miksch, S., Lucas, P.: Computer-based medical guidelines and protocols: a primer and current trends, vol. 139. IOS Press (2008)
28. Van De Wetering, G., Woertman, W.H., Adang, E.M.: Time to incorporate time in cost-effectiveness analysis. European Journal of Health Economics **13**(3), 223–226 (2012). DOI 10.1007/s10198-011-0374-3
29. Van der Heijden, M., Lucas, P.: Describing disease processes using a probabilistic logic of qualitative time. Artificial Intelligence in Medicine **59**(3), 143–155 (2013)
30. Visscher, S., Lucas, P., Schurink, C., Bonten, M.: Modelling treatment effects in a clinical bayesian network using boolean threshold functions. Artificial Intelligence in Medicine **46**(3), 251–266 (2009)
31. Weber, P., Medina-Oliva, G., Simon, C., Iung, B.: Overview on bayesian networks applications for dependability, risk analysis and maintenance areas. Engineering Applications of Artificial Intelligence **25**(4), 671 – 682 (2012)
32. Witten, I., Frank, E., Hall, M.: Data mining: practical machine learning tools and techniques, 3 edn. Morgan Kaufmann, San Francisco (2011)

A Table of Abbreviations and Symbols

Table 1 Abbreviations and symbols used in the definitions and procedures of the Framework for Speculative Computation with Default Revision

Abbreviation / Symbol	Meaning
Σ	Set containing the agent identifiers representing the information sources
Δ	The default answer set
APS	The set of active process processes
AAQ	The set of already asked questions
$body(R)$	The body of rule R
C_d	A default constraint
C_{newd}	A new default constraint
CBS	The current belief state
C_r	A fact constraint
$const(R)$	The constraints in the body of rule R
$\mathcal{E}$	Set containing external predicates representing askable atoms
GS	The goal set in a process
$head(R)$	The head of rule R
$New\Delta$	The set of changed default constraint rules
$NewAAQ$	An updated version of AAQ resulting from a process reduction phase
$NewAPS$	An updated version of APS resulting from a phase in Speculative Computation with Default Revision
$NewCBS$	An updated version of CBS resulting from an answer arrival phase or a default revision phase
$NewRF$	An updated version of RF resulting from an answer arrival phase
$NewSPS$	An updated version of SPS resulting from a phase in Speculative Computation with Default Revision
$\mathcal{P}$	The constraint logic program
$\mathcal{R}$	The reply set
RF	The set of returned facts from the agent information sources
SAS	The suspended atom set
SF_{DR}	Framework for Speculative Computation with Default Revision
SPS	The set of suspended processes
UD	The used default set of a process

B Algorithms for Speculative Computation with Default Revision

The following algorithms specify the operations involved in the different phases of Speculative Computation with Default Revision.

Algorithms 1 and 2 refer to the initial step and iteration steps respectively of *process reduction*. In this phase, changes occur in the process sets. In the algorithms, changed APS, SPS, and AAQ are represented as $NewAPS$, $NewSPS$ and $NewAAQ$. The last three correspond to updated versions of the previous after the step of *process reduction* is applied.

Algorithm 1 Initial step of process reduction

Data: GS: the initial goal set
Data: Δ: the default answer set

1: give $\langle \hookleftarrow C \parallel GS, \emptyset \rangle$ to the proof procedure
2: $APS \leftarrow \{\langle \hookleftarrow C \parallel GS, \emptyset \rangle\}$
3: $SPS \leftarrow \emptyset$
4: $CBS \hookleftarrow \Delta$
5: $AAQ \leftarrow \emptyset$
6: $RF \leftarrow \emptyset$

Algorithm 2 Iteration step of process reduction

Data: Σ: the set of agent information sources
Data: APS: the set of active processes
Data: SPS: the set of suspended processes
Data: CBS: the current belief state
Data: AAQ: the set of already asked questions
Data: RF: the set of returned facts

if there is an active process $\langle \hookleftarrow C \parallel \emptyset, UD\rangle$ w.r.t. CBS in APS **then** ▷ Case 1
  output constraints C and return used defaults UD
else ▷ Case 2
  select an active process $\langle \hookleftarrow C \parallel GS, UD\rangle$ from APS w.r.t. CBS
  select an atom L in GS
  $APS' \leftarrow APS - \{\langle \hookleftarrow C \parallel GS, UD\rangle\}$
  $GS' \leftarrow GS - \{L\}$
  for the selected atom L **do**
    if L is a non-askable atom **then** ▷ Case 2.1
      $NewAPS \leftarrow APS' \cup \{\langle \hookleftarrow (C \wedge \{B_i = head(R)\} \wedge const(R) \parallel (body(R) \cup GS'), UD\rangle \mid C \wedge \{B_i = head(R)\} \wedge const(R)$ is consistent$\}$
    else if L is an askable atom $Q@S$ **then** ▷ Case 2.2
      if $L \notin AAQ$ **then**
        send question Q to agent S in which $S \in \Sigma$
        $NewAAQ \leftarrow AAQ \cup \{L\}$
        $AAQ \leftarrow NewAAQ$
      end if
      if $L \in UD$ **then**
        $NewAPS \leftarrow APS' \cup \{\langle \hookleftarrow C \parallel GS', UD\rangle\}$
      else if $(L \hookleftarrow C_r \parallel) \in RF$ **then**
        if $C \wedge C_r$ is consistent **then**
          $NewAPS \leftarrow APS' \cup \{\langle \hookleftarrow C \wedge C_r \parallel GS', UD\rangle\}$
        else
          $NewAPS \leftarrow APS'$
        end if
      else if there is a default constraint C_d for L **then**
        if $C \wedge C_d$ is consistent **then**
          $NewAPS \leftarrow APS' \cup \{\langle \hookleftarrow C \wedge C_d \parallel GS', UD \cup \{L\}\rangle\}$
        else
          $NewAPS \leftarrow APS'$
        end if
        if $C \wedge \neg C_d$ is consistent **then**
          $NewSPS \leftarrow SPS \cup \{\langle L, \hookleftarrow \alpha \parallel GS', UD\rangle\}$ where $C \wedge \neg C_d \models \alpha$
          $SPS \leftarrow NewSPS$
        end if
      end if
    end if
    $APS \leftarrow NewAPS$
  end for
end if

Algorithm 3 describes *fact arrival*. In the algorithm, $NewRF$, $NewAPS$, $NewSPS$ and $NewCBS$ correspond to changed versions of RF, APS, SPS and CBS after *fact arrival*. The prefixes "*Added*−" and "*Deleted*−" indicate portions of a set that are being added or removed respectively.

Algorithm 3 Fact arrival

Data: APS: the set of active processes
Data: SPS: the set of suspended processes
Data: CBS: the current belief state
Data: RF: the set of returned facts
Data: $Q@S \hookleftarrow C_r \parallel$: the returned fact constraint

$NewRF \leftarrow RF \cup \{Q@S \hookleftarrow C_r \parallel\}$
$RF \leftarrow NewRF$
$NewCBS \leftarrow CBS - \{Q@S \hookleftarrow C_d \parallel\} \cup \{Q@S \hookleftarrow C_r \parallel\}$
$CBS \leftarrow NewCBS$
if C_r entails C_d **then** ▷ Case 1
▷ revision of the set of active processes
  $DeletedAPS \leftarrow \{\langle \hookleftarrow \parallel GS, UD\rangle \in APS \mid Q@S \in UD\}$
  $AddedAPS \leftarrow \{\langle \hookleftarrow (C \wedge C_r) \parallel GS, UD\rangle \mid \langle \hookleftarrow C \parallel GS, UD\rangle \in DeletedAPS$ and $C \wedge C_r$ is consistent$\}$
  $NewAPS \leftarrow APS - DeletedAPS \cup AddedAPS$
▷ revision of the set of suspended processes
  $DeletedSPS \leftarrow \{\langle SAS, \hookleftarrow C \parallel GS, UD\rangle \in SPS \mid Q@S \in SAS$ or $Q@S \in UD\}$
  $AddedSPS \leftarrow \{\langle SAS, \hookleftarrow (C \wedge C_r) \parallel GS, UD\rangle \mid \langle SAS, \hookleftarrow C \parallel GS, UD\rangle \in DeletedSPS$, $Q@S \in UD$, and $C \wedge C_r$ is consistent$\}$
  $NewSPS \leftarrow SPS - DeletedSPS \cup AddedSPS$
else if C_r contradicts C_d **then** ▷ Case 2
▷ revision of the set of active processes
  $DeletedAPS \leftarrow \{\langle \hookleftarrow C \parallel GS, UD\rangle \in APS \mid Q@S \in UD\}$
  $ResumedSPS \leftarrow \{\langle \hookleftarrow (C \wedge C_r) \parallel GS, UD\rangle \mid \langle \{Q@S\}, \hookleftarrow C \parallel GS, UD\rangle \in SPS$, and $C \wedge C_r$ is consistent$\}$
  $NewAPS \leftarrow APS - DeletedAPS \cup ResumedSPS$
▷ revision of the set of suspended processes
  $DeletedSPS \leftarrow \{\langle SAS, \hookleftarrow C \parallel GS, UD\rangle \in SPS \mid Q@S \in SAS$ or $Q@S \in UD\}$
  $AddedSPS \leftarrow \{\langle SAS - \{Q@S\}, \hookleftarrow (C \wedge C_r) \parallel GS, UD \cup \{Q@S\}\rangle \mid \langle SAS, \hookleftarrow C \parallel GS, UD\rangle \in DeletedSPS$, $Q@S \in SAS$, $SAS - \{Q@S\} \neq \emptyset$, and $C \wedge C_r$ is consistent$\}$
  $NewSPS \leftarrow SPS - DeletedSPS \cup AddedSPS$
else if C_r does not entail C_d or contradict C_d **then** ▷ Case 3
▷ revision of the set of active processes
  $DeletedAPS \leftarrow \{\langle \hookleftarrow C \parallel GS, UD\rangle \in APS \mid Q@S \in UD\}$
  $AddedAPS \leftarrow \{\langle \hookleftarrow (C \wedge C_r) \parallel GS, UD\rangle \mid \langle \hookleftarrow C \parallel GS, UD\rangle \in DeletedAPS$ and $C \wedge C_r$ is consistent$\}$
  $ResumedSPS \leftarrow \{\langle \hookleftarrow (C \wedge C_r) \parallel GS, UD\rangle \mid \langle \{Q@S\}, \hookleftarrow C \parallel GS, UD\rangle \in SPS$ and $C \wedge C_r$ is consistent$\}$
  $NewAPS \leftarrow APS - DeletedAPS \cup AddedAPS \cup ResumedSPS$
▷ revision of the set of suspended processes
  $DeletedSPS \leftarrow \{\langle SAS, \hookleftarrow (C \wedge C_r) \parallel GS, UD\rangle \in SPS \mid Q@S \in SAS$ or $Q@S \in UD\}$
  $AddedSPS1 \leftarrow \{\langle SAS, \hookleftarrow (C \wedge C_r) \parallel GS, UD\rangle \mid \langle SAS, \hookleftarrow C \parallel GS, UD\rangle \in DeletedSPS$, $Q@S \in UD$, and $C \wedge C_r$ is consistent$\}$
  $AddedSPS2 \leftarrow \{\langle SAS - \{Q@S\}, \hookleftarrow (C \wedge C_r) \parallel GS, UD \cup \{Q@S\}\rangle \mid \langle SAS, \hookleftarrow C \parallel GS, UD\rangle \in DeletedSPS$, $Q@S \in SAS$, $SAS - \{Q@S\} \neq \emptyset$, and $C \wedge C_r$ is consistent$\}$.
  $NewSPS \leftarrow SPS - DeletedSPS \cup AddedSPS1 \cup AddedSPS2$
end if
$APS \leftarrow NewAPS$
$SPS \leftarrow NewSPS$

Finally, Algorithms 4 and 5 describe *default revision*. In the procedures given in the algorithms, $NewAPS$, $NewSPS$, and $NewCBS$ correspond to changed versions of APS, SPS and CBS. The prefixes "$Added-$" and "$Deleted-$" indicate portions of a set that are being added or removed respectively.

Algorithm 4 Default revision

Data: APS: the set of active processes
Data: SPS: the set of suspended processes
Data: CBS: the current belief state
Data: $New\Delta$: the set of changed default constraints
Data: $Q_d@S \hookleftarrow C_{newd} \|$: the new default constraint

1: **if** $New\Delta \neq \emptyset$ **then**
2: **for** each $Q_d@S \hookleftarrow C_{newd} \|. \in new\Delta$ **do**
3: $NewCBS \leftarrow CBS - \{Q_d@S \hookleftarrow C_d \|\} \cup \{Q_d@S \hookleftarrow C_{newd} \|\}$
4: **if** C_{newd} entails C_d **then** ▷ Case 1
5: ▷ revision of the set of active processes
6: $DeletedAPS \leftarrow \{\langle \hookleftarrow C \| GS, UD\rangle \in APS \mid Q_d@S \in UD\}$
7: $AddedAPS \leftarrow \{\langle \hookleftarrow (C \wedge C_{newd}) \| GS, UD\rangle \mid \langle \hookleftarrow C \| GS, UD\rangle \in DeletedAPS$ and $C \wedge C_{newd}$ is consistent$\}$
8: $NewAPS \leftarrow APS - DeletedAPS \cup AddedAPS$
9: ▷ revision of the set of suspended processes
10: $DeletedSPS \leftarrow \{\langle SAS, \hookleftarrow C \| GS, UD\rangle \in SPS \mid Q_d@S \in UD\}$
11: $AddedSPS1 \leftarrow \{\langle SAS, \hookleftarrow (C \wedge C_{newd}) \| GS, UD\rangle \mid \langle SAS, \hookleftarrow C \| GS, UD\rangle \in DeletedSPS$, $Q_d@S \in UD$, and $C \wedge C_{newd}$ is consistent$\}$
12: $AddedSPS2 \leftarrow \{\langle SAS \cup \{Q_d@S\}, \hookleftarrow C \| GS, UD - \{Q_d@S\}\rangle \mid \langle SAS, \hookleftarrow C \| GS, UD\rangle \in DeletedSPS$ and $Q_d@S \in UD\}$
13: $SuspendedAPS \leftarrow \{\langle \{Q_d@S\}, \hookleftarrow (C \wedge \neg C_{newd}) \| GS, UD\rangle \mid \langle \hookleftarrow C \| GS, UD\rangle \in APS$ and $C \wedge \neg C_{newd}$ is consistent$\}$
14: $NewSPS \leftarrow SPS - DeletedSPS \cup AddedSPS1 \cup AddedSPS2 \cup AddedSPS3 \cup SuspendedAPS$
15: **else if** C_{newd} contradicts C_d **then** ▷ Case 2
16: ▷ revision of the set of active processes
17: $DeletedAPS \leftarrow \{\langle \hookleftarrow C \| GS, UD\rangle \in APS \mid Q_d@S \in UD\}$
18: $ResumedSPS \leftarrow \{\langle \hookleftarrow (C \wedge C_{newd}) \| GS, UD\rangle \mid \langle \{Q_d@S\}, \hookleftarrow C \| GS, UD\rangle \in SPS$ and $C \wedge C_{newd}$ is consistent$\}$
19: $NewAPS \leftarrow APS - DeletedAPS \cup ResumedSPS$
20: ▷ revision of the set of suspended processes
21: $DeletedSPS \leftarrow \{\langle SAS, \hookleftarrow C \| GS, UD\rangle \in SPS \mid Q_d@S \in SAS$ or $Q_d@S \in UD\}$
22: $AddedSPS1 \leftarrow \{\langle SAS \cup \{Q_d@S\}, \hookleftarrow (C \wedge \neg C_{newd}) \| GS, UD - \{Q_d@S\}\rangle \mid \langle SAS, \hookleftarrow C \| GS, UD\rangle \in DeletedSPS$, $Q_d@S \in UD$ and $C \wedge \neg C_{newd}$ is consistent$\}$
23: $AddedSPS2 \leftarrow \{\langle SAS \cup \{Q_d@S\}, \hookleftarrow C \| GS, UD - \{Q_d@S\}\rangle \mid \langle SAS, \hookleftarrow C \| GS, UD\rangle \in DeletedSPS$ and $Q_d@S \in UD\}$
24: $AddedSPS3 \leftarrow \{\langle SAS - \{Q_d@S\}, \hookleftarrow (C \wedge C_{newd}) \| GS, UD \cup \{Q_d@S\}\rangle \mid \langle SAS, \hookleftarrow C \| GS, UD\rangle \in DeletedSPS$, $Q_d@S \in SAS$, $SAS - \{Q_d@S\} \neq \emptyset$, and $C \wedge C_{newd}$ is consistent$\}$
25: $SuspendedAPS \leftarrow \{\langle \{Q_d@S\}, \hookleftarrow C \| GS, UD\rangle \mid \langle \hookleftarrow C \| GS, UD\rangle \in APS$ and $Q_d@S \in UD\}$
26: $NewSPS \leftarrow SPS - DeletedSPS \cup AddedSPS1 \cup AddedSPS2 \cup AddedSPS3 \cup SuspendedAPS$

Algorithm 5 Default revision - continued

else if C_{newd} does not entail C_d or contradict C_d **then** ▷ Case 3
▷ revision of the set of active processes
$DeletedAPS \leftarrow \{\langle \hookleftarrow C \parallel GS, UD\rangle \in APS \mid Q_d@S \in UD\}$
$AddedAPS \leftarrow \{\langle \hookleftarrow (C \wedge C_{newd}) \parallel GS, UD\rangle \mid \langle \hookleftarrow C \parallel GS, UD\rangle \in DeletedAPS$ and $C \wedge C_{newd}$ is consistent$\}$
$ResumedSPS \leftarrow \{\langle \hookleftarrow (C \wedge C_{newd}) \parallel GS, UD\rangle \mid \langle \{Q_d@S\}, \hookleftarrow C \parallel GS, UD\rangle \in SPS$ and $C \wedge C_{newd}$ is consistent$\}$
$NewAPS \leftarrow APS - DeletedAPS \cup AddedAPS \cup ResumedSPS$
▷ revision of the set of suspended processes
$DeletedSPS \leftarrow \{\langle SAS, \hookleftarrow C \parallel GS, UD\rangle \in SPS \mid Q_d@S \in SAS$ or $Q_d@S \in UD\}$
$AddedSPS1 \leftarrow \{\langle SAS, \hookleftarrow (C \wedge C_{newd}) \parallel GS, UD\rangle \mid \langle SAS, \hookleftarrow C \parallel GS, UD\rangle \in DeletedSPS$, $Q_d@S \in UD$, and $C \wedge C_{newd}$ is consistent$\}$
$AddedSPS2 \leftarrow \{\langle SAS \cup \{Q_d@S\}, \hookleftarrow (C \wedge \neg C_{newd}) \parallel GS, UD - \{Q_d@S\}\rangle \mid \langle SAS, \hookleftarrow C \parallel GS, UD\rangle \in DeletedSPS$, $Q_d@S \in UD$, and $C \wedge \neg C_{newd}$ is consistent$\}$
$AddedSPS3 \leftarrow \{\langle SAS, \hookleftarrow (C \wedge \neg C_{newd}) \parallel GS, UD\rangle \mid \langle SAS, \hookleftarrow C \parallel GS, UD\rangle \in DeletedSPS$, $Q_d@S \in SAS$, and $C \wedge \neg C_{newd}$ is consistent$\}$
$AddedSPS4 \leftarrow \{\langle SAS - \{Q_d@S\}, \hookleftarrow (C \wedge C_{newd}) \parallel GS, UD \cup \{Q_d@S\}\rangle \mid \langle SAS, \hookleftarrow C \parallel GS,\ UD\rangle \in DeletedSPS$, $Q_d@S \in SAS$, $SAS - \{Q_d@S\} \neq \emptyset$, and $C \wedge C_{newd}$ is consistent$\}$
$SuspendedAPS \leftarrow \{\langle \{Q_d@S\}, \hookleftarrow (C \wedge \neg C_{newd}) \parallel GS, UD\rangle \mid \langle \hookleftarrow C \parallel GS, UD\rangle \in APS$ and $C \wedge \neg C_{newd}$ is consistent$\}$
$NewSPS \leftarrow SPS - DeletedSPS \cup AddedSPS1 \cup AddedSPS2 \cup AddedSPS3 \cup AddedSPS4 \cup SuspendedAPS$
end if
$CBS \leftarrow NewCBS$
$APS \leftarrow NewAPS$
$SPS \leftarrow NewSPS$
end for
end if

C Sketch of Proof for Claim 1

In the following sketch of proof, a round is defined as the execution of operations in the iteration step part of *process reduction*, the whole of *fact arrival*, or the whole of *default revision*, from their beginning to their end, without returning to the beginning and without transferring to another phase.

It is shown that a more general property holds for active or suspended processes at each round. The property, represented as λ, is the following: at any k-th round, for any active process or suspended process $\mathcal{P}$ without negation of constraints, there exists a sequence of reductions

" $\hookleftarrow \parallel GS''_{init}$, ..., " $\hookleftarrow C'_P \parallel GS''_P$

w.r.t. $\mathcal{P}$ and

$\mathcal{R}_P^{(k)} = RF_k \cup \{\delta_k(Q@S) | Q@S \in UD_P\}$

s.t. $\pi_V(C_P)$ entails $\pi_V(C'_P)$,

where $C_P = C_{P_a}$, $GS_P = GS_{P_a}$, and $UD_P = UD_{P_a}$ if P is an active process $P_a = \langle \hookleftarrow C_{P_a} \parallel GS_{P_a}, UD_{P_a} \rangle$, and $C_P = C_{P_s}$, $GS_P = SAS_{P_s} \cup GS_{P_s}$, and $UD_P = UD_{P_s}$, if P is a suspended process $P_s = \langle SAS_{P_s} \hookleftarrow C_{P_s} \parallel GS_{P_s}, UD_{P_s} \rangle$, and RF_k is the reply set of constraints returned at the k-th round.

Property λ is proven by mathematical induction for the progress of the three phases of Speculative Computation with Default Revision: *process reduction*, *fact arrival*, and *default revision*. The induction base and the induction step are the following:

- *Induction base.* For $k = 0$, which is the initial round of Speculative Computation with Default Revision and corresponds to the initial step of *process reduction*, an active process $\langle \hookleftarrow \parallel GS_{init}, \emptyset \rangle$ is created, which satisfies property λ.

- *Induction step.* Assume that at the k-th round property λ holds. Now, considering the $(k+1)$-th round, it is shown that λ holds in the following way:
  1. In the case that the round is *a step of process reduction:*

     It is straightforward to show that λ holds for a step of *process reduction* because no constraint is added to the reply set from the previous round. It is the normal reduction of a process according to the operational model.

  2. In the case that the round is a *fact arrival*:

     In this phase, consider the processing of a returned answer $Q@S \hookleftarrow C_r$ in the *fact arrival* phase and the existence of a default constraint C_d for $Q@S$.

     Let $P_a = \langle \hookleftarrow C_{P_a} \parallel GS_{P_a}, UD_{P_a} \rangle$ be any existing active process s.t. $Q@S \in UD_{P_a}$. In this round, P_a is deleted. For the newly added active process created from P_a one can have the following three cases:

     (a) In the case that C_r entails C_d. If $C_{P_a} \wedge C_r$ is consistent, the active process $P'_a = \langle \hookleftarrow C_{P_a} \wedge C_r \parallel GS_{P_a}, UD_{P_a} - \{Q@S\} \rangle$ is created and one has $\mathcal{R}^{(k+1)}_{P'_a} = R^{(k)}_{P_a} \cup \{Q@S \hookleftarrow C_r \parallel\} \backslash \{Q@S \hookleftarrow C_d \parallel\}$. By the induction hypothesis, P_a satisfies λ for some C'_{P_a}, i.e, there exists a sequence of reductions " $\hookleftarrow \parallel GS''_{init}$, ..., " $\hookleftarrow C_1 \parallel \{Q@S\} \cup GS''$, " $\hookleftarrow C_1 \wedge C_d \parallel GS''$, ..., " $\hookleftarrow C_1 \wedge C_d \wedge C_2 \parallel GS''_{P_a}$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k)}_{P_a}$ s.t. $\pi_V(C_{P_a})$ entails $\pi_V(C_1 \wedge C_d \wedge C_2)$, where C_1 and C_2 are the constraints obtained before and after processing $Q@S$ respectively, and $C_1 \wedge C_d \wedge C_2 = C'_{P_a}$. Then one can consider the sequence of reductions " $\hookleftarrow \parallel GS''_{init}$, ..., " $\hookleftarrow C_1 \parallel \{Q@S\} \cup GS''$, " $\hookleftarrow C_1 \wedge C_r \parallel GS''$, ..., " $\hookleftarrow C_1 \wedge C_r \wedge C_2 \parallel GS''_{P_a}$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k+1)}_{P'_a}$. Since C_r entails C_d and $\pi_V(C_{P_a})$ entails $\pi_V(C_1 \wedge C_d \wedge C_2)$, then $\pi_V(C_{P_a} \wedge C_r)$ entails $\pi_V(C_1 \wedge C_r \wedge C_2)$. Thus λ holds for this new process.

     (b) In the case that C_r contradicts C_d. No active process is created from P_a.

     (c) In the case that C_r does not entail or contradict C_d. If $C_{P_a} \wedge C_r$ is consistent, an active process is created from P_a, for which one can show λ in a similar way to case (a).

     Now, let $P_s = \langle SAS_{P_s} \hookleftarrow C_{P_s} \parallel GS_{P_s}, UD_{P_s} \rangle$ be an existing suspended process s.t. $Q@S \in SAS_{P_s}$ or $Q@S \in UD_{P_s}$. P_s is deleted at this step.

     Consider the newly added active process that is created from P_s s.t. $Q@S \in SAS_{P_s}$ and $SAS_{P_s} - \{Q@S\} = \emptyset$ (which corresponds to the resumed process). One can have the following cases:

     (a) In the case that C_r entails C_d. No active process is created from P_s.

(b) In the case that C_r contradicts C_d. If $C_{P_s} \wedge C_r$ is consistent, the active process $P'_a = \langle \hookleftarrow C_{P_s} \wedge C_r \parallel GS_{P_s}, UD_{P_s} \cup \{Q@S\} \rangle$ is created and one has $\mathcal{R}^{(k+1)}_{P'_a} = R^{(k)}_{P_s} \cup \{Q@S \hookleftarrow C_r \parallel\} \backslash \{Q@S \hookleftarrow C_d \parallel\}$. By the induction hypothesis, P_s satisfies λ for some C'_{P_s}, i.e, there exists a sequence of reductions " $\hookleftarrow \parallel GS''_{init}$, ..., " $\hookleftarrow C'_{P_s} \parallel \{Q@S\} \cup GS''_{P_s}$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k+1)}_{P'_a}$ s.t. $\pi_V(C_{P_s})$ entails $\pi_V(C'_{P_s})$. Then one can consider the sequence of reductions " $\hookleftarrow \parallel GS''_{init}$, ..., " $\hookleftarrow C'_{P_s} \parallel \{Q@S\} \cup GS''$, " $\hookleftarrow C'_{P_s} \wedge C_r \parallel GS''_{P_s}$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k+1)}_{P'_a}$. Since $\pi_V(C_{P_s})$ entails $\pi_V(C'_{P_s})$, then $\pi_V(C_{P_s} \wedge C_r)$ entails $\pi_V(C'_{P_s} \wedge C_r)$. Thus λ holds for this new process.

(c) In the case that C_r does not entail or contradict C_d. If $C_{P_s} \wedge C_r$ is consistent, an active process is created from P_s, for which one can show λ in a similar way to case (b).

Consider the newly added suspended process that is created from P_s so that $Q@S \in SAS_{P_s}$ and $SAS_{P_s} - \{Q@S\} \neq \emptyset$. One can have the following cases:

(a) In the case that C_r entails C_d. No new suspended process is created from P_s.

(b) In the case that C_r contradicts C_d. If $C_{P_s} \wedge C_r$ is consistent, the suspended process $P'_s = \langle SAS_{P_s} - \{Q@S\}, \hookleftarrow C_{P_s} \wedge C_r \parallel GS_{P_s}, UD_{P_s} \cup \{Q@S\} \rangle$ is created and one has $\mathcal{R}^{(k+1)}_{P'_s} = R^{(k)}_{P_s} \cup \{Q@S \hookleftarrow C_r \parallel\} \backslash \{Q@S \hookleftarrow C_d \parallel\}$. By the induction hypothesis, P_s satisfies λ for some C'_{P_s}, i.e, there exists a sequence of reductions " $\hookleftarrow \parallel GS''_{init}$, ..., " $\hookleftarrow C'_{P_s} \parallel \{Q@S\} \cup GS''_{P_s}$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k+1)}_{P'_s}$ so that $\pi_V(C_{P_s})$ entails $\pi_V(C'_{P_s})$. Then one can consider the sequence of reductions " $\hookleftarrow \parallel GS''_{init}$, ..., " $\hookleftarrow C'_{P_s} \parallel \{Q@S\} \cup GS''$, " $\hookleftarrow C'_{P_s} \wedge C_r \parallel GS''_{P_s}$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k+1)}_{P'_s}$. Since $\pi_V(C_{P_s})$ entails $\pi_V(C'_{P_s})$, then $\pi_V(C_{P_s} \wedge C_r)$ entails $\pi_V(C'_{P_s} \wedge C_r)$. Thus λ holds for this new process.

(c) In the case that C_r does not entail or contradict C_d. If $C_{P_s} \wedge C_r$ is consistent, a suspended process is created from P_s, for which one can show λ in a similar way to case (b).

Now, consider the newly added suspended process that is created from P_s s.t. $Q@S \in UD_{P_s}$. One may have the following three cases:

(a) In the case that C_r entails C_d. If $C_{P_s} \wedge C_r$ is consistent, the suspended process $P'_s = \langle SAS_{P_s} \hookleftarrow C_{P_s} \wedge C_r \parallel GS_{P_s}, UD_{P_s} \rangle$ is created and one has $\mathcal{R}^{(k+1)}_{P'_s} = R^{(k)}_{P_s} \cup \{Q@S \hookleftarrow C_r \parallel\} \backslash \{Q@S \hookleftarrow C_d \parallel\}$. By the induction hypothesis, P_s satisfies λ for some C'_{P_s}, i.e, there exists a sequence of reductions " $\hookleftarrow \parallel GS''_{init}$, ..., " $\hookleftarrow C_1 \parallel \{Q@S\} \cup GS''$, " $\hookleftarrow C_1 \wedge C_d \parallel GS''$, ..., " $\hookleftarrow C_1 \wedge C_d \wedge C_2 \parallel GS''_{P_s}$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k)}_{P_s}$ so that $\pi_V(C_{P_s})$ entails $\pi_V(C_1 \wedge C_d \wedge C_2)$, where C_1 and C_2 are the constraints obtained before and after processing $Q@S$ respectively, and $C_1 \wedge C_d \wedge C_2 = C'_{P_s}$. Then one can consider the sequence of reductions $\langle \hookleftarrow \parallel GS_{init} \rangle$, ..., $\langle \hookleftarrow C_1 \parallel \{Q@S\} \cup GS \rangle$, $\langle \hookleftarrow C_1 \wedge C_r \parallel GS \rangle$, ..., $\langle \hookleftarrow C_1 \wedge C_r \wedge C_2 \parallel \{SAS_{P_s}\} \cup GS_{P_s} \rangle$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k+1)}_{P'_s}$. Since C_r entails C_d and $\pi_V(C_{P_s})$ entails $\pi_V(C_1 \wedge C_d \wedge C_2)$, then $\pi_V(C_{P_s} \wedge C_r)$ entails $\pi_V(C_1 \wedge C_r \wedge C_2)$. Thus λ holds for this new process.

(b) In the case that C_r contradicts C_d. No suspended process is created from P_s.

(c) In the case that C_r does not entail or contradict C_d. If $C_{P_s} \wedge C_r$ is consistent, a suspended process is created from P_s, for which one can show λ in a similar way to case 1.

As a result of *fact arrival*, no suspended process is created from an active process since the constraints from the replies of the agent information sources are regarded as definitive.

3. In the case that the round is a *default revision*:

   Let $P_a = \langle \hookleftarrow C_{P_a} \parallel GS_{P_a}, UD_{P_a} \rangle$ be an existing active process so that $Q@S \in UD_{P_a}$. In this step P_a is deleted. For the newly added active process created from P_a one can have the following three cases:

   (a) In the case that C_{newd} entails C_d. If $C_{P_a} \wedge C_{newd}$ is consistent, the active process $P'_a = \langle \hookleftarrow C_{P_a} \wedge C_{newd} \parallel GS_{P_a}, UD_{P_a} \rangle$ is created and one has $\mathcal{R}^{(k+1)}_{P'_a} = R^{(k)}_{P_a} \cup \{Q@S \hookleftarrow C_{newd} \parallel\} \backslash \{Q@S \hookleftarrow C_d \parallel\}$. By the induction hypothesis, P_a satisfies λ for some C'_{P_a}, i.e, there exists a sequence of reductions " $\hookleftarrow \parallel GS''_{init}$, ..., " $\hookleftarrow C_1 \parallel \{Q@S\} \cup GS''$, " $\hookleftarrow C_1 \wedge C_d \parallel GS''$, ..., " $\hookleftarrow C_1 \wedge C_d \wedge C_2 \parallel GS''_{P_a}$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k)}_{P_a}$ so that $\pi_V(C_{P_a})$ entails $\pi_V(C_1 \wedge C_d \wedge C_2)$, where C_1 and C_2 are the constraints obtained before and after processing $Q@S$ respectively, and $C_1 \wedge C_d \wedge C_2 = C'_{P_a}$. Then one can consider the sequence of reductions " $\hookleftarrow \parallel GS''_{init}$, ..., " $\hookleftarrow C_1 \parallel \{Q@S\} \cup GS''$, " $\hookleftarrow C_1 \wedge C_{newd} \parallel GS''$, ..., " $\hookleftarrow C_1 \wedge C_{newd} \wedge C_2 \parallel GS''_{P_a}$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k+1)}_{P'_a}$. Since C_{newd} entails C_d and $\pi_V(C_{P_a})$ entails $\pi_V(C_1 \wedge C_{newd} \wedge C_2)$, $\pi_V(C_{P_a} \wedge C_d)$ entails $\pi_V(C_1 \wedge C_{newd} \wedge C_2)$. Thus λ holds for this new process.

   (b) In the case that C_{newd} contradicts C_d. No active process is created from P_a.

   (c) In the case that C_{newd} does not entail or contradict C_d. If $C_{P_a} \wedge C_{newd}$ is consistent, an active process is created from P_a, for which one can show λ in a similar way to case (a).

   Now, let $P_s = \langle SAS_{P_s} \hookleftarrow C_{P_s} \parallel GS_{P_s}, UD_{P_s} \rangle$ be an existing suspended process so that $Q@S \in SAS_{P_s}$ or $Q@S \in UD_{P_s}$. P_s is deleted at this step.

   Consider the newly added active process that is created from P_s so that $Q@S \in SAS_{P_s}$ and $SAS_{P_s} - \{Q@S\} = \emptyset$ (which corresponds to the resumed process). One can have the following cases:

   (a) In the case that C_{newd} entails C_d. No active process is created from P_s.

   (b) In the case that C_{newd} contradicts C_d. If $C_{P_s} \wedge C_{newd}$ is consistent, the active process $P'_a = \langle \hookleftarrow C_{P_s} \wedge C_{newd} \parallel GS_{P_s}, UD_{P_s} \cup \{Q@S\} \rangle$ is created and one has $\mathcal{R}^{(k+1)}_{P'_a} = R^{(k)}_{P_s} \cup \{Q@S \hookleftarrow C_{newd} \parallel\} \backslash \{Q@S \hookleftarrow C_d \parallel\}$. By the induction hypothesis, P_s satisfies λ for some C'_{P_s}, i.e, there exists a sequence of reductions " $\hookleftarrow \parallel GS''_{init}$, ..., " $\hookleftarrow C'_{P_s} \parallel \{Q@S\} \cup GS''_{P_s}$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k+1)}_{P'_s}$ so that $\pi_V(C_{P_s})$ entails $\pi_V(C'_{P_s})$. Then one can consider the sequence of reductions " $\hookleftarrow \parallel GS''_{init}$, ..., " $\hookleftarrow C'_{P_s} \parallel \{Q@S\} \cup GS''$, " $\hookleftarrow C'_{P_s} \wedge C_{newd} \parallel GS''_{P_s}$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k+1)}_{P'_a}$. Since $\pi_V(C_{P_s})$ entails $\pi_V(C'_{P_s})$, $\pi_V(C_{P_s} \wedge C_{newd})$ entails $\pi_V(C'_{P_s} \wedge C_{newd})$. Thus λ holds for this new process.

   (c) In the case that C_{newd} does not entail or contradict C_d. If $C_{P_s} \wedge C_{newd}$ is consistent, an active process is created from P_s, for which one can show λ in a similar way to case (b).

   Consider the newly added suspended process that is created from P_s so that $Q@S \in SAS_{P_s}$ and $SAS_{P_s} - \{Q@S\} \neq \emptyset$. One can have the following cases:

   (a) In the case that C_{newd} entails C_d. No new suspended process is created from P_s.

(b) In the case that C_{newd} contradicts C_d. If $C_{P_s} \wedge C_{newd}$ is consistent, the suspended process $P'_s = \langle SAS_{P_s} - \{Q@S\}, \hookleftarrow C_{P_s} \wedge C_{newd} \parallel GS_{P_s}, UD_{P_s} \cup \{Q@S\}\rangle$ is created and one has $\mathcal{R}^{(k+1)}_{P'_s} = R^{(k)}_{P_s} \cup \{Q@S \hookleftarrow C_{newd} \parallel\} \backslash \{Q@S \hookleftarrow C_d \parallel\}$. By the induction hypothesis, P_s satisfies λ for some C'_{P_s}, i.e, there exists a sequence of reductions " $\hookleftarrow\parallel GS''_{init}$, ..., " $\hookleftarrow C'_{P_s} \parallel \{Q@S\} \cup GS''_{P_s}$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k+1)}_{P'_s}$ so that $\pi_V(C_{P_s})$ entails $\pi_V(C'_{P_s})$. Then one can consider the sequence of reductions " $\hookleftarrow\parallel GS''_{init}$, ..., " $\hookleftarrow C'_{P_s} \parallel \{Q@S\} \cup GS''$, " $\hookleftarrow C'_{P_s} \wedge C_{newd} \parallel GS''_{P_s}$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k+1)}_{P'_s}$. Since $\pi_V(C_{P_s})$ entails $\pi_V(C'_{P_s})$, $\pi_V(C_{P_s} \wedge C_{newd})$ entails $\pi_V(C'_{P_s} \wedge C_{newd})$. Thus λ holds for this new process.

(c) In the case that C_{newd} does not entail or contradict C_d. If $C_{P_s} \wedge C_{newd}$ is consistent, a suspended process is created from P_s, for which one can show λ in a similar way to case (b).

Now, consider the newly added suspended process that is created from P_s so that $Q@S \in UD_{P_s}$. One may have the following three cases.

(a) In the case that C_{newd} entails C_d. If $C_{P_s} \wedge C_{newd}$ is consistent, the suspended process $P'_s = \langle SAS_{P_s} \hookleftarrow C_{P_s} \wedge C_{newd} \parallel GS_{P_s}, UD_{P_s}\rangle$ is created and one has $\mathcal{R}^{(k+1)}_{P'_s} = R^{(k)}_{P_s} \cup \{Q@S \hookleftarrow C_{newd} \parallel\} \backslash \{Q@S \hookleftarrow C_d \parallel\}$. By the induction hypothesis, P_s satisfies λ for some C'_{P_s}, i.e, there exists a sequence of reductions $\langle\hookleftarrow\parallel GS_{init}\rangle$, ..., $\langle\hookleftarrow C_1 \parallel \{Q@S\} \cup GS\rangle$, $\langle\hookleftarrow C_1 \wedge C_d \parallel GS\rangle$, ..., $\langle\hookleftarrow C_1 \wedge C_d \wedge C_2 \parallel GS_{P_s}\rangle$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k)}_{P_s}$ so that $\pi_V(C_{P_s})$ entails $\pi_V(C_1 \wedge C_d \wedge C_2)$, where C_1 and C_2 are the constraints obtained before and after processing $Q@S$ respectively, and $C_1 \wedge C_d \wedge C_2 = C'_{P_s}$. Then one can consider the sequence of reductions $\langle\hookleftarrow\parallel GS_{init}\rangle$, ..., $\langle\hookleftarrow C_1 \parallel \{Q@S\} \cup GS\rangle$, $\langle\hookleftarrow C_1 \wedge C_{newd} \parallel GS\rangle$, ..., $\langle\hookleftarrow C_1 \wedge C_{newd} \wedge C_2 \parallel \{SAS_{P_s}\} \cup GS_{P_s}\rangle$ w.r.t. $\mathcal{P}$ and $\mathcal{R}^{(k+1)}_{P'_s}$. Since C_{newd} entails C_d and $\pi_V(C_{P_s})$ entails $\pi_V(C_1 \wedge C_d \wedge C_2)$, then $\pi_V(C_{P_s} \wedge C_{newd})$ entails $\pi_V(C_1 \wedge C_{newd} \wedge C_2)$. Thus λ holds for this new process.

(b) In the case that C_{newd} contradicts C_d. No suspended process is created from P_s.

(c) In the case that C_{newd} does not entail or contradict C_d. If $C_{P_s} \wedge C_{newd}$ is consistent, a suspended process is created from P_s, for which one can show λ in a similar way to case (a).

Processes that fall outside the scope of this proof still hold property λ since $Q@S$ has not been used in their reduction. As such, property λ holds for any active or suspended process without negation of constraints after the *fact arrival* and *default revision* phases.

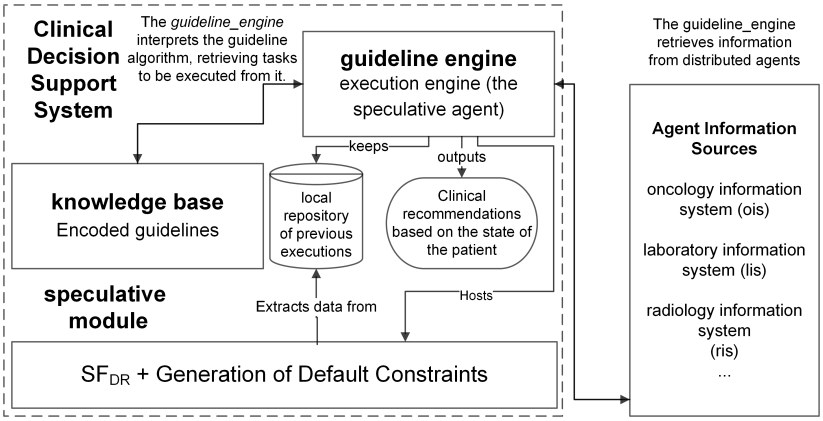

Fig. 1 Architecture of the clinical decision support system with its basic components.

Conditions	T (t)	N (n)	M (m)	Task	Recommendation
A	t0 *or* tis	--	--	Action 1	No adjuvant therapy, colonoscopy in 1 year
B	t1 *or* t2	n0	m0	Action 1	No adjuvant therapy, colonoscopy in 1 year
C	t3	--	m0	Action 2	Clinical trial or observation or consider capecitabine or 5-FU/leucovorin
D	t4	n0	m0	Action 3	Capecitabine or 5-FU/leucovorin or FOLFOX or CapeOX
E	t1 *or* t2 *or* t3 *or* t4	n1 *or* n2	m0	Action 4	FOLFOX or CapeOX or FLOX
F	t1 *or* t2 *or* t3 *or* t4	n0 *or* n1 *or* n2	m1	Action 5	Colonoscopy, chest and abdominal/pelvic CT, platelets, chemistry profile

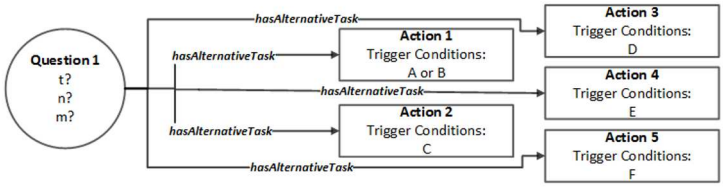

Fig. 2 Clinical setting for the example. The trigger conditions are expressed in terms of three variables, according to the TNM staging system for colon cancer. The possible values for T are $t0, tis, t1, t2, t3$, and $t4$. The possible values for N are $n0, n1$, and $n2$. The possible values for M are $m0$ and $m1$

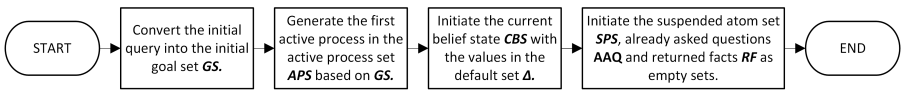

Fig. 3 Flowchart with the procedures for the initial step of the *process reduction* phase.

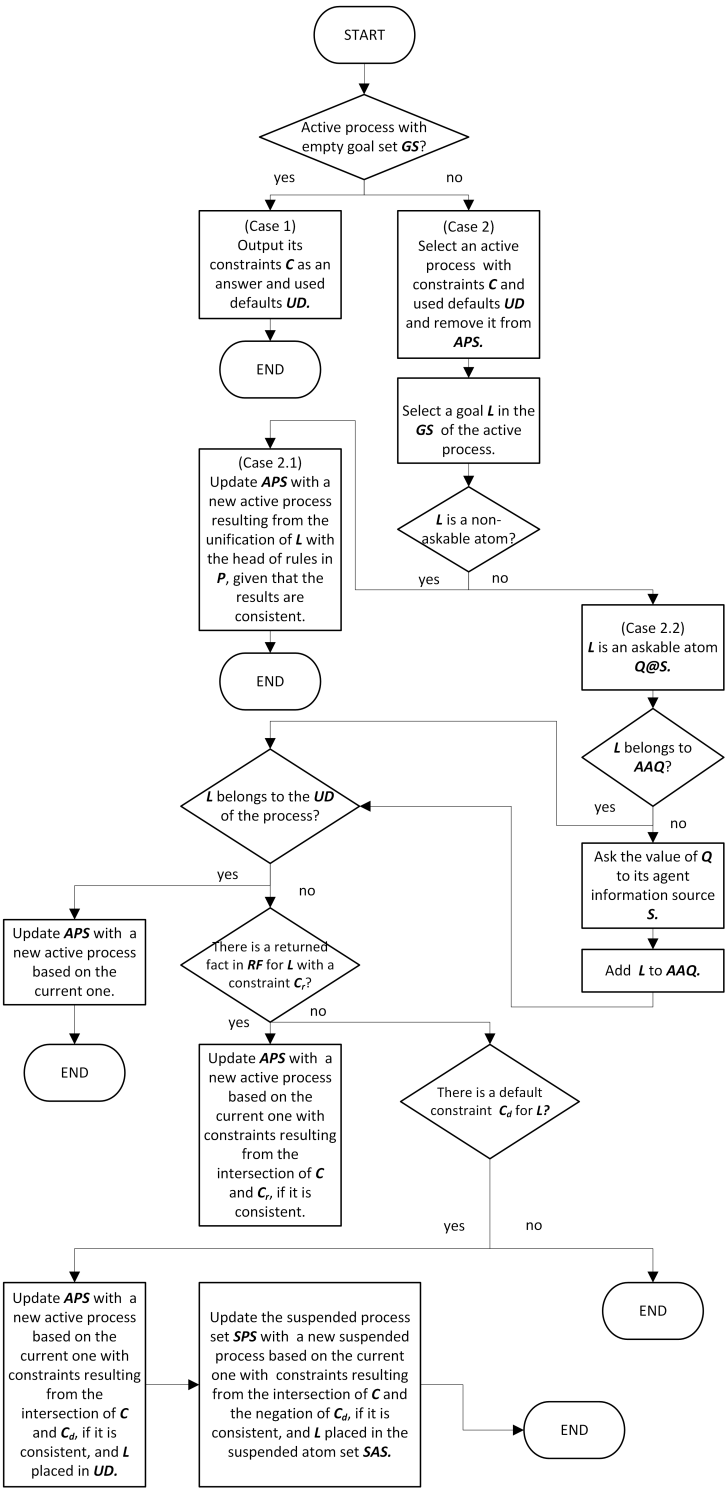

Fig. 4 Flowchart with the procedures for the iteration step of the *process reduction* phase.

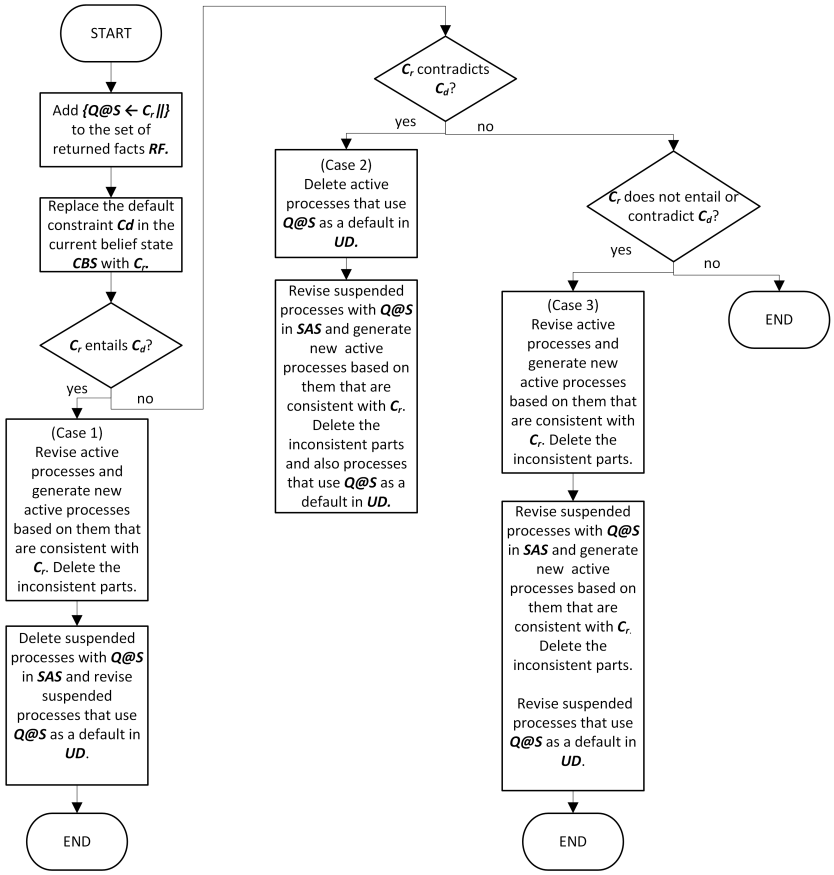

Fig. 5 Flowchart with the procedures for the *fact arrival* phase.

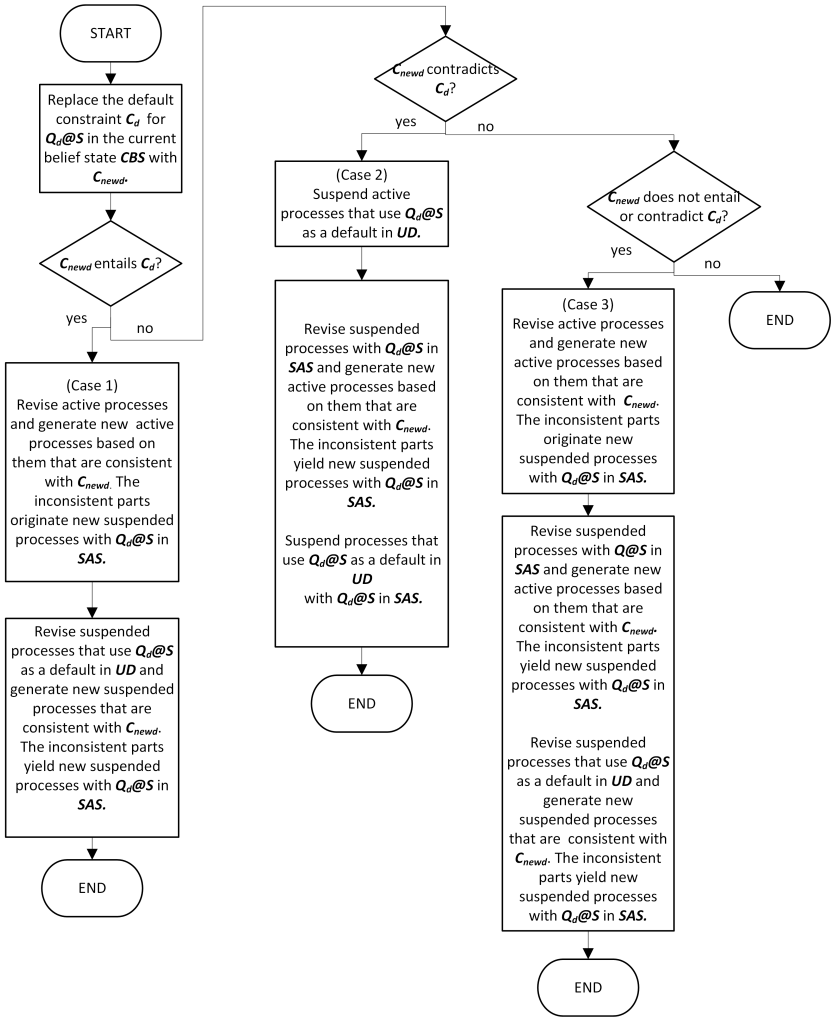

Fig. 6 Flowchart with the procedures for the *default revision* phase.

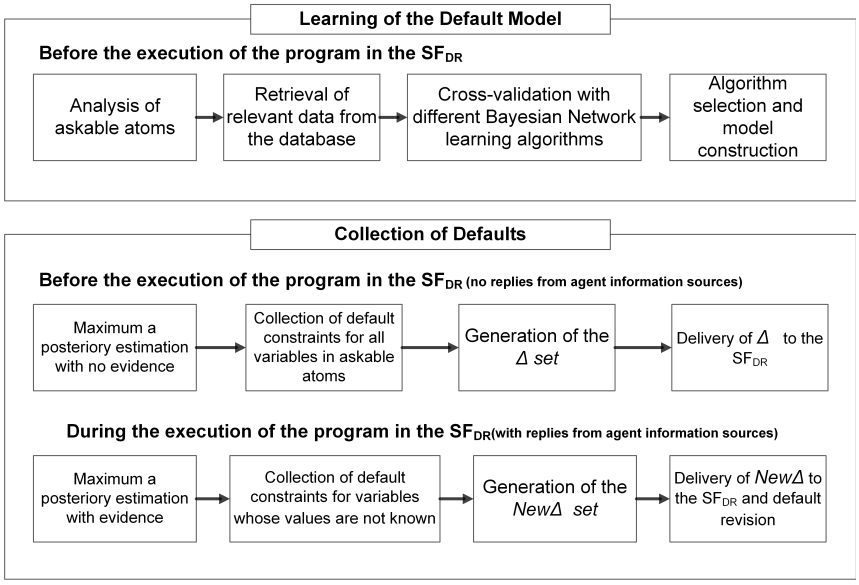

Fig. 7 Procedures in the Generation of Default Constraints. The two types of procedures, Learning of the Default Model and Collection of Defaults, comprise steps to construct a probabilistic model with the variables in the problem the speculative agent has to address and to collect the default constraints from that model

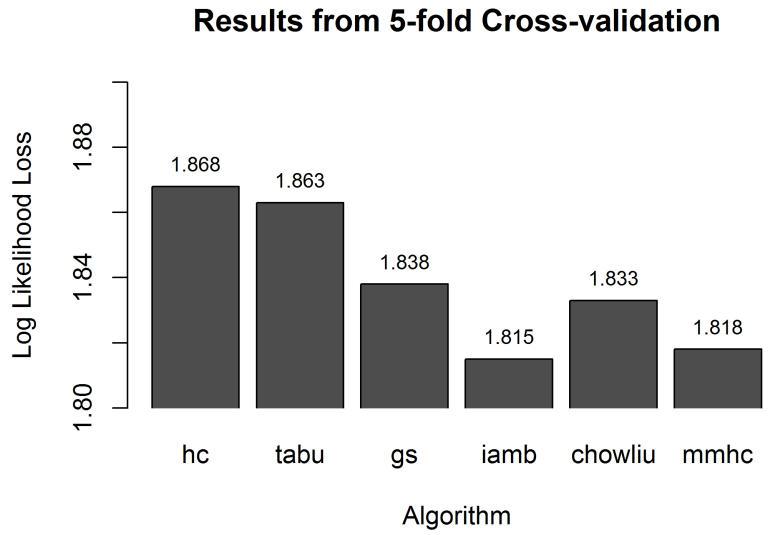

Fig. 8 Results from 5-fold cross-validation for the six BN learning algorithms in terms of log likelihood loss. The set of learning algorithms used in this step includes: two score-based learning algorithms, the Hill-Climbing (*hc*) and the Tabu-Search (*tabu*); three constraint-based search algorithms, the Grow-Shrink (*gs*), the Incremental Association (*iamb*), and the Chow-Liu (*chowliu*); and one hybrid algorithm, the Max-Min Hill-Climbing (*mmhc*)

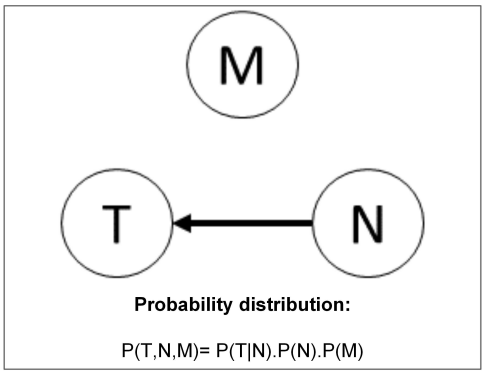

Fig. 9 Network structure produced by the *iamb* learning algorithm about the TNM staging of colon cancer and respective probability distribution