Annals of Mathematics and Artificial Intelligence manuscript No.
(will be inserted by the editor)

Fabrizio Angiulli · Gianluigi Greco · Luigi Palopoli

Detecting and Repairing Anomalous Evolutions in Noisy Environments

Logic Programming Formalization and Complexity Results

Abstract In systems where agents are required to interact with a partially known and dynamic world, sensors can be used to obtain knowledge about the environment. However, sensors may be unreliable, that is, they may deliver wrong information (due, e.g., to hardware or software malfunctioning) and, consequently, they may cause agents to take wrong decisions, which is a scenario that should be avoided.

The paper considers the problem of reasoning in noisy environments in a setting where no (either certain or probabilistic) data is available in advance about the reliability of sensors. Therefore, assuming that each agent is equipped with a background theory encoding its general knowledge about the world, a concept of detecting an anomaly perceived in sensor data and the related concept of agent recovering to a coherent status of information are defined. In this context, the complexities of various anomaly detection and anomaly recovery problems are studied. Finally, rewriting algorithms are proposed that transform recovery problems into equivalent inference problems under answer set semantics, thereby making them effectively realizable on top of available answer set solvers.

Keywords Logic Programming · Computational Complexity · Nonmonotonic Reasoning

Fabrizio Angiulli and Luigi Palopoli
DEIS, Università della Calabria, 87036 Rende, Italy, E-mail: {angiulli,palopoli}@deis.unical.it
Gianluigi Greco
Dip. di Matematica, Università della Calabria, 87036 Rende, Italy, E-mail: ggreco@mat.unical.it

1 Introduction

Consider an agent operating in a dynamic context according to an internal background theory (the agent's *trustable knowledge*) which is enriched, over time, through sensing the environment. The agent acts on the environment through operators, which generally cause the environment to evolve from one state to the next one. To this aim, the agent exploits an internal image of the environment (and its evolution over time) which is constructed on the basis of its background theory and sensed information. Therefore, corresponding to environment evolution is the agent's perceived evolution of it. Were sensors completely reliable, in a fully observable environment, the agent could gain a perfectly correct perception of environment evolution. However, in general, sensors might be unreliable, in that they may deliver erroneous observations to the agent. Thus, the agent's perception about environment evolution might be erroneous and this, in turn, might cause wrong decisions are taken.

In order to deal with the uncertainty that arises from noisy sensors, probabilistic approaches have been proposed (see, e.g., [9,14,15,54,60,69,55,10, 73,23,24]) where evolutions are represented by means of dynamic systems in which transitions among possible states are determined in terms of probability distributions. Other approaches refer to some *logic* formalization (see, e.g., modal logics, action languages, logic programming, and situation calculus [5, 74,39,72,44,42]) in which a logical theory is augmented to deal quantitatively and/or qualitatively with reliability information about sensors.

In this paper we take a different perspective instead, by assuming that no information (neither probabilistic nor qualitative) about reliability of sensors is available in advance. Therefore, evidence for the failure of the sensors can only be gained from singling out some form of discrepancy between the observations sensed through them and the internal trustable knowledge of the agent. This is precisely the approach proposed and investigated in the paper, which is next briefly illustrated with the help of a running example.

1.1 Example of Faulty Sensors Identification

Assume that an agent is in charge of parking cars in a parking lot (see Figure 1). The parking lot consists of two buildings, each with several floors. The floors are reached via a single elevator which runs in the middle in between the two buildings (so, there is a building to the *left* and one to the *right* of the elevator door). A number of sensors are used to inform the agent about parking place availability at different levels of the two buildings. In particular, the sensors tell the agent:

(a) if there is any available parking place at some level in any of the two buildings (sensor s_1);
(b) given the floor where the agent is currently located, if there is any available parking place in the left or the right building at that floor (sensor s_2);
(c) given the floor and the building (left or right) where the agent is currently located, whether parking places are available at that floor in that building (sensor s_3).

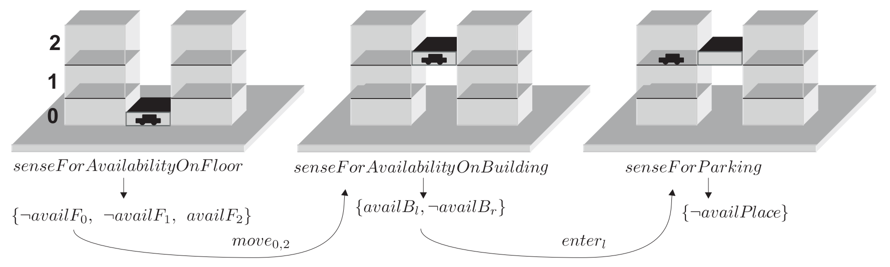
2
1
0
$senseForAvailabilityOnFloor$
$\{\neg availF_0,\ \neg availF_1,\ availF_2\}$
$move_{0,2}$
$senseForAvailabilityOnBuilding$
$\{availB_l, \neg availB_r\}$
$enter_l$
$senseForParking$
$\{\neg availPlace\}$

Fig. 1 Parking lot example.

Also, the agent uses a background theory that tells him that if he is at floor i of the building x and sensors s_1, when queried, signalled parking availability at level i and sensor s_2, when queried, signalled a parking availability in building x, then there must be indeed at least one parking place available at his current position.

Now, assume that the agent had actually queried sensor s_1 that notified parking availability at level 2, and sensor s_2 that notified parking availability in the *left* building. Assume, moreover, that the agent is at level 2, in the left building. If sensor s_3 returns the information that no place is available at the current agent's position, then a disagreement will emerge with the internal state of the agent that tells that there should be indeed at least one place available in that position. This disagreement implies that some anomalies came into play.

In particular, the agent might doubt about the reliability of sensor s_3 (that is, there actually are available parking places at the agent position, but s_3 tells that none is available). Similarly, the agent might also suspect that sensor s_1 is reliable while s_2 is not; by this way, the output of s_3 would come at no surprise.

1.2 Contribution and Organization

By focusing on scenarios as the one discussed above, it is our aim to introduce techniques that are able to automatically identify discrepancies between the observations and the internal knowledge and that may suggest possible fixes for them. In more detail, the contribution of the paper is as follows:

- ► We propose the concept of anomaly in state evolutions of a dynamic environment, as perceived by an agent sensing that environment through (possibly) noisy sensors. The formalization is, to a large extent, independent of the specific formalism used to encode the agent's trustable knowledge, but details are provided for agents whose reasoning capabilities are modeled as extended logic programs, and where the description of the planning domain is formalized via the $\mathcal{A}_K$ action language [10, 73, 74].
- ► In the case where a discrepancy is actually found, a suitable repair should be computed. A contribution of the paper is to introduce a notion of repair for an anomaly, i.e., of an evolution where sensed values are fixed

to comply with the trustable knowledge. Thus, by repairing an evolution we simply mean to equip the agent with some novel possible perception of the status of the world, where discrepancies between observations and background knowledge are resolved.

- ▶ For the several detection and repair problem variants we have defined, we develop a thorough study of the underlying computational complexities. This is relevant since complexity results are quite useful for gaining knowledge of the structure of the problems the framework comprises and, above all, to be able to realize effective rewriting and optimizations needed to implement them. The study considers different kinds of extended logic programs used to encode the agent knowledge about the environment, ranging from negation-by-default-free programs to general programs under both the brave and the cautious semantics.
- ▶ We propose and analyze variants for the notion of repair where, for instance, suitable minimality conditions are taken into account, in order to express preferences on the way the revision of the evolution should be carried out. In particular, this analysis is contextualized to the case of negation-by-default-free programs, for which nice links between the existence of anomalies and the emergence of incoherences in the agent's theory are established.
- ▶ Finally, having realized that the reasoning problems related to the identification and the repairing of an anomaly are confined within the second level of the polynomial hierarchy (for the case of negation-by-default-free programs) has paved the way towards the design of an implementation based on rewritings in terms of disjunctive logic programs under the answer set semantics. Details on the rewriting algorithms are also reported and discussed.

The rest of the paper is organized as follows. Section 2 describes the main complexity classes that will be referred to in the paper, and introduces some preliminaries on the language $\mathcal{A}_K$ and on logic programming. Section 3 presents the framework for detecting and repairing anomalies in evolutions. The complexity analysis of the proposed setting is carried out throughout Section 4 and Section 5. In particular, the latter section investigates a few variants for the notion of repair, where some minimality conditions are possibly taken into account, and where the focus is on negation-by-default-free logic programs. In Section 6, the rewriting approach is discussed to implement repair problems on top of answer set engines. Finally, some relevant related works are discussed in Section 7, while some concluding remarks are reported in Section 8.

2 Preliminaries

2.1 Computational Complexity

Some basic definitions about complexity theory are recalled next. The reader is referred to [64,46] for more on this.

Decision problems are maps from strings (encoding the input instance over a suitable alphabet) to the set $\{$"*yes*", "*no*"$\}$. A (possibly nondeterministic) Turing machine M answers a decision problem if on a given input x, (i) M has a computation that halts in an accepting state iff x is a "yes" instance, and (ii) all the computations of M halt in some rejecting state iff x is a "no"instance.

The class P is the set of decision problems that can be answered by a deterministic Turing machine in polynomial time. The class of decision problems that can be solved by a nondeterministic Turing machine in polynomial time is denoted by NP, while the class of decision problems whose complementary problem is in NP, is denoted by co-NP.

The classes $\Sigma_{\rm k}^{\rm P}$ and $\Pi_{\rm k}^{\rm P}$, forming the *polynomial hierarchy*, are defined as follows: $\Sigma_0^{\rm P} = \Pi_0^{\rm P} = {\rm P}$ and for all $k \geq 1$, $\Sigma_{\rm k}^{\rm P} = {\rm NP}^{\Sigma_{\rm k-1}^{\rm P}}$, $\Delta_{\rm k}^{\rm P} = {\rm P}^{\Sigma_{k-1}^{P}}$, and $\Pi_{\rm k}^{\rm P} = \text{co-}\Sigma_{\rm k}^{\rm P}$ where co-$\Sigma_{\rm k}^{\rm P}$ denotes the class of problems whose complementary version is in $\Sigma_{\rm k}^{\rm P}$, and where $\Sigma_{\rm k}^{\rm P}$ (resp. $\Delta_{\rm k}^{\rm P}$) models computability by a nondeterministic (resp. deterministic) polynomial-time Turing machine that may use an oracle that is, loosely speaking, a subprogram, that can be run with constant computational cost, for solving a problem in $\Sigma_{\rm k-1}^{\rm P}$. Note that $\Sigma_1^{\rm P} = {\rm NP}$ and $\Pi_1^{\rm P} = \text{co-NP}$. The class ${\rm D}_{\rm k}^{\rm P}$, $k \geq 1$, is the class of problems defined as a conjunction of two independent problems, one from $\Sigma_{\rm k}^{\rm P}$ and one from $\Pi_{\rm k}^{\rm P}$, respectively. Note that for all $k \geq 1$, $\Sigma_{\rm k}^{\rm P} \subseteq {\rm D}_{\rm k}^{\rm P} \subseteq \Sigma_{\rm k+1}^{\rm P}$.

Functions (also *computation problems*) are (partial) maps from strings to strings, which can be computed by suitable Turing machines, called *transducers*, which have an output tape. In particular, a transducer T computes a string y on input x, if some branch of the computation of T on x halts in an accepting state and, in that state, y is on the output tape of T. Thus, a function f is computed by T, if (i) T computes y on input x iff $f(x) = y$, and (ii) all the branches of T halt in some rejecting state iff $f(x)$ is undefined.

In this paper, some classes of computation problems will be referred to which are illustrated next (see, also, [50, 70]). For each class of decision problems, say $\mathcal{C}$, F$\mathcal{C}$ denotes its functional version; for instance, FNP denotes the class of functions computed by nondeterministic transducers in polynomial time, ${\rm F}\Sigma_2^{\rm P}$ denotes the class of functions computed in polynomial time by nondeterministic transducers which use an NP oracle, and ${\rm F}\Delta_2^{\rm P}$ denotes the functions computed, in polynomial time, by a deterministic transducer which uses an NP oracle. In particular, ${\rm F}\Delta_2^{\rm P}[\mathcal{O}(\log {\rm n})]$ denotes the functions that can be computed in polynomial time by a deterministic transducer making logarithmically many calls (in the size n of the input) to an NP oracle.

In conclusion, the notion of reduction for decision and computation problems should be recalled. A decision problem A_1 is *polynomially reducible* to a decision problem A_2 if there is a polynomial time computable function h such that for every x, $h(x)$ is defined and A_1 outputs "yes" on input x iff A_2 outputs "yes" on input $h(x)$. A decision problem A is *complete* for the class $\mathcal{C}$ of the polynomial hierarchy iff A belongs to $\mathcal{C}$ and every problem in $\mathcal{C}$ is polynomially reducible to A. Moreover, a function f_1 is *reducible* to a function f_2 if there is a pair of polynomial-time computable functions h_1, h_2 such that for every x, $h_1(x)$ is defined, and $f_1(x) = h_2(x, w)$ where $w = f_2(h_1(x))$. A function f is hard for a class of functions F$\mathcal{C}$, if every $f' \in \mathcal{F}$ is polynomially reducible to f, and is complete for F$\mathcal{C}$, if it is hard for F$\mathcal{C}$ and belongs to F$\mathcal{C}$.

2.2 The Action Language $\mathcal{A}_K$

Planning capabilities for the agent will be modelled in the paper by means of the well-known action language $\mathcal{A}_K$ [10,73,74], which has been obtained by extending $\mathcal{A}$ [36] with suitable mechanisms for reasoning in the presence of sensing actions. Below, we shall recall the main peculiarities of this language.

We assume the existence of two disjoint sets of propositional letters $\mathcal{F}$ and $\mathcal{A}$, denoting the *fluents* of interest in the modelling of the world and the *actions* available to the agent, respectively. For any letter $f \in \mathcal{F}$, a fluent *literal* is either f itself or its negation denoted by $\neg f$. For a literal l, $\neg\, l$ denotes the complementary literal; thus, for a letter $f \in \mathcal{F}$, $\neg(\neg f) = f$. Moreover, for any set S of literals, we denote by $S^{\neg}$ the set $\{\neg f \mid f \in S\}$.

Syntax. A description in $\mathcal{A}_K$ consists of propositions of four kinds:

(1) A *v-proposition* is an expression of the form:

$$\textbf{initially } f$$

where f is a literal in $\mathcal{F} \cup \mathcal{F}^{\neg}$. A v-proposition specifies the truth value of a fluent in the initial situation.

(2) An *ef-proposition* is an expression of the form:

$$a \textbf{ causes } f \textbf{ if } p_1, ..., p_n$$

where a is an action in $\mathcal{A}$ and $f, p_1, ..., p_n$ $(n \geq 0)$ are literals in $\mathcal{F} \cup \mathcal{F}^{\neg}$[1]. An ef-proposition describes the effect of an action a on the truth value of a fluent. The conjunction $p_1, ..., p_n$ is the precondition of the proposition.

(3) A *k-proposition* is an expression of the form:

$$a \textbf{ determines } f \textbf{ if } q_1, ..., q_n$$

where a is an action in $\mathcal{A}$ and $f, q_1, ..., q_n$ $(n \geq 0)$ are literals in $\mathcal{F} \cup \mathcal{F}^{\neg}$. A k-proposition describes that an action a has the effect of sensing the truth value of a fluent f.

(4) An *ex-proposition* is an expression of the form:

$$\textbf{executable } a \textbf{ if } l_1, ..., l_n$$

where a is an action in $\mathcal{A}$ and $l_1, ..., l_n$ $(n \geq 0)$ are literals in $\mathcal{F} \cup \mathcal{F}^{\neg}$. An ex-proposition states an executability condition for action a.

[1] If $n = 0$ the symbol **if** is omitted, also for all the other kinds of propositions.

Two ef-propositions a **causes** f **if** $p_1, ..., p_n$ and a **causes** $\neg f$ **if** $p'_1, ..., p'_m$ are *contradictory* if $\{p_1, ..., p_n\} \cap \{\neg p'_1, ..., \neg p'_m\} = \emptyset$. A *domain description* $\mathcal{D}$ (over fluents $\mathcal{F}$ and actions $\mathcal{A}$) is a set of propositions of the form above that does not contain contradictory ef-propositions.

Semantics. There are many possible ways to assign semantics to a domain description $\mathcal{D}$. In fact, when one needs to capture the difference between what is true in the world and what is known about the world, modalities and Kripke models can be considered (e.g, [71]). In this paper, we shall resort instead to approximations of this approach enjoying nicer computational properties [7].

In particular, we shall consider the semantics that has firstly been discussed in [10] based on the notion of *0-approximation.*

In the 0-approximation semantics a state S for the agent is a set of fluent literals that is *coherent*, i.e., such that $S \cap S^{\neg} = \emptyset$. The semantics assumes a three-valued interpretation of the world, i.e., S may possibly be *incomplete.* Thus, a literal f is true in S if $f \in S$; f is false in S if $\neg f \in S$; and, f is unknown in S if neither f nor $\neg f$ belongs to S.

The initial state S_0 consists of all fluent literals f for which the proposition **initially**f occurs in $\mathcal{D}$. Then, the domain description $\mathcal{D}$ specifies how the agent perception of the environment may evolve as a consequence of executing actions. Thus, the core of the semantics consists in specifying a transition function Φ_a for each action a, mapping each state S to the sets of states that are reachable from S when applying a.

An action a is *executable* in S if there is an ex-proposition **executable** a **if** $l_1, ..., l_n$ and $\{l_1, ..., l_n\} \subseteq S$. Then, for a given state S, and for an action a executable in S:

- Let $e_a(S)$ be the set containing each literal f such that

  $$a \textbf{ causes } f \textbf{ if } p_1, ..., p_n$$

  occurs in the domain description and $\{p_1, ..., p_n\} \subseteq S$ (we say that the precondition is true in S).
- Let $F_a(S)$ be the set containing each literal f such that

  $$a \textbf{ causes } f \textbf{ if } p_1, ..., p_n$$

  occurs in the domain description and $\{\neg p_1, ..., \neg p_n\} \cap S = \emptyset$ (we say that the precondition is possibly true in S).
- Let $K_a(S)$ be the set containing each fluent f such that

  $$a \textbf{ determines } f \textbf{ if } q_1, ..., q_n$$

  occurs in the domain description and $\{q_1, ..., q_n\} \subseteq S$.

W.l.o.g., we assume that actions in $\mathcal{A}$ are partitioned into sensing actions (not involved in ef-propositions) and non-sensing ones (not involved in k-propositions). Then:

– For a non-sensing action a, $\Phi_a(S)$ is a singleton set $\{S'\}$ where S' contains all the fluent literals that either are effects of some ef-proposition caused by a and whose precondition is true in S, or are in S and their opposite is not entailed by some ef-proposition caused by a and whose precondition is possibly true in S. Formally:

$$\Phi_a(S) = \{S \cup e_a(S) \setminus F_a(S)^{\neg}\} \tag{1}$$

– For a sensing action a, $\Phi_a(S)$ is instead the set of all the possible states each of which is obtained by adding to S the fluent literals that can be true or false as a result of the sensing. Formally,

$$\Phi_a(S) = \{S \cup v \mid v \subseteq K_a(S) \cup K_a(S)^{\neg} \text{ and } S \cup v \text{ is coherent}\} \tag{2}$$

A more formal treatment for the 0-approximation semantics can be found in [10,73,7,74], where the reader is also referred to for examples and further discussions. In particular, we omit from here discussing actual planning mechanisms, since in monitoring frameworks such as ours, the focus is on reasoning on top of plans that are assumed to have already been executed.

2.3 Extended Logic Programs

Throughout the paper we will assume that agents' internal trustable knowledge is modeled by means of propositional Extended Logic Programs (short: ELPs). We briefly recall that a propositional ELP is a set of rules of the form

$$L_0 \leftarrow L_1, \ldots, L_m,\ \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_n.$$

where $n \geq m \geq 0$, the symbol "*not*" denotes negation by default, and each L_i is a literal, i.e. an expression of the form p or $\neg p$ with p being a propositional letter and the symbol "$\neg$" denoting classical negation. By $\mathbf{h}(r)$ we denote the head L_0 of the rule r, and by $\mathbf{b}(r)$ its body $L_1, \ldots, L_m, \mathit{not}\ L_{m+1}, \ldots, \mathit{not}\ L_n$. An ELP is *positive* if classical negation does not occur in the program.

In the following, we consider the *answer set* semantics for ELPs [35]. *Answer sets* of an ELP P are defined as follows.

Let $U(P)$ denote the set of all the literals obtained using the propositional letters occurring in P. Let a *context* be any subset of $U(P)$.

Let P be a *negation-by-default-free* ELP. Call a context S *closed under* P iff for each rule $L_0 \leftarrow L_1, \ldots, L_m$ in P, if $L_1, \ldots, L_m \in S$, then $L_0 \in S$. An *answer set* of P is any minimal context S such that (1) S is closed under P and (2) if S is *incoherent*, that is, if there exists a propositional letter p such that both $p \in S$ and $\neg p \in S$, then $S = U(P)$.

An answer set of a general ELP is defined as follows. Let the *reduct of P w.r.t the context S*, denoted by $\mathit{Red}(P, S)$, be the ELP obtained from P by deleting (i) each rule that has *not* L in its body for some $L \in S$, and (ii) all subformulae of the form *not* L of the bodies of the remaining rules. Any context S which is an answer set of $\mathit{Red}(P, S)$ is an *answer set* of P.

By $\mathrm{ANSW}(P)$ we denote the collection of all the answer sets of an ELP P. An ELP P is *inconsistent* iff $\mathrm{ANSW}(P) = \emptyset$. An ELP P is *incoherent* iff it has a unique answer set that is incoherent.

An ELP P *cautiously* entails a literal l, written $P \models_c l$, iff for each $S \in \mathrm{ANSW}(P)$, $l \in S$. An ELP P *bravely* entails a literal l, written $P \models_b l$, iff there exists $S \in \mathrm{ANSW}(P)$ such that $l \in S$. Whenever the specific form of entailment is not relevant, we shall simply write $P \models l$.

We conclude by recalling some well-known complexity results pertaining extended logic programs.

A negation-by-default-free ELP has one and only one answer set. Computing the answer set of a negation-by-default-free ELP is a P-complete problem. Moreover, for this class of ELPs there is no difference between the brave and the cautious semantics, and deciding whether a negation-by-default-free ELP entails a literal is a P-complete problem.

Given a context S and an ELP P, deciding whether $S \in \mathrm{ANSW}(P)$ can be done in polynomial time by computing $\mathit{Red}(P, S)$ and then checking whether S is the unique answer set of the negation-by-default-free ELP $\mathit{Red}(P, S)$.

Finally, we recall that deciding whether an ELP bravely entails a literal is an NP-complete problem, while deciding whether an ELP cautiously entails a literal is a co-NP-complete problem.

3 Formal Framework

In this section, we formally define the problem of reasoning about possibly faulty sensors. In particular, we present some techniques that an agent might exploit to identify "anomalous" observations (and, hence, faulty sensors), and a "repair approach" in execution monitoring accommodating the uncertainty on the outcome of the sensors.

In fact, all these techniques will be accommodated in a framework where an agent A is represented as a tuple $\langle \mathcal{F}, \mathcal{S}, \mathcal{D}, \mathcal{K} \rangle$ such that $\mathcal{F}$ denotes the set of propositional letters used to describe the state of the world; $\mathcal{S}$ is a set of available sensors; $\mathcal{D}$ is a domain description that is meant to characterize the set of possible state evolutions determined by the applications of actions; and, $\mathcal{K}$ denotes the internal background knowledge A may use to reason about the current state of the world. Thus, we start the exposition by providing details on each of the components above.

3.1 States and Sensors ($\mathcal{F}$ and $\mathcal{S}$)

The first ingredient in the agent formalization is a set $\mathcal{F}$ of propositional letters that are meant to denote the fluents of interest in the modelling of the world. For notational convenience (given our interest w.r.t. those fluents that can be sensed), we shall assume that $\mathcal{F}$ is partitioned into two (disjoint) sets of letters:

(i) *beliefs* $\mathcal{B}$, denoting the agent beliefs about the status of the world;
(ii) *observables* $\mathcal{O}$, modeling the status of the world as returned by a set $\mathcal{S}$ of environment sensors. In more detail, for each sensor $s \in \mathcal{S}$, $\lambda(s) \subseteq \mathcal{O}$ denotes the set of propositional letters that are sensed by s, where $\lambda(s_1) \cap \lambda(s_2) = \emptyset$ for each pair of sensors s_1 and s_2 in $\mathcal{S}$—thus, we assume that any observable can be sensed by one sensor at most.

$$\mathcal{B}: \begin{cases} floor_0,\ floor_1,\ floor_2 \\ building_l,\ building_r \end{cases} \qquad \mathcal{S}: \begin{cases} \lambda(s_1) = \{availF_0,\ availF_1,\ availF_2\} \\ \lambda(s_2) = \{availB_l,\ availB_r\} \\ \lambda(s_3) = \{availPlace\} \end{cases}$$

$$\mathcal{O}: \begin{cases} availF_0,\ availF_1,\ availF_2 \\ availB_l,\ availB_r \\ availPlace \end{cases} \qquad \mathcal{K}: \begin{cases} availPlace \leftarrow availF_i,\ availB_x, \\ \qquad\qquad\qquad floor_i,\ building_x. \end{cases}$$

Fig. 2 Formalization of the parking lot example.

Example 1 In the *parking lot* application, fluents are associated with the existence of parking places available at a certain level and building of the parking lot as well as with the current location of the car to be parked.

Accordingly, the sensors $\mathcal{S}$, the observables $\mathcal{O}$ and the beliefs $\mathcal{B}$ are reported in Figure 2, where, for instance, $availF_1$ means that there are parking places available at level 1, $availB_l$ that at the current floor there are places available in the left building, and $building_r$ that the car is currently in the building on the right. $\lhd$

In order to help distinguishing beliefs from observables in a given state S, we shall represent S as a pair of sets $\langle S \cap (\mathcal{B} \cup \mathcal{B}^{\neg}), S \cap (\mathcal{O} \cup \mathcal{O}^{\neg})\rangle$. Moreover, $S \cap (\mathcal{B} \cup \mathcal{B}^{\neg})$ (resp. $S \cap (\mathcal{O} \cup \mathcal{O}^{\neg})$) will be denoted by $\mathcal{B}(S)$ (resp. $\mathcal{O}(S)$).

Recalling that we exploit a 3-valued interpretation of the domain (cf. Section 2.2), $l \in \mathcal{B}(S)$ (resp., $l \in \mathcal{O}(S)$) means that l is known to be true in S; $\neg\, l \in \mathcal{B}(S)$ (resp., $\neg\, l \in \mathcal{O}(S)$), where l is a letter, means that l is known to be false in S; and, $l \in \mathcal{F}$ such that neither l nor $\neg\, l$ belongs to $\mathcal{B}(S)$ (resp. $\mathcal{O}(S)$) means that l is unknown in S.

3.2 Transitions and Evolutions ($\mathcal{D}$)

An agent A operating in the environment (modeled over $\mathcal{F}$) has to be equipped with some mechanisms for executing *actions* that cause *transitions* between states. In this respect, we note that several ways to define transitions between states in the presence of sensing actions have been proposed in the literature, accounting, e.g., for non-deterministic effects, causal effects, probabilities, and so on (see, e.g., [44,54,73,74] and references therein).

In fact, the concept of anomaly in state evolution is completely orthogonal with respect to the specific approach being adopted. Hence, in principle, it would just suffice to assume that a set of actions is given together with a function Φ_a, for each action a, mapping A's current state into the set of states that A may reach when applying a. For the sake of concreteness, we shall contextualize our proposal to the case of the language $\mathcal{A}_K$ and its 0-approximation semantics.

Example 2 Consider again the parking lot example. Then, the actions of sensing s_1, s_2, and s_3 can be represented via the following k-propositions:

$$\begin{array}{l} s_1 \textbf{ determines } availF_0 \\ s_1 \textbf{ determines } availF_1 \\ s_1 \textbf{ determines } availF_2 \\ \textbf{executable } s_1 \textbf{ if } floor_0 \\ s_2 \textbf{ determines } availB_l \\ s_2 \textbf{ determines } availB_r \\ s_3 \textbf{ determines } availPlace \end{array}$$

where action names coincide with the name of the sensors—action names will coincide with sensor names also in the following.

Moreover, besides sensing, the agent may act in the environment through the actions $move_{i,j}$ and $enter_x$, where indexes i, j and x are used as placeholders of actual indexes $0, 1, 2$ and l, r, respectively. In particular, movements of the agent can be formalized via the following ef-propositions:

$$\begin{array}{l} move_{i,j} \textbf{ causes } \neg floor_i \textbf{ if } floor_i \\ move_{i,j} \textbf{ causes } floor_j \textbf{ if } floor_i \\ enter_x \textbf{ causes } building_x \end{array}$$

Intuitively, $move_{i,j}$ determines moving from floor i to floor j, while $enter_x$ determines entering into the building x. $\lhd$

The repeated application of actions defines an evolution for the agent, which can be seen as an actual plan that the agent is executing in order to achieve a given goal starting from the initial state $\langle S_B^0, S_O^0 \rangle$. In this paper, we are not interested in investigating on suitable planning mechanisms for the agent. Rather, we are interested in analyzing how reasoning mechanisms applied on the events registered during an evolution can possibly lead to singling out malfunctioning of the sensors. Thus, we consider a monitoring perspective (e.g., [61,38,31,33,25]) focused on the identification of noisy sensors.

As in classical monitoring frameworks, we assume that the various actions performed by the agent (and their consequences on the state of the world) are registered and available for analysis purposes.

Formally, for a state $\langle S_B, S_O \rangle$ and an action a, let us denote by $\langle S_B, S_O \rangle \rightarrow_a \langle S'_B, S'_O \rangle$ the fact that the application of a on $\langle S_B, S_O \rangle$ led the agent to the state $\langle S'_B, S'_O \rangle \in \Phi_a(\langle S_B, S_O \rangle)$. In fact, note that since we assume a 3-valued interpretation over states, $\langle S'_B, S'_O \rangle$ may not completely specify the truth value for all the possible fluents in $\mathcal{F}$. Thus, in general $\langle S'_B, S'_O \rangle$ succinctly represents a set of alternatives on the actual world, rather than a concrete one. Moreover, note that since we are considering the case of the 0-approximation semantics, if a is a non-sensing action, then $\langle S'_B, S'_O \rangle$ is univocally determined by $\langle S_B, S_O \rangle$ and a (cf. Equation (1)). To the contrary, if a is a sensing action, then $\langle S'_B, S'_O \rangle$ precisely stores the values that have been sensed by the agent in the transition (cf. Equation (2)).

Definition 1 (Evolution) An *evolution* H for A is a succession of states of the form:

$$\langle S_B^0, S_O^0\rangle \rightarrow_{a_1} \langle S_B^1, S_O^1\rangle \rightarrow_{a_2} \dots \rightarrow_{a_n} \langle S_B^n, S_O^n\rangle,$$

where:

(i) $\langle S_B^0, S_O^0\rangle$ satisfies all the v-propositions of A's domain description;
(ii) a_i is executable in $\langle S_B^i, S_O^i\rangle$ for each $1 \leq i \leq n$; and,
(iii) $\langle S_B^i, S_O^i\rangle \in \Phi_{a_i}(\langle S_B^{i-1}, S_O^{i-1}\rangle)$, for each $1 \leq i \leq n$. □

Note that the final state of an evolution not involving sensing actions is univocally determined by the actions on which it is defined. Moreover, an evolution possibly involving sensing actions precisely reports the values that have been sensed. Thus, provided the knowledge about these values and the initial state, the final state is again univocally determined.

In our framework, we assume that all the values sensed by the agent during the evolution are registered, so that the actual evolution H is available and constitutes the basis for identifying noisy sensors.

Example 3 Consider again the domain description discussed in Example 2, and enrich it with the proposition:

$$\textbf{initially } floor_0$$

Consider, now, the evolution H consisting of the following five actions: $a_1 : s_1$; $a_2 : move_{0,2}$; $a_3 : s_2$; $a_4 : enter_l$; and $a_5 : s_3$, and where sensed values in H are such that $\mathcal{O}(state_1(H)) \supseteq \{\neg availF_0, \neg availF_1, availF_2\}$, $\mathcal{O}(state_3(H)) \supseteq \{availB_l, \neg availB_r\}$, and $\mathcal{O}(state_5(H)) \supseteq \{\neg availPlace\}$.

An illustration for this evolution is reported in Figure 1, which formally is as follows:

$\langle\{floor_0\}, \{\}\rangle \rightarrow_{s_1}$
$\langle\{floor_0\}, \{\neg availF_0, \neg availF_1, availF_2\}\rangle \rightarrow_{move_{0_2}}$
$\langle\{floor_2\}, \{\neg availF_0, \neg availF_1, availF_2\}\rangle \rightarrow_{s_2}$
$\langle\{floor_2\}, \{\neg availF_0, \neg availF_1, availF_2, availB_l, \neg availB_r\}\rangle \rightarrow_{enter_l}$
$\langle\{floor_2, building_l\}, \{\neg availF_0, \neg availF_1, availF_2, availB_l, \neg availB_r\}\rangle \rightarrow_{s_3}$
$\langle\{floor_2, building_l\}, \{\neg availF_0, \neg availF_1, availF_2, availB_l, \neg availB_r, \neg availPlace\}\rangle$

Intuitively, the agent starts in H from the state $\langle\{floor_0\}, \emptyset\rangle$ and senses s_1 (which checks for a parking place on the various floors).

Based on the outcome of the sensing, he then plans to park at the second floor, and eventually after sensing s_2 he enters into the left building, and checks for the actual parking there. ◁

In the following, $len(H)$ denotes the number of transitions occurring in the evolution H; $state_i(H)$ denotes the ith state of the evolution H; $state(H)$ denotes $state_{len(H)}(H)$; $act_i(H)$ denotes the ith action performed in the evolution; $H[i]$ denotes the evolution $state_0(H) \rightarrow_{act_1(H)} \dots \rightarrow_{act_i(H)} state_i(H)$.

3.3 Anomalies and Repairs ($\mathcal{K}$)

The last ingredient of the agent formalization is the internal background knowledge $\mathcal{K}$ that allows him to reason about the current state of the world. As discussed in Section 2.3, $\mathcal{K}$ is assumed to be formalized as an extended logic program (over the alphabet of $\mathcal{F}$).

Note that, in principle, we may use $\mathcal{K}$ to entail the truth value of some fluents that are possibly unknown in the given state, thereby affecting the definition of transition functions. This is, for instance, the approach pursued in [75], where $\mathcal{K}$ is enriched with static causal laws. In fact, this is not our focus and we shall use $\mathcal{K}$ under a different perspective.

Indeed, the agent background theory will be used as a sort of repository of integrity constraints expressed over the values returned from the sensors. Therefore, this theory allows the agent, at each step, to check whether some anomalous situations occur, i.e., whether a discrepancy between the trustable knowledge and the result of sensing emerges. It is worthwhile noticing that this perspective has received less attention in the context of monitoring frameworks, even though it has deeply been investigated in related fields such as belief revision, inconsistency management, and diagnosis, just to cite a few (see Section 7, for an overview of related approaches). A formalization of the concept of "disagreement" specifically tailored for dealing with evolutions in noisy environments is discussed below.

Definition 2 (Anomaly) Let H be an evolution for an agent A with internal knowledge $\mathcal{K}$. A set of observations $W \subseteq \mathcal{O}(state(H))$ is an *anomaly* for A in H if:

$$\exists w \in W, \text{ such that } th(A,H) \setminus W \models \neg w,$$

where $th(A,H)$ denotes the theory $\mathcal{K} \cup \mathcal{B}(state(H)) \cup \mathcal{O}(state(H))$. □

Intuitively, in the definition above, anomalies are characterized as those observations whose removal from the current perception of the environment allows to entail precisely the opposite of one of them. This is an evidence of a disagreement between the trustable knowledge and the observations gained from the sensors, even though a classical incoherence need not to necessarily occur.

Example 4 Consider the evolution H in Example 3 and the knowledge base $\mathcal{K}$ that is reported in Figure 2. Basically, $\mathcal{K}$ tells the agent that if he is at floor i of the building j and sensors s_1, when queried, signalled parking availability at level i and sensor s_2, when queried, signalled a parking availability in building j, then there must be indeed at least one parking place available at his current position.

However, according to H, the agent is planning to park at the second floor in the left building after sensing s_1 and s_2. But, the result of sensing s_3 is anomalous, as it disagrees with its beliefs (in $\mathcal{K}$) according to which $availPlace$ should be true there. ◁

Given an anomaly, we are interested in finding possible fixes for it, i.e., "alternative" evolutions defined over the same set of transitions in which, however, the result of the sensing actions may differ from the evolution in which the anomaly has been singled out. In other words, by fixing an evolution, the agent modifies (its perception of) the values returned by some sensors which are, therefore, implicitly regarded as faulty, when such sensors' values causes anomalies to occur in the evolution at hand. This is formalized next with the notion of *repair* for an evolution.

Definition 3 (Repair) An evolution H' for A is a *repair* for H w.r.t. an anomaly W if:
(1) $len(H) = len(H')$,
(2) $state_0(H) = state_0(H')$,
(3) $act_i(H) = act_i(H')$, for each $1 \leq i \leq len(H)$, and
(4) $\forall w \in W \cap \mathcal{O}(state(H'))$, $th(A, H') \setminus W \not\models \neg w$. □

Note that, differently from classical goal-oriented monitoring approaches, our notion of repair does not consist in undoing some of the actions performed so far. Rather, by repairing an evolution H, we just mean equipping the agent with the capability of constructing an alternative perception of the world that is consistent with all the actions performed from the given initial state (cf. conditions (1), (2), and (3) in Definition 3) and that resolves the conflict with the anomaly W (cf. condition (4) in Definition 3).

Moreover, we note that a repair H' may be defined by just modifying the values sensed by sensing actions, since non-sensing actions play deterministically on the given states. Thus, by repairing an evolution, we basically aim at determining values for the sensors that will eliminate the discrepancy with agent background knowledge. This is further illustrated below.

Example 5 A repair for our running example is obtained by replacing the value returned by sensor s_2 with $\{\neg availB_l, availB_r\}$ while keeping the values returned by s_1 and s_3. This represents the scenario in which the available place is in the opposite building of the same floor, and in which the output of s_3 is no longer perceived as anomalous. ◁

4 Reasoning with Noisy Sensors: General Results

Now that we have defined the formal framework for anomaly detection and repairing of an agent's belief state evolution, we turn to the problem of defining relevant agent's reasoning tasks. Moreover, as already stated in the Introduction, it is important to pinpoint the computational complexity characterizing such tasks, since this is an important premise towards devising effective and optimized implementations of the framework.

Specifically, in this section we consider the basic notions of anomaly and repair, and the following relevant problems:

Anomaly-Existence: Given an agent A and an evolution H for A, is there any anomaly W for A in H?

Repair-Existence: Given an agent A and an anomaly W for A in an evolution H, does there exist a repair H' for H w.r.t. W?

Anomaly&Repair-Checking: Let A be an agent and H an evolution. Given an evolution H' and a set of observables $W \subseteq \mathcal{O}(state(H))$, is W an anomaly for A in H, and H' a repair for H w.r.t. W?

Other problems related to the notion of "full repair" and to variants where minimality conditions are taken into account are investigated in Section 5.

	not -free	cautious	brave
Anomaly-Existence	P-c	Σ_2^P-c	NP-c
Repair-Existence	NP-c	NP-c	Σ_2^P-c
Anomaly&Repair-Checking	P-c	in Δ_2^P	D^P-c

Fig. 3 Complexity of Basic Problems.

Overview of the Results. Complexity results concerning the problems defined above are depicted in Figure 3, where analysis is conducted for the setting of negation-by-default-free (short: not-free) knowledge bases as well as for general knowledge bases under both the brave and the cautious semantics.

It turns out that all the problems for not-free programs are confined within the first level of the polynomial hierarchy and, in particular, that Anomaly-Existence and Anomaly&Repair-Checking are efficiently solvable, while Repair-Existence appears intrinsically more complex.

Dealing with general knowledge bases leads, instead, to an increase in complexity but with some subtle differences between brave and cautious semantics. Indeed, while with most reasoning tasks swapping from brave to cautious semantics moves the complexity to complementary classes, in our case seemingly unrelated results are obtained. In particular, Repair-Existence is more difficult under brave reasoning, while Anomaly-Existence is more difficult under the cautious semantics.

While at a first glance these results may seem counter-intuitive, they can be readily understood by taking a closer look to the two main conditions to be checked while deciding for the aforementioned problems. These conditions appear to be, loosely speaking, specular to each other. As a matter of fact, checking for anomaly existence (see Definition 2) amounts to verify whether there exists a set of observations W and an observation w in W such that the theory $th(A,H) \setminus W$ entails the literal $\neg w$. To the contrary, checking for the existence of a repair (see Definition 3) amounts basically to check whether for each observation w occurring both in the anomaly W and in the current agent state, the theory $th(A,H') \setminus W$ does not entail the literal $\neg w$. The dual behavior of the two investigated semantics for the entailment operator, which are the cautious and brave one, then completes the picture, explaining why complexity results for Anomaly-Existence and Repair-Existence are mirrored when switching from the brave to the cautious semantics and vice-versa, and also why, within the same problem, there is a gap between the levels of the polynomial hierarchy associated with complexity results under cautious and brave semantics. The reader is referred to the proofs reported next for the formal details.

4.1 Proofs of Complexity Results

Next we present details on the results depicted in Figure 3. To this end, we find it useful to introduce some additional notations and definitions that will be used throughout the proofs.

Let L be a coherent set of literals. We denote with $\mathcal{T}_L$ the truth assignment on the set of letters occurring in L such that, for each positive literal $p \in L$, $\mathcal{T}_L(p) = \mathbf{true}$, and for each negative literal $\neg p \in L$, $\mathcal{T}_L(p) = \mathbf{false}$.

Let L be a set of literals. Then we denote with L^+ the set of positive literals occurring in L, and with L^- the set of negative literals in L.

Let T be a truth assignment of the set $\{x_1, \ldots, x_n\}$ of boolean variables. Then we denote with $Lit(T)$ the set of literals $\{\ell_1, \ldots, \ell_n\}$, such that ℓ_i is x_i if $T(x_i) = \mathbf{true}$ and is $\neg x_i$ if $T(x_i) = \mathbf{false}$, for $i = 1, \ldots, n$.

In the following, we will denote by

Φ: the boolean formula $\Phi = C_1 \wedge \ldots \wedge C_m$ in conjunctive normal form, with $C_j = t_{j,1} \vee t_{j,2} \vee t_{j,3}$, where each of $t_{j,1}$, $t_{j,2}$ and $t_{j,3}$ is a literal on the set of boolean variables $X = x_1, \ldots, x_n$;

Ψ: the quantified boolean formula $\Psi = \exists X \forall Y f(X,Y)$, where $X = x_1, \ldots, x_n$, $Y = y_1, \ldots, y_m$, $f(X,Y) = D_1 \vee \ldots \vee D_l$ is in conjunctive normal form, with $D_k = t_{k,1} \wedge t_{k,2} \wedge t_{k,3}$, and each of $t_{k,1}, t_{k,2}, t_{k,3}$ is a literal on the set of boolean variables X and Y.

Armed with these notations, we can now start our investigation by discussing the complexity of the Anomaly-Existence problem.

Theorem 1 Anomaly-Existence *is*

(1) P*-complete, for negation-by-default-free ELPs,*
(2) NP*-complete, for general ELPs under brave semantics, and*
(3) Σ_2^{P}*-complete, for general ELPs under the cautious semantics.*

Proof

(1) (Membership) First of all, we recall that for negation-by-default-free ELPs the *monotonicity property* holds, that is: if $P \models w$ then, for any ELP P' such that $P' \supseteq P$, $P' \models w$. Furthermore, we recall that given a negation-by-default-free ELP P and a literal w, deciding whether $P \models w$ is a P-complete problem, hence solvable in polynomial time.

Now we prove that there exists an anomaly W for A in H iff there exists an observable $w \in \mathcal{O}(state(H))$ such that $\{w\}$ is an anomaly for A in H. Let W be an anomaly for A in H. Then there exists a literal $w \in W$ such that $th(A,H) \setminus W \models \neg w$, and, from the monotonicity property, it follows that $th(A,H) \setminus \{w\} \models \neg w$. We can conclude that $\{w\}$ is an anomaly for A in H. The reverse direction is immediately verified.

Thus, it can be decided whether there exists an anomaly W in H for A as follows. For each literal $w \in \mathcal{O}(state(H))$, check whether $th(A,H) \setminus \{w\} \models \neg w$. If at least one of these checks holds true, then return "yes", otherwise return "no". The overall procedure can be accomplished in polynomial time, since the number $|\mathcal{O}(state(H))|$ of observables sensed in H

is polynomially related to the size of H, while each entailment check is polynomial time solvable.

(Hardness) Given a positive logic program P and a propositional letter w, consider the agent $A(P,w)$ with the set of observables $\mathcal{O} = \{w\}$, the knowledge base P, the sensor $s(P,w)$ and the domain description:

$$s(P,w) \textbf{ determines } w$$

Also, consider the evolution:

$$H(P,w) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(P,w)} \langle \emptyset, \{\neg w\} \rangle.$$

As already recalled above, given a negation-by-default-free ELP P and a literal w, deciding whether $P \models w$ is a P-complete problem. In order to complete the proof, now we prove that there exists an anomaly for $A(P,w)$ in $H(P,w)$ iff $P \models w$.

($\Rightarrow$) Assume that there exists an anomaly W for $A(P,w)$ in $H(P,w)$. Then $W \subseteq \mathcal{O}(state(H(P,w))) = \{\neg w\}$. Since W cannot be empty, it is the case that W is the set $\{\neg w\}$. Thus, by definition of anomaly, it holds that $th(A(P,w), H(P,w)) \setminus \{\neg w\} \models w$. The result follows by noticing that $th(A(P,w), H(P,w)) \setminus \{\neg w\} = (P \cup \emptyset \cup \{\neg w\}) \setminus \{\neg w\} = P$, and, consequently, that $P \models w$.

($\Leftarrow$) Assume that $P \models w$. Then $W = \{\neg w\}$ is an anomaly for $A(P,w)$ in $H(P,w)$. Indeed, $th(A(P,w), H(P,w)) \setminus \{\neg w\} = P$, and, consequently, $th(A(P,w), H(P,w)) \setminus \{\neg w\} \models w$.

(2) *(Membership)* The problem can be solved by a polynomial time nondeterministic Turing machine that guesses a subset $W \subseteq \mathcal{O}(state(H))$ together with a literal $w \in W$ and a context S such that $\neg w \in S$, and then checks, in polynomial time, that S is an answer set of $Red(th(A,H) \setminus W, S)$ and, hence, of $th(A,H) \setminus W$.

(Hardness) Given the boolean formula Φ, consider the agent $A(\Phi)$ with the set of observables $\mathcal{O}(\Phi) = \{x_0, x_1, \ldots, x_n\}$ (where x_0 is a new variable not occurring in Φ), the sensor $s(\Phi)$, the knowledge base $\mathcal{K}(\Phi)$:

$$\begin{array}{ll}
r_0 : sat \leftarrow c_1, \ldots, c_m. & \\
r_{1,j} : c_j \leftarrow \gamma(t_{j,1}). & (1 \leq j \leq m) \\
r_{2,j} : c_j \leftarrow \gamma(t_{j,2}). & (1 \leq j \leq m) \\
r_{3,j} : c_j \leftarrow \gamma(t_{j,3}). & (1 \leq j \leq m) \\
r_{4,i} : \neg x_i \leftarrow not\ x_i, sat. & (0 \leq i \leq n)
\end{array}$$

where $\gamma(x_i) = x_i$ and $\gamma(\neg x_i) = not\ x_i$, and the domain description:

$$s(\Phi) \textbf{ determines } x_i \quad (0 \leq i \leq n)$$

Consider the evolution:

$$H(\Phi) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(\Phi)} \langle \emptyset, \mathcal{O}(\Phi) \rangle.$$

We prove that there exists an anomaly W for $A(\Phi)$ in $H(\Phi)$ iff Φ is satisfiable.

($\Rightarrow$) Assume that there exists an anomaly $W \subseteq \mathcal{O}(\Phi)$ for $A(\Phi)$ in $H(\Phi)$. Then there exists $x_i \in W$ such that $th(A(\Phi), H(\Phi)) \setminus W \models_b \neg x_i$. As the negation of x_i can be implied only by rule $r_{4,i}$, then it is the case that there exists an answer set S^Φ of $th(A(\Phi), H(\Phi)) \setminus W$ such that $sat \in S^\Phi$. Since $sat \in S^\Phi$, by rule r_0 it follows that, for each $1 \leq j \leq m$, it holds $c_j \in S^\Phi$. But, $c_j \in S^\Phi$ if and only if $\mathcal{T}^\Phi = \mathcal{T}_{(X \setminus W) \cup (W \setminus \{x_0\})^\neg}$ is a truth assignment to the variables of Φ that makes the clause C_j true (see rules $r_{1,j}$, $r_{2,j}$, and $r_{3,j}$). It can be eventually concluded that the truth assignment $\mathcal{T}^\Phi$ makes the formula $C_1 \wedge \ldots \wedge C_m$ true, or, equivalently, that Φ is satisfiable.

($\Leftarrow$) Assume that Φ is satisfiable, and let $\mathcal{T}^\Phi$ be a truth value assignment to the variables in X that makes Φ true. Then $W = \{x_0\} \cup \{x_i \mid \mathcal{T}^\Phi(x_i) = \textbf{false}\}$ is an anomaly for $A(\Phi)$ in $H(\Phi)$. Indeed, the context $S^\Phi = \{sat, c_1, \ldots, c_m\} \cup (X \setminus W) \cup W^\neg$ is an answer set of the program $th(A(\Phi), H(\Phi)) \setminus W$—in this respect, it is worth observing that rule r_{4_i} prevents possible incoherences in the theory. Then, the result follows since $\neg x_0 \in S^\Phi$, by construction.

(3) (Membership) The problem can be solved by a polynomial time nondeterministic Turing machine with an NP oracle that guesses a subset $W \subseteq \mathcal{O}(state(H))$ and an observable $w \in W$, and then decides whether $th(A, H) \setminus W \models_c \neg w$ by calling the NP oracle (co-NP check).

(Hardness) Given the formula Ψ, consider the agent $A(\Psi)$ with the set of observables $\mathcal{O}(\Psi) = \{x_0, x_1, \ldots, x_n\}$ (where x_0 is a new variable not occurring in Ψ), the sensor $s(\Psi)$, the knowledge base $\mathcal{K}(\Psi)$:

$$\begin{array}{rll}
r_0: & sat \leftarrow \delta(t_{k,1}), \delta(t_{k,2}), \delta(t_{k,3}). & (1 \leq k \leq l) \\
r_{1,j}: & y_j \leftarrow not\ \neg y_j. & (1 \leq j \leq m) \\
r_{2,j}: & \neg y_j \leftarrow not\ y_j. & (1 \leq j \leq m) \\
r_{3,i}: & \neg x_i \leftarrow not\ x_i, sat. & (0 \leq i \leq n)
\end{array}$$

where $\delta(x_i) = x_i$, $\delta(\neg x_i) = not\ x_i$, $\delta(y_j) = y_j$, and $\delta(\neg y_j) = \neg y_j$, and the domain description:

$$s(\Psi)\ \textbf{determines}\ x_i \quad (0 \leq i \leq n)$$

Consider also the evolution:

$$H(\Psi) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(\Psi)} \langle \emptyset, \mathcal{O}(\Psi) \rangle.$$

Now we prove that there exists an anomaly $W \subseteq \mathcal{O}(\Psi)$ for $A(\Psi)$ in $H(\Psi)$ iff Ψ is satisfiable.

($\Leftarrow$) First of all, it must be noticed that, for each $W \subseteq \mathcal{O}(\Psi)$ the ELP $th(A(\Psi), H(\Psi)) \setminus W$ is consistent. In particular, there exists a bijection between the truth value assignments $\mathcal{T}^Y$ to the set of variables Y and the answer sets of $th(A(\Psi), H(\Psi)) \setminus W$. Indeed, for each assignment $\mathcal{T}^Y$, either the set $(X \setminus W) \cup Lit(\mathcal{T}^Y)$ or the set $(X \setminus W) \cup W^\neg \cup \{sat\} \cup Lit(\mathcal{T}^Y)$ is an answer set of $th(A(\Psi), H(\Psi)) \setminus W$. Say $S^{\mathcal{T}^Y}$ the answer set associated with $\mathcal{T}^Y$.

Assume that there exists an anomaly W for $A(\Psi)$ in $H(\Psi)$. Then there exists $x_i \in W$ such that $th(A(\Psi), H(\Psi)) \setminus W \models_c \neg x_i$. As $\neg x_i$ can be implied only by rule $r_{3,i}$, then it is the case that, for each answer set S of $th(A(\Psi), H(\Psi)) \setminus W$, the literal sat is in S and, consequently, that the set $(X \setminus W) \cup W^{\neg}$ is contained in S. Hence, it follows from what we have noticed above that, for each truth value assignment $\mathcal{T}^Y$ to the set of variables Y, there exists one and only one answer set $S^{\mathcal{T}^Y}$ of $th(A(\Psi), H(\Psi)) \setminus W$, and viceversa, and $S^{\mathcal{T}^Y}$ is of the form $(X \setminus W) \cup W^{\neg} \cup \{sat\} \cup Lit(\mathcal{T}^Y)$. Since, for each $\mathcal{T}^Y$, $sat \in S^{\mathcal{T}^Y}$, by rule r_0 it can be concluded that W encodes a truth value assignment, that is $\mathcal{T}_{(X \setminus W) \cup (W \setminus \{x_0\})^{\neg}}$, to the set of variables X such that, for each truth value assignment $\mathcal{T}^Y$ to the set of variables Y, the formula $f(X,Y)$ holds true, i.e. that Ψ is satisfiable.

($\Leftarrow$) Assume that Ψ is satisfiable. Then there exists a truth value assignment $\mathcal{T}^X$ to the variables X that makes Φ true. Then $W = \{x_0\} \cup \{x_i \mid \mathcal{T}^{\Psi}(x_i) = \mathbf{false}\}$ is an anomaly for $A(\Psi)$ in $H(\Psi)$. Indeed, for each truth value assignment $\mathcal{T}^Y$ to the set of variables Y, the context $S^{\mathcal{T}^Y} = (X \setminus W) \cup W^{\neg} \cup \{sat\} \cup Lit(\mathcal{T}^Y)$ is an answer set of the program $th(A(\Phi), H(\Phi)) \setminus W$, and $\neg x_0 \in S^{\mathcal{T}^Y}$. As there is no other answer set for $th(A(\Psi), H(\Psi)) \setminus W$, it can be concluded that $th(A(\Psi), H(\Psi)) \models_c \neg w_0$, and W is an anomaly for $H(\Psi)$ in $A(\Psi)$. $\Box$

Next, we turn to the investigation of the Repair-Existence problem.

Theorem 2 Repair-Existence *is*

(1) NP*-complete, for negation-by-default-free ELPs,*
(2) Σ_2^{P}*-complete, for general ELPs under the brave semantics, and*
(3) NP*-complete, for general ELPs under the cautious semantics.*

Proof

(1) (Membership) The problem can be solved by a polynomial time nondeterministic Turing machine that guesses a succession of states H', and then checks that H' is an evolution and that H' is a repair. These checks (for conditions in Definition 1 and Definition 3) are feasible in polynomial time. In particular, since the unique answer set of a negation-by-default-free ELP can be computed in polynomial time, condition (4) in Definition 3 can be checked in polynomial time.

(Hardness) Given the formula Φ, consider the agent $A(\Phi)$ with the set of observables $\mathcal{O}(\Phi) = \{unsat, w, x_1, \ldots, x_n\}$, the sensors $s_1(\Phi)$ and $s_2(\Phi)$, the knowledge base $\mathcal{K}(\Phi)$:

$$\begin{aligned} r_0 &: \neg unsat. \\ r_{1,j} &: unsat \leftarrow \neg t_{j,1}, \neg t_{j,2}, \neg t_{j,3}.\ (1 \leq j \leq m) \end{aligned}$$

and the domain description:

$s_1(\Phi)$ **determines** w
$s_2(\Phi)$ **determines** x_i $\quad (1 \leq i \leq n)$
$s_2(\Phi)$ **determines** $unsat$
executable $s_2(\Phi)$ **if** w

Consider the evolution:

$$H(\Phi) = \langle \emptyset, \emptyset \rangle \rightarrow_{s_1(\Phi)} \langle \emptyset, \{w\} \rangle \rightarrow_{s_2(\Phi)} \langle \emptyset, \{unsat, w, x_1, \ldots, x_n\} \rangle$$

and the anomaly $W = \{w\}$. The set $W = \{w\}$ is an anomaly since, being both *unsat* and $\neg unsat$ in $th(A(\Phi), H(\Phi))$, the unique answer set of the negation-by-default-free program $th(A(\Phi), H(\Phi)) \setminus \{w\}$ is $S^H = \mathcal{O}(\Phi) \cup \mathcal{O}(\Phi)^{\neg}$, and $\neg w \in S^H$.
Now we prove that there exists a repair $H'(\Phi)$ for $H(\Phi)$ w.r.t. W iff Φ is satisfiable.

($\Rightarrow$) Assume that there exists a repair $H'(\Phi)$ for $H(\Phi)$ w.r.t. $W = \{w\}$. First of all, it must be noticed that the value sensed by sensor $s_1(\Phi)$ in the evolution $H'(\Phi)$ is $\{w\}$, for otherwise the action $s_2(\Phi)$ is not executable in $state(H'[1])$.
Then, in order to be the case that $th(A(\Phi), H'(\Phi)) \setminus \{w\} \not\models \neg w$, the program $th(A(\Phi), H'(\Phi)) \setminus \{w\}$ must be coherent and, by rule r_0, the value for the observable *unsat* in the evolution $H'(\Phi)$ must be $\neg unsat$. Since *unsat* can now be derived only by rules $r_{1,j}$, it is the case that the value for the observables $x_1, \ldots, x_n$ in the evolution $H'(\Phi)$ represents a truth value assignment to the variables $x_1, \ldots, x_n$ that makes the formula Φ true.

($\Leftarrow$) Assume that Φ is satisfiable, and let $\mathcal{T}^X$ be a truth value assignment to the variables $x_1, \ldots, x_n$ that makes Φ true. Then

$$H'(\Phi) = \langle \emptyset, \emptyset \rangle \rightarrow_{s_1(\Phi)} \langle \emptyset, \{w\} \rangle \rightarrow_{s_2(\Phi)} \langle \emptyset, \{\neg unsat, w\} \cup Lit(\mathcal{T}^X) \rangle$$

is a repair for $H(\Phi)$ w.r.t. W. Indeed, since $\mathcal{T}^X$ makes Φ true, the body of rules $r_{1,j}$ is false. Consequently, the unique answer set of $th(A(\Phi), H'(\Phi)) \setminus \{w\}$ is $\{\neg unsat, w\} \cup Lit(\mathcal{T}^X)$, and $th(A(\Phi), H'(\Phi)) \setminus \{w\} \not\models \neg w$.

(2) (Membership) The problem can be solved by a polynomial time nondeterministic Turing machine with an NP oracle. The machine guesses a succession of states H', and then checks that H' is an evolution and that H' is a repair. All the checks for conditions in Definition 1 and Definition 3 are feasible in polynomial time, except for condition (4) in Definition 3. Indeed, to verify that $th(A, H') \setminus W \not\models_b \neg w$, $\forall w \in W \cap \mathcal{O}(state(H'))$, we may check in non-deterministic polynomial time the complementary condition, i.e., the existence of a letter w and an answer set S for $th(A, H') \setminus W$ such that $\neg w \in S$.

(Hardness) Given the formula Ψ, consider the agent $A(\Psi)$ with the set of observables $\mathcal{O}(\Psi) = \{unsat, w, x_1, \ldots, x_n\}$, the sensors $s_1(\Psi)$ and $s_2(\Psi)$, the knowledge base $\mathcal{K}(\Psi)$:

$$\begin{array}{rll}
r_0 : & \neg unsat. & \\
r_1 : & unsat \leftarrow \neg d_1, \ldots, \neg d_l. & \\
r_{2,k} : & \neg d_k \leftarrow \neg t_{k,1}. & (1 \leq k \leq l) \\
r_{3,k} : & \neg d_k \leftarrow \neg t_{k,2}. & (1 \leq k \leq l) \\
r_{4,k} : & \neg d_k \leftarrow \neg t_{k,3}. & (1 \leq k \leq l) \\
r_{5,j} : & y_j \leftarrow not\ \neg y_j. & (1 \leq j \leq m) \\
r_{6,j} : & \neg y_j \leftarrow not\ y_j. & (1 \leq j \leq m)
\end{array}$$

and the domain description:

$s_1(\Psi)$ **determines** w
$s_2(\Psi)$ **determines** $x_i \quad (1 \le i \le n)$
$s_2(\Psi)$ **determines** $unsat$
executable $s_2(\Phi)$ **if** w

Consider the evolution:

$$\langle \emptyset, \emptyset \rangle \rightarrow_{s_1(\Psi)} \langle \emptyset, \{w\} \rangle \rightarrow_{s_2(\Psi)} \langle \emptyset, \{unsat, w, x_1, \ldots, x_n\} \rangle$$

and the anomaly $W = \{w\}$. The set $W = \{w\}$ is an anomaly since, being both $unsat$ and $\neg unsat$ in $th(A(\Psi), H(\Psi))$, then $S^H = \mathcal{O}(\Psi) \cup \mathcal{O}(\Psi)^{\neg} \cup Y \cup Y^{\neg} \cup \{d_k, \neg d_k \mid 1 \le k \le l\}$ is an answer set of the program $th(A(\Psi), H(\Psi)) \setminus \{w\}$ such that $\neg w \in S^H$, and, consequently, $th(A(\Psi), H(\Psi)) \setminus \{w\} \models_b \neg w$.
Now we prove that there exists a repair $H'(\Psi)$ for $H(\Psi)$ w.r.t. W iff Ψ is satisfiable.

($\Rightarrow$) Assume that there exists a repair $H'(\Psi)$ for $H(\Psi)$ w.r.t. W. First of all, it must be noticed that the value sensed by sensor $s_1(\Phi)$ in the evolution $H'(\Phi)$ is $\{w\}$, for otherwise the action $s_2(\Phi)$ is not executable in $state(H'[1])$.
Then, in order to be the case that $th(A(\Psi), H'(\Psi)) \setminus \{w\} \not\models_b \neg w$, then program $th(A(\Psi), H'(\Psi)) \setminus \{w\}$ must have at least a coherent answer set and, hence, the value for the observable $unsat$ sensed by the sensor $s_2(\Psi)$ in the evolution $H'(\Psi)$ must be $\neg unsat$.
Let $\mathcal{T}^X$ be the truth value assignment associated with the repair $H'(\Psi)$, that is the truth assignment $\mathcal{T}^X = \mathcal{O}(state(H'(\Psi))) \cap (X \cup X^{\neg})$.
Now, for the sake of contradiction, assume that there exists a truth value assignments $\mathcal{T}^Y$ to the set of variables Y such that $\mathcal{T}^X \cup \mathcal{T}^Y$ makes $f(X, Y)$ false. Then, by rules $r_1, r_{2,k}, r_{3,k}, r_{4,k}$, it follows that S^H is an answer set of $th(A(\Psi), H'(\Psi)) \setminus \{w\}$, and, consequently, that $th(A(\Psi), H'(\Psi)) \models_b \neg w$. Thus, it is the case that $\mathcal{T}^X$ is a truth value assignment to the set of variables X that makes the formula Ψ true.

($\Leftarrow$) Assume that Ψ is satisfiable, and let $\mathcal{T}^X$ be a truth value assignment to the set of variables X that makes Ψ true. Then the evolution

$$H'(\Psi) = \langle \emptyset, \emptyset \rangle \rightarrow_{t_1(\Psi)} \langle \emptyset, \{w\} \rangle \rightarrow_{t_2(\Psi)} \langle \emptyset, \{\neg unsat, w\} \cup Lit(\mathcal{T}^X) \rangle$$

is a repair for $H(\Psi)$ w.r.t. W. Indeed, the program $th(A(\Psi), H'(\Psi)) \setminus \{w\}$ bravely entails $\neg w$ if and only if it has an incoherent answer set. Incoherence can be introduced only by rule r_1. Notice that, by rules $r_{5,j}, r_{6,j}$ for each truth value assignments $\mathcal{T}^Y$ to the set of variables Y, there exists only one answer set $S^{\mathcal{T}^Y}$ of $th(A(\Psi), H'(\Psi)) \setminus \{w\}$ such that $S^{\mathcal{T}^Y} \supset Lit(\mathcal{T}^Y)$. Since $\mathcal{T}^X$ makes Ψ true, it the case that the body $\neg d_1, \ldots, \neg d_l$ of rule r_1 is such that $\{\neg d_1, \ldots, \neg d_l\} \not\subseteq S^{\mathcal{T}^Y}$. As there are no other answer sets for $th(A(\Psi), H'(\Psi)) \setminus \{w\}$, it is the case that $th(A(\Phi), H'(\Phi)) \setminus \{w\} \not\models_b \neg w$.

(3) (Membership) The problem can be solved by a polynomial time nondeterministic Turing machine. Let W be the set $\{w_1, \ldots, w_n\}$. Then, the machine guesses a succession of states H' together with $S_1, \ldots, S_n$ answer sets of $th(A, H') \setminus W$. Then, it checks in polynomial time that all the conditions in Definition 1 and Definition 3 are satisfied. In particular, to check for condition (4) in Definition 3, it suffices to verify that either $\neg w_i \notin S_i$ or $w_i \notin \mathcal{O}(H')$.

(Hardness) It immediately follows from (1) above. □

We can finally conclude this section by proving complexity results for Anomaly&Repair-Check.

Theorem 3 Anomaly&Repair-Check *is*

(1) P*-complete, for negation-by-default-free ELPs,*
(2) D^{P}*-complete, for general ELPs under brave semantics, and*
(3) in Δ_2^{P}*, for general ELPs under cautious semantics.*

Proof

(1) (Membership) The problem can be solved by checking that, (i) $len(H') = len(H)$, (ii) $state_0(H') = state_0(H)$, (iii) for each $1 \leq i \leq len(H')$, $act_i(H') = act_i(H)$, (iv) there exists $w \in W$ such that $th(A, H) \setminus W \models \neg w$ (P check) and (v) for each $w \in W \cap \mathcal{O}(state(H'))$, $th(A, H') \setminus W \not\models \neg w$ (P check). Thus, the overall procedure can be accomplished in polynomial time.

(Hardness) Given a positive logic program P and a propositional letter q, consider the agent $A(P, q)$ with the set of observables $\mathcal{O} = \{p, q\}$ (where p is a novel letter not occurring in P), the sensor $s(P, q)$, the knowledge base $\mathcal{K}(P, q) = P \cup \{\neg p \leftarrow p\}$, and the domain description:

$$s(P,q)\ \textbf{determines}\ p$$
$$s(P,q)\ \textbf{determines}\ q$$

Also, consider the evolutions

$$H(P,q) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(P,q)} \langle \emptyset, \{p, \neg q\} \rangle$$
$$H'(P,q) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(P,q)} \langle \emptyset, \{\neg p, \neg q\} \rangle$$

and the anomaly $W = \{\neg q\}$. Given a negation-by-default-free ELP P and a literal q, deciding whether $P \not\models q$ is a P-complete problem. In order to complete the proof, now we prove that W is an anomaly for $A(P, q)$ in $H(P, q)$ and $H'(P, q)$ is a repair for $H(P, q)$ w.r.t. W iff $P \not\models q$.

($\Rightarrow$) Assume that $W = \{\neg q\}$ is an anomaly for $A(P, q)$ in $H(P, q)$ and $H'(P, q)$ is a repair for $H(P, q)$ w.r.t. W. Then it is the case that $th(A(P, q), H'(P, q)) \setminus \{\neg q\} \not\models q$. Let S' be the unique answer set of $th(A(P, q), H'(P, q)) \setminus \{\neg q\}$. The answer set S' must be coherent and such that $q \notin S'$. Since S' is coherent and since the value for the observable p in evolution $H'(P, q)$ is $\neg p$, then it follows that $S' \setminus \{\neg p\}$ is the unique answer set of P, and, consequently, that $P \not\models q$.

($\Leftarrow$) Assume that $P \not\models q$. Then the unique answer set S of P is coherent and such that $q \notin S$. Hence, the set $S' = S \cup \{\neg p\}$ is the unique answer set of $th(A(P,q), H'(P,q)) \setminus \{\neg q\}$ and this program does not entail q, so that $H'(P,q)$ is a repair for $H(P,q)$ w.r.t. $W = \{\neg q\}$.
As for $W = \{\neg q\}$, the fact that it is an anomaly for $A(P,q)$ in $H(P,q)$ is immediately verified by noticing that p is the value sensed in evolution $H(P,q)$ for the observable p and, hence, being $\neg p \leftarrow p$ a rule of $\mathcal{K}(P,q)$, the program $th(A(P,q), H(P,q)) \setminus \{\neg q\}$ is incoherent regardless of the structure of the logic program P.

(2) (Membership) The problem can be solved by a polynomial time deterministic Turing machine with an NP oracle that executes exactly two oracle calls.
First of all, the machine checks in polynomial time that (i) $len(H') = len(H)$, (ii) $state_0(H') = state_0(H)$, and (iii) for each $1 \leq i \leq len(H')$, $act_i(H') = act_i(H)$. Then, it performs the two oracle calls.
During the first oracle call, the machine guesses a literal $w \in W$ and an answer set S of $th(A,H) \setminus W$ such that $\neg w \in S$ (NP check), so that $th(A,H) \setminus W \models_b \neg w$.
During the second oracle call, the machine verifies that, for each $w \in W \cap \mathcal{O}(state(H'))$ and for each answer set S of $th(A,H') \setminus W$, $\neg w \notin S$ (co-NP check), so that $th(A,H') \setminus W \not\models_b \neg w$.

(Hardness) Given two consistent ELPs P_1 and P_2, having no propositional letters in common, and the letters q_1, occurring in P_1, and q_2, occurring in P_2, let Ω denote the problem

$$(P_1 \models_b q_1) \wedge (P_2 \not\models_b q_2).$$

Consider the agent $A(\Omega)$ with the set of observables $\mathcal{O}(\Omega) = \{p_1, q_1\}$ (where p_1 is a new letter not occurring either in P_1 or in P_2), the sensor $s(\Omega)$, the knowledge base $\mathcal{K}(\Omega)$:

$$\begin{array}{rl} r_0 : & q_1 \leftarrow q_2. \\ r_{1,r'} : & \mathbf{h}(r') \leftarrow \mathbf{b}(r'), p_1. \quad (r' \in P_1) \\ r_{2,r''} : & \mathbf{h}(r'') \leftarrow \mathbf{b}(r''), \neg p_1. \ (r'' \in P_2) \end{array}$$

and the domain description:

$$s(\Omega) \textbf{ determines } o \qquad \forall o \in \mathcal{O}(\Omega)$$

Consider also the evolutions

$$\begin{array}{rl} H(\Omega) = & \langle \emptyset, \emptyset \rangle \rightarrow_{s(\Omega)} \langle \emptyset, \{p_1, \neg q_1\} \rangle \\ H'(\Omega) = & \langle \emptyset, \emptyset \rangle \rightarrow_{s(\Omega)} \langle \emptyset, \{\neg p_1, \neg q_1\} \rangle \end{array}$$

and the anomaly $W = \{\neg q_1\}$.
The problem Ω is the conjunction of two independent problems: an NP-complete problem $(P_1 \models_b q_1)$ and a co-NP-complete problem $(P_2 \not\models_b q_2)$. Hence, problem Ω is complete for the class D^{P}.

Now we prove that W is an anomaly for $A(\Omega)$ in $H(\Omega)$ and $H'(\Omega)$ is a repair for $H(\Omega)$ w.r.t. W iff Ω is true. First of all, we need to prove the two following intermediate results.

Claim A: $th(A(\Omega), H(\Omega)) \setminus \{\neg q_1\} \models_b q_1$ if and only if $P_1 \models_b q_1$.

Proof

($\Rightarrow$) Recall that the value for the literal p_1 sensed by sensor $s(\Omega)$ in evolution $H(\Omega)$ is p_1. Let S be an answer set of $th(A(\Omega), H(\Omega)) \setminus \{\neg q_1\}$. Then, either (a) there does not exist $r'' \in P_2$ such that the body $\{\mathbf{b}(r''), \neg p_1\}$ of rule $r_{2,r''}$ is contained in S, or (b) both p_1 and $\neg p_1$ are in S and S is incoherent.

Assume that $th(A(\Omega), H(\Omega)) \setminus \{\neg q_1\} \models_b q_1$. Then there exists an answer set S of $th(A(\Omega), H(\Omega)) \setminus \{\neg q_1\}$ such that $q_1 \in S$. Consider case (a): recall that the letter q_2 does not occur in P_1; then the literal q_1 can be entailed only by using rules $r_{1,k}$, hence $P_1 \models_b q_1$. As for case (b): $\neg p_1$ is a new letter not occurring either in P_1 or in P_2, hence the literal $\neg p_1$ does not appear in the head of any rule of $th(A(\Omega), H(\Omega)) \setminus \{\neg q_1\}$; then, the only possibility is that P_1 has an incoherent answer set and, thus, that $P_1 \models q_1$.

($\Leftarrow$) Assume that $P_1 \models_b q_1$. Then there exists an answer set S_1 of P_1 such that $q_1 \in S_1$. If S_1 is coherent, then $S = S_1 \cup \{p_1\}$ is an answer set of $th(A(\Omega), H(\Omega)) \setminus \{\neg q_1\}$ such that $q_1 \in S$. Indeed, recall that p_1 ($\neg p_1$, resp.) is (is not, resp.) a fact of the program $th(A(\Omega), H(\Omega)) \setminus \{\neg q_1\}$, and that q_2 is a letter not occurring in P_1.

On the contrary, if S_1 is incoherent, then, regardless of the program P_2 having at least an answer set (i.e., that P_2 is consistent), the program $th(A(\Omega), H(\Omega)) \setminus \{\neg q_1\}$ must have an incoherent answer set S, hence such that $q_1 \in S$, and $th(A(\Omega), H(\Omega)) \setminus \{\neg q_1\} \models_b q_1$. □

Claim B: $th(A(\Omega), H'(\Omega)) \setminus \{\neg q_1\} \not\models_b q_1$ if and only if $P_2 \not\models_b q_2$.

Proof

($\Rightarrow$) Assume that $th(A(\Omega), H'(\Omega)) \setminus \{\neg q_1\} \not\models_b q_1$. Then the program $P'' = th(A(\Omega), H'(\Omega)) \setminus \{\neg q_1\}$ cannot have an incoherent answer set, and either (a) P'' is inconsistent or (b) P'' has at least one answer set and each answer set S of P'' is such that $q_1 \notin S$.

Consider case (a): since $\neg p_1$ (p_1, resp.) is (is not, resp.) a fact of $th(A(\Omega), H'(\Omega)) \setminus \{\neg q_1\}$, p_1 is a new letter not occurring either in P_1 or P_2, while q_1 does not occur in P_2, it is the case that P_2 is inconsistent, and, hence, that $P_2 \not\models_b q_2$.

As for case (b): let S be a generic answer set of $th(A(\Omega), H'(\Omega)) \setminus \{\neg q_1\}$. It is known that $\neg p_1 \in S$ and that $q_1 \notin S$. Since S is coherent, it follows that $p_1 \notin S$ and it can be concluded that $S \setminus \{\neg p_1\}$ is also an answer set of the program P_2. As P_2 has no other answers sets, $P_2 \not\models_b q_2$.

($\Leftarrow$) Assume that $P_2 \not\models_b q_2$. Then P_2 cannot have an incoherent answer set, and either (a) P_2 is inconsistent, or (b) P_2 has at least one answer set and each answer S_2 of P_2 is such that $q_2 \notin S_2$. Consider case (a): it follows that the program P'' is inconsistent, and, hence, that

$P'' \not\models_b q_1$. As for case (b): let S_2 be a generic answer set of P_2. Recall that $\neg p_1$ (p_1, resp.) is (is not, resp.) a fact of $th(A(\Omega), H'(\Omega)) \setminus \{\neg q_1\}$. Since S_2 is coherent, it can be concluded that $S = S_2 \cup \{q_1\}$ is also an answer set of P''. As P'' has no other answer sets, then $P'' \not\models_b q_1$. □

Now we can resume the main proof. The two following facts are now known:

1. "W is an anomaly for $A(\Omega)$ in $H(\Omega)$" $\Longleftrightarrow$ (by Definition 2) "$th(A(\Omega), H(\Omega)) \setminus \{\neg q_1\} \models_b q_1$" $\Longleftrightarrow$ (by Claim A) "$P_1 \models_b q_1$", and
2. "$H'(\Omega)$ is a repair for $H(\Omega)$ w.r.t. W" $\Longleftrightarrow$ (by Definition 2) "$th(A(\Omega), H'(\Omega)) \setminus \{\neg q_1\} \not\models_b q_1$" $\Longleftrightarrow$ (by Claim B) "$P_2 \not\models_b q_2$".

Hence, by combining the two above results, it is proved that W is an anomaly for $A(\Omega)$ in $H(\Omega)$ and $H'(\Omega)$ is a repair for $H(\Omega)$ w.r.t. W if and only if Ω is true.

(3) The problem can be solved by a polynomial time deterministic Turing machine with an NP oracle.
First of all, the machine checks in polynomial time that, (i) $len(H') = len(H)$, (ii) $state_0(H') = state_0(H)$, and (iii) for each $1 \leq i \leq len(H')$, $act_i(H') = act_i(H)$.
Let $W = \{w_1, \ldots, w_n\}$. Next, for each $w_i \in W$, the machine executes an oracle call to verify whether $th(A, H) \setminus W \models_c \neg w_i$ (co-NP check). If the oracle always returns the answer "no", then W is not an anomaly in H for A and the machine stops its execution and returns the answer "no". Otherwise, the machine continues its work.
In order to check whether for each $w \in \mathcal{O}(state(H'))$, $th(A, H') \setminus W \not\models_c \neg w$, an additional oracle call is performed. Let $\{w_{i_1}, \ldots, w_{i_m}\} = W \cap \mathcal{O}(state(H'))$. During this oracle call, the machine guesses m answer sets $S_1, \ldots, S_m$ of $th(A, H') \setminus W$ such that, for $j = 1, \ldots, m$, $\neg w_{i_j} \notin S_j$ (NP check).
In order to conclude the proof, we note that the machine executes exactly $|W| + 1$ oracle calls, that is a number of calls which is polynomial in the size of the input. Since the machine is deterministic and overall executes a polynomially bounded number of steps, then the problem can be solved in $\mathrm{FP}^{\mathrm{NP}} = \mathrm{F}\Delta_2^{\mathrm{P}}$. □

5 Not-free Programs: Full Repairs and Minimality Conditions

A crucial aspect in the definition of repair considered in the previous sections lies in the identification of the underlying anomaly W that occurred in a given evolution. As a matter of fact, however, when fixing an evolution, it is often convenient to eliminate *all* the possible anomalies rather than some pre-identified one. This observation leads to the following definition, whose study will be the main subject of this section.

Definition 4 (Full Repair) An evolution H' for A is a *full repair* for H if:

(1) $len(H) = len(H')$,
(2) $state_0(H) = state_0(H')$,

(3) $act_i(H) = act_i(H')$, for each $1 \leq i \leq len(H)$, and
(4) $\forall i \in \{1, \ldots, len(H)\}$, there is no anomaly in $H[i]$. □

Thus, a full repair corrects all the anomalies not only in the current state, but also in any other intermediate state of the evolution. For instance, in our running example, the reader might check that the repair informally discussed in Example 5 is also a full repair.

Minimal/Minimum Full Repairs. As a further remark, we note that even though full repairs remove all the possible anomalies in a given evolution, they can in principle also update all the possible observations, thereby resulting in possibly rather unnatural evolutions. To avoid this, which might be undesirable in several circumstances, a viable way consists in constraining full repairs to satisfy some additional requirements. In particular, singling out full repairs is next formalized, where a minimal/minimum number of observations is required to be updated w.r.t. the original evolution.

Definition 5 Let H' be a full repair for H in A. Then, H' is said to be:

– *minimal*, if there does not exist a full repair H'' for H in A such that $\Delta(H'', H) \subset \Delta(H', H)$;
– *minimum*, if there does not exist a full repair H'' for H in A such that $|\Delta(H'', H)| < |\Delta(H', H)|$;

where $\Delta(H', H) = (\mathcal{O}(state(H')) \setminus \mathcal{O}(state(H))) \cup (\mathcal{O}(state(H)) \setminus \mathcal{O}(state(H')))$. The size $|\Delta(H', H)|$ of the set $\Delta(H', H)$ is also called the *size of the repair H' for H in A.* □

5.1 Anomalies with not-free Programs

Throughout this section, we shall investigate on the notion of full repair (possibly under the above additional minimality requirements) in the important case where knowledge bases are expressed as negation-by-default-free programs, for which the entailment relation is efficiently decidable.

Interestingly, for this class of programs, a very natural characterization for the occurrence of anomalies can be given in terms of the classical notion of (in)coherence. In fact, the careful reader may have already argued that an anomaly W may emerge for an agent A in an evolution H, even though the associated theory $th(A, H)$ is coherent. And, similarly, an incoherent theory may well not admit any anomaly in it. The reason underlying this behavior is that our concept of anomaly is very focused on the identification of noisy observations; while the causes underlying the emergence of an incoherence in $th(A, H)$ might be of very different kinds and related, for instance, to the arising of conflicts over beliefs. Thus, in general it is not possible to establish any relation between these notions.

However, this is not the case for not-free-programs. Indeed, let $th(A, H) = \mathcal{K} \cup \mathcal{B}(state(H)) \cup \mathcal{O}(state(H))$ be the not-free theory associated with an agent A and evolution H. If there is an anomaly for A in H, i.e., if there exists $w \in W$ such that $th(A, H) \setminus W \models \neg w$ then, because of the monotonicity of

$th(A,H)$, we have that $th(A,H) \setminus \{w\} \models \neg w$ and, hence, that $th(A,H)$ is incoherent.

For the converse, things are trivial since the incoherence of $th(A,H)$ entails that any observable that does not play a role in the knowledge base $\mathcal{K}$ is an anomaly—w.l.o.g., one can always assume that one such observable exists. Thus, the following result can be established.

Proposition 1 *Let H be an evolution for the agent A with knowledge base $\mathcal{K}$ expressed as a negation-by-default-free ELP. Then, there is an anomaly for A in H if and only if $th(A,H)$ is incoherent.*

And, an immediate corollary is as follows.

Corollary 1 *Let H be an evolution for the agent A with knowledge base $\mathcal{K}$ expressed as a negation-by-default-free ELP. Then, an evolution H' satisfying the first three conditions in Definition 4 is a full repair for H in A if and only if $th(A,H'[i])$ is coherent, for each $i \in \{1,...,len(H)\}$.*

5.2 Relevant Reasoning Tasks

In this section, we continue the complexity analysis initiated in Section 4 by focusing on the notion of full repairs and on the case of not-free programs. In particular, we investigate the intrinsic difficulty of the following decision problems:

FULL-REPAIR-CHECKING: Given an agent A and two evolutions H and H' for A, is H' a full repair for H in A ?
FULL-REPAIR-EXISTENCE: Given an agent A and an evolution H for A, does there exist a full repair H' for H in A ?
FULL-REPAIR-MEMBERSHIP: Given an agent A, an evolution H and a literal o, is o contained in $\mathcal{O}(state(H'))$ for some full repair H' for H in A ?

In fact, we shall consider the problems above also in the cases where full repairs are constrained to be minimal or minimum. In addition, we shall also consider the following computation problem which is very peculiar for minimum full repairs:

MINIMAL-FULL-REPAIR-SIZE-COMPUTATION: Given an agent A and an evolution H, compute the size of the minimal full repair H' for H in A.

Complexity results concerning the problems defined above are depicted in Figure 4. For the implementation issues discussed in Section 6, it is relevant to note that all the problems are confined within the first two levels of the polynomial hierarchy.

Proof of Complexity Results: Full Repairs. We start the discussion of our results with the FULL-REPAIR-CHECKING problem.

Theorem 4 FULL-REPAIR-CHECKING *is* P*-complete.*

	not-free
Full-Repair-Checking	P-c
Full-Repair-Existence	NP-c
Full-Repair-Membership	NP-c
Minimal-Full-Repair-Checking	co-NP-c
Minimal-Full-Repair-Existence	NP-c
Minimal-Full-Repair-Membership	Σ_2^{P}-c
Minimum-Full-Repair-Checking	co-NP-c
Minimum-Full-Repair-Existence	NP-c
Minimum-Full-Repair-Membership	$\Delta_2^{\mathrm{P}}[\mathcal{O}(\log \mathrm{n})$-c
Minimum-Full-Repair-Size-Computation	$\mathrm{F}\Delta_2^{\mathrm{P}}[\mathcal{O}(\log \mathrm{n})]$-c

Fig. 4 Complexity of full repairs for not-free programs.

Proof (Membership) Given an agent A and two evolutions H and H' for A, the problem can be solved by a polynomial time Turing machine as follows. The machine checks in polynomial time that, (1) $len(H') = len(H)$, (2) $state_0(H') = state_0(H)$, (3) for each $1 \leq i \leq len(H')$, $act_i(H') = act_i(H)$, and (4) for each $1 \leq i \leq len(H')$, the evolution $H'[i]$ has no anomalies. Checking whether the agent A has an anomaly in the evolution $H'[i]$ corresponds to solving the Anomaly-Existence problem. Since the background knowledge of the agent A is a negation-by-default-free logic program, by Theorem 1, Point 1, the problem is solvable in polynomial time. The overall procedure can be thus accomplished in polynomial time.

(Hardness) Analogous to the hardness part of Theorem 1, Point 1. □

Turning from Full-Repair-Checking to Full-Repair-Existence, things appear intrinsically more complex as shown below.

Theorem 5 Full-Repair-Existence *is* NP-*complete.*

Proof (Membership) The problem can be solved by a polynomial time nondeterministic Turing machine that guesses an evolution H' for A and then checks in polynomial time that H' is a full repair for H in A by solving the Full-Repair-Checking problem.

(Hardness) Given the formula Φ, consider the agent $A(\Phi)$ with the set of observables $\mathcal{O}(\Phi) = \{w, x_1, \ldots, x_n\}$, the sensor $s(\Phi)$, the knowledge base $\mathcal{K}(\Phi)$:

$$r_0 : \neg unsat.$$
$$r_{1,j} : unsat \leftarrow \neg t_{j,1}, \neg t_{j,2}, \neg t_{j,3}. \ (1 \leq j \leq m)$$

and the domain description:

$$s(\Phi) \ \mathbf{determines} \ o \qquad \forall o \in \mathcal{O}(\Phi)$$

Then, consider the evolution:

$$H(\Phi) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(\Phi)} \langle \emptyset, \{w, x_1, \ldots, x_n\} \rangle.$$

We prove that there exists a full repair $H'(\Phi)$ for $H(\Phi)$ in $A(\Phi)$ iff Φ is satisfiable.

($\Rightarrow$) Assume there exists a full repair $H'(\Phi)$ for $H(\Phi)$ in $A(\Phi)$. Then, it is the case that there is no anomaly W for $A(\Phi)$ in $H'(\Phi)$. Hence, given that $th(A(\Phi), H'(\Phi))$ is a negation-by-default-theory, it must be the case that the unique answer set S' of this theory is coherent. Consequently, there does not exist a rule $r_{1,j}$ whose body $\{\neg t_{j,1}, \neg t_{j,2}, \neg t_{j,3}\}$ is contained in S'. Let X' be the set $\mathcal{O}(state(H'(\Phi))) \cap (X \cup X^{\neg})$ of the values associated with the observables in the set $X = \{x_1, ..., x_n\}$ in the evolution $H'(\Phi)$. Then, the answer set S' is the set $X' \cup \{\neg unsat\}$. Notice that the body of the rule $r_{1,j}$ $(1 \leq j \leq m)$ corresponds to the negation $\neg C_j$ of the j-th clause C_j composing the boolean formula $\Phi = C_1 \wedge \ldots \wedge C_m$. Hence, it is the case that the truth value assignment $\mathcal{T}_{X'}$ to the set of variables X associated with the answer set S' makes the formula Φ true. Therefore, Φ is satisfiable.

($\Leftarrow$) Assume that Φ is satisfiable. Then there exists a truth value assignment $\mathcal{T}^X$ to the set of variables X which makes the formula Φ true. Consider the repair

$$H'(\Phi) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(\Phi)} \langle \emptyset, \{w\} \cup Lit(\mathcal{T}^X) \rangle.$$

From the one-to-one correspondence between rules $r_{1,j}$ and formulas $\neg C_j$, it follows that none of the bodies $\{\neg t_{j,1}, \neg t_{j,2}, \neg t_{j,3}\}$ is contained in $Lit(\mathcal{T}^X)$. Then, the unique answer set S' of $th(A(\Phi), H'(\Phi))$ is $Lit(\mathcal{T}^X) \cup \{w, \neg unsat\}$, and S' is coherent. Thus, $th(A(\Phi), H'(\Phi))$ is coherent and, hence, there is no anomaly for $A(\Phi)$ in $H'(\Phi)$. □

We conclude the analysis of full repairs with the Full-Repair-Membership problem.

Theorem 6 Full-Repair-Membership *is* NP-*complete.*

Proof (Membership) The problem can be solved by a polynomial time nondeterministic Turing machine that guesses an evolution H' for A and then checks in polynomial time that H' is a full repair for H in A by solving the Full-Repair-Checking problem and that the observable o occurs in $\mathcal{O}(state(H'))$.

(Hardness) Consider again the proof of Theorem 5, where the knowledge base $\mathcal{K}(\Phi)$ is modified to include the rule r_{new} : $unsat \leftarrow \neg h.$, where h is a novel observable. Then, it can be noticed that h has to occur in any full repair for $H(\Phi)$ in $A(\Phi)$. Thus, checking whether h occurs in *some* full repair is equivalent to checking whether there is any full repair at all. □

Proof of Complexity Results: Minimal Full Repairs. Next, the problems above are reconsidered while adding a minimality requirement.

Theorem 7 Minimal-Full-Repair-Checking *is* co-NP-*complete.*

Proof (Membership) Given an agent A and two evolutions H and H' for A, a polynomial time nondeterministic Turing machine can determine if H' is not a minimal full repair for H in A as follows.

The machine checks in polynomial time (by solving the Full-Repair-Checking problem, see Theorem 4, Point 1) that H' is indeed a full repair for H in A.
In the negative case, the machine terminates its work replying "yes", otherwise it guesses a full repair H'' for H in A such that $\Delta(H'', H) \subset \Delta(H', H)$ and then checks in polynomial time that H'' is a full repair for H in A (see Theorem 4, Point 1). Since the negation of the problem Minimal-Full-Repair-Checking is in NP, then this problem is in co-NP.

(Hardness) Consider the boolean formula $\Phi = C_1 \wedge \ldots \wedge C_m$ in conjunctive normal form, with $C_j = t_{j,1} \vee t_{j,2} \vee t_{j,3}$, where each $t_{j,k}$ is a literal on the set of boolean variables $X = x_1, \ldots, x_n$. W.l.o.g., assume that the assignment where all the variables are evaluated false does not satisfy Φ.

Given the formula Φ, consider the agent $A(\Phi)$ with the set of observables $\mathcal{O}(\Phi) = \{unsat, x_1, \ldots, x_n\}$, the sensor $s(\Phi)$, the knowledge base $\mathcal{K}(\Phi)$:

$$\begin{array}{ll} r_0 : \neg unsat. & \\ r_{1,j} : unsat \leftarrow \neg t_{j,1}, \neg t_{j,2}, \neg t_{j,3}, ok. & (1 \leq j \leq m) \\ r_{2,j} : ok \leftarrow x_i. & (1 \leq j \leq m) \end{array}$$

and the domain description:

$$s(\Phi) \textbf{ determines } o \quad \forall o \in \mathcal{O}(\Phi)$$

Consider the evolution

$$H(\Phi) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(\Phi)} \langle \emptyset, \{unsat, x_1, \ldots, x_n\} \rangle,$$

and the evolution

$$H'(\Phi) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(\Phi)} \langle \emptyset, \{\neg unsat, \neg x_1, \ldots, \neg x_n\} \rangle.$$

We first claim that $H'(\Phi)$ is a full repair, i.e., that the theory $th(A(\Phi), H'(\Phi))$ is coherent. Indeed, let S be the unique answer set of $th(A(\Phi), H'(\Phi))$. Because of the rule $r_{2,j}$, it is the case that $ok \notin S$. Thus, $unsat$ does not belong to S, which is the only possible source of incoherence in the theory $\mathcal{K}(\Phi)$ above.

Next, we show that $H'(\Phi)$ is, in fact, a *minimal* full repair for $H(\Phi)$ in $A(\Phi)$ iff Φ is unsatisfiable. But, beforehand, we need to observe that each possible full repair for $H(\Phi)$, which is different from $H'(\Phi)$, must be of the form:

$$H''(\Phi) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(\Phi)} \langle \emptyset, \{\neg unsat\} \cup Lit(\mathcal{T}^X) \rangle$$

where $\mathcal{T}^X$ is a truth value assignment to the set of variables X that satisfies Φ. Indeed, whenever $\mathcal{T}^X$ does not satisfy Φ and $X \neq \{\neg x_1, ..., \neg x_n\}$, we have that $unsat$ is entailed.

($\Rightarrow$) Assume that $H'(\Phi)$ is a minimal full repair and, for the sake of contradiction, that Φ is satisfiable. Let $\mathcal{T}^X$ be a satisfying assignment for Φ. Then, consider the full repair $H''(\Phi) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(\Phi)} \langle \emptyset, \{\neg unsat\} \cup Lit(\mathcal{T}^X) \rangle$. Given that $Lit(\mathcal{T}^X) \neq \{\neg x_1, ..., \neg x_n\}$, it holds: $\Delta(H''(\Phi), H(\Phi)) \subset \Delta(H'(\Phi), H(\Phi))$. Contradiction, since $H'(\Phi)$ is minimal.

($\Leftarrow$) Assume that Φ is not satisfiable and, for the sake of contradiction, that $H'(\Phi)$ is not minimal. Then, there exists a full repair $H''(\Phi)$ such that $\Delta(H''(\Phi), H(\Phi)) \subset \Delta(H'(\Phi), H(\Phi))$. We recall that $H''(\Phi)$ must be of the form: $H''(\Phi) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(\Phi)} \langle \emptyset, \{\neg unsat\} \cup Lit(\mathcal{T}^X) \rangle$, where $\mathcal{T}^X$ is a truth value assignment to the set of variables X that satisfies Φ, hence a contradiction is implied. □

Clearly enough, the existence of a full repair immediately entails the existence of a minimal full repair. Thus, the following is immediate from Theorem 5.

Theorem 8 MINIMAL-FULL-REPAIR-EXISTENCE *is* NP*-complete.*

Yet, working with minimal full repairs is intrinsically more complex, as implied from the hardness result for the second level of the polynomial hierarchy discussed below.

Theorem 9 MINIMAL-FULL-REPAIR-MEMBERSHIP *is* Σ_2^{P}*-complete.*

Proof (Membership) Given an agent A, an evolution H and a literal o, we can check whether o is contained in $\mathcal{O}(state(H'))$ for some minimal full repair H' for H in A as follows. A polynomial-time non-deterministic Turing machine first guesses an evolution H' for H in A, and then checks that: (i) H' is indeed a minimal full repair, and (ii) $o \in \mathcal{O}(state(H'))$. While the second condition can be checked in polynomial time, the first one requires the exploitation of a co-NP oracle (cf. Theorem 7).

(Hardness) Consider the quantified boolean formula $\Psi = \exists X \forall Y f(X,Y)$, where $X = x_1, \ldots, x_n$, $Y = y_0, y_1, \ldots, y_m$, $f(X,Y) = D_1 \vee \ldots \vee D_l$ is in conjunctive normal form, with $D_k = t_{k,1} \wedge t_{k,2} \wedge t_{k,3}$, and each $t_{k,1}, t_{k,2}, t_{k,3}$ is a literal on the set of boolean variables X and Y. W.l.o.g., assume that y_0 being false satisfies Ψ.

Based on Ψ, consider the agent $A(\Psi)$ with the set of observables $\mathcal{O}(\Psi) = \{w, x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n, ok, y_0, y_1, ..., y_m\}$, the sensor $s(\Psi)$, the knowledge base $\mathcal{K}(\Psi)$:

$$\begin{array}{rll}
r_0 : & \neg unsat. & \\
r_{1,i} : & unsat \leftarrow x_i, \bar{x}_i. & (1 \leq i \leq n) \\
r_{2,i} : & unsat \leftarrow \neg x_i, \neg \bar{x}_i. & (1 \leq i \leq n) \\
r_3 : & ok \leftarrow \neg y_0, \neg y_1, ..., \neg y_m. & \\
r_{4,j} : & \neg ok \leftarrow y_j. & (0 \leq j \leq m) \\
r_{5,k} : & true \leftarrow t_{k,1}, t_{k,2}, t_{k,3}. & (1 \leq k \leq l) \\
r_{6,j} : & \neg y_j \leftarrow true. & (0 \leq j \leq m)
\end{array}$$

and the domain description:

$$s(\Psi) \textbf{ determines } o \qquad \forall o \in \mathcal{O}(\Psi)$$

Consider the evolution:

$$H(\Psi) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(\Psi)} \langle \emptyset, \{w, x_1, \bar{x}_1, ..., x_n, \bar{x}_n, \neg ok, y_0, y_1, ..., y_m\} \rangle.$$

We now claim that *ok* occurs in some minimal full repair H' for $H(\Psi)$ if and only if Ψ is satisfiable.

($\Rightarrow$) Assume that ok occurs in a minimal full repair H' for $H(\Psi)$. Since $ok \in \mathcal{O}(state(H'))$ and since $H'[1]$ has to be coherent—otherwise would have an anomaly over w (cf. Proposition 1)—, it is the case that $Y^{\neg} \subset \mathcal{O}(state(H'))$ because of rules $r_{4,j}$.
Consider, now, the set $S^X = \mathcal{O}(state(H')) \cap (X \cup X^{\neg})$ that encodes an assignment $\mathcal{T}^X = \mathcal{T}_{S^X}$ for the existentially quantified variables in Ψ. Also, consider the set $\bar{S}^X = \{\neg\bar{x}_i \mid x_i \in S^X\} \cup \{\bar{x}_i \mid \neg x_i \in S^X\}$ and note that, due to rules $r_{1,i}$ and $r_{2,i}$, $\mathcal{O}(state(H')) \supset S^X \cup \bar{S}^X$.
We claim that $\mathcal{T}^X$ witnesses the validity of Ψ. Indeed, assume for the sake of contradiction, that there is an assignment $\mathcal{T}^Y$ for the universally quantified variables such that $\mathcal{T}^X$ and $\mathcal{T}^Y$ do not satisfy Ψ—note that the variable y_0 has to evaluate true in $\mathcal{T}^Y$. Then, we build the evolution

$$H'' = \langle\emptyset,\emptyset\rangle \rightarrow_{s(\Psi)} \langle\emptyset, \{w, \neg ok\} \cup S^X \cup \bar{S}^X \cup Lit(\mathcal{T}^Y)\rangle.$$

By construction, $th(A(\Psi), H'')$ is coherent (in particular, note that $true$ is not entailed from $r_{5,k}$) and, hence, H'' is a full repair.
To conclude this part of the proof it is, then, sufficient to observe that: $\Delta(H'', H) \subset \Delta(H', H)$, a contradiction with the fact that H' is a minimal full repair. To see why this is the case, note that $\mathcal{O}(state(H'))$ and $\mathcal{O}(state(H''))$ coincide over w and $S^X \cup \bar{S}^X$, while there is certainly an advantage in taking $Lit(\mathcal{T}^Y)$ and $\neg ok$ (in H'') w.r.t. $Y^{\neg}$ and ok (in H').

($\Leftarrow$) Assume that $\mathcal{T}^*$ witnesses the satisfiability of Ψ. Consider the evolution H' of the form $\langle\emptyset,\emptyset\rangle \rightarrow_{s(\Psi)} \langle\emptyset, \mathcal{O}(state(H'))\rangle$, where:

$$\begin{aligned}\mathcal{O}(state(H')) = \{&w, ok, \neg y_0, \neg y_1, ..., \neg y_m\} \cup \\ \{&x_i, \neg\bar{x}_i \mid x_i \text{ is true in } \mathcal{T}^*\} \cup \\ \{&\neg x_i, \bar{x}_i \mid x_i \text{ is false in } \mathcal{T}^*\}.\end{aligned}$$

Note that H' is a repair for $H(\Psi)$. Then, we claim that H' is a minimal full repair. To this end, assume for the sake of contradiction, that there is a full repair H'' such that $\Delta(H'', H) \subset \Delta(H', H)$.
Let $S'_X = \mathcal{O}(state(H')) \cap (X \cup X^{\neg})$ and $S''_X = \mathcal{O}(state(H'')) \cap (X \cup X^{\neg})$. First, we note that S'_X must be equal to S''_X. Indeed, due to rules $r_{1,i}$ and $r_{2,i}$, any full repair H^R for $H(\Psi)$ must be such that $x_i \in \mathcal{O}(state(H^R))$ ($\neg x_i \in \mathcal{O}(state(H^R))$, resp.) if and only if $\neg\overline{x}_i \in \mathcal{O}(state(H^R))$ ($\overline{x}_i \in \mathcal{O}(state(H^R))$, resp.). Thus, in order to have $S'_X \neq S''_X$ it must be the case that either

(a) there exists $x_i \in X$ such that $x_i \in S'_X$ (and, consequently, $\neg\overline{x}_i \in S'_X$) and $\neg x_i \in S''_X$ (and, consequently, $\overline{x}_i \in S''_X$), or
(b) there exists $x_i \in X$ such that $\neg x_i \in S'_X$ (and, consequently, $\overline{x}_i \in S'_X$) and $x_i \in S''_X$ (and, consequently, $\neg\overline{x}_i \in S''_X$).

But, in both cases we have that:

(a) $\Delta(H', H) \supset (\{x_i, \neg\overline{x}_i\} \setminus \{x_i, \overline{x}_i\}) \cup (\{x_i, \overline{x}_i\} \setminus \{x_i, \neg\overline{x}_i\}) = \{\overline{x}_i, \neg\overline{x}_i\}$ and neither x_i nor $\neg x_i$ belong to $\Delta(H', H)$, while $\Delta(H'', H) \supset (\{\neg x_i, \overline{x}_i\} \setminus \{x_i, \overline{x}_i\}) \cup (\{x_i, \overline{x}_i\} \setminus \{\neg x_i, \overline{x}_i\}) = \{\neg x_i, x_i\}$, and, hence, $\Delta(H'', H) \not\subseteq \Delta(H', H)$, or
(b) $\Delta(H', H) \supset (\{\neg x_i, \overline{x}_i\} \setminus \{x_i, \overline{x}_i\}) \cup (\{x_i, \overline{x}_i\} \setminus \{\neg x_i, \overline{x}_i\}) = \{\neg x_i, x_i\}$ and neither $\overline{x}_i$ nor $\neg\overline{x}_i$ belong to $\Delta(H', H)$, while $\Delta(H'', H) \supset$

$(\{x_i, \neg\overline{x}_i\} \setminus \{x_i, \overline{x}_i\}) \cup (\{x_i, \overline{x}_i\} \setminus \{x_i, \neg\overline{x}_i\}) = \{\neg\overline{x}_i, \overline{x}_i\}$, and, hence, $\Delta(H'', H) \not\subseteq \Delta(H', H)$.

We can eventually conclude that S'_X is equal to S''_X.

In addition, since $H'' \neq H'$, it must be the case that there is a variable y_j such that $y_j \in \mathcal{O}(state(H''))$, and consequently, because of rules $r_{4,j}$, that $\neg ok$ is in $\mathcal{O}(state(H''))$. Consider, now, the assignment $\mathcal{T}_{S^Y}$, where $S^Y = \mathcal{O}(state(H'')) \cap (Y \cup Y^\neg)$. Since, $\mathcal{T}^*$ witnesses the validity of Ψ, we have that Ψ evaluates true over $\mathcal{T}_{S^X}$ (conducing with $\mathcal{T}^*$ when restricted over X) and $\mathcal{T}_{S^Y}$. Then, because of the rules $r_{5,k}$ and $r_{6,k}$, it must be the case that $Y^\neg \subset \mathcal{O}(state(H''))$. But, this is impossible because $y_j \in \mathcal{O}(state(H''))$. $\square$

Proof of Complexity Results: Minimum Full Repairs. We close this section by presenting complexity results regarding minimum full repairs. Interestingly, our first observation is that the complexity of the checking problem is not altered when swapping from minimal to minimum requirements.

Theorem 10 Minimum-Full-Repair-Checking *is* co-NP-*complete.*

Proof (Membership) Given an agent A and two evolutions H and H' for A, a polynomial time nondeterministic Turing machine can determine if H' is not a minimum full repair for H in A as follows.

The machine checks in polynomial time (by solving the Full-Repair-Checking problem, see Theorem 4, Point 1) that H' is indeed a full repair for H in A.

In the negative case, the machine terminates its work replying "yes", otherwise it guesses a full repair H'' for H in A such that $|\Delta(H'', H)| < |\Delta(H', H)|$ and then checks in polynomial time that H'' is a full repair for H in A (see Theorem 4, Point 1). Since the negation of the problem Minimum-Full-Repair-Checking is in NP, then this problem is in co-NP.

(Hardness) The reduction is identical to that described in the hardness part of Theorem 7. In particular in Theorem 7, an agent $A(\Phi)$ and two evolutions $H(\Phi)$ and $H'(\Phi)$ for $A(\Phi)$ are associated with the boolean formula Φ, such that $H'(\Phi)$ is a minimal full repair for $H(\Phi)$ in $A(\Phi)$ if and only if Φ is unsatisfiable. Since, by construction of the reduction, all the evolutions $H''(\Phi)$ that are candidates to be full repairs for $H(\Phi)$ in $A(\Phi)$ satisfy the property $\Delta(H''(\Phi), H(\Phi)) \subseteq \Delta(H'(\Phi), H(\Phi))$, it follows that $H'(\Phi)$ is a minimal full repair for $H(\Phi)$ in $A(\Phi)$ if and only if $H'(\Phi)$ is a minimum full repair for $H(\Phi)$ in $A(\Phi)$. $\square$.

As with the case of minimal full repairs, the existence of a full repair immediately entails the existence of a minimum full repair. Thus, the following is immediate from Theorem 5.

Theorem 11 Minimum-Full-Repair-Existence *is* NP-*complete.*

Next, the membership problem is analyzed, which turns out to be confined within the first level of the polynomial hierarchy. Beforehand, we assess the exact complexity of computing the size of the minimum full repair, which is useful to better understand our membership result.

Theorem 12 Minimum-Full-Repair-Size-Computation *is complete for* $\mathrm{F}\Delta_2^{\mathrm{P}}[\mathcal{O}(\log \mathrm{n})]$.

Proof (Membership) Let n^* be the minimum size associated with a full repair H' for H in A. The value n^* can be computed by a deterministic polynomial time transducer with an oracle in NP executing a binary search in the interval of integer numbers $[1, n]$, where n is the number $|\mathcal{O}(state(H))|$ of observables sensed in evolution H. At each step of the search, a threshold $\overline{n} \in [1, n]$ is given, and it is decided, by calling the NP oracle, whether there exists a full repair H' for H in A such that $|\Delta(H', H)| \leq \overline{n}$. The oracle guesses an evolution H' for A such that $|\Delta(H', H)| \leq \overline{n}$ and then checks in polynomial time that H' is a full repair for H in A by solving the Full-Repair-Checking problem. Since the overall procedure is feasible by executing at most $\mathcal{O}(\log n)$ oracle calls, the function can be computed in $\mathrm{F}\Delta_2^{\mathrm{P}}[\mathcal{O}(\log \mathrm{n})]$.

(Hardness) Recall that computing the size of the maximum clique in a graph is $\mathrm{F}\Delta_2^{\mathrm{P}}[\mathcal{O}(\log n)]$-complete [19]. Let $\mathcal{G} = (V, E)$ be an undirected graph, where $V = \{v_1, \ldots, v_n\}$ is the set of the nodes and E is the set of the edges $\{v_j, v_j\}$ $(v_i, v_j \in V)$ of the graph $\mathcal{G}$.

Given the graph $\mathcal{G}$, consider the agent $A(\mathcal{G})$ with the set of observables $\mathcal{O}(\mathcal{G}) = \{clique, v_1, \ldots, v_n\}$, the sensor $s(\mathcal{G})$, the knowledge base $\mathcal{K}(\mathcal{G})$:

$$\begin{aligned} r_0 &: clique. \\ r_{1,i,j} &: \neg ok \leftarrow v_i, v_j.\ (\forall \{v_i, v_j\} \notin E) \\ r_2 &: ok. \end{aligned}$$

and the domain description:

$$s(\mathcal{G})\ \textbf{determines}\ o \qquad \forall o \in \mathcal{O}(\mathcal{G})$$

Consider the evolution:

$$H(\mathcal{G}) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(\mathcal{G})} \langle \emptyset, \{\neg clique, v_1, \ldots, v_n\} \rangle.$$

Now we prove that there is a one-to-one correspondence between cliques $\mathcal{C}$ of the graph $\mathcal{G}$ and full repairs $H^{\mathcal{C}}(\mathcal{G})$ for $H(\mathcal{G})$ in $A(\mathcal{G})$ such that $|\Delta(H^{\mathcal{C}}(\mathcal{G}), H(\mathcal{G}))| = 2(n + 1 - |\mathcal{C}|)$. Indeed, maximum cliques will then correspond to minimum full repairs.

($\Rightarrow$) Let $\mathcal{C}$ be a clique in the graph $\mathcal{G}$. Consider the evolution

$$H^{\mathcal{C}}(\mathcal{G}) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(\mathcal{G})} \langle \emptyset, \{clique\} \cup \mathcal{C} \cup (V \setminus \mathcal{C})^{\neg} \rangle.$$

Now we prove that the unique answer set $S^{\mathcal{C}}$ of the program $th(A(\mathcal{G}), H(\mathcal{G}))$ is coherent. By contradiction, assume that $S^{\mathcal{C}}$ is incoherent. Then, it is the case that at least the body $\{v_i, v_j\}$ of a rule $r_{1,i,j}$ is contained in $\mathcal{C} \cup (V \setminus \mathcal{C})^{\neg}$. Since there does not exist in $\mathcal{G}$ an edge connecting nodes v_i and v_j, this contradicts the fact that $\mathcal{C}$ is a clique.
Hence, it follows that $H^{\mathcal{C}}(\mathcal{G})$ is a full repair for $H(\mathcal{G})$ in $A(\mathcal{G})$. As for the size of the set $\Delta(H^{\mathcal{C}}(\mathcal{G}), H(\mathcal{G}))$, this is $|\Delta(H^{\mathcal{C}}(\mathcal{G}), H(\mathcal{G}))| = 2|V \setminus \mathcal{C}| + 2 = 2(n - |\mathcal{C}|) + 2 = 2(n + 1 - |\mathcal{C}|)$.

($\Leftarrow$) Let $H'(\mathcal{G})$ be a full repair for $H(\mathcal{G})$ in $A(\mathcal{G})$. Then, $th(A(\mathcal{G}), H'(\mathcal{G}))$ is coherent, and hence its unique answer set S', by rules $r_{1,i,j}$ and r_2, is such that $\neg ok \notin S'$. Therefore, it is the case that there does not exist a pair of positive literals v_j, v_j in $\mathcal{O}(state(H'(\mathcal{G})))$ such that $\{v_i, v_j\} \notin E$, or, equivalently, that the set of nodes

$$\mathcal{C} = V \cap \mathcal{O}(state(H'(\mathcal{G})))$$

is a clique of the graph $\mathcal{G}$. From what we have stated above, it follows that $\mathcal{O}(state(H'(\mathcal{G}))) = \{clique\} \cup \mathcal{C} \cup (V \setminus \mathcal{C})^{\neg}$, and the size of the set $\Delta(H'(\mathcal{G}), H(\mathcal{G}))$ is $|\Delta(H'(\mathcal{G}), H(\mathcal{G}))| = 2|V \setminus \mathcal{C}| + 2 = 2(n + 1 - |\mathcal{C}|)$. □

We are now in the position of proving the result about the MINIMUM-FULL-REPAIR-MEMBERSHIP problem.

Theorem 13 MINIMUM-FULL-REPAIR-MEMBERSHIP *is* $\Delta_2^{\mathrm{P}}[\mathcal{O}(\log \mathrm{n})]$-*complete.*

Proof (Membership) Given an agent A, an evolution H and a literal o, we can check whether o belongs to $\mathcal{O}(state(H'))$ for some minimum full repair H' for H in A as follows. First of all, the machine computes the size n^* of the minimum full repair for H. This can be accomplished by calling $\mathcal{O}(\log n)$ times an NP oracle, where n is the number $|\mathcal{O}(state(H))|$ of observables sensed in evolution H, as detailed in the membership part of Theorem 12.

Then, the machine performs an additional query to the oracle in order to check whether there exists a full repair H' for H in A of size n^* such that the observable o occurs in $\mathcal{O}(state(H'))$. Thus, this time the oracle guesses an evolution H' for A such that $|\Delta(H', H)| = n^*$ and $o \in \mathcal{O}(state(H'))$, and then checks in polynomial time that H' is indeed a full repair for H in A.

(Hardness) Let $\Phi = C_1 \wedge \ldots \wedge C_m$ be a boolean formula in conjunctive normal form, with $C_j = t_{j,1} \vee t_{j,2} \vee t_{j,3}$, where each $t_{j,k}$ is a literal on the set of boolean variables $X = x_1, \ldots, x_n$. An assignment $\mathcal{T}$ for X is *csat*-maximum for Φ iff it satisfies the maximum number of clauses over all the assignments for X. The problem of deciding whether every csat-maximum assignment satisfies C_1 is $\Delta_2^{\mathrm{P}}[\mathcal{O}(\log \mathrm{n})]$-complete [30].

Since $\Delta_2^{\mathrm{P}}[\mathcal{O}(\log \mathrm{n})]$ is closed under complement, the problem of deciding whether there is a csat-maximum assignment which does not satisfy C_1 is also complete for this class. We shall exploit a reduction for this latter problem.

Given the formula Φ, consider the agent $A(\Phi)$ with the set of observables $\mathcal{O}(\Phi) = \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n, c_1, ..., c_m\}$, the sensor $s(\Phi)$, the knowledge base $\mathcal{K}(\Phi)$:

$$\begin{array}{lll} r_0 : \neg unsat. & \\ r_{1,i} : unsat \leftarrow x_i, \bar{x}_i. & (1 \leq i \leq n) \\ r_{2,i} : unsat \leftarrow \neg x_i, \neg \bar{x}_i. & (1 \leq i \leq n) \\ r_{3,j} : \neg c_j \leftarrow \neg t_{j,1}, \neg t_{j,2}, \neg t_{j,3}. & (1 \leq j \leq m) \end{array}$$

and the domain description:

$$s(\Phi) \textbf{ determines } o \qquad \forall o \in \mathcal{O}(\Phi)$$

Consider the evolution:

$$H(\Phi) = \langle \emptyset, \emptyset \rangle \rightarrow_{s(\Phi)} \langle \emptyset, \{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n, c_1, ..., c_m\} \rangle.$$

Let us firstly investigate on the structure of minimum full repairs. For each full repair H', we note that $|\{x_i, \bar{x}_i\} \cap \mathcal{O}(state(H'))| = 1$ holds for each $1 \leq i \leq n$, because of rules r_0, $r_{1,i}$ and $r_{2,j}$ and the fact that $H'[1]$ must not lead to incoherent information. Moreover, let S_X be the set $\{x_1, \neg x_1, ..., x_n, \neg x_n\} \cap \mathcal{O}(state(H'))$. Then, $\neg c_j$ is in $\mathcal{O}(state(H'))$, for each clause C_j that is not satisfied by $\mathcal{T}_{S_X}$, because of rules $r_{3,j}$. Thus, minimum full repairs are in one-to-one correspondence with csat-maximum assignments for Φ.

Armed with this observation, we can conclude that $\neg c_1$ occurs in $\mathcal{O}(state(H'))$ for some minimum full repair H' for $H(\Phi)$ in $A(\Psi)$ if and only if there is a csat-maximum assignment that does not satisfy C_1. □

6 Repair Computation over Answer Set Engines

Now that the framework for identifying and repairing anomalous executions in noisy environments has been illustrated and its complexity has been investigated, attention can be focused on the problem of devising effective strategies for its implementation.

To this end, sound and complete algorithms are next exhibited that transform any pair (A, H) of an agent and an evolution for it, respectively, into a suitable (extended) logic program $\mathcal{P}(A, H)$ such that its answer sets are in one-to-one correspondence with the full repairs for H in A.

The most interesting aspect of this transformation is that, since answer sets represent the solution of the reasoning problems, it is possible to implement a prototype tool for finding repairs in anomalous evolutions with the support of efficient engines such as GnT [45], DLV [51] and *Smodels* [62]. In fact, this tool might act as a front-end for any engine supporting the computation of answer sets of logic programs, as is graphically sketched in Figure 5: The basic idea is that a rewriting module implements the algorithm producing the specification $\mathcal{P}(A, H)$, and that a wrapping module is subsequently interfaced with the output of the engine, so that answer sets are parsed and translated into the repairs for H in A. In fact, reformulations in terms of logic programs have already been exploited in the literature for prototypically implementing other reasoning tasks such as abduction (see, e.g., [47, 52]), planning (see, e.g., [27, 74]), and diagnosis (see, e.g., [20, 26]).

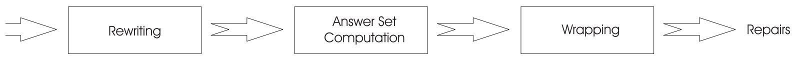

Fig. 5 Conceptual architecture for implementation on top of answer set engines.

In order to show the key ingredients of the approach, we shall next focus our attention on those scenarios where the background knowledge is formalized by means of negation-by-default-free extended logic programs. Indeed, these scenarios are very simple and intuitive, but yet expressive enough to encode problems that are complete for the first and the second level of the polynomial hierarchy, as it emerged from the complexity analysis carried out in the previous sections. In fact, despite the absence of negation by default, the emergence of problems complete for the second level of the polynomial hierarchy makes it impossible to devise polynomially-bounded encodings in terms of extended logic programs, since this formalism is known to capture the complexity class NP only (see, e.g., [21]). Therefore, even in the case of this basic setting, we need to resort to a more powerful formalism; and, our choice here is to consider extended logic programs enhanced with *disjunction*, which are able to model any problem that is complete for the second level of the polynomial hierarchy under the answer set semantics (cf. [21]). Investigating different implementation strategies for general background knowledge bases is left as subject for further research.

A few notes on disjunctive programs. For the sake of completeness, we conclude this overview of the implementation approach by recalling that disjunctive extended logic programs allow clauses to have both disjunction (denoted by $\vee$) in their heads and negation in their bodies. More formally, a *disjunctive rule* r is a clause of the form:

$$L_0^1 \vee \cdots \vee L_0^h \leftarrow L_1, \cdots, L_m, not\ L_{m+1}, \cdots, not\ L_n.$$

where $h > 0$, $n \geq m \geq 0$ and each L_i (resp., L_0^j) is a literal, i.e., a propositional letter or its classical negation.

Then, answer sets for disjunctive programs are defined precisely as answer sets for disjunction-free programs, provided the extension of the notion of closure for contexts. Formally, let P be a negation-by-default-free disjunctive program; a context S is closed under P iff for each rule r of the form $L_0^1 \vee \cdots \vee L_0^h \leftarrow L_1, \cdots, L_m.$, if $L_1, \cdots, L_m$ are in S, then at least one of the literals in $\{L_0^1, ..., L_0^h\}$ is in S as well.

It is well-known that deciding the existence of a coherent answer set is Σ_2^{P}-complete for this class of programs [29].

6.1 Basic Rewriting

In order to implement repair problems for negation-by-default-free background knowledge bases, we make use of the rewriting algorithm *Repair-ToANSW* shown in Figure 6. The algorithm takes as input an agent A and an evolution H, and outputs a logic program $\mathcal{P}(A, H)$, whose salient aspects are discussed below. Beforehand, we shall discuss the notation used in $\mathcal{P}(A, H)$ and, in particular, the three ingredients in it.

First, the program $\mathcal{P}(A, H)$ is built by avoiding the use of classical negation as well as of negation by default, which is an implementation choice suggested by the need of being flexible in the way inconsistency is dealt with—for instance, deriving a letter p and its negation $\neg p$ should sometimes be allowed for implementing the "saturation" technique we shall discuss to deal with minimal repairs. Therefore, for any propositional letter p in $\mathcal{P}(A, H)$, we just denote its classical negation $\neg p$ by means of a novel and distinguished

```
  Input:  An agent $A$ and $H : \langle S_B^0, S_O^0 \rangle \rightarrow_{a_1} \langle S_B^1, S_O^1 \rangle \rightarrow_{a_2} \dots \rightarrow_{a_n} \langle S_B^n, S_O^n \rangle$;
 Output:  A logic program $\mathcal{P}(A, H)$;
Method:  Perform the following steps:
      1. $\mathcal{P}(A, H) := \emptyset$;
      2. /*——— Initialization ———*/
         **for each** literal $l \in S_B^0 \cup S_O^0$, **insert into** $\mathcal{P}(A, H)$ the fact
           (a) $\sigma(l)^0$.
         **for each** letter $l \notin (S_B^0 \cup S_O^0) \cup (S_B^0 \cup S_O^0)^\neg$, **insert into** $\mathcal{P}(A, H)$ the fact
           (b) $u\_l^0$.
      3. /*——— (not-free) $\mathcal{K}$ rewriting ———*/
         **for each** rule $r \in \mathcal{K}$ of the form $h \leftarrow b_1, \dots, b_m$, **insert into** $\mathcal{P}(A, H)$ the rules
           (a) $\sigma(h)_k^i \leftarrow \sigma(b_1)_k^i, \dots, \sigma(b_m)_k^i$., where $i = 1..n$
         **for each** literal $l \in \mathcal{B} \cup \mathcal{O}$, **insert into** $\mathcal{P}(A, H)$ the rules
           (b) $l_k^i \leftarrow l^i$.
           (c) $\bar{l}_k^i \leftarrow \bar{l}^i$.
      4. /*——— Beliefs update ———*/
         **for each** ef-prop. $a_i$ **causes** $l$ **if** $p_1, \dots, p_k$, **insert into** $\mathcal{P}(A, H)$ the rules
           (a) $\sigma(l)^i \leftarrow \sigma(p_1)^{i-1}, \dots, \sigma(p_k)^{i-1}$.
           (b) $\sigma(l)^i \leftarrow \sigma(l)^{i-1},\ \sigma(\neg p_j)^{i-1}$., where $j = 1..k$
           (c) $u\_l^i \leftarrow u\_l^{i-1},\ \sigma(\neg p_j)^{i-1}$., where $j = 1..k$
           (d) $\sigma(\neg l)^i \leftarrow \sigma(\neg l)^{i-1},\ \sigma(\neg p_j)^{i-1}$., where $j = 1..k$
           (e) $u_i \leftarrow u\_\gamma(p_j)^{i-1}$., where $j = 1..k$
           (f) $ut\_\gamma(p_j)^i \leftarrow p_j^{i-1}$., where $j = 1..k$
           (g) $ut\_\gamma(p_j)^i \leftarrow u\_\gamma(p_j)^{i-1}$., where $j = 1..k$
           (h) $\sigma(l)^i \leftarrow \sigma(l)^{i-1}, ut\_\gamma(p_1)^{i-1}, \dots, ut\_\gamma(p_k)^{i-1}, u_i$.
           (i) $u\_\gamma(l)^i \leftarrow u\_\gamma(l)^{i-1}, ut\_\gamma(p_1)^{i-1}, \dots, ut\_\gamma(p_k)^{i-1}, u_i$.
           (j) $u\_\gamma(l)^i \leftarrow \sigma(\neg l)^{i-1}, ut\_\gamma(p_1)^{i-1}, \dots, ut\_\gamma(p_k)^{i-1}, u_i$.
         **for each** letter $l \in \mathcal{B}$ not caused by $a_i$, **insert into** $\mathcal{P}(A, H)$ the rules
           (k) $l^i \leftarrow l^{i-1}$.
           (l) $\bar{l}^i \leftarrow \bar{l}^{i-1}$.
           (m) $u\_l^i \leftarrow u\_l^{i-1}$.
      5. /*——— Sensing ———*/
         **for each** k-prop. $a_i$ **determines** $o$ **if** $q_1, \dots, q_k$, **insert into** $\mathcal{P}(A, H)$ the rules
           (a) $o^i \vee \bar{o}^i \leftarrow \sigma(q_1)^{i-1}, \dots, \sigma(q_k)^{i-1}$.
         **for each** $o \in \mathcal{O}$ not determined by $a_i$ **insert into** $\mathcal{P}(A, H)$ the rules
           (b) $o^i \leftarrow o^{i-1}$.
           (c) $\bar{o}^i \leftarrow \bar{o}^{i-1}$.
           (d) $u\_o^i \leftarrow u\_o^{i-1}$.
         **for each** $l \in \mathcal{B} \cup \mathcal{O}$ **insert into** $\mathcal{P}(A, H)$ the rules
           (e) $bad \leftarrow l_k^i, \bar{l}_k^i$.
           (f) $bad \leftarrow l_k^i, \bar{l}_k^i$.
      6. /*——— Executability ———*/
         **for each** ex-prop. **executable** $a_i$ **if** $l_1, \dots, l_k$ **insert into** $\mathcal{P}(A, H)$ the rules
           (a) $bad \leftarrow \sigma(\neg l_j)^{i-1}$., where $j = 1..k$
           (b) $bad \leftarrow u\_\gamma(l_j)^{i-1}$., where $j = 1..k$
```

Fig. 6 Algorithm *RepairToANSW*.

predicate symbol $\bar{p}$; and, for notational convenience, we make use of the function σ such that $\sigma(p) = p$ and $\sigma(\neg p) = \bar{p}$. In addition, for each literal l, by $\gamma(l)$ we shall denote the letter on top of which it is built. Also, since we are dealing with 3-valued interpretations, we shall use the symbol u_p to denote that the truth value of p is currently unknown.

The second notational ingredient in $\mathcal{P}(A, H)$ serves the aim of keeping track of the actions $a_1, \dots, a_n$ in the evolution H. Thus, each literal l is now equipped with an index ranging in $1..n$ denoting the state of the evolution in which its truth value is being considered; accordingly, l^i is used to denote the literal l in the i-th step of the evolution.

Finally, the last notational ingredient is that all the rules in $\mathcal{K}$ will be evaluated over distinguished predicates, as to avoid conflicts with the other rules in $\mathcal{P}(A, H)$. Thus, for any letter l^i, the corresponding letter in $\mathcal{K}$ is denoted by l^i_k.

We are now ready to discuss the five steps illustrated in Figure 6. Step 1 and Step 2 are responsible for the initialization of the program $\mathcal{P}(A, H)$. In particular, in rule 2.(a) we add a fact for each literal (belief or observable) that occurs in the first step of the evolution (and, hence, in the v-propositions of the domain description), while in rule 2.(b) we state that all the other literals are unknown in the initial state.

Rules inserted in Step 3 serve the purpose of evaluating the knowledge base $\mathcal{K}$, which is instantiated for each step $i = 1..n$ of the evolution. In particular, 3.(b) and 3.(c) copy the values for the fluents over the corresponding letters in $\mathcal{K}$ (only those that are known to be true or false in the i-th state), while 3.(a) is responsible for the actual evaluation.

Rules inserted in Step 4 and Step 5 serve to model the domain description, and in fact they are inspired by the logic programming translation of the 0-semantics in [10, 73]. In more detail, rules inserted in the Step 4 aim at updating the beliefs based on each non-sensing action a_i occurred in H. Indeed, rule 4.(a) asserts the effect l of the proposition whenever preconditions hold. Rule 4.(b), 4.(c), and 4.(d) state that the truth value of l remains unchanged if the precondition is false. Eventually, rules 4.(e)...4.(j) deal with the scenario where the precondition is possibly true, but not actually true. Finally, rule 4.(k), 4.(l), and 4.(m) take care of those beliefs that are not affected by a_i, and whose values remain unchanged by inertia.

Rules inserted in Step 5 serve the crucial role of handling the sensing of the environment. In particular, 5.(a), 5.(b), 5.(c), and 5.(d) act similarly to the corresponding rules added in Step 4. The main difference is that a disjunctive rule is now used to state that at each step i where a sensing occurs for a letter o, either o or its negation $\neg o$ (here represented as $\bar{o}$) should hold. Also, observables that are not sensed by a_i keep their previous value. In fact, if some problems occur with the sensing and an incoherence emerges from $\mathcal{K}$, then *bad* has to be derived (cf. 5.(e) and 5.(f)).

Finally, rule 6.(a) and rule 6.(b) are used to enforce the entailment of *bad*, whenever the action a_i is not executable.

Summarizing the construction above, we may say that the program $\mathcal{P}(A, H)$ simulates the evolution H with the only difference that sensing values are left uncertain, so that each possible answer set will correspond to a possible way of sensing the sensors. Clearly enough, only correct sensing leads to repairs for H; therefore $\mathcal{P}(A, H)$ should be further constrained as to avoid the entailment of *bad*. The correctness of the rewriting is stated next.

Theorem 14 *Let A be an agent with negation-by-default-free normalized knowledge base and $H : \langle S_B^0, S_O^0\rangle \rightarrow_{t_1} \langle S_B^1, S_O^1\rangle \rightarrow_{t_2} \dots \rightarrow_{t_n} \langle S_B^n, S_O^n\rangle$ be an evolution for A. Let P' be the program $\mathcal{P}(A, H) \cup \{c \leftarrow bad, not\ c.\}$, where c is a distinguished letter and where $\mathcal{P}(A, H)$ is the rewriting obtained by the algorithm in Figure 6. Then,*

(1) for each answer set S of P', $H(S)$ is a full repair for H in A, and
(2) for each full repair $\tilde{H}$ for H in A, there is an answer set S of P' such that $\tilde{H} = H(S)$,

where $H(S) : \langle S_B(S)^0, S_O(S)^0\rangle \rightarrow_{t_1} ... \rightarrow_{t_n} \langle S_B(S)^n, S_O(S)^n\rangle$, with
- *$S_B(S)^i = \{l \mid l^i \in S \wedge l \in \mathcal{B}\} \cup \{\neg\, l \mid \bar{l}^i \in S \wedge l \in \mathcal{B}\}$, and*
- *$S_O(S)^i = \{o \mid o^i \in S \wedge o \in \mathcal{O}\} \cup \{\neg\, o \mid \bar{o}^i \in S \wedge o \in \mathcal{O}\}$.*

Proof

(1) Let S be an answer set of P'. Beforehand, we note that the rule

$$c \leftarrow bad, not\ c.$$

acts as a constraint, since it forces *bad* not to occur in S; otherwise, S would not be an answer set. It follows that, at each step, the result of the application of a_i conforms with the semantics of the 0-approximation, because of the rules 4.(a)...4.(m), 5.(a)...5.(d), 6.(a), and 6.(b) that implement Equation (1) and Equation (2). In particular, each action a_i is executable, for otherwise *bad* would be entailed by rules 6.(a) and 6.(b). Thus, we have that $H(S)$ is indeed an evolution. In fact, by construction, the first three conditions in Definition 4 are satisfied and, it only remains to show that for each $i \in \{1, ..., n\}$, $H[i]$ contains no anomalies. To this end, we exploit Proposition 1 and notice that to avoid anomalies, at each step i, the theory $th(A, H[i])$ has to be coherent. Eventually, if $th(A, H[i])$ were incoherent, we would conclude that *bad* is entailed in S as well, due to rules 5.(e) and 5.(f), which is impossible.

(2) Let $\tilde{H} : \langle \tilde{S}_B^0, \tilde{S}_O^0\rangle \rightarrow_{t_1} \langle \tilde{S}_B^1, \tilde{S}_O^1\rangle \rightarrow_{t_2} ... \rightarrow_{t_n} \langle \tilde{S}_B^n, \tilde{S}_O^n\rangle$ be a full repair for H in A, and consider the context S obtained by closing the set:

$$\begin{array}{l}
\{l^i \mid l \in \tilde{S}_B^i \cup \tilde{S}_O^i \wedge l \in \mathcal{B} \cup \mathcal{O}\} \cup \\
\{\bar{l}^i \mid \neg l \in \tilde{S}_B^i \cup \tilde{S}_O^i \wedge l \in \mathcal{B} \cup \mathcal{O}\} \cup \\
\{u\_l^i \mid (l \cup \neg l) \cap (\tilde{S}_B^i \cup \tilde{S}_O^i) = \emptyset \wedge l \in \mathcal{B} \cup \mathcal{O}\} \cup \\
\{l_k^i \mid th(A, \tilde{H}[i]) \models l \wedge l \in \mathcal{B} \cup \mathcal{O}\} \cup \\
\{\bar{l}_k^i \mid th(A, \tilde{H}[i]) \models \neg l \wedge l \in \mathcal{B} \cup \mathcal{O}\}
\end{array}$$

under the inference w.r.t. rules 4.(e)...4.(j).
In fact, by construction it holds that $H(S) = \tilde{H}$. Thus, we have only to show that S is an answer set of P'. Indeed, if some of the rules were not satisfied by S, one may immediately conclude that $\tilde{H}$ is not a full repair. Thus, we have to show that S is minimal. Assume, for the sake of contradiction, that there exists a context $S' \subset S$ that is closed under $Red(P', S) = \mathcal{P}(A, H)$. Since $bad \notin S'$, we can use the same line of reasoning as in the proof of point *(1)* above and conclude that $H(S')$ is a full repair for H. Then, when comparing $H(S')$ with $H(S)$, since $S' \subset S$ there must be a state, say i^*, such that either $S_B(S')^{i^*} \subset S_B(S)^{i^*}$ or $S_O(S')^{i^*} \subset S_O(S)^{i^*}$, while $S_B(S')^i = S_B(S)^i$ and $S_O(S')^i = S_O(S)^i$, for each $i < i^*$. But, this is impossible because of the definition of evolution and because of the fact that $H(S) = \tilde{H}$ is a repair, and hence an evolution, by hypothesis. □

6.2 Minimum repairs

In order to deal with the problem of singling out minimum full repairs for noisy evolutions, we shall next exploit an approach to encode optimization problems that is used, for instance, in the DLV system [51]. The approach relies on the use of weak constraints, i.e., of rules of the form $:\sim\ b_1, \cdots, b_k, not\ b_{k+1}, \cdots, not\ b_{k+m}$, expressing a set of desired conditions that may be however violated (the constraint is violated if the expression evaluates to true). Their informal semantics is to minimize the number of violated instances. In fact, in [17] it is proved that the introduction of weak constraints allows the solution of optimization problems since each weak constraint can be regarded as an "objective function" of an optimization problem.

Given a disjunctive logic program P and a set W of weak constraints, the idea is to order the answer sets of P w.r.t. the number of weak constraints that are not satisfied: *best stable models* are those that minimize this number [17].

Example 6 Given a graph $G = \langle V, E\rangle$, denoted by the unary predicate *node* and the binary predicate *edge*, we can model the `MAX_CLIQUE` problem, asking for the clique of G having maximum size, by means of the following program:

$$\begin{array}{l} c(X) \leftarrow not\ \ nc(X),\ node(X). \\ nc(X) \leftarrow not\ \ c(X),\ node(X). \\ p \leftarrow c(X), c(Y), X \neq Y, not\ \ edge(X,Y), not\ p. \\ :\sim\ \ nc(X). \end{array}$$

where the first two rules are used for creating all the possible partitions of nodes into `c` and `nc`, the third one is used for ensuring that nodes in `c` form a clique, i.e., each pair of nodes in `c` is connected by an edge, while the weak constraint minimizes the number of vertices that are not in the clique, or equivalently it maximizes the size of the clique. Then, the best stable models are in one-to-one correspondence with maximum-size cliques in G. ◁

Thus, the algorithm in Figure 6 can be modified by inserting the constraint $:\sim\ \ \sigma(\neg o)^n.$ into $\mathcal{P}(A, H)$, for each observable o in $\mathcal{O}(state(H))$. Then, letting $\mathcal{P}^{\sim}(A, H)$ be the transformed program resulting from applying the modified algorithm, we have that minimum repairs are in one-to-one correspondence with best stable models, as shown by the following theorem.

Theorem 15 *Let A be an agent with negation-by-default-free normalized knowledge base and $H : \langle S_B^0, S_O^0\rangle \rightarrow_{t_1} \langle S_B^1, S_O^1\rangle \rightarrow_{t_2} \ldots \rightarrow_{t_n} \langle S_B^n, S_O^n\rangle$ be an evolution for it. Let P' be the program $\mathcal{P}^{\sim}(A, H) \cup \{c \leftarrow bad, not\ c.\}$. Then,*

(1) for each best answer set S of P', $H(S)$ is a minimum full repair for H in A, and
(2) for each minimum full repair $\tilde{H}$ for H in A, there is a best answer set S of P' such that $\tilde{H} = H(S)$,

```
 Input:  An agent $A$ and $H : \langle S_B^0, S_O^0 \rangle \rightarrow_{t_1} \langle S_B^1, S_O^1 \rangle \rightarrow_{t_2} \dots \rightarrow_{t_n} \langle S_B^n, S_O^n \rangle$;
Output:  A logic program $\mathcal{P}^{\vee}(A, H)$;
Method:  Perform the following steps:
    1. /*——— Initialization ———*/
       let $\mathcal{P}_w(A, H)$ be a copy of $\mathcal{P}(A, H)$ where each letter $p$ is replaced by $p_w$
    2. $\mathcal{P}^{\vee}(A, H) := \mathcal{P}(A, H) \cup \mathcal{P}_w(A, H)$;
    3. /*——— Saturation ———*/
       insert into $\mathcal{P}^{\vee}(A, H)$ the rules
        (a) $w \leftarrow \sigma(o)^n, \sigma(\neg o)^n_w.$, where $o \in \mathcal{O}(state(H))$
        (b) $eq\_o_w \leftarrow o^n, o^n_w.$, where $o \in \mathcal{O}(state(H))$
        (c) $eq\_o_w \leftarrow \bar{o}^n, \bar{o}^n_w.$, where $o \in \mathcal{O}(state(H))$
        (d) $w \leftarrow eq\_l_{1\,w}, \dots, eq\_l_{h\,w}.$, where $\{l_1, \dots l_h\} = \mathcal{O}(state(H))$
        (e) $w \leftarrow bad_w.$
        (f) $w \leftarrow not\ w.$
        (g) $p_w \leftarrow w.$, where $p_w$ is a letter in $\mathcal{P}_w(A, H)$
```

Fig. 7 Algorithm *MinimalRepairToANSW*.

Proof Given A and H, the answer sets of $\mathcal{P}(A, H) \cup \{c \leftarrow bad, not\ c\}$ are in one-to-one correspondence with the full repairs for H. By definition these answer sets are to be ordered w.r.t. the number of violated weak constraints. In fact, each weak constraint is of the form $:\sim\ \sigma(\neg o)^n$. where o is in $\mathcal{O}(state(H))$; thus, violating this constraint means that the value of o differs between $H(S)$ and H. Minimizing the number of violated constraints in a best model S amounts, then, to minimizing the differences over the values for the observables between the full repair $H(S)$ and the evolution H. □

6.3 Minimal full repairs

We conclude this section by discussing how minimal full repairs can be identified. In this case, we observe that the power of the disjunctive encoding has to be fully exploited in order to deal with the reasoning task that lies at the second level of the polynomial hierarchy.

Indeed, the careful reader may have noticed that the program $\mathcal{P}(A, H)$ does not make any substantial use of the disjunctive connective, since the guesses of a suitable value for the observations performed in 5.(a) could have been equivalently obtained by using negation-by-default rules.

To model minimal full repairs, instead, disjunction has to be used in a more elaborate way. To this end, we shall apply a "saturation" technique on top of the rewriting $\mathcal{P}(A, H)$. The resulting algorithm is reported in Figure 7.

The basic idea in it is to consider the rewriting $\mathcal{P}(A, H)$ together with a copy of it, denoted by $\mathcal{P}_w(A, H)$, where each letter p in $\mathcal{P}(A, H)$ is consistently replaced with a fresh letter p_w. The role of $\mathcal{P}(A, H)$ plus the constraint $c \leftarrow bad, not\ c$. is to encode the derivation of full repairs for H.

The role of $\mathcal{P}_w(A, H)$ is instead to guarantee that such computed repairs are in fact minimal ones. To this end, we note that because of step 3.(f) (in Figure 7), any answer set of $\mathcal{P}^{\vee}(A, H)$ must contain w and in turn all the letters of the form p_w, because of 3.(g)—thus, these letters are saturated. Then, the other rules are defined to guarantee that w is entailed as long as the full repair is a minimal one.

Theorem 16 *Let A be an agent with negation-by-default-free normalized knowledge base and $H : \langle S_B^0, S_O^0\rangle \rightarrow_{t_1} \langle S_B^1, S_O^1\rangle \rightarrow_{t_2} \dots \rightarrow_{t_n} \langle S_B^n, S_O^n\rangle$ be an evolution for it. Let P' be the program $\mathcal{P}^{\vee}(A, H) \cup \{c \leftarrow bad, not\ c.\}$. Then,*

(1) for each answer set $S^{\vee}$ of P', $H(S^{\vee})$ is a minimal full repair for H in A, and
(2) for each minimal full repair $\tilde{H}$ for H in A, there is an answer set $S^{\vee}$ of P' such that $\tilde{H} = H(S^{\vee})$.

Proof

(1) Consider an answer set $S^{\vee}$ of P' and let $S^{\vee} = S \cup S_w$, where S only contains literals defined in $\mathcal{P}(A, H)$, while S_w only contains literals of the form p_w (defined in $\mathcal{P}_w(A, H)$ plus rules in Step 3) and possibly the letter w. In fact, note that because of the rule 3.(f), it is the case that $w \in S^{\vee}$ and, hence, because of the rule 3.(g), S_w contains *all* the letters of the form p_w as well as w.
Now, consider the evolution $H(S^{\vee})$ that coincides with $H(S)$, since each state in it is defined over predicates in $\mathcal{P}(A, H)$. This evolution is, indeed, a full repair for H in A, since we can precisely apply the same line of reasoning as in Theorem 14. Thus, we only need to show that $H(S)$ is a minimal full repair.
Assume for the sake of contradiction that $H(S)$ is not minimal and let H^* be the full repair such that $\Delta(H^*, H) \subset \Delta(H(S), H)$. Consider the answer set S^* for the program $\mathcal{P}(A, H) \cup \{c \leftarrow bad, not\ c.\}$ such that $H(S^*) = H^*$ (cf. Theorem 14). And, eventually, let us build the set $S_w^* = \{l_w \mid l \in S^*\}$. We claim that $S' = S \cup S_w^* \cup \{eq\_o_w \mid (o \in S \wedge o \in S_w^*) \vee (\neg o \in S \wedge \neg o \in S_w^*)\}$ is closed under $Red(P', S^{\vee})$.
To this aim, note that rules in $\mathcal{P}(A, H)$ and $\mathcal{P}_w(A, H)$ are trivially satisfied by S and S_w, respectively. Thus, it remains to show that the rules added in the Step 3 are also satisfied. In fact, rules 3.(a) are satisfied since $\Delta(H^*, H) \subset \Delta(H(S), H)$ and, hence, their body evaluates to false. Rule 3.(b), 3.(c), and 3.(d) are satisfied by construction of S' and since $H^* \neq H$ and, hence, we would not need to entail w. Rule 3.(e) does not apply since H^* is a full repair and, hence, no incoherences occur with it. Rule 3.(f) does not occur in the reduct of P', since w evaluates to true in $S^{\vee}$. And, finally, rule 3.(g) is satisfied since the body is trivially false. It follows that $S' \subset S^{\vee}$ is closed under $Red(P', S^{\vee})$, which is impossible, since $S^{\vee}$ is an answer set for P'.

(2) Let $\tilde{H}$ be a minimal full repair for H in A. Consider the context $S \cup S_w$ such that: S_w contains all the letters of the form p_w as well as w; and, S is such that $H(S) = \tilde{H}$ according to the construction in Theorem 14. We claim that $S \cup S_w$ is an answer set for P'. Actually, the fact that $S \cup S_w$ is closed under P' derives from the fact that S is closed under $\mathcal{P}(A, H) \cup \{c \leftarrow bad, not\ c\}$ and that S_w contains all the literals in $\mathcal{P}_w(A, H)$.
Then, assume for the sake of contradiction that $S \cup S_w$ is not a minimal context for $Red(P', S \cup S_w)$. Note that S is a minimal context for the program $\mathcal{P}(A, H)$, because of the arguments in the proof of Theorem 14. Thus, we have to assume that there is another context of the form $S \cup S'_w$

that is closed under $Red(P', S \cup S_w)$ and such that: $S'_w \subset S_w$. Then, because of the rule 3.(g), we have to assume that w is not in S'_w. Consider now the set $S' = \{l \mid l_w \in S'_w\}$. According to the rules 3.(a), 3.(d), and 3.(e), the fact that $w \notin S'_w$ entails that $H(S_w)$ is a full repair such that $\Delta(H(S_w), H) \subset \Delta(H(S), H)$, a contradiction with the fact that $H(S) = \tilde{H}$ is a minimal full repair. □

7 Related Work

Several areas of research are related with the contribution of this paper ranging, for instance, from representing the uncertainty arising from noisy sensors to repairing inconsistent knowledge bases. Below we discuss some of the main features our framework shares with this related literature.

Monitoring frameworks. We start the discussion by pointing out that the problem of detecting and recovering anomalies in dynamic environments can naturally be perceived as an execution plan monitoring problem that is focused on the identification of noisy sensors. In this respect, we note however that two major differences characterize the proposed approach w.r.t. previous approaches in the literature (see, e.g., [61,38,31,33,25,22]).

First, for most of the earlier approaches, the reliability of the sensors is not a crucial issue, since anomalies emerge as discrepancies between the actual execution and a set of intended or preferred trajectories that are given to hand (possibly, in an implicit way after having fixed the goal), rather than as discrepancies between the observations gained through sensors and the available knowledge. And, second, by repairing an evolution we aimed at equipping the agent with some novel possible perception of the status of the world (without affecting the transitions already performed by the agent), whereas classical monitoring frameworks aim at finding an alternative plan ensuring the reachability of the goal and possibly undo some of the actions.

Belief change. Besides the marginal influence coming from earlier monitoring approaches, the careful reader may have noticed that our formalization is reminiscent of the growing body of research focusing on reasoning in the presence of incoherent information. Indeed, anomalies emerge as a form of disagreement between the background knowledge and the observations, so that the repairing process is just a method for resuming the agent's view of the world to a coherent state.

A classical reference in this context is the well-known *AGM theory of belief revision* originated by [1] and further developed by [2,3], in which some methods are proposed to revise a background theory when evidences from the world happen to be in conflict with it (*success axiom*). While in the classical model of belief revision the input sentence is always accepted, several models of belief change have been proposed subsequently, in which no absolute priority is assigned to the new information due to its novelty [43] (*non-prioritized belief revision*). However, these approaches are only loosely related to the work done here.

Indeed, as far as the comparison with the belief revision in presence of the success axiom is concerned, rather than being interested in revising the agent theory in order to entail the new information provided by the environment, we are interested in singling out environmental manifestations to be doubted about, while taking the background knowledge for granted.

Moreover, in the more general scenario in which no absolute priority is assigned to the new information due to its novelty, it must be said that the form of agreement usually taken into account by non-prioritized belief revision approaches is the consistency one. For example, in the *decision + revision* approach [58], the input sentence can be taken into account only if it is consistent with a set of pre-defined core beliefs. Moreover, in order to incorporate part of the input in the belief state, a strategy to remove inconsistency is generally defined. On the contrary, we have already noticed that according to our approach an anomaly may emerge for an agent in an evolution even though the agent's theory is coherent and, also, that an incoherent theory may well not admit any anomaly in it.

More importantly, when repairing an evolution, we determine values for the sensors that eliminate the discrepancy with the agent background knowledge, and this is done by flipping the actual truth value of some of the associated observations, while in belief change, the belief set is revised by adding or deleting beliefs.

(Consistency-based) Diagnosis. Other classical references for reasoning with inconsistent knowledge bases come from the study of the diagnosis problem, which is the problem of finding what is wrong with some possibly malfunctioning systems based on knowledge about the design of that system and observations about its actual behavior (cf. [66,65]). In particular, in the consistency-based approach to diagnosis, the problem is that of isolating components that are not consistent with all other components acting normally. Formally, there is a set H of *hypotheses*, a background theory T, and a set O of observations; then, the problem is to single out a set $\Delta \subseteq H$ so that $T \cup O \cup \Delta$ is consistent in the classical sense. In classical approaches, there is no knowledge as to how malfunctioning occurs and manifests itself, and only the "normal" behavior is axiomatized [37,48,67]. For instance, in [67], a hypothesis $\neg ab(C)$ is introduced for each component C that can possibly be faulty, and what follows from the assumptions of normality is cast into rules. In other approaches, faulty behavior of components can be constrained by including fault models, that is, functional descriptions of the components in case they are broken (see, e.g., [77,49]).

Clearly enough, the major difference with our approach is again in the focus on revising the observations. Indeed, our notion of repair differs from those usually exploited in diagnostic approaches (see, e.g., [8,59,6,79,63]) because of the focus on recovering values from faulty sensors, rather than on identifying hypotheses that may explain a malfunctioning of the system.

Repairs. We conclude by observing that the notion of "repair" has been widely used in different contexts in the last few years (sometimes under different names), with the ultimate goal of recovering a system, a knowledge base, or a database to a consistent state. For instance, in [78] the reader may find a general framework for reasoning about inconsistency over a broad class of nonmonotonic logics.

Similarly, in the context of database applications, the notion of repair [4] has been observed to play a crucial role for dealing with violations of integrity constraints that may have happened for different reasons, for instance, when pre-existing data are re-organized under a new schema that has integrity constraints describing semantic aspects of the new scenario. In most of the approaches in the literature (see, e.g., [12,18,16,28,4,40,53,34,32,57]), a repair is a new database that satisfies the constraints in the schema and that is obtained by deleting and/or adding tuples in the original database, depending on the underlying semantics adopted for the inconsistent database and on the kinds of integrity constraints which are allowed on the schema. In particular, repairs are usually defined as those databases that minimize (w.r.t. some given order such as subset minimality or minimum cardinality, to cite a few) the number of modifications needed to restore consistency.

Note that in our framework anomalies in observations are not dropped in order to be coherent with the rest of the observations in an evolution, but rather they are are detected and modified. Thus, the perspective adopted in this paper is slightly different from that of the works discussed above, and is in fact closer to the approaches in [13,11], where fine-grained methods are exploited to define repairs by attribute-value modifications. In these latter contexts, the optimality criteria is not hard-coded in the definition (precisely as in our case), and various notions of cost for attribute-value modifications are introduced and their complexity is investigated. In particular, the existence of a repair within a given distance to the original database instance turns out to be NP-complete is such frameworks, thereby agreeing with the results presented in this paper. Eventually, for the sake of completeness note that [80] has recently shown how to combine tuple-based and attribute-based repairs in a database context with a single framework.

We leave the section by noticing that in some approaches in the literature (see, e.g., [41,76]), repairs have been extend to take into account (user) preference criteria to improve the quality of the result. In particular, [76] uses a priority relation among tuples as the basis for ranking repairs, and then for selecting the "best" among them. Following this perspective, an interesting avenue of further research is to exploit user preferences in our framework to incorporate extra knowledge that a user can have w.r.t. specific sensors, by adding it on top of our current notion of repairs as to rank them.

8 Conclusion

In this paper we have defined a formal framework good for reasoning about agents' belief state evolution in environments sensed through possibly unreliable sensors. In our framework, no information whatsoever (neither certain nor probabilistic) is assumed to be available in advance about the reliability of sensors. However, the agent's perception can be maintained correct through the identification and the resolution of discrepancies occurring between sensor delivered data and the agent's internal trustable knowledge, encoded in the form of an ELP under answer set semantics. After having defined the formal framework, in order to pinpoint main computational complexity sources implied in the implementation of the anomaly detection and repairing agent's belief state evolution, several reasoning problems have been considered and their intrinsic difficulty has been studied. Finally, rewriting algorithms have been also proposed that make the framework implementable on top of available answer set engines (e.g., [51,62]).

We believe that our investigation is a step towards providing capabilities for dynamic plan monitoring and repairing in noisy environments, where it can be useful for an agent that is trying to achieve its goals to be able to monitor, identify anomalies and fix a plan while evolving [20,26,33]. In this respect, the implementation of anomaly identification primitives based on our rewriting algorithms deserves further work to make them available within conditional planning environments (e.g, [56,68]).

All the rewriting algorithms discussed in the paper as well as some specific complexity studies refer to the case of negation-by-default-free ELPs. Thus, an interesting avenue of further research would be completing the complexity analysis carried out in Section 5 for the case of general background knowledge bases, and investigating different implementation strategies for them.

The theoretical machinery we have developed in this paper accounts for environments where no information is available about sensors' reliability nor such information can be gained by agents while exploiting them, which is indeed significant in many scenarios. However, one might also consider strategies whereby an agent having looked at a specific sensor s as a faulty one may safely assume that s is going to provide wrong information in possible subsequent agent's reads. This would have influence on the agent's planning strategies in future steps. In particular, plan formation and monitoring algorithms might be constructed as to take into account sensors to "become", in the agent's perception, faulty. Also, numerical "reliability levels" might be associated to sensors to be exploited during plan formation and updates when plan faults and recovery take place. Finally, sensor reliability evaluation might be conducted by running agents cooperatively in order to be able to collect more useful information. We leave this much-interesting avenue of research as future work.

Finally, we note that even though the sensors under consideration can only report binary states, the framework we propose is not limited to the management of binary environmental measures, as many-valued discrete signals can be indeed simulated by sets of binary signals. However, investigating the impact of enriching the framework with sensors delivering real-valued data might also be an interesting direction for further research.

Acknowledgements The authors thank the anonymous referees for their useful comments and suggestions, which helped improving the quality of the paper.

References

1. Alchourrón, C.E., Gärdenfors, P., Makinson, D.: On the logic of theory change: Partial meet contraction and revision functions. Journal of Symbolic Logic **50**(2), 510–530 (1985)
2. Alchourrón, C.E., Makinson, D.: The logic of theory change: Contraction functions and their associated revision functions. Theoria **48**, 14–37 (1982)

3. Alchourrón, C.E., Makinson, D.: On the logic of theory change: Safe contraction. Studia Logica **44**, 405–422 (1985)
4. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS), pp. 68–79 (1999)
5. Bacchus, F., Halpern, J.Y., Levesque, H.J.: Reasoning about noisy sensors and effectors in the situation calculus. Artificial Intelligence, 111(1-2): 171–208 (1999)
6. Balduccini, M., Gelfond, M.: Diagnostic reasoning with a-prolog. Journal of Theory and Practice of Logic Programming **3**(4-5), 425–461 (2003)
7. Baral, C., Kreinovich, V., Trejo, R.: Computational complexity of planning and approximate planning in the presence of incompleteness. Artificial Intelligence **122**(1-2), 241–267 (2000)
8. Baral, C., McIlraith, S.A., Son, T.C.: Formulating diagnostic problem solving using an action language with narratives and sensing. In: Proc. of the 7th Int. Conf. of Principles of Knowledge Representation and Reasoning (KR), pp. 311–322 (2000)
9. Baral, C., Tran, N., Tuan, L.C.: Reasoning about actions in a probabilistic setting. In: Proc. of the 18th Conf. on Artificial Intelligence and 14th Conf. on Innovative Applications of Artificial Intelligence (AAAI/IAAI), pp. 507–512 (2002)
10. Baral, C., Son, T.C.: Approximate reasoning about actions in presence of sensing and incomplete information. In: Proc. of the 1997 International Logic Programming Symposium (ILPS), pp. 387–401 (1997).
11. Bertossi, L., Bravo, L., Franconi, E., Lopatenko, A.: The complexity and approximation of fixing numerical attributes in databases under integrity constraints. Information Systems **33**(4-5), 407–434 (2008)
12. Bertossi, L., Chomicki, J., Cortes, A., Gutierrez, C.: Consistent answers from integrated data sources. In: Proc. of the 6th Int. Conf. on Flexible Query Answering Systems (FQAS), pp. 71–85 (2002)
13. Bohannon, P., Flaster, M., Fan, W., Rastogi, R.: A Cost-Based Model and Effective Heuristic for Repairing Constraints by Value Modification, In: Proc. of the ACM SIGMOD Int. Conf. on Management of Databooktitle (SIGMOD), pp. 143–154 (2005)
14. Boutilier, C., Dean, R., Hanks. S.: Planning under uncertainty: structural assumptions and computational leverage. JAIR, 11: 1-94 (1999)
15. Boutilier, C., Reiter, R., Price. B.: Symbolic dynamic programming for first-order MDPs. In: Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 690–700 (2001)
16. Bravo, L., Bertossi, L.: Logic programming for consistently querying data integration systems. In: Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 10–15 (2003)
17. Buccafurri, F., Leone, N., Rullo, P.: Enhancing disjunctive datalog by constraints. IEEE Transactions on Knowledge and Data Engineering **12**(5), 845–860 (2000)
18. Calì, A., Lembo, D., Rosati, R.: Query rewriting and answering under constraints in data integration systems. In: Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 16–21 (2003)
19. Chen, Z., Toda, S.: The complexity of selecting maximal solutions. Information and Computation **119**(2), 231–239 (1995)
20. Damasio, C., Pereira, L.M., Schroeder, M.: Revise: Logic programming and diagnosis. In: Proc. of the 4th International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR), pp. 354–363. Dagstuhl, Germany (1997)
21. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Computing Survey **33**(3), 374–425 (2001)
22. Dix, J., Either, T., Fink, M., Polleres, A., Zhang, Y.: Monitoring Agents using Declarative Planning. Fundamenta Informaticae, **57**(2-4), 345–370 (2003)
23. Dix, J., Kraus, S., Subrahmanian, V.S.: Heterogenous Temporal Probabilistic Agents, ACM Transactions of Computational Logic **7**(1), 151–198 (2006)

24. Dix, J., Nanni, M., Subrahmanian, V.S.: Probabilistic Agent Reasoning, ACM Transactions of Computational Logic **2**(1), 201–245 (2000)
25. Eiter, T., Erdem, E., Faber, W.: Plan reversals for recovery in execution monitoring. In: Proc. of the 10th Int. Work. on Non-Monotonic Reasoning (NMR), pp. 147–154 (2004)
26. Eiter, T., Faber, W., Leone, N., Pfeifer, G.: The diagnosis frontend of the dlv system. AI Communications **12**(1-2), 99–111 (1999)
27. Eiter, T., Faber, W., Leone, N., Pfeifer, G., Polleres, A.: A logic programming approach to knowledge-state planning: Semantics and complexity. ACM Transaction on Computational Logic **5**(2), 206–263 (2004)
28. Eiter, T., Fink, M., Greco, G., Lembo., D.: Repair localization for query answering from inconsistent databases. ACM Transactions on Database Systems **33**(2), (2008)
29. Eiter, T., Gottlob, G.: On the Computational Cost of Disjunctive Logic Programming: Propositional Case. Ann. Math. Artif. Intell. **15**(3-4): 289-323 (1995)
30. Eiter. T., Gottlob. G.: The Complexity of Logic-Based Abduction. Journal of the ACM **42**(1), 3–42 (1995)
31. Eiter, T., Mascardi, V., Subrahmanian, V.: Error-tolerant agents. In: A.C. Kakas, F. Sadri (eds.) Computational Logic. Logic Programming and Beyond, pp. 586–625 (2002)
32. Faber, W., Greco, G., Leone, N.: Magic Sets and their application to data integration. Journal of Computer and System Sciences **73**(4), 584–609 (2007)
33. Fichtner, M., Großmann, A., Thielscher, M.: Intelligent execution monitoring in dynamic environments. Fundamenta Infformaticae **57**(2-4), 371–392 (2003)
34. Franconi, E., Palma, A.L., Leone, N., Perri, S., Scarcello, S.: Census Data Repair: A Challenging Application of Disjunctive Logic Programming. In: Proc. of the 8yh Int. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR), pp. 561–578 (2001)
35. Gelfond, M., Lifschitz. V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing, 9(3-4): 365–386 (1991).
36. Gelfond, M., Lifschitz. V.: Representing actions and change by logic programming. Journal of Logic Programming, 17(2-4): 301–323 (1993).
37. Genesereth, M.R.: The use of design descriptions in automated diagnosis. Artificial Intelligence **24**, 411–436 (1984)
38. Giacomo, G.D., Soutchanski, M., Reiter, R.: Execution monitoring of high-level robot programs. In: Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR), pp. 453–464 (1998)
39. Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., Turner, H.: Nonmonotonic causal theories. Artificial Intelligence **153**, 49–104 (2004)
40. Greco, G., Greco, S., Zumpano, E.: A Logical Framework for Querying and Repairing Inconsistent Databases. IEEE Transactions on Knowledge and Data Engineering **15**(6), 1389–1408 (2003)
41. Greco, G., Lembo, D.: Data Integration with Preferences Among Sources. In: Proc. of the 23rd International Conference on Conceptual Modeling (ER), pp. 231–244 (2004)
42. Halpern, J.Y., Tuttle, M.R.: Knowledge, probability, and adversaries. Journal of the ACM **40**(4), 917–962 (1993)
43. Hansson, S.O.: A Survey of Non-Prioritized Belief Revision. Erkenntnis **50**, 413–427 (1999)
44. Iocchi, L., Lukasiewicz, T., Nardi, D., Rosati, R.: Reasoning about actions with sensing under qualitative and probabilistic uncertainty. In: Proc. of the 6th European Conf. on Artificial Intelligence (ECAI), pp. 818–822 (2004)
45. Janhunen, T., Niemelä, I., Simons, P., You, J.H.: Unfolding partiality and disjunctions in stable model semantics. In: Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR), pp. 411–419 (2000)
46. Johnson, D.S.: Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity, chap. A catalog of complexity classes, pp. 67–161. Elsevier and The MIT Press (co-publishers) (1990)

47. Kakas, A.C., Mancarella, P.: Database updates through abduction. In: Proc of. the 16th Int. Conf. on Very Large Data Bases (VLDB), pp. 650–661. Brisbane, Queensland, Australia (1990)
48. de Kleer, J., Williams, B.C.: Diagnosing multiple faults. Artificial Intelligence **32(1)**, 97–130 (1984)
49. de Kleer, J., Williams, B.C.: Diagnosis with behavioral modes pp. 124–130 (1992)
50. Krentel, M.W.: The complexity of optimization problems. Journal of Computer and System Sciences **36**(3), 490–509 (1988)
51. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The dlv system for knowledge representation and reasoning. ACM Transactions on Computational Logic, **7**(3), 499-562 (2006)
52. Lin, F., You, J.H.: Abduction in logic programming: A new definition and an abductive procedure based on rewriting. Artificial Intelligence **140**(1-2), 175–205 (2002)
53. Lin, J., Mendelzon, A.O.: Merging databases under constraints. International Journal of Cooperative Information Systems **7**(1), 55–76 (1998)
54. Littman, M.L., Goldsmith, J., Mundhenk. M.: The Computational Complexity of Probabilistic Planning. JAIR 9:1–36 (1998).
55. Lobo, J., Mendez, G., Taylor, S.R.: Adding knowledge to the action description language a. In: Proc. of the 14th Conf. on Artificial Intelligence and 9th Conf. on Innovative Applications of Artificial Intelligence (AAAI/IAAI), pp. 454–459 (1997).
56. Lobo. J.: COPLAS: a COnditional PLannner with Sensing Actions. In: FS-98-02, AAAI (1998).
57. Lopatenko, A., Bertossi, L.: Complexity of Consistent Query Answering in Databases Under Cardinality-Based and Incremental Repair Semantics. In: Proc. of 11th Int. Conf. on Database Theory (ICDT), pp. 179–193 (2007)
58. Makinson, D.: Screened Revision. Theoria **63**, 14–23 (1997)
59. McIlraith, S.: Representing action and state constraints in model-based diagnosis. In: Proc. of the 14th Conf. on Artificial Intelligence and 9th Conf. on Innovative Applications of Artificial Intelligence (AAAI/IAAI), pp. 43–49 (1997).
60. Moore, R.C.: A formal theory of knowledge and action. In: J.R. Hobbs, R.C. Moore (eds.) Formal Theories of the Common Sense World, pp. 319–358 (1985)
61. Nebel, B., Koehler, J.: Plan reuse versus plan generation: a theoretical and empirical analysis. Artificial Intelligence **76**(1-2), 427–454 (1995).
62. Niemelä, I., Simons, P.: Smodels: An implementation of the stable model and well-founded semantics for normal LP. In: Proc. of the 4th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR), pp. 420–429 (1997)
63. Otero, M., Otero, R.P.: Using causality and actions for diagnosis. In: Proc. of 11th Int. Workshop on Principles of Diagnosis, pp. 171–176 (2000)
64. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley, Reading, Mass. (1994)
65. Poole, D.: Representing knowledge for logic-based diagnosis. In: Proc. of the Int. Conf. on Fifth Generation Computing Systems, pp. 1282–1290 (1988)
66. Poole, D.: Normality and faults in logic-based diagnosis. In: Proc. of the 11th Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 1304–1310 (1989)
67. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence **32**(1), 57–96 (1987)
68. Rintanen. J.: Constructing conditional plans by a theorem prover. JAIR, 10:323–352 (2000).
69. Scherl, R.B., Levesque, H.J.: Knowledge, action, and the frame problem. Artificial Intelligence **144**(1-2), 1–39 (2003)
70. Selman, A.L.: A taxonomy of complexity classes of functions. Journal of Computer and System Sciences **48**(2), 357–381 (1994)
71. Scherl, R., Levesque, H.J.: Knowledge Producing Actions. In: Proc. of the 9th Int. Conf. on Knowledge Representation and Reasoning (KR), pp. 1139–1146 (1994).

72. Son, T., Tu, P., Baral, C.: Planning with sensing actions and incomplete information using logic programming. In: Proc. of the 7th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR), pp. 261–274 (2004)
73. Son, T.C., Baral, C.: Formalizing sensing actions a transition function based approach. Artificial Intelligence **125**(1-2), 19–91 (2001)
74. Son, T.C., Tu, P.H., Baral, C.: Planning with sensing actions and incomplete information using logic programming. In: Proc. of the 7th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR), pp. 261–274 (2004)
75. Son, T.C., Tu, P.H., Baral, C.: Reasoning and Planning with Sensing Actions, Incomplete Information, and Static Causal Laws using Answer Set Programming. Theory and Practice of Logic Programming **7**(4), 377–450 (2007)
76. Staworko, S., Chomicki, J., Marcinkowski, J.: Preference-Driven Querying of Inconsistent Relational Databases. In: Proc. of EDBT Workshops, pp. 318–335 (2006)
77. Struss, P., Dressler, O.: "physical negation" - integrating fault models into the general diagnosis engine. In: Proc. of the 11th Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 1318–1323 (1989)
78. Subrahmanian, V.S., Amgoud, L.: A General Framework for Reasoning about Inconsistency. In: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence (IJCAI), pp. 599–504 (2007)
79. Thielscher, M.: A theory of dynamic diagnosis. Electronic Transactions on Artificial Intelligence **1**, 73–104 (1997)
80. Wijsen, J.: Database Repairing Using Updates. ACM Transactions on Database Systems **30**(3), 722–768 (2005).