∃-ASP for Computing Repairs with Existential Ontologies*

Jean-François Baget\textsuperscript{1}, Zied Bouraoui\textsuperscript{2}, Farid Nouioua\textsuperscript{3}, Odile Papini\textsuperscript{3}, Swan Rocher\textsuperscript{4}, and Eric Würbel\textsuperscript{3}

\textsuperscript{1} INRIA Montpellier, France - baget@lirmm.fr
\textsuperscript{2} Cardiff University, UK - BouraouiZ@cardiff.ac.uk
\textsuperscript{3} Aix-Marseille Univ, France - \{farid.nouioua,odile.papini,eric.wurbel\}@univ-amu.fr
\textsuperscript{4} Univ Montpellier, France - rocher@lirmm.fr

Abstract. Repair-based techniques are a standard way of dealing with inconsistency in the context of ontology-based data access where several inconsistency-tolerant semantics have been mainly proposed for lightweight description logics. In this paper we present a generic transformation from knowledge bases expressed within existential rules formalism into an ASP program. We propose different strategies for this transformation, and highlight the ones for which answer sets of the generated program correspond to various kinds of repairs used in inconsistency-tolerant inferences.

1 Introduction

Dealing with inconsistency in ontology-based query answering is one of the challenging problems that received a lot of attention in recent years, (e.g. \cite{2,7,11,17}). In such a setting, inconsistency problem comes from the data, i.e. occurs when assertional facts contradict constraints imposed by the ontological knowledge. In case of inconsistency, standard inference is meaningless: All queries would be positively answered. In this paper we focus on the mainstream approach that considers that the ontology, built by experts, is correct, and that only data has to be repaired. Other approaches (e.g. \cite{20}) rely upon the assumption that the database is reliable but the rules are not. The latter assumption will not be explored in this paper and left for future work.

Many works (e.g. \cite{12,16,18}), basically inspired by the approaches proposed in database area (e.g. \cite{19}) and in propositional logic (e.g. \cite{8}), deal with inconsistency by proposing several inconsistency-tolerant inferences, called semantics. These semantics are based on the notion of assertional base repair which is closely related to the notion of database repair \cite{16} or maximally consistent subbase used in the propositional logic setting. An ABox repair is simply an assertional subbase which is consistent with an ontology. Ontology-based consistent query answering (AR-semantics) \cite{16} comes down first to compute the set of repairs (i.e. all possible maximally consistent subsets of facts consistent with the ontological knowledge) and then to check to which extent a query can be entailed using these repairs. As shown in \cite{16,10}, the AR-semantics (also called

* This work was supported by the projects ASPIQ (ANR-12-BS02-0003) and the ERC Starting Grant 637277.
universal entailment) is a hard task (co-NP complete) for lightweight DLs [16,19]. In
fact, inconsistency-tolerant semantics were introduced for the lightweight description
logics DL-Lite (e.g. [16]), and later extended to other description logics (e.g. [19]) or
existential rules (e.g. [17]). In this paper, we use existential rules (e.g. [5]) (also called
Datalog+/-) as ontology language that generalizes lightweight description logics, such
as DL-Lite and EL by allowing the use of any predicate arity as well as cyclic structures.

Recently the ASP framework [6], a convenient paradigm for knowledge represen-
tation and reasoning, especially when information is incomplete, has been enriched in
order to deal with existential variables [13]. ∃-ASP is a fragment of ASP that generalises
skolemized existential rules. It allows for enriching lightweight description logics with
non-monotonic features, and benefits from decidability results obtained for existential
rules. ∃-ASP has been naturally implemented on top of the ASP solver ASPeRiX1
which does not rely on preliminary grounding to compute answer sets [15,14].

The paper first recalls the logical frameworks used in this paper: Existential rules
in Section 2 ∃-ASP in Section 3 and the best known notions of repair in Section 4.
Our contribution is presented in Section 5. We present a generic transformation from
knowledge bases expressed within existential rules formalism into an ASP program. We
propose different strategies for this transformation, and highlight the ones for which
answer sets of the generated program correspond to various kinds of repairs used in
inconsistency-tolerant inferences. The sound and complete ∃-ASP algorithm which is
central to ASPeRiX computations will be used to prove the one-to-one correspondence
between the answer sets of the generated program and the knowledge base repairs.

2 Existential Rules

We consider a vocabulary V consisting of three disjoint sets, the set P of predicate
names, the set F of function symbols (each provided with an arity) and the set C of
constants. Disjoint with V, we also consider a set X of variables. In what follows,
constants will be notated in lowercase and variables in uppercase. The set of terms is
defined inductively as follows: Constants and variables alike are terms, and if f ∈ F is
a function symbol of arity k and t1, . . . , tp are terms, then f(t1, . . . , tp) is also a term.
An atom is an object of form p(t1, . . . , tk), where p is a predicate name of arity k and
the ti are terms. An atom is said basic when none of its terms involve any function
symbol, and is said grounded when no variable is used to define any of its terms. A set
of atoms is said basic (resp. grounded) when all its atoms are basic (resp. grounded).

Homomorphisms A substitution σ is a mapping σ from a set of variables to a set
of terms. If A is a set of atoms and σ is a substitution, we note σ(A) the set of
atoms obtained, for each variable x appearing both in an atom of A and the domain
of σ, by replacing non-recursively each occurrence of x in A by σ(x). For example,
let A = {p(f(X,Y),Z),q(X,a)} and σ : X → f(X,a), Y → X. Then σ(A) =
{p(f(f(X,a),X),Z),q(f(X,a),a)}. Let F and Q be two sets of atoms. A homomor-
phism from Q to F is a substitution σ such that σ(Q) ⊆ F. If we note φ(A) the first-
order logics (FOL) formula obtained by the conjunction of the atoms in A, and Φ(A)

1 available at http://www.info.univ-angers.fr/pub/claire/asperix/
the existential closure of \( \phi(A) \), it is well known that \( \Phi(F) \models \Phi(Q) \) iff there exists a homomorphism from \( Q \) to \( F \). Let \( \sigma \) be a bijective substitution from the variables of \( F \) to a fresh set of constants (that appear neither in \( F \) nor in \( Q \)). The ground set of atoms \( \sigma(F) \) is called a grounding of \( F \) and it holds that \( \Phi(F) \models \Phi(Q) \) iff \( \Phi(\sigma(F)) \models \Phi(Q) \).

**Existential Rules**  
An existential rule is of form \( B \rightarrow H \) where both the body \( B \) and the head \( H \) are sets of basic atoms. We often note such a rule \( B[X,Y] \rightarrow H[Y,Z] \), where the variables in \( X \) are those that appear only in the body, the variables in \( Y \) (called the frontier) are those that appear both in the body and the head, and those in \( Z \) (called existential variables) are those that appear only in the head. The FOL formula associated with this existential rule is \( \forall X \forall Y (\phi(B) \rightarrow (\exists Z \phi(H))) \). For example, the FOL formula associated with \( p(X,Y), r(X,Y',a) \rightarrow r(Y,Y',Z) \), \( p(Z,Z') \) is \( \forall X \forall Y \forall Y' (p(X,Y) \land r(X,Y',a) \rightarrow (\exists Z \exists Z' r(Y,Y',Z) \land p(Z,Z'))) \). Let \( R = B \rightarrow H \) be a rule with frontier \( Y \) and existential variables \( Z \). Let us consider a substitution \( \sigma_R \) that maps each existential variable \( Z \in Z \) to a functional term \( f^R_z(Y) \). Then we say that \( sk(R) = B \rightarrow \sigma_R(H) \) is a skolemization of \( R \). Let \( R \) be the rule given in the example, then \( sk(R) = p(X,Y), r(X,Y',a) \rightarrow r(Y,Y',f^R_z(Y,Y'),p(f^R_z(Y,Y'),f^R_z(Y,Y'))) \).

**Derivations**  
Consider now a set of atoms \( B \) and a skolemized existential rule \( R = B \rightarrow H \). We say that \( R \) is applicable to \( F \) when there exists a homomorphism \( \sigma \) from \( B \) to \( F \). In that case, the application of \( R \) on \( F \) according to \( \sigma \) produces a set of atoms \( \alpha(F,R,\sigma) = F \cup \sigma(H) \). Note that when \( F \) is ground, \( \alpha(F,R,\sigma) \) is also ground. Let \( R \) be the rule given in the previous example, and \( F = p(a,g(b)), r(a,g(b),a) \). The substitution \( \sigma : X \mapsto a, Y \mapsto g(b), Y' \mapsto g(b) \) is a homomorphism from \( B \) to \( F \) and \( \alpha(F,R,\sigma) = F \cup \{r(g(b),g(b),f^R_z(g(b),g(b))), p(f^R_z(g(b),g(b)), f^R_z(g(b),g(b)))\} \).

Let \( F \) be a set of atoms and \( \mathcal{R} \) be a set of rules. A \( \mathcal{R} \)-derivation from \( F \) is a (possibly infinite) sequence \( F = F_0, F_1, \ldots, F_i, \ldots \) such that, for any \( i > 0 \), there exists a rule \( R = B \rightarrow H \in \mathcal{R} \) and an homomorphism \( \sigma \) from \( B \) to \( F_{i-1} \) such that \( F_i = \alpha(F_{i-1}, R, \sigma) \). The result of a finite derivation \( F_0, \ldots, F_k \) is the set of atoms \( F_k \), when it is infinite we define it as the (infinite) union of all \( F_i \). A derivation is said full when, for every rule \( R = B \rightarrow H \in \mathcal{R} \), for every homomorphism \( \sigma \) from \( B \) to its result, there exists some \( F_i \) in the derivation such that \( F_{i+1} = \alpha(F_i, R, \sigma) \). Any full \( \mathcal{R} \)-derivation on \( F \) produces the same result, and we call that result the \( \mathcal{R} \)-closure of \( F \) and note it \( Cl_\mathcal{R}(F) \) (or simply \( Cl(F) \) when there is no ambiguity on \( \mathcal{R} \)). When we consider a set \( \Pi = F \cup \mathcal{R} \) as a program (as in Section 3), we note \( Cl(\Pi) = Cl_\mathcal{R}(F) \).

**Theorem 1.** Let \( F \) and \( Q \) be two set of atoms, and \( \mathcal{R} \) be a set of existential rules. We note \( F_g \) a grounding of \( F \) and \( \mathcal{R}_{sk} \) the skolemization of \( \mathcal{R} \). Then \( Cl_{\mathcal{R}_{sk}}(F_g) \) is a universal model, i.e., \( F, \mathcal{R} \models Q \) iff there is a homomorphism from \( Q \) to \( Cl_{\mathcal{R}_{sk}}(F_g) \).

**Skolem Chase**  
Deciding whether or not \( F, \mathcal{R} \models Q \) is undecidable. However, for all positive instances of the problem, a homomorphism from \( Q \) to \( Cl_{\mathcal{R}_{sk}}(F_g) \) can be found after finitely many steps of a breadth first derivation. Such a derivation is called the skolem chase. For a more precise relationship between the skolem chase and other chases found in the literature, the reader can refer to [4]. A lot of work has been devoted to predicting that the chase will stop. Acyclicity conditions on a set of existential rules such as the ones presented in [3] ensure that the closure \( Cl_{\mathcal{R}_{sk}}(F_g) \) will be finite.
3 Existential ASP

Syntax An existential ASP (∃-ASP) rule is of form $H \leftarrow B^+, \neg B_1^-, \ldots, \neg B_k^-$, where the positive body $B^+$, the negative bodies $B_i^-$ and the head $H$ are sets of basic atoms. Intuitively, such a rule means "if the positive body is verified, and none of the negative bodies are, then we can conclude with the head". To make our definitions easier to read, and without loss of generality (see the safety condition in [13]), we consider that all variables appearing in negative bodies also appear in the positive body.

∀-ASP program is a set $\Pi_F$ of basic atoms and a set $\Pi_R$ of ∃-ASP rules. As for existential rules, we can skolemize ∃-ASP rules respecting the safety condition as follows: The skolemization of the previous rule results in $\sigma(H) \leftarrow B^+, \neg B_1^-, \ldots, \neg B_k^-$, where $\sigma(H) \leftarrow B^+$ is the skolemization of $H \leftarrow B^+$, as defined for existential rules. The skolemization of an ∃-ASP program is defined by the grounding of $\Pi_F$ and the skolemization of $\Pi_R$. For example, let $r(X,Z) \leftarrow p(X,Y), \neg q(X), \neg (r(Y,a),r(a,b))$ be an existential ASP rule. Its skolemization is $r(X,f^G_1(X)) \leftarrow p(X,Y), \neg q(X), \neg (r(Y,a),r(a,b))$.

Note that the skolemization of an existential ASP program (without function symbol) is a standard ASP program with function symbols.

Semantics In what follows we consider $\Pi$ an ASP program obtained from a skolemized existential ASP program. Let $C_H$ be the set of constants appearing in $\Pi$ and $F_H$ be the set of function symbols appearing in $\Pi$. The Herbrand domain of $\Pi$ is the minimal set of ground terms $H_H$ such that $C_H \subseteq H_H$ and, if $f \in F_H$ is a function symbol of arity $k$ and $h_1, \ldots, h_k$ are in $H_H$, then $f(h_1, \ldots, h_k)$ is also in $H_H$. If $R = H \leftarrow B^+, \neg B_1^-, \ldots, \neg B_k^-$ is a rule in $\Pi$ and $\sigma$ is a substitution from all its variables in to $H_H$, then the rule $\sigma(R) = \sigma(H) \leftarrow \sigma(B^+), \neg \sigma(B_1^-), \ldots, \neg \sigma(B_k^-)$ is a grounding of $R$. The grounding of a program $\Pi$ is the program obtained from all possible groundings of all rules in $\Pi$. Not that the Herbrand domain (and thus the grounding) of a finite $\Pi$ is infinite as soon as $\Pi$ contains a constant and a predicate symbol of arity $\geq 1$. Let us now consider the grounding $\Pi^G_H$ of $\Pi$ and a (possibly infinite) set of ground atoms $E$. The reduct of $\Pi^G_H$ with respect to $E$, denoted $\Pi^G_{H|E}$, is the minimal set that contains all (ground) atoms of $\Pi^G_H$ and, for each skolemized ∃-ASP rule $R = H \leftarrow B^+, \neg B_1^-, \ldots, \neg B_k^-$ in $\Pi^G_H$, if there is no $B_i^-$ such that $B_i^- \subseteq E$, then $H \leftarrow B^+$ (called the positive part of $R$) is a skolemized existential rule of $\Pi^G_{H|E}$.

Finally, $E$ is an answer set (stable model) of $\Pi$ when $E = Cl(\Pi^G_{H|E})$. We define the answer sets of an existential ASP program as the answer sets of its skolemization. Note that it is not a neutral choice, for a semantic point of view (see the discussion in [4] where using different chases can lead to different semantics and different answer sets).

Computation Given an ASP program $\Pi$, most solvers rely upon a 2-step algorithm that first compute the grounding $\Pi^G_H$ of $\Pi$, then use $\Pi^G_H$ to build an answer set $E$ (using for instance a SAT solver). However, the grounding becomes infinite as soon as function symbols (such as the ones obtained from our skolemization) are involved. Some solvers can try to extract from the grounding rules that have no chance to be involved in the second step, but doing that optimally would require to compute that second step, making
Now we say that \((\text{case nor} B)\) if there exists a negative body \(B\) in \(R\) such that there exists a homomorphism \(\sigma\) from \(B\) to \(\text{IN}(n)\) and \((R, \sigma)\) has not already been evaluated on \(n\) nor on any of its ancestors. Now we say that \((R, \sigma)\) is evaluated on \(n\) and there is 3 possible outcomes. Blocked case: If there exists a negative body \(B_i\) in \(R\) such that \(\sigma(B_i) \subseteq \text{IN}(n)\), meaning that one of the negative bodies appears in \(\text{IN}(n)\), then this step produces nothing (but marks this evaluation as done). Positive case: If \(R = H \leftarrow B^+\) contains no negative body, then we update \(\text{IN}(n)\) with the result of the rule application, and do not change \(\text{OUT}\) nor \(\text{MBT}\). Then \(\text{IN}(n) = \alpha(\text{IN}(n), R, \sigma) = \text{IN}(n) \cup \sigma(H)\). Choice case: otherwise we create two children \(n_1\) and \(n_2\) of \(n\). In \(n_1\) we effectively apply the rule and forbid its negative bodies to appear in the final result, in \(n_2\) we must prove that we have the right not to apply it by finding one of the negative bodies in the final result. Then \(\text{IN}(n_1) = \alpha(\text{IN}(n), R, \sigma) \cap \text{IN}(n) \cup \sigma(H), \text{OUT}(n_1)\) is the set of sets of atoms whose elements are those of \(\text{OUT}(n)\) and the \(k\) sets of atoms \(\sigma(B_i^-)\), for \(1 \leq i \leq k\), \(\text{MBT}(n_1) = \text{MBT}(n)\) and \(\text{IN}(n_2) = \text{IN}(n), \text{OUT}(n_2) = \text{OUT}(n),\) and \(\text{MBT}(n_2)\) is the set of disjunctions of sets of atoms whose elements are those of \(\text{MBT}(n)\) and the disjunction \(\bigvee_{1 \leq i \leq k} \sigma(B_i^-)\).

Consider a (possibly infinite) branch of this tree. Similarly to what was done for derivations, we define the result of that branch as the (possibly infinite) union, for all nodes \(n\) in that branch, of the \(\text{IN}(n)\). When such a branch is finite, its result is \(\text{IN}(l)\), where \(l\) is the leaf of the branch. A branch is said full when, for every rule \(R\) and every homomorphism \(\sigma\) from \(B^+\) to the result of the branch, \((R, \sigma)\) has been evaluated on some node of the branch. If \(n\) is a node of a branch and \(B\) is a set of atoms, we say that \(B\) satisfies \(\text{OUT}(n)\) when, for every set of atoms \(O \subseteq \text{OUT}(n)\), \(O \subseteq B\). In the same way, we say that \(B\) satisfies \(\text{MBT}(n)\) when, for every disjunction \(M_1 \lor \ldots \lor M_k \in \text{MBT}(n)\), there exists a \(M_i\) such that \(M_i \subseteq B\). A branch is said \(\text{OUT-valid}\) (resp. \(\text{MBT-valid}\)) when its result satisfies \(\text{OUT}(n)\) (resp. \(\text{MBT}(n)\)) for every node \(n\) in the branch. A branch that is both \(\text{OUT-valid}\) and \(\text{MBT-valid}\) is said \(\text{valid}\).

**Theorem 2.** Let \(\Pi\) be a skolemized existential ASP program. Then \(A\) is an answer set of \(\Pi\) iff \(A\) is the result of a full valid branch in the computation of \(\Pi\).

**Properties** It is first important to note that, when the positive part of rules satisfy the acyclicity conditions presented in [4], then the computation produces a finite tree. In that case, validity of a branch with leaf \(l\) admits a simpler characterization: A branch is \(\text{OUT-valid}\) (resp. \(\text{MBT-valid}\)) when \(\text{IN}(l)\) satisfies \(\text{OUT}(l)\) (resp. \(\text{MBT}(l)\)).
Then we point out the monotonic increase of the field \textsc{in}: If a node \( n' \) is a descendant of a \( n \), then \( \textsc{in}(n) \subseteq \textsc{in}(n') \). It follows that if there is a node \( n \) such that \( \textsc{in}(n) \) does not satisfy \( \textsc{out}(n) \), then no branch containing \( n \) is \( \textsc{out} \)-valid, so we can cut the development of the computation tree for node \( n \). Such an optimization is more difficult to achieve using the \textsc{mbt} field, to stop the development of the computation tree for node \( n \), we have to prove that there exists a disjunction \( M_1 \lor \ldots \lor M_k \in \textsc{mbt}(n) \) and a set of atoms \( M_i \) that will never be contained in the \textsc{in} field of any descendant of \( n \). Simple arguments achieve that goal in the ASP programs we generate in Section 5.

4 The Notion of Repair

We now recall the definitions of repairs [11,16,10] rephrased within the framework of existential rules. Let \( \mathcal{K} = (F, \mathcal{R}, \mathcal{N}) \) be a knowledge base where \( F \) is a set of ground atoms, \( \mathcal{R} \) is a set of existential rules, and \( \mathcal{N} \) is a set of negative constraints, i.e. a set of rules of form \( \bot \leftarrow B \) where \( B \) is a set of basic atoms and \( \bot \) is the absurd symbol. We say that a set of atoms \( Y \) is consistent w.r.t. \((\mathcal{R}, \mathcal{N})\) when \((F, \mathcal{R}, \mathcal{N}) \not\models \bot\), i.e. when \( \text{cl}(Y, \mathcal{R} \cup \mathcal{N}) \) does not contain \( \bot \). Our knowledge base is thus consistent when \( F \) is consistent w.r.t. \((\mathcal{R}, \mathcal{N})\). Different kind of repairs can be considered when the knowledge base is inconsistent. (Standard) repairs: A repair of \( \mathcal{K} \) is an inclusion-maximal subset \( F' \) of \( F \) that is consistent w.r.t. \((\mathcal{R}, \mathcal{N})\), and we note \( F' \in R(\mathcal{K}) \). Closed repairs: If \( X \) is a set of atoms, we call ground positive closure of \( X \) and note \( g^+\text{cl}(X) \) the restriction of \( \text{cl}(X, \mathcal{R}) \) to basic ground atoms (whose terms are only constants, and not obtained with function symbols). A closed repair of \( \mathcal{K} \) is a set of basic ground atoms \( F'' = g^+\text{cl}(F') \), where \( F' \) is a standard repair of \( \mathcal{K} \), and we note \( F'' \in CR(\mathcal{K}) \). Repairs of closure: A repair of the closure of \( \mathcal{K} \) is a standard repair \( F' \) of \((g^+\text{cl}(F, \mathcal{R}), \mathcal{R}, \mathcal{N})\), and we note \( F' \in RC(\mathcal{K}) \).

Recently a unified framework combining modifiers (way of computing the repairs) and inferences strategies has been proposed for querying ontological knowledge bases represented with existential rules [2]. This framework covers the best known semantics and introduces new ones. The semantics are denoted by \( \langle \phi_i, s \rangle \) where \( \phi_i \) is a modifier and \( s \in \{\forall, \exists, \cap, \text{maj} \} \) is an inference strategy. Within this framework \( \phi_1 \) computes the set of repairs, \( \phi_5 \) computes the closed repairs and \( \phi_7 \) computes the repairs of the closure.

5 Computing Repairs with \( \exists \)-ASP

In this section we describe the transformation from a knowledge base \( \mathcal{K} \) into a generic \( \exists \)-ASP program \( \Pi \). Though this program computes “repairs” in the broad sense, two configurable modules (namely selection and display) are used to obtain the intended behaviour. In particular, we show that, given specific rules, this program can compute the repairs, the closed repairs or the repairs of the closure of \( \mathcal{K} \). This transformation relies upon the following steps: 1) \( \mathcal{K} \) is put into its skolemized form, 2) the user selects either the select or the display transformation scheme, 3) the transformation builds the program \( \Pi \), using an extended vocabulary, 4) we use an ASP solver to compute the...
answer sets of $I$, 5) the restriction of those answer sets to the original vocabulary provides the “repairs”.

5.1 Transformation Into $\exists$-ASP

Our knowledge base is built upon an original vocabulary $\mathcal{V}$. For every predicate name $p \in \mathcal{V}$, we consider different versions of $p$ that will be used in the extended vocabulary of our $\exists$-ASP program: $p_i$ for initial predicate, $p_p$ for possible predicate, $p_n$ for forbidden predicate, $p_c$ for chosen predicate, $p_s$ for may be selected predicate, $p_v$ for valid predicate, $p_g$ for ground predicate, and $p_d$ for display predicate. If $A$ is a set of atoms built upon the original vocabulary, we note $A_x$ the set of atoms $p_x(t)$ built upon the extended vocabulary where $p(t)$ is an atom of $A$. The $\exists$-ASP program $I$ is obtained as follows:

**Encoding of initial facts**: $I$ contains $F_i$ (every atom of $F$ is considered as an initial fact of the program $I$).

**Encoding of positive closure**: For every predicate name in $\mathcal{V}$, we have a rule of form $[P_{1i}: p_p(X) \leftarrow p_i(X)]$, those rules assert that every initial atom is possible; and for every rule $E(X) \rightarrow H(X, Y)$ in $\mathcal{R}_{sk}$, we have a rule of form $[R_{1i}: H_p(X, Y), fct(Y_1), \cdots, fct(Y_k) \leftarrow B_p(X)]$ where the $Y_i$ are the functional terms of the head of the skolemized rule, those rules are used to encode the positive closure $Cl(F, \mathcal{R})$ with possible atoms, and to “mark” functional terms. Finally, for every predicate name $p \in \mathcal{V}$, we have a rule of form $[P_{2i}: p_g(X) \leftarrow p_p(X), not fct(X_1), \cdots, not fct(X_k)]$ asserting that every possible atom using no functional term is ground.

**Selection strategy**: Those configurable rules provide the user strategy to define which atoms (of form $p_s$) are selectable, i.e. can appear or not in the “repairs”. We provide here two such strategies: $SEL1$ says that every initial atom is selectable. For every predicate name $p \in \mathcal{V}$, we have a rule $[S_{1i}: p_s(X) \leftarrow p_i(X)]$. $SEL2$ says that every ground possible atom is selectable. For every predicate name $p \in \mathcal{V}$, we have a rule $[S_{2i}: p_s(X) \leftarrow p_p(X)]$.

**Choice rules**: These rules are the core of our program, since they will build all possible subsets of selectable atoms. They say that every atom that is selectable and not forbidden must be chosen. $[P_{3i}: p_c(X) \leftarrow p_s(X), not p_n(X)]$.

**Definition of contexts**: For every atom $p(t)$, the atom $p_v(t, c)$ asserts that $p(t)$ is valid in the context $c$. All chosen atoms are valid in the base context. This is encoded, for each predicate name $p \in \mathcal{V}$, by the rule $[P_{4i}: p_v(X, base) \leftarrow p_c(X)]$. An atom $p(t)$ that is not chosen will be valid in its own context, encoded by the term $ctx(p, t)$. This is encoded, for each predicate name $p \in \mathcal{V}$, by the rule $[P_{5i}: p_v(X, ctx(p, X)), context(ctx(p, X)) \leftarrow p_s(X), not p_c(X)]$. Finally, we say that every atom valid in the base context is also valid in any other context. For each predicate name $p \in \mathcal{V}$, we have the rule $[P_{6i}: p_v(X, C) \leftarrow p_v(X, base), context(C)]$. The base context encodes the chosen atoms. Every other context encodes the adding of one particular unchosen atom to the already chosen ones. Intuitively, to obtain a repair we will have to prove that the base context is consistent and that all other contexts are not, meaning that the base context is maximal.

**Context closure**: Every atom that can be deduced from those valid in a particular context will also be valid in that context. For every skolemized existential rule of the
form $B(X) \rightarrow H(X)$, we obtain the rule $[R_2:] H_v(X, C) \leftarrow B_v(X, C)$. Then we say that if a constraint is violated in a given context, then that context is absurd. For any constraint in $\mathcal{N}$ of the form $p^1(X_1), \ldots, p^k(X_k) \rightarrow \bot$ we add the rule of form $[C_1:]$ absurd$(C) \leftarrow p^1_v(X_1, C), \ldots, p^k_v(X_k, C)$.

**Retropropagation of absurd contexts** : Finally, we say that if the base context is absurd, then every atom valid in that context is forbidden. For every predicate $p \in \mathcal{V}$, we have the rule $[C_2:] p_v(X) \leftarrow p_v(X), \text{absurd}(base)$. For other absurd contexts, only selected unchosen atoms of that specific context are forbidden. For every predicate $p \in \mathcal{V}$, we have the rule $[C_3:]$ $p_v(X) \leftarrow \text{not} p_v(X), p_s(X), p_v(X, C), \text{context}(C)$, absurd$(C)$.

**Visualization strategy** : Those configurable rules provide the user strategy to define which atoms (of form $p_d$) are displayable, i.e. can appear or not in the visualization of the “repairs”. Whatever the strategy chosen, only displayable atoms that are valid in the base context will be displayed (i.e. added using the original vocabulary). This is encoded, for each predicate name $p \in \mathcal{V}$, by the rule $[D:]$ $p_d(X) \leftarrow p_v(X), p_v(X, \text{base})$. We provide here two such strategies: DISP1 says that every initial atom is displayable. For every predicate name $p \in \mathcal{V}$, we have a rule $[V_1:]$ $p_d(X) \leftarrow p_v(X)$. DISP2 says that every ground possible atom is displayable. For every predicate name $p \in \mathcal{V}$, we have a rule $[V_2:]$ $p_d(X) \leftarrow p_g(X)$.

**Example 1.** Let $\mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base such that $F = \{p(a), q(a)\}$, $\mathcal{R}_{st}$ = $\{p(X) \rightarrow r(X, f(X)), q(X) \rightarrow s(X), r(X, Y) \rightarrow t(X)\}$ and $\mathcal{N}$ = $\{r(X, Y), q(X) \rightarrow \bot\}$. The original vocabulary of $\mathcal{K}$ contains the predicate names $\{p, q, r, v\}$.

The **initial facts** are $p_i(a)$ and $q_i(a)$.

The rules encoding the **positive closure** are those of form $P_1$ for initialization (we restricted those to the predicates appearing in initial form): $p_p(X) \leftarrow p_i(X)$ and $q_p(X) \leftarrow q_i(X)$, those of form $R_1$ for propagation: $r_p(X, f(X))$, $fct(f(X)) \leftarrow p_p(X)$. $s_p(X) \leftarrow q_p(X)$. $t_p(X) \leftarrow r_p(X, Y)$ and those of form $P_2$ to detect ground atoms: $p_g(X) \leftarrow p_p(X)$, $not$ $fct(X)$. $q_g(X) \leftarrow q_p(X)$, $not$ $fct(X)$. $r_g(X, Y) \leftarrow r_p(X, Y)$, $not$ $fct(X)$, $not$ $fct(Y)$, $s_g(X) \leftarrow s_p(X)$. $not$ $fct(X)$. $t_g(X) \leftarrow t_p(X)$, $not$ $fct(X)$.

Two **selection strategies** are possible. With SEL1 we have: $p_s(X) \leftarrow p_i(X)$, and $q_s(X) \leftarrow q_i(X)$. With SEL2 we have: $p_s(X) \leftarrow p_g(X)$. $q_s(X) \leftarrow q_g(X)$. $r_s(X, Y) \leftarrow r_g(X, Y)$, $s_s(X) \leftarrow s_g(X)$, and $t_s(X) \leftarrow t_g(X)$.

The **choice rules** are: $p_c(X) \leftarrow p_s(X)$, $not$ $p_n(X)$. $q_c(X) \leftarrow q_s(X)$, $not$ $q_n(X)$. $r_c(X, Y) \leftarrow r_s(X, Y)$, $not$ $r_n(X, Y)$. $s_c(X) \leftarrow s_s(X)$, $not$ $s_n(X)$. $t_c(X) \leftarrow t_s(X)$, $not$ $t_n(X)$.

For the **definition of contexts**, we have the rules of form $P_3$ defining the base context: $p_c(X, \text{base}) \leftarrow p_c(X)$. $q_c(X, \text{base}) \leftarrow q_c(X)$. $r_c(X, Y, \text{base}) \leftarrow r_c(X, Y)$. $s_c(X, \text{base}) \leftarrow s_c(X)$. $t_c(X, \text{base}) \leftarrow t_c(X)$. the rules of form $P_5$ defining other contexts: $p_c(X, \text{ctx}(p, X)) \leftarrow p_c(X)$, $not$ $p_n(X)$. $q_c(X, \text{ctx}(q, X)) \leftarrow q_c(X)$, $not$ $q_n(X)$. $r_c(X, Y, \text{ctx}(r, X, Y)) \leftarrow r_c(X, Y)$, $not$ $r_n(X, Y)$. $s_c(X, \text{ctx}(s, X)) \leftarrow s_c(X)$, $not$ $s_n(X)$. $t_c(X, \text{ctx}(t, X)) \leftarrow t_c(X)$, $not$ $t_n(X)$ and the rules of form $P_6$ encoding inheritance of base context: $p_b(X, C) \leftarrow p_v(X, \text{base})$. $q_b(X, C) \leftarrow q_v(X, \text{base})$. $r_b(X, Y, C) \leftarrow r_v(X, Y, \text{base})$. $s_b(X, C)$
The context closure will be computed with the rules of form \( R_2: r_v(X, f(X), C) \leftarrow p_v(X, C). s_v(X, C) \leftarrow q_v(X, C). t_v(X, C) \leftarrow r_v(Y, C). \) and inconsistencies will be detected by the rule of form \( C_1: absurd(C) \leftarrow r_v(Y, C, q_v(X, C). \)

Retropropagation of absurd contexts is handled by rules of form \( C_2: p_n(X) \leftarrow p_v(X). absurd(base). q_n(X) \leftarrow q_v(X). absurd(base). r_n(X, Y) \leftarrow r_v(X, Y). absurd(base). s_n(X) \leftarrow s_v(X). absurd(base). t_n(X) \leftarrow t_v(X). absurd(base). \) and \( C_3: p_n(X) \leftarrow not p_v(X). p_s(X). p_v(X, C). context(C). absurd(C). q_n(X) \leftarrow not q_v(X). q_s(X). q_v(X, C). context(C). absurd(C). r_n(X, Y) \leftarrow not r_v(X, Y). r_s(X, Y). r_v(X, Y, C). context(C). absurd(C). s_n(X) \leftarrow not s_v(X). s_s(X). s_v(X, C). context(C). absurd(C). t_n(X) \leftarrow not t_v(X). t_s(X). t_v(X, C). context(C). absurd(C). \)

Finally, display rules contain the rules of form \( D: p(X) \leftarrow p_d(X). p_v(X, base). q(X) \leftarrow q_d(X). q_v(X, base). r(X, Y) \leftarrow r_d(X, Y). r_v(X, Y, base). s(X) \leftarrow s_d(X). s_v(X, base). t(X) \leftarrow t_d(X). t_v(X, base). \) And the choice of strategy DISP1 with rules: \( p_d(X) \leftarrow p(X). q_d(X) \leftarrow q(X). r_d(X, Y) \leftarrow r(X, Y). s_d(X) \leftarrow s(X). t_d(X) \leftarrow t(X). \) or of strategy DISP2 with rules: \( p_d(X) \leftarrow p_g(X). q_d(X) \leftarrow q_g(X). r_d(X, Y) \leftarrow r_g(X, Y). s_d(X) \leftarrow s_g(X). t_d(X) \leftarrow t_g(X). \)

It is important to note that when the skolem chase halts for the original existential rules KB (such fragments have been studied for instance in [4]) then the Skolem chase also halts on the positive part of the generated ASP program, and thus (see properties in Section [3]) the ASPeRiX computation generates all answer sets in finite time.

5.2 General Form of the Computation Tree of \( \Pi \)

Let us now examine what is happening during a computation of such a program \( \Pi \). We first point out that we can evaluate rules in a particular order: 1) the positive closure rules of form \( P_1 \), 2) those of form \( R_1 \), 3) those of form \( P_2 \), 4) the selection rules, 5) the choice rules \( P_3 \), 6) the definitions of contexts of form \( P_4 \), 7) those of form \( P_5 \), 8) those of form \( P_6 \), 9) the context closure of form \( R_2 \), 10) and those of form \( C_1 \), 11) the retropropagation rules \( C_2 \) and 12) \( C_3 \), and 13) the visualisation rules. Indeed, we can check that, if \( i < j \) are two of those steps, no rule evaluated at step \( j \) can trigger a new application of a rule that was evaluated at step \( i \). This will not always be the case with any selection rules provided by the user, but this property is satisfied by the strategies SEL1 and SEL2 presented here. Among all equivalent computation trees, we will thus consider those that respect that particular order: The natural computations of \( \Pi \).

**Proposition 1.** Let \( \mathcal{K} = (\mathcal{F}, \mathcal{R}, \mathcal{N}) \) be a knowledge base, and let \( \Pi \) be the \( \exists \)-ASP program obtained from the above encoding. At the end of Step 3 the natural computation tree corresponding to \( \Pi \) only has one finite branch that could lead to a full valid branch.

**Proof.** The computation of \( \Pi \) is a binary tree. Initially the root is s.t \( IN(root) = \mathcal{F}_1 \), \( OUT(root) = \emptyset, MBT(root) = \emptyset \). After \( \mathcal{F}_1 \) applications of the rule \( P_1, IN(root) = \mathcal{F}_1 \cup \mathcal{F}_p \), since the rule \( P_1 \) is positive (the negative body of \( P_1 \) is empty) \( OUT(root) \) and \( MBT(root) \) are unchanged. (In the following in case of positive rule we do not specify that the fields \( OUT \) and \( MBT \) do not change.) After a possible infinite number of applications of the rule \( R_1, IN(root) = \mathcal{F}_1 \cup (Cl(\mathcal{F}, R))_p \cup \{fact(t) \} \) basic terms of
We develop the computation tree using the rule $P_2$, starting from the root, for each node $n$ we look for a homomorphism $\sigma$ in $IN(root)$ s.t $\sigma(X_i)=t_i$ where $t_i$ is a grounded term. Two cases hold:

- case 1: $\exists t_i$ such that $fct(t_i) \in IN(root)$. This is the blocked case of the computation tree given in Section 3. The node is not changed.
- case 2: $\forall t_i$ such that $fct(t_i) \in IN(root)$. This is the choice case in the computation tree given in Section 3. The node $n$ has two children $n_1$ and $n_2$ such that $IN(n_1) = IN(n) \cup \{p_1(t_1, \ldots , t_k)\}$, $OUT(n_1) = OUT(n) \cup \{\{fct(t_1)\}$, $\ldots$, $\{fct(t_k)\}\}$, $MBT(n_1) = MBT(n)$ and $IN(n_2) = IN(n)$, $OUT(n_2) = OUT(n)$, $MBT(n_2) = MBT(n) \cup \{fct(t_1) \lor \ldots \lor fct(t_k)\}$.

Note that we get all $fct(t_i)$ that could be generated and there will be no other way to obtain others. According to the properties in Section 3 none of the $fct(t_i)$ in $MBT(n_2)$ can be proved therefore this branch cannot lead to a valid branch. At the end of Step 3, the computation tree only has one branch that could lead to a valid branch and therefore to an answer set. Since there is a finite number of atoms without function symbol, this only branch is finite and $l$ denotes its leaf and $IN(l) = F \cup (Cl(F, R)) \cup \{fct(t)\}$ where $t$ is a functional term of $Cl(F, R)$ or $\{a \in Cl(F, R) | a$ is a basic atom}\}, $OUT(l) = \{\{fct(t)\}$ where $t$ is a functional term of $Cl(F, R)$ and $MBT(l) = MBT(n)$. As no further development of the computation tree can add any atom with predicate name $fct(t)$, the result of any branch having the node $l$ as ancestor will satisfy $OUT(l)$. Thus, in the following, we will ignore $OUT(l)$.

**Example 2.** (Example 1 continued) At the end of Step 3 the computation tree has only one branch and $l$ denotes its leaf. We have $IN(l) = \{p_1(a), q_1(a), p_2(a), q_2(a), r_1(a, f(a)), fct(f(a)), r_2(a, t_1(a), p_3(a), q_3(a), s_2(a), t_2(a))\}$, $OUT(l) = \{\{fct(a)\}$ and $MBT(l) = \{fct(f(a))\}$. Note that this branch may lead to a full valid branch since $IN(l)$ satisfies $OUT(l)$ and $IN(l)$ satisfies $MBT(l)$.

**Proposition 2.** Let $K=(F, R, N)$ be a knowledge base, and let $\Pi$ be the 3-ASP program obtained from the above encoding. Let $X$ be the finite set of selectable atoms obtained after Step 4. At the end of Step 5 the natural computation tree corresponding to $\Pi$ has $2^{|X|}$ finite branches (each one determined by the subset $Y$ of the chosen atoms in $X$).

**Proof.** As shown in Proposition 1 the computation tree corresponding to $\Pi$ obtained at the end of Step 3 only has one finite branch and $l$ denotes its leaf. We start from $l$ where $IN(l)$, $OUT(l)$ and $MBT(l)$ are given at the end of the proof of Proposition 1. Step 4 proposes two strategies for selecting the predicates, in order to handle both cases, we consider the set of atoms $X$ provided by the selection rules and the field $IN$ is updated with $X$. Thanks to the proposed selection rules $X$ is always finite. The application of the rules $P_3$ leads to the development of $2^{|X|}$ sub-branches from $l$, each one encoding a subset $Y \subseteq X$. The branch associated with $Y$ has a leaf denoted by $l_Y$ such that $IN(l_Y) = IN(l) \cup Y$ denoted by $INY$, $OUT(l_Y) = OUT(l) \cup \{p_n(t) \mid p_n(t) \in Y\}$ and $MBT(l_Y) = MBT(l) \cup \{(X \setminus Y)_n\}$. 


Proposition 3. Let $K = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base, and let $\Pi$ be the $\exists$-ASP program obtained from the above encoding. Let $l_{Y_3}$ be the leaf of a branch obtained after Step 5 of the natural computation tree. Then $l_{Y_3}$ can lead to at most one valid full branch, which is finite.

Proof. We now consider the development of the computation tree from $l_{Y_3}$. The application of the rules $P_2$ introduces the base context and since they are positive only the field $IN$ is updated, thus $IN(l_{Y_3}) = IN1 \cup \{p_0(t, \text{base}) \mid p(t) \in Y\}$. The rules $P_5$ introduce the contexts different from the base context. These are choice rules however like in the case of rules $P_2$, no other application of rules can generate chosen predicates $(p_i(t))$ therefore there is only one branch that can eventually lead to a valid branch and $l_{Y_3}$ denotes its leaf. Note that this branch is finite because $X$ is finite. Thus $IN(l_{Y_3}) = IN(l_{Y_3}) \cup \{p_0(t, ctx(p,t)) \mid p(t) \in X \cup \{\text{context}(ctx(p,t)), \mid p(t) \in X \} \cup \{\text{context}(ctx(p,t)), \mid p(t) \in X \}$, denoted by $INY2$, the fields $OUT$ and $MBT$ are unchanged. The application of the rules $P_6$ updates the field $IN$, thus $IN(l_{Y_3}) = INY2 \cup \{p_0(t,c) \mid p(t) \in Y\}$, denoted by $NY3$, where $c$ is a constant different from $base$. The application of rules $R_2$ updates the field $IN$, thus $IN(l_{Y_3}) = INY3 \cup \{p_0(t, base) \mid p(t) \in Cl(Y,R) \} \cup \{p_0(t,c) \mid c = ctx(q,u), c \neq base \text{ and } p(t) \in Cl(Y \cup \{q(u)\}, R)\}$, denotes $INY4$. The application of the rules $C_1$ updates the field $IN$, thus $IN(l_{Y_3}) = INY4 \cup \{\text{absurd}(base) \mid Cl(Y,R) \text{ violates a constraint} \} \cup \{\text{absurd}(c) \mid c = ctx(q,u), c \neq base \text{ and } Cl(Y \cup \{q(u)\}, R) \text{ violates a constraint} \}$, denoted by $INY5$. The application of the rules $C_2$ updates the field $IN$, thus $IN(l_{Y_3}) = INY5 \cup \{p_0(t,c) \mid p(t) \in \mathcal{F}\}$ if $Cl(Y,R)$ violates a constraint or $IN(l_{Y_3}) = INY5$ otherwise. The rules $C_3$ introduce the forbidden predicates These are choice rules however like in the case of rules $P_2$, no other application of rules can generate chosen predicates $(p_i(t))$ therefore there is only one branch that can eventually lead to a valid branch. $l_{Y_3}$ denotes its leaf. Note that this branch is finite because $X$ is finite. Thus $IN(l_{Y_3}) = IN(l_{Y_3}) \cup \{p_0(t, base) \mid Cl(Y \cup p_i(t), R) \text{ violates a constraint} \}$ denoted by $INY6$, the fields $OUT$ and $MBT$ are unchanged. So by Step 13 proposes two strategies for visualizing the predicates, with the strategy $\text{DISP1}$ the field $IN$ is updated such that $IN(l_{Y_3}) = INY6 \cup \{p_6(t) \mid p_i(t) \in \mathcal{F}\}$, while the strategy $\text{DISP2}$ the field $IN$ is updated such that $IN(l_{Y_3}) = INY6 \cup \{p_6(t) \mid p_6(t) \in Cl(F,R)\}$ Finally the display rule $D$ updates the field $IN$, thus $IN(l_{Y_3}) = INY7 \cup \{p(t) \mid p \text{ is valid in the base context and } p_6(t) \text{ has been selected by a visualization strategy. At the end of Step 13, the branch associated with } Y \text{ is full. The computation is finite even if its nodes can require an infinite derivation.}$

5.3 Computation Tree of $\Pi$ and Repairs

As a preliminary remark, and since all the branches of the computation tree are finite, let us point out that we can thus use the characterization of the validity given in the properties of Section 3 using the leaves of that tree. The branch associated with $Y$ is $OUT - valid$ if and only if $IN(l_{Y_3})$ satisfies $OUT(l_{Y_3})$. Moreover, the branch associated with $Y$ is $MBT - valid$ if and only if $IN(l_{Y_3})$ satisfies $MBT(l_{Y_3})$.

Theorem 3. Let $K = (\mathcal{F}, \mathcal{R}, \mathcal{N})$ be a knowledge base. Let $\Pi$ be the $\exists$-ASP program obtained from $K$ according to the above encoding. Let $Y$ be a subset of the set of selectable
atoms \(X\). The full branch of the computation tree corresponding to \(\Pi\), associated with \(Y\) is valid if and only if \(Y\) is a maximal subset of \(X\) such that \(\text{Cl}(Y, R \cup N) \not\models \bot\).

**Proof.** By hypothesis \(Y \subseteq X\), thus by Proposition 3 the computation tree provides a full branch associated with \(Y\) and \(l\) denotes its leaf. We prove the first the direction by contraposition. If \(\text{Cl}(Y, R \cup N) \models \bot\) then \(\exists N \in N\) such that \(\text{Cl}(Y, R) \models N\) thus \(\text{absurd}(\text{base}) \in \text{IN}(l)\), thus \(\forall p(t) \in Y\) we have \(p_n(t) \in \text{IN}(l)\) and \(p_n(t) \in \text{OUT}(l)\) therefore the branch associated with \(Y\) is not \(\text{OUT} - \text{valid}\). Suppose now that \(\text{Cl}(Y, R \cup N) \not\models \bot\) but there exists \(p(t) \in X \setminus Y\) s.t. \(\text{Cl}(Y \cup \{p(t)\}, R \cup N) \not\models \bot\) thus \(p_n(t, ctx(p, t)) \in \text{IN}(l)\) and we cannot obtain \(\text{absurd}(\text{ctx}(p, t))\). However \(p_n(t)\) could only be obtained from \(\text{absurd}(\text{ctx}(p, t))\). Therefore the branch associated with \(Y\) is \(\text{OUT} - \text{valid}\).

We now prove the other direction. Let \(Y\) be a maximal subset of \(X\) such that \(\text{Cl}(Y, R \cup N) \not\models \bot\). Thus \(\text{absurd}(\text{base}) \not\in \text{IN}(l)\) and \(\forall p(t) \in Y, p_n(t) \not\in \text{IN}(l)\). Since \(\text{OUT}(l) = \{p_n(t) \mid p(t) \in Y\}\) then the branch associated with \(Y\) is \(\text{OUT} - \text{valid}\). \(Y\) is maximal w. r. t. set inclusion thus \(\forall q(u) \in X \setminus Y\) we have \(\text{Cl}(Y \cup \{q(u)\}, R \cup N) \models \bot\). Thus \(\text{absurd}(\text{ctx}(q, u)) \not\in \text{IN}(l)\) since \(q(u) \not\in \text{IN}(l)\). Therefore \(q_n(u) \in \text{IN}(l)\) and \(q_n(u) \not\in \text{MBT}(l)\) therefore the branch associated with \(Y\) is \(\text{MBT}-\text{valid}\).

We did not discuss yet the effects of the selection and visualization strategies on the results of our program. If we select the atoms with Strategy \(\text{SEL1}\) then \(X\) is exactly the set \(F\). If we select the atoms with Strategy \(\text{SEL2}\) then \(X\) is exactly the ground closure of \(F\). According to Theorem 1 using Strategy \(\text{SEL1}\) the result of the branch associated with \(Y\) is an answer if and only if \(Y\) is maximal consistent subset of \(F\) while using Strategy \(\text{SEL2}\) the result of the branch associated with \(Y\) is an answer if and only if \(Y\) is maximal consistent subset of the ground closure of \(F\). When displaying atoms with Strategy \(\text{DISP1}\) the restriction of the answer set associated with a branch \(Y\) to the predicates of the original vocabulary is exactly \(\text{Cl}(Y, R) \cap F\) while displaying the atoms with Strategy \(\text{DISP2}\) the restriction of the answer set associated with a branch \(Y\) to the predicates of the original vocabulary is exactly \(\text{Cl}(Y, R)\). Let \(\Pi_1\) be an \(\exists\text{-ASP}\) program obtained from the above encoding. Let \(\text{AS}\) be an answer set of \(\Pi\), \(\rho(\text{AS})\) denotes the restriction of \(\text{AS}\) to the original vocabulary \(\mathcal{V}\) and \(\rho(\Pi) = \{\rho(\text{AS}) \mid \text{AS} \in \text{AS}(\Pi)\}\).

**Corollary 1.** Let \(K = (\mathcal{F}, R, N)\) be knowledge base. Let \(\Pi_1\) be the \(\exists\text{-ASP}\) program obtained from the above encoding using strategies \(\text{SEL1}\) and \(\text{DISP1}\). Then \(\rho(\Pi_1)\) is the set of repairs of \(K\). Let \(\Pi_5\) be the \(\exists\text{-ASP}\) program obtained from the above encoding using strategies \(\text{SEL1}\) and \(\text{DISP2}\). Then \(\rho(\Pi_5)\) is the set of closed repairs of \(K\). Let \(\Pi_7\) be the \(\exists\text{-ASP}\) program obtained from the above encoding using strategies \(\text{SEL2}\) and \(\text{DISP2}\). Then \(\rho(\Pi_7)\) is the set of repairs of the closure of \(K\).

**Example 3.** The selection strategy \(\text{SEL1}\) allows one to select the predicates in \(F\) and provides the set \(X = \{p_s(a), q_a(a)\}\). The computation tree develops 4 branches, each one encoding a subset of \(Y\) of \(X\). Only two of them are full valid branches. The selection strategy \(\text{SEL2}\) allows one to select the predicates in the grounded closure of \(F\) and provides the set \(X = \{p_s(a), q_s(a), s_s(a), t_s(a)\}\). The computation tree develops 16 branches, each one encoding a subset of \(Y\) of \(X\). Only two of them are full valid branches. The visualization strategy \(\text{DISP1}\) allows one to display valid predicates.
within the base context which belong to $F$ while the visualization strategy DISP2 allows one to display valid predicates within the base context which belong to grounded closure of $F$. Using strategies SEL1 and DISP1 we obtain an $\exists$-ASP program denoted by $Π_1$ such that the answer sets restricted to the original vocabulary are $\{p(a)\}$ and $\{q(a)\}$. Note that they correspond to the repairs of $K$. Using strategies $SEL1$ and $DISP2$ we obtain an $\exists$-ASP program denoted by $Π_7$ such that the answer sets restricted to the original vocabulary are $\{p(a), t(a)\}$ and $\{q(a), s(a)\}$. Note that they correspond to the closed repairs of $K$. Using strategies $SEL2$ and $DISP1$ we obtain an $\exists$-ASP program denoted by $Π_7$ such that the answer sets restricted to the original vocabulary are $\{p(a), s(a), t(a)\}$ and $\{q(a), s(a), t(a)\}$. Note that they correspond to the repairs of the closure of $K$.

5.4 Other Strategies

We have presented here a generic encoding of a knowledge base $K$ into an ASP program that computes different kind of repairs of $K$, according to the different selection rules and display rules we have chosen in that encoding. This generic ASP program could take into account other possible select/display rules to achieve different outcome. For instance, let us consider the following set of rules. **Selection rules:** The user defines all “optional” atoms with rules of form $p_1(X) \leftarrow p_1(X)$. where all atoms of $F$ with predicate name $p$ are optional and $p_i(a) \leftarrow p_i(a)$, where the atom $p(a)$ of $F$ is optional and then asserts that every atom of $F$ that is not optional is mandatory. For every predicate name $p$, there is a rule of form $p_1(X, base) \leftarrow p_1(X)$, not $p_i(X)$. **Display rules:** The user can use rules similar to the selection rules to display only optional atoms of $F$. With such a set of select/display rules, the program $Π$ will admit an answer set only when the subset $M$ of mandatory atoms of $F$ (i.e. those that are not declared optional) is consistent w.r.t. $(R, N)$, and in that case, if $AS$ is an answer set of $Π$, $p(AS)$ will be an inclusion-maximal subset $F'$ of $F$ such that $M \cup F'$ is consistent w.r.t. $(R, N)$.

6 Conclusion

This paper presented a generic encoding in $\exists$-ASP of repair-based techniques for inconsistent knowledge bases expressed within the formalism of existential rules. We focused on three kinds of repairs that allow for computing query answering with the following semantics proposed in [2]: $(\circ_1, \forall)$ (corresponds to AR-semantics [16]), $(\circ_1, \forall)$ (corresponds to IAR-semantics [16]), $(\circ_7, \forall)$ (close to CAR-semantics [16]), $(\circ_7, \forall)$ (close to ICAR-semantics [16]) and $(\circ_5, \forall)$ (corresponds to ICR-semantics [10]). Indeed these semantics can be rephrased in our framework as follows. Let $K$ be a knowledge base and let $q$ and $q_v$ be first order formulas, where $q_v$ is obtained from $q$ by replacing each predicate $p(t)$ occurring in $q$ by $p_v(t, base)$ we have: 1) $K = (\circ_1, \forall)q$ iff $\forall AS \in AS(Π_1), q_v \in AS. 2) K = (\circ_1, \forall)q$ iff $q_v \in \cap AS_i, AS(Π_1) AS_i. 3) K = (\circ_7, \forall)q$ iff $\forall AS \in AS(Π_7), q_v \in AS. 4) K = (\circ_7, \forall)q$ iff $q_v \in \cap AS_i, AS(Π_7) AS_i. 5) K = (\circ_5, \forall)q$ iff $q_v \in \cap AS_i, AS(Π_5) AS_i.$

A future work will be dedicated to the implementation and experimentation of the proposed encoding with ASPeRiX [14]. Another interesting issue is the extension of this encoding to the modifiers proposed within the unified framework for inconsistency-tolerant query answering stemming from the selection modifier based on cardinality.
References