Abduction with Negation as Failure for Active and Reactive Rules

Fariba Sadri and Francesca Toni

Department of Computing, Imperial College, London SW7 2AZ, UK
{fs,ft}@doc.ic.ac.uk

Abstract. Recent work has suggested abductive logic programming as a suitable formalism to represent active databases and intelligent agents. In particular, abducibles in abductive logic programs can be used to represent actions, and integrity constaints in abductive logic programs can be used to represent active rules of the kind encountered in active databases and reactive rules incorporating reactive behaviour in agents. One would expect that, in this approach, abductive proof procedures could provide the engine underlying active database management systems and the behaviour of agents. We analyse existing abductive proof procedures and argue that they are inadequate in handling these applications. The inadequacy is due to the inappropriate treatment of negative literals in integrity constraints. We propose a new abductive proof procedure and give examples of how this proof procedure can be used to achieve active behaviour in (deductive) databases and reactivity in agents. Finally, we prove some soundness and completeness results for the new proof procedure.

1 Introduction

Abductive logic programming is a powerful knowledge representation framework, that can be used to realise many diverse applications [9]. An **abductive logic program** $\langle P, \mathcal{A}, I \rangle$ consists of

- a logic program, P, seen as an "incomplete" theory,
- a set of **abducible predicates**, $\mathcal{A}$,
  whose (variable-free) **abducible atoms** are used to expand the theory,
- a set of first-order sentences, I, the integrity constraints,
  that must be "satisfied" by any sets of abducibles expanding the theory.

Earlier work [11, 16] shows promise in using abductive logic programming to understand and implement active databases (e.g. see [17]) and intelligent agents (e.g. see [7]). In particular, *beliefs* and *desires* of agents as conceptualised in the BDI architecture [15] can be understood as logic programs and integrity constraints, respectively.

In general, (many) *active rules* in active databases and rules determining the reactive behaviour of agents in multi-agent systems (*reactive rules*) can be understood as integrity constraints in abductive logic programming. For example, the active/reactive rule

on *rise-temperature-to(T)*, if $T{>}90$ then *raise-alarm*
can be understood declaratively as an integrity constraint

$\forall T$ [*rise-temperature-to(T)* $\wedge$ $T{>}90 \rightarrow$ *raise-alarm*], or, more accurately

$\forall T, T'$ [*rise-temperature-to*(T, T') $\wedge$ $T{>}90 \rightarrow$ *raise-alarm*$(T' + 1)$].

Reactive rules like the one above can be interpreted as production rules.

Given a goal (conjunction of literals) G, a set of abducible atoms $\mathcal{D}$, and a variable substitution θ for the variables in G, a pair $(\mathcal{D}, \theta)$ is an **abductive answer** for G, wrt an abductive logic program $\langle P, \mathcal{A}, I \rangle$, iff

1) $P \cup \mathcal{D}$ entails $G\theta$, and

2) $P \cup \mathcal{D}$ satisfies I.

Various notions of entailment and satisfaction can be adopted (see [9]). Fulfillment of other properties is sometimes required, e.g. minimality, wrt set inclusion, of the set of abducibles in the answer.

Under the abductive interpretation of multi-agent systems and active databases, abductive answers are understood as acceptable sets of actions to be performed as a consequence of given transactions, in the case of active databases, and of given observations and goals, in the case of agents. In the above example, given the observation *rise-temperature-to(95)* (or more accurately *rise-temperature-to(95,now)*), the only acceptable set of actions (abductive answer) is {*raise-alarm*} (or more accurately {*raise-alarm(now + 1)*}). In general the actions can be internal, for example changing the database, or external, affecting the environment, for example communication actions. Any selected abductive answer corresponds to an *intention* in the BDI architecture for agents [15].

A number of proof procedures for the computation of abductive answers (**abductive proof procedures**) have been proposed, e.g. [10, 2, 5, 3, 12, 14, 16]. [1] Reactivity in active database management systems and agents can be obtained by interleaving the execution of the proof procedures, the selection of abducibles (actions) in partially computed abductive answers (plans) and the processing of outside observations, that might trigger active/reactive rules. This interleaving gives rise to an observe-think-act cycle [11, 16].

Our investigation indicates that none of the existing procedures produces the required active/reactive behaviour, due to the inadequate treatment of negative literals in integrity constraints. We propose an alternative abductive proof procedure, obtained by modifying the proof procedure of [5]. We prove soundness of the new proof procedure wrt the semantics of abductive answers in [5] and, in a special case, soundness and completeness wrt the stable model semantics [6].

The paper is organised as follows. In section 2 we describe some preliminary notions. In section 3, through a number of examples, we analyse the behaviour of some existing abductive proof procedures, in particular [5], [10] and [3]. On the basis of the examples, we demonstrate the inadequacy of these proof procedure to be applied in the context of active databases and multi-agent systems. In section 4 we revise the proof procedure of [5] to define the new abductive proof procedure, illustrating its application to examples. In section 5 we prove some

[1] Theorist, the procedure presented in [14], computes answers wrt more general logic programs where P, in addition to I, is a set of first-order sentences.

formal properties of the proof procedure, in particular its relationshoip to the proof procedure of [5] and its soundness. In section 6 we conclude.

2 Preliminaries

A **logic program** is a set of rules of the form

$A \leftarrow L_1 \wedge \ldots \wedge L_n$,

with A atom, $L_1, \ldots, L_n$ (positive or negative) literals, and $n \geq 0$. All variables in A, L_i are implicitly universally quantified, with scope the entire rule. If $A = p(t)$, for some vector of terms t, the rule is said to *define* p. In the sequel, t, s and t_i represent vectors of terms and X represents a vector of variables.

Our proof procedure relies upon the **completion** [1] of logic programs. The completion of a predicate p defined by the set of rules

$\{p(t_1) \leftarrow D_1, \ldots, p(t_k) \leftarrow D_k\}$

is the **iff-definition**

$p(X) \leftrightarrow [X = t_1 \wedge D_1] \vee \ldots \vee [X = t_k \wedge D_k]$

where X is a vector of variables, implicitly universally quantified with scope the entire definition, and every variable in t_j and D_j not in X is implicitly existentially quantified on the righthand side of the definition. The left and righthand side are called the *head* and the *body*, respectively. The completion of a predicate p not defined in a given logic program is $p(X) \leftrightarrow false$.

The **selective completion** $comp_S(P)$ of a logic program P wrt a set of predicates S in the language of P is the union of the completions of all the predicates in S.

In this paper, **integrity constraints** are implications of the form

$L_1 \wedge \ldots \wedge L_n \rightarrow A$

with $L_1, \ldots, L_n$ literals, possibly $true$, $n > 0$, and A atom, possibly $false$. All variables in A, L_i are implicitly universally quantified, with scope the entire implication. A is called the *head* and $L_1 \wedge \ldots \wedge L_n$ is called the *body*.

Without loss of generality [9] we can assume that the abducible predicates in an abductive logic program $\langle P, \mathcal{A}, I \rangle$ are not defined in P. In the sequel, $\overline{\mathcal{A}}$ will refer to the complement of the set $\mathcal{A}$ wrt the set of all predicates in the language of P, i.e. $\overline{\mathcal{A}}$ is the set of non-abducible predicates.

3 Motivating examples

A number of abductive proof procedures have been defined, e.g. [14, 10, 2, 5, 3]. In this section, we will examine the behaviour of the proof procedures of [10, 5, 3], here referred to as KM, FK and SLDNFA, respectively, for some simple abductive logic programs. Analogous considerations apply for the remaining proof procedures. Note that KM and SLDNFA employ only (*denial*) integrity constraints with the special atom $false$ in the head, and assume that integrity constraints $L_1 \wedge \ldots \wedge L_n \rightarrow A$, with $A \neq false$, are rewritten as $L_1 \wedge \ldots \wedge L_n \wedge not\, A \rightarrow false$. Also, the literal $not\, false$ is implicitly conjoined to every initial goal in SLDNFA. Finally, similarly to [2], FK uses $comp_{\overline{\mathcal{A}}}(P)$ rather than P.

In the absence of negation, KM, FK and SLDNFA all produce the required results, as illustrated by the following example.

Example 1. Let $\langle P, \mathcal{A}, I \rangle$ be the abductive logic program with

$P = \{\ emp(mary), emp(joe), tax\text{-}form(mary, 32B98), tax\text{-}form(joe, nil)\}$

$\mathcal{A} = \{print, date\}$

$I = \{date(may) \wedge emp(E) \wedge tax\text{-}form(E, nil) \rightarrow print(\text{"E to fill tax form"})\}$.

The program P represents a database for the personnel of a department, defined in terms of the relations $emp(Employee)$ and $tax\text{-}form(Employee, Code)$. The integrity constraint in I represents the active rule

on $date(may)$, if $emp(E)$ and $tax\text{-}form(E, nil)$ then $print(\text{"E to fill tax form"})$

whose procedural interpretation is: when the event $date(may)$ takes place, the rule is triggered, the condition "$emp(E)$ and $tax\text{-}form(E, nil)$" is evaluated, and, for each instance of E satisfying the condition, the printing action is executed.

Given the input $date(may)$, KM, FK and SLDNFA compute the only intuitively correct answer $\{print(\text{"joe to fill tax form"})\}$.

However, in the presence of negation KM, FK and SLDNFA can produce unintuitive answers, as illustrated by the following examples.

Example 2. Let $\langle P, \mathcal{A}, I \rangle$ be the abductive logic program with $P = \emptyset$ and

$\mathcal{A} = \{cleaning\text{-}day, temperature\text{-}rise, dust, sound\text{-}alarm, evacuate\}$

$I = \{\ cleaning\text{-}day \wedge not\ sound\text{-}alarm \rightarrow dust,$
$\quad temperature\text{-}rise \rightarrow sound\text{-}alarm,$
$\quad sound\text{-}alarm \rightarrow evacuate\}$.

Intuitively, the integrity constraints represent the reactive rules

on $cleaning\text{-}day$, if $not\ sound\text{-}alarm$, then $dust$
on $temperature\text{-}rise$, then $sound\text{-}alarm$
on $sound\text{-}alarm$, then $evacuate$

Given only the observation $cleaning\text{-}day$, KM, FK and SLDNFA compute the intuitively correct answer $\{cleaning\text{-}day, dust\}$ as well as the unintuitive answer $\{cleaning\text{-}day, sound\text{-}alarm, evacuate\}$. The second answer arbitrarily abduces $sound\text{-}alarm$, and is incorrect according to the active database and multi-agent system interpretations. (Note that both answers are minimal wrt set inclusion.)

Example 3. Let $\langle P, \mathcal{A}, I \rangle$ be the abductive logic program with

$P = \{\ pregnant \leftarrow naturally\text{-}inseminated,$
$\quad pregnant \leftarrow artificially\text{-}inseminated\}$

$\mathcal{A} = \{weight\text{-}gain, naturally\text{-}inseminated, diet, artificially\text{-}inseminated\}$

$I = \{weight\text{-}gain \wedge not\ pregnant \rightarrow diet\}$.

Intuitively, the integrity constraint represents the reactive rule

on $weight\text{-}gain$, if $not\ pregnant$, then $diet$

whose procedural interpretation is: when the event $weight\text{-}gain$ occurs and the condition $pregnant$ does not hold, the action $diet$ is executed. Given the observation $weight\text{-}gain$, KM, FK and SLDNFA compute three alternative (minimal) answers, $\{weight\text{-}gain, diet\}$, $\{weight\text{-}gain, naturally\text{-}inseminated\}$, and $\{weight\text{-}gain, artificially\text{-}inseminated\}$. Only the first is intuitively correct.

The unintuitive behaviour of KM, FK and SLDNFA is due to the inappropriate treatment of the negative literals, occurring in the body of integrity constraints, whose atoms are either abducibles or are defined via abducibles. Indeed, these proof procedures seem to treat negations, implicitly, classically rather than as negation as failure, and the implication $\rightarrow$ as material implication. In example 3, the classical interpretation of negation occurring in the body of integrity constraints leads to treating $weight\text{-}gain \wedge not\,pregnant \rightarrow diet$ as the material implication $weight\text{-}gain \rightarrow [diet \vee pregnant]$ rather than as the reactive rule the integrity constraint aims at representing.

We will define a modification of FK computing only the intuitively correct answers to goals and observations in an active database/agent perspective.

4 The proof procedure

Like FK, our proposed proof procedure uses the selective completion $comp_{\overline{\mathcal{A}}}(P)$ of P in the given abductive logic program $\langle P, \mathcal{A}, I\rangle$. Moreover, like FK, it derives a sequence of goals, starting from an **initial goal**, which is a conjunction of literals, conjoined with the integrity constraints in I. All variables in the literals in the initial goal are free. All variables in the integrity constraints in the initial goal are universally quantified. Goals in the sequence are derived by applying a number of inference rules. To simplify the treatment of quantifiers, both iff-definitions in $comp_{\overline{\mathcal{A}}}(P)$ and integrity constraints in I are assumed to be range-restricted, i.e. all variables in the head must appear in at least one (positive) atom in the body, and this atom must not be an equality between two variables.

In the simplest case, goals derived by the proof procedure are disjunctions of **simple goals**, which are conjunctions of the form

$A_1 \wedge \ldots \wedge A_n \wedge I_1 \wedge \ldots \wedge I_m \wedge D_1 \wedge \ldots \wedge D_k$

where $n, m, k \geq 0$, $n+m+k > 0$, the A_i are atoms, the I_i are **implications**, with the same syntax of integrity constraints, except that in addition to universally quantified variables, they can also contain existentially quantified or free variables, occurring elsewhere in the goal, and the D_i are disjunctions of conjunctions of literals and implications. Implications are obtained by repeatedly applying the inference rules of the proof procedure to either integrity constraints in the given $\langle P, \mathcal{A}, I\rangle$ or to the result of rewriting negative literals $not\,A$ as $A \rightarrow false$.

FK assumes that the inference rules below are applied in such a way that every goal in a sequence derived by the proof procedure is a *disjunction of simple goals*. (Note that every initial goal is a simple goal with the integrity constraints and the rewriting of negative literals as the only implications.) Our proof procedure will make the same assumption.

4.1 FK proof procedure

Given an abductive logic program $\langle P, \mathcal{A}, I\rangle$ and an initial goal G, a **derivation** for G wrt $comp_{\overline{\mathcal{A}}}(P), I$ is a sequence of goals, $G_1, \ldots, G_n$, such that $G_1 = G \wedge I$ and each G_{i+1} is obtained from the previous formula G_i by application of one of the following (basic) inference rules:

Unfolding: given an atom $p(t)$ and $p(X) \leftrightarrow D_1 \vee \ldots \vee D_n$ in $comp_{\overline{\mathcal{A}}}(P)$

- if the atom is a conjunct of a simple goal in G_i, then G_{i+1} is G_i with the atom replaced by $(D_1 \vee \ldots \vee D_n)\theta$, where $\theta = \{X/t\}$;
- if the atom is a conjunct L_i in the body of an implication $L_1 \wedge \ldots \wedge L_m \rightarrow A$, $m \geq 1$ which is a conjunct of a simple goal of G_i, then G_{i+1} is G_i with the implication replaced by the conjunction $[L_1 \wedge \ldots \wedge D_1\theta \wedge \ldots \wedge L_m \rightarrow A] \wedge \ldots \wedge [L_1 \wedge \ldots \wedge D_n\theta \wedge \ldots \wedge L_m \rightarrow A]$, where $\theta = \{X/t\}$.

Propagation: given an atom $p(s)$ and an implication $L_1 \wedge \ldots \wedge L_m \rightarrow A$, $m \geq 1$, with $L_i = p(t)$, both conjuncts of the same simple goal in G_i, then, if the implication $L_1 \wedge \ldots L_{i-1} \wedge t = s \wedge L_{i+1} \wedge \ldots \wedge L_m \rightarrow A$ is not already a conjunct of the simple goal, then G_{i+1} is G_i with the new implication conjoined to the simple goal.

Negation elimination: given an implication $not\, A_1 \wedge \ldots \wedge not\, A_m \rightarrow A$ which is a conjunct of a simple goal in G_i, then G_{i+1} is G_i with the implication replaced by the disjunction $A \vee A_1 \vee \ldots \vee A_m$. [2]

Logical simplification replaces:
$[B \vee C] \wedge E$ *by* $[B \wedge E] \vee [C \wedge E]$ (**splitting**)
$B \wedge true$ *by* B; $B \wedge B$ *by* B; $B \wedge false$ *by* $false$;   *etc.*

Equality rewriting applies the equality rewrite rules, simulating the unification algorithm [13] and the application of substitution.

Case analysis: given an implication $B \wedge X = t \rightarrow A$, where X is a free or existentially quantified variable and t does not contain X and is not a universally quantified variable, if the implication is a conjunct in a simple goal in G_i, then G_{i+1} is G_i with the implication replaced by the disjunction $[X = t \wedge [B \rightarrow A]] \vee [X = t \rightarrow false]$. Any variables in t which are universally quantified in the implication correctly become existentially quantified in the first disjunct and universally quantified in the second disjunct.

Factoring: given a simple goal $a(t) \wedge a(s) \wedge C$ in G_i, with a abducible, G_{i+1} is G_i with the given simple goal replaced by the disjunction
$[a(t) \wedge a(s) \wedge C \wedge [t = s \rightarrow false]] \vee [a(t) \wedge C \wedge t = s]$.

Note that for goals in a derivation to be guaranteed to be disjunctions of simple goals, it is sufficient that every step of unfolding, negation elimination and case analysis is followed by a step of splitting.

Given an initial goal G, an FK-derivation $(G_1 = G \wedge I), \ldots (G_n = N \vee Rest)$, for some conjunction N of literals and implications, is a **successful FK-derivation** for G iff no inference rule can be applied to N and $N \neq false$. Given such a successful FK-derivation, an **FK-answer extracted from** N is a pair $(\mathcal{D}, \sigma)$ such that

- σ' is a substitution replacing all free and existentially quantified variables in N by ground terms, and σ' satisfies all equalities and disequalities in N,
- σ is the restriction of σ' to the variables in G,
- $\mathcal{D}$ is the set of all abducible atoms in $N\sigma$.

[2] In [5], negation elimination is defined as follows: $not\, A_1 \wedge Rest \rightarrow A$ is replaced by $Rest \rightarrow A \vee A_1$. Operationally, our definition is equivalent to the one in [5].

Every FK-answer extracted from a disjunct in a successful FK-derivation for an initial goal G is called an **FK-computed answer** for G.

In example 3, a possible FK-derivation for *weight-gain* is

$G_1 = \mathit{weight\text{-}gain} \wedge [\mathit{weight\text{-}gain} \wedge \mathit{not\,pregnant} \rightarrow \mathit{diet}]$

$G_2 = G_1 \wedge [\mathit{not\,pregnant} \rightarrow \mathit{diet}]$ by propagation

$G_3 = G_1 \wedge [\mathit{pregnant} \vee \mathit{diet}]$ by negation elimination

$G_4 = [G_1 \wedge \mathit{pregnant}] \vee [G_1 \wedge \mathit{diet}]$ by splitting

$G_5 = [G_1 \wedge [\mathit{naturally\text{-}inseminated} \vee \mathit{artificially\text{-}inseminated}]] \vee [G_1 \wedge \mathit{diet}]$
by unfolding in the first disjunct

$G_6 = [G_1 \wedge \mathit{naturally\text{-}inseminated}] \vee$
$[G_1 \wedge \mathit{artificially\text{-}inseminated}] \vee$ by splitting
$[G_1 \wedge \mathit{diet}]$.

Since no inference rule can be applied to any of the resulting disjuncts, the derivation $G_1, \ldots, G_6$ is a successful derivation. The FK-answers

$\{\mathit{weight\text{-}gain}, \mathit{naturally\text{-}inseminated}\}$,
$\{\mathit{weight\text{-}gain}, \mathit{artificially\text{-}inseminated}\}$ and
$\{\mathit{weight\text{-}gain}, \mathit{diet}\}$

can be extracted from the first, second and third disjunct of G_6, respectively. Note that the derivation $G_1, \ldots, G_4$ is also a successful derivation, since no inference rule can be applied to the second disjunct in G_4. The answer $\{\mathit{weight\text{-}gain}, \mathit{diet}\}$ can be extracted from this disjunct.

4.2 Modified proof procedure

In the modified proof procedure, implications in goals are either **marked** (by means of the * symbol) or **non-marked**. In the initial goal, integrity constraints are marked whereas implications obtained by rewriting negative literals are non-marked. All inference rules of FK except negation elimination can be applied to marked implications. We define a new inference rule to handle negation in the conditions of marked implications.

Negation rewriting: a *marked* implication
$*[\mathit{not}\, A_1 \wedge \ldots \wedge \mathit{not}\, A_m \rightarrow A]$, $m \geq 1$,
is replaced by the disjunction $[\mathit{provable}(A_1)] \vee \ldots \vee [\mathit{provable}(A_m)] \vee [*[A_1 \rightarrow \mathit{false}] \wedge \ldots \wedge *[A_m \rightarrow \mathit{false}] \wedge A]$.

All inference rules of FK leave the marked/non-marked status unchanged, except that $*[\mathit{true} \rightarrow A]$ is automatically rewritten as A.

Example 4. In example 2, a possible derivation for *cleaning-day* is

$G_1 = \mathit{cleaning\text{-}day} \wedge *[\mathit{cleaning\text{-}day} \wedge \mathit{not\,sound\text{-}alarm} \rightarrow \mathit{dust}] \wedge$
$*[\mathit{temperature\text{-}rise} \rightarrow \mathit{sound\text{-}alarm}] \wedge *[\mathit{sound\text{-}alarm} \rightarrow \mathit{evacuate}]$

$G_2 = G_1 \wedge *[\mathit{not\,sound\text{-}alarm} \rightarrow \mathit{dust}]$ by propagation

$G_3 = G_1 \wedge [[[*[\mathit{sound\text{-}alarm} \rightarrow \mathit{false}] \wedge \mathit{dust}] \vee [\mathit{provable}(\mathit{sound\text{-}alarm})]]$
by negation rewriting

$G_4 = [G_1 \wedge [*[\mathit{sound\text{-}alarm} \rightarrow \mathit{false}] \wedge \mathit{dust}] \vee [G_1 \wedge \mathit{provable}(\mathit{sound\text{-}alarm})]$
by splitting

to which no inference rule can be applied.

Given an initial goal G, a derivation $(G_1 = G \wedge I), \ldots, (G_n = N \vee Rest)$, for some conjunction N of literals and implications, is a **successful derivation** for G iff no inference rule can be applied to N. Given such a successful derivation, an **answer extracted from** N is a pair $(\mathcal{D}, \sigma)$ such that

- if no conjunct in N is of the form $provable(A)$ then $(\mathcal{D}, \sigma)$ is an FK-answer extracted from N,
- otherwise, let
  1. σ' be a Skolemisation for free and existentially quantified variables in N,
  2. $\mathcal{D}'$ be the set of all abducible atoms in $N\sigma'$, and T be $comp_{\mathcal{A}}(\mathcal{D}') \cup comp_{\overline{\mathcal{A}}}(P)$,
  3. $provable(A)$ be a conjunct of $N\sigma'$,
  4. $G'_1 = A, \ldots, G'_k$ be a derivation wrt $T, \emptyset$ such that $G'_k = N' \vee Rest'$ for some conjunction $N' \neq false$ to which no inference rule can be applied,
  5. N'' be the deSkolemisation of N' wrt σ'.

  If N'' satisfies all (in)equalities in N, $(\mathcal{D}, \sigma)$ is an *answer extracted from* the conjunction obtained by replacing $provable(A)$ in N by N''.

Every answer extracted from a disjunct in a successful derivation for an initial goal G is called a **computed answer** for G.

Given the derivation in example 4, $(\{dust\}, \emptyset)$ is the only (FK-)answer extracted from the first disjunct in G_4. No answer can be extracted from the second disjunct in G_4. Indeed, $provable(sound\text{-}alarm)$ belongs to it, and the only derivation from $sound\text{-}alarm$, wrt $T = comp_{\overline{\mathcal{A}}}(\emptyset) \cup comp_{\mathcal{A}}(\{cleaning\text{-}day\}) = \{dust \leftrightarrow false, sound\text{-}alarm \leftrightarrow false, temperature\text{-}rise \leftrightarrow false, cleaning\text{-}day \leftrightarrow true\}$ and wrt the empty set of integrity constraints, is the derivation

$sound\text{-}alarm, false.$

To illustrate the requirement that equalities and inequalities must be satisfied during the answer extraction process, let $\langle P, \mathcal{A}, I \rangle$ be

$P = \{p(a) \leftarrow r(a)\}$

$\mathcal{A} = \{r, s, t\}$

$I = \{t(X) \wedge not\, p(X) \rightarrow s(X)\}.$

A possible derivation for $G = r(Y) \wedge t(Y) \wedge [Y = a \rightarrow false]$ is

$G_1 = G \wedge *[t(X) \wedge not\, p(X) \rightarrow s(X)]$

$G_2 = G_1 \wedge *[not\, p(Y) \rightarrow s(Y)]$

$G_3 = [G_1 \wedge *[p(Y) \rightarrow false] \wedge s(Y)] \vee [G_1 \wedge provable(p(Y))].$

G_1, G_2, G_3 is a successful derivation, as no inference rule can be applied to the second disjunct in G_3. To extract an answer from the second disjunct, let

1. σ' be the Skolemisation $\{Y/\alpha\}$,
2. T be $comp_{\overline{\mathcal{A}}}(P) \cup comp_{\mathcal{A}}(\{r(\alpha), t(\alpha)\})=$ $\{p(X) \leftrightarrow r(X) \wedge X = a, r(X) \leftrightarrow X = \alpha, t(X) \leftrightarrow X = \alpha, s(X) \leftrightarrow false\}$
3. $provable(p(\alpha))$ be the chosen conjunct of the Skolemised disjunct,
4. $G'_1 = p(\alpha)$, $G'_2 = [r(\alpha) \wedge \alpha = a]$, $G'_3 = [\alpha = \alpha \wedge \alpha = a]$, $G'_4 = [\alpha = a]$ be a derivation, wrt $T, \emptyset$,
5. $Y = a$ be the deSkolemisation of $\alpha = a$ wrt σ'.

Since $Y = a$ does not satisfy the denial $Y = a \rightarrow false$ in the second disjunct in G_3, correctly no answer for G can be extracted from that disjunct.

5 Formal results

Our proof procedure is a specialisation of FK in the sense that every answer computed by our modified proof procedure is also computed by FK.

Theorem 1. *If (D, σ) is a computed answer for an initial goal G then (D, σ) is an FK-computed answer for G.*

Proof (sketch): we construct a successful FK-derivation from the given successful derivation by mapping every step except negation rewriting and logical simplification onto itself, every negation rewriting step replacing an implication $*[not\, A_1 \wedge \ldots \wedge not\, A_m \rightarrow A]$ with $A \neq false$ by the disjunction

$[provable(A_1)] \vee \ldots \vee [provable(A_m)] \vee [*[A_1 \rightarrow false] \wedge \ldots \wedge *[A_m \rightarrow false] \wedge A]$
onto a negation elimination step replacing $not\, A_1 \wedge \ldots \wedge not\, A_m \rightarrow A$ by the disjunction $A_1 \vee \ldots \vee A_m \vee A$, and every negation rewriting + logical simplification steps replacing an implication $*[not\, A_1 \wedge \ldots \wedge not\, A_m \rightarrow false]$ first (negation rewriting) by the disjunction

$[provable(A_1)] \vee \ldots \vee [provable(A_m)] \vee$

$[*[A_1 \rightarrow false] \wedge \ldots \wedge *[A_m \rightarrow false] \wedge false]$
and then (logical simplification) by the disjunction $[provable(A_1)] \vee \ldots \vee [provable(A_m)] \vee false$, onto a single negation elimination step replacing $not\, A_1 \wedge \ldots \wedge not\, A_m \rightarrow false$ by the disjunction $A_1 \vee \ldots \vee A_m \vee false$.

It is easy to see that, if $A \wedge *[I'] \wedge provable(p_1) \wedge \ldots \wedge provable(p_m)$ is the disjunct (from which the answer has been extracted) of the final goal in the given successful derivation, with A a set of (abducible) atoms, $*[I']$ a set of (marked and non-marked) implications and each p_i an atom ($m \geq 0$), then $A \wedge I' \wedge p_1 \wedge \ldots \wedge p_m$ is a disjunct of the final goal in the constructed FK-derivation. This might not be a successful derivation yet, since some p_i might be non-abducible and additional steps might have to be performed on the p_is. However, the constructed FK-derivation can be extended to a successful FK-derivation from which $(\mathcal{D}, \sigma)$ can be extracted, by mapping all operations in the process of answer extraction onto FK operations. In particular, every step except unfolding is mapped onto itself, every step of unfolding of non-abducible atoms at step 4 is mapped onto a step of unfolding in the FK-derivation, and every step of unfolding of abducible atoms at step 4 is mapped onto a step of logical simplification ($p \wedge p$ is replaced by p) or factoring. □

The converse of this theorem does not hold, e.g. for example 2.

FK is sound wrt the semantics of abductive answers in [5], i.e. every FK-computed answer $(\mathcal{D}, \sigma)$ for an initial goal G satisfies the following conditions:

1) $comp(P \cup \mathcal{D}) \cup CET \models G\sigma$, and

2) $comp(P \cup \mathcal{D}) \cup CET \models I$,

where $\models$ is either ordinary two-valued logical consequence or logical consequence in Kunen's three-valued logic. We will refer to abductive answers as defined above as **FK-abductive answers**. The following soundness result wrt the semantics of FK-abductive answers is a direct corollary of theorem 1:

Corollary 1 *If $(\mathcal{D}, \sigma)$ is a computed answer for an initial goal G then $(\mathcal{D}, \sigma)$ is an FK-abductive answer for G.*

FK is complete (in a weak sense) wrt the semantics of FK-abductive answers: for every FK-abductive answer $(\mathcal{D}, \sigma)$ for an initial goal G, there exists a subset $\mathcal{D}'$ of $\mathcal{D}$ such that $(\mathcal{D}', \sigma)$ is an FK- computed answer for G. Our procedure is not complete, even in the weak sense, wrt the semantics of FK- abductive answers. Indeed, it was specifically designed not to compute the unintuitive FK-computed answers. So, it is the semantics, rather than the proof procedure, that is at fault here. However, our procedure is complete wrt the semantics of FK- abductive answers in some cases, e.g when no negation occurs in integrity constraints and goals (in this case our procedure is equivalent to FK).

In the special case in which the given logic program is empty and all atoms occurring in the integrity constraints and the initial goal are abducibles, then our proof procedure computes the **stable model semantics** [6] of the integrity constraints and the goal, both seen as logic programs. In the agent setting, this special case corresponds to purely reactive agents. Every ground negative literal $not\, p$ in the goal is seen as a logic programming rule $false \leftarrow p$.

Theorem 2. *Let $\langle P, \mathcal{A}, I\rangle$ be an abductive logic program with $\mathcal{P} = \emptyset$ and G be an initial goal, and let all predicates occurring in I and G belong to $\mathcal{A}$. Then:*

- *if $(\mathcal{D}, \sigma)$ is a computed answer for G then $\mathcal{D}$ is a stable model of $I \cup G\sigma$;*
- *if M is a stable model of $I \cup G\sigma$, for some variable-free substitution σ for the variables of G, and $false \notin M$, then (M, σ) is a computed answer for G.*

Note that $false$ might be the head of some rule in $I \cup G\sigma$ and thus belong to some stable model of $I \cup G\sigma$.

This theorem is a corollary of theorem 3 below, relating our procedure, in the special case considered in theorem 2, and the procedure of [8], which is proved sound and complete wrt the stable model semantics. Given a logic program T, the proof procedure of [8] applies the model generation theorem prover (MGTP) to the disjunctive logic program $tr(T)$ obtained from T by replacing every rule of the form $A \leftarrow A_1 \wedge \ldots \wedge A_n \wedge not\, A_{n+1} \wedge \ldots \wedge not\, A_{n+m}$ with A, A_i atoms and $n \geq 0$, $m \geq 1$, by a (disjunctive) rule of the form

$[A, \neg KA_{n+1} \wedge \ldots \wedge \neg KA_{n+m}] \vee [KA_{n+1}] \vee \ldots \vee [KA_{n+m}] \leftarrow A_1 \wedge \ldots \wedge A_n$.
[8] computes the set $MGTP(tr(T), \{\emptyset\})$ of all (minimal) models of $tr(T)$ satisfying the denials $A \wedge \neg KA \rightarrow false$ and $KA \wedge \neg KA \rightarrow false$, for all atoms A, and ignoring the models containing KA but not containing A, for some atom A. [8] proves that for each model M in $MGTP(tr(T), \{\emptyset\})$, the set of all atoms in M is a stable model of T and, conversely, every stable model of T is the set of all atoms of some model in $MGTP(tr(T), \{\emptyset\})$.

Theorem 3. *Let $\langle P, \mathcal{A}, I\rangle$ and G be as in theorem 2.*

- *if $(\mathcal{D}, \sigma)$ is a computed answer for G then $\mathcal{D} \in MGTP(tr(I \wedge G\sigma), \{\emptyset\})$;*
- *if $M \in MGTP(tr(I \wedge G\sigma), \{\emptyset\})$, for some variable-free substitution σ for the variables of G, and $false \notin M$, then (M, σ) is a computed answer for G.*

Proof (sketch): $MGTP(tr(I \wedge G\sigma), \{\emptyset\})$ is constructed as follows, starting with $S_0 = \{\emptyset\}$:

1. if there exists a (disjunctive) rule $H_1 \vee \ldots \vee H_m \leftarrow B \in tr(I \wedge G\sigma)$ and a substitution θ such that, for some $M \in S_i$, $B\theta \subseteq M$ and no $H_i\theta \subseteq M$, then $S_{i+1} = S_i - \{M\} \cup \{M \cup \{H_1\theta\}\} \cup \ldots \cup \{M \cup \{H_m\theta\}\}$;
2. if there exists an atom A such that, for some $M \in S_i$, either
   (a) both A and $\neg KA$ belong to M, or
   (b) both KA and $\neg KA$ belong to M,
   then $S_{i+1} = S_i - \{M\}$.

until $S_{i+1} = S_i$. Then

3. $MGTP(tr(I \wedge G\sigma), \{\emptyset\}) = S_i - \{M | KA \in M \text{ but } A \notin M\}$.

It is easy to see that the construction of every M not containing $false$ in $MGTP(tr(I \wedge G\sigma), \{\emptyset\})$ corresponds to a derivation from G to a disjunct $N \vee$ *Rest* such that $(\mathcal{D}, \sigma)$ can be extracted from N, where $\mathcal{D}$ is the set of all "non-K-atoms" in M. Indeed:

• Every step of type 1 for $H_1 \vee \ldots \vee H_m \leftarrow B$ with B non-empty and $m = 1$ corresponds to a series of propagation steps (one for each atom in B) with the integrity constraint (without negation) $B \rightarrow H_1$;

• Every step of type 1 for $H_1 \vee \ldots \vee H_m \leftarrow B$ with B empty and $m = 1$ corresponds to no step in our proof procedure (H_1 is an atom in every disjunct of any goal in any derivation from G));

• Every step of type 1 for $H_1 \vee \ldots \vee H_m \leftarrow B$ with B non-empty and $m > 1$ corresponds to a series of propagation steps (one for each atom in B) with the integrity constraint from which $H_1 \vee \ldots \vee H_m \leftarrow B$ has been obtained by tr, followed by a step of negation rewriting and a step of splitting (in this sequence); for every atom A, $\neg KA$ corresponds to $A \rightarrow false$ in the derivation and KA corresponds to $provable(A)$ in the derivation;

• Every step of type 2a corresponds to a step of propagation of $false$ from $A \rightarrow false$ and A;

• Every step of type 3 corresponds to eliminating disjuncts from successful derivations by answer extraction, which, in the special case considered, amounts to checking that if $provable(A)$ belongs to the disjunct then so does A.

The corresponding steps of type 2b in the proof procedure is that $provable(A)$ and $A \rightarrow false$ do not both belong to any computed answer. Such steps do not correspond to any explicit steps in the proof procedure. However, answer extraction guarantees that $provable(A)$ and $A \rightarrow false$ do not both belong to any computed answer. □

6 Conclusions and future work

We have defined a new abductive proof procedure and compared it to a number of existing similar proof procedures. We have argued that the new procedure is suitable as the engine underlying active database management systems as well as (rational and reactive) logic-based agents. Indeed, our proof procedure can be used as the underlying procedure in the agent architecture of [11].

Our integrity constraints seem closely related to the production rules with negation as failure proposed by [4]. The formal study of the relationships between the two is a subject for future work.

Our proof procedure needs to be extended to handle built-ins such as $<$, e.g. as in [12]. This extension is needed, in particular, to cater for temporal reasoning in agents.

Finally, future work includes refining existing prototype implementations of the proof procedure.

Acknowledgements

This reasearch was supported by the UK EPSRC project "Logic-based multi-agent systems". The authors are grateful to Bob Kowalski for challenging discussions and to the anonymous referees for their comments.

References

1. Clark, K.L.; 1978. Negation as failure. *Logic and Data Bases*, 293–322, Plenum, NY.
2. Console, L.; Theseider Dupré, D.; Torasso, P.; 1991. On the relationship between abduction and deduction. *Journal of Logic and Computation* 2(5): 661–690, OUP.
3. Denecker, M.; De Schreye, D.; 1997. SLDNFA: an abductive procedure for abductive logic programs. *Journal of Logic Programming* 34(2): 111–167, Elsevier.
4. Dung, P.M.; Mancarella, P.; 1996. Production systems need negation as failure. *Proc. 13th AAAI,* AAAI Press.
5. Fung, T.H.; Kowalski, R.A.; 1997. The iff procedure for abductive logic programming. *Journal of Logic Programming* 33(2):151–165, Elsevier.
6. Gelfond, M.; Lifschitz, V.; 1988. The stable model semantics for logic programming. *Proc. 5th ICLP* 1070–1080, MIT Press.
7. Huhns, M.N.; Singh M.P. (eds); 1997. *Readings in Agents*, Morgan Kaufman.
8. Inoue, K.; Koshimura, M.; Hasegawa, R.; 1992. Embedding negation as failure into a model generation theorem prover. *Proc. 11th CADE* , LNAI 607:400–415, Springer.
9. Kakas, A.C.; Kowalski, R.A.; Toni, F.; 1998. The role of abduction in logic programming. *Handbook of Logic in AI and Logic Programming* 5:235–324, OUP.
10. Kakas, A.C.; Mancarella, P.; 1990. Abductive Logic Programming. *Workshop on Non-Monotonic Reasoning and Logic Programming.*
11. Kowalski, R.A.; Sadri, F.; 1999. From Logic Programming to Multi-agent Systems. *Annals of Mathemathics and Artificial Intelligence* (Forthcoming).
12. Kowalski, R.A.; Toni, F.; Wetzel, G.; 1998. Executing suspended logic programs. *Fundamenta Informaticae* 34(3):203–224, ISO Press.
13. Martelli, A.; Montanari, U.; 1982. An efficient unification algorithm. *ACM Trans. on Prog. Lang. and Systems* 4(2):258–282.
14. Poole, D.; 1988. A logical framework for default reasoning. *Artificial Intelligence* 36:27–47, Elsevier.
15. Rao, A.S.; Georgeff, M.P.; 1992. An abstract architecture for rational agents. *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning* .
16. Shanahan, M.; 1997. Event calculus planning revisited. *Proc. 4th European Conference on Planning* 390–402, LNAI 1348, Springer.
17. Widom, J.; Ceri, S.; 1996. *Active Database Systems: Triggers and Rules for Advanced Database Processing*, Morgan Kaufmann.