Interleaving belief updating and reasoning in abductive logic programming

Fariba Sadri and **Francesca Toni** [1]

Abstract. Most existing work on knowledge representation and reasoning assumes that the updating of beliefs is performed off-line, and that reasoning from the beliefs is performed either before or after the beliefs are changed. This imposes that, if an update occurs while reasoning is performed, reasoning has to be stopped and re-started anew so that the update is taken into account, with an obvious wastage of reasoning effort. In this paper, we tackle the problem of performing belief updating on-line, while reasoning is taking place by means of an abductive proof procedure.

1 INTRODUCTION

Traditionally, the area of knowledge representation and reasoning has focused on identifying techniques for drawing correct conclusions efficiently from beliefs. Beliefs can be held by intelligent agents, for example, and reasoning can aim at identifying plans for the agents' goals or answering queries on the agents' beliefs. Such beliefs can be subject to updates. The updates can be generated by a number of different means, for example by learning, or, in a multi-agent setting, by observation or by communication amongst agents.

If the updating of beliefs takes place while reasoning is performed, the problem arises as to which parts of the already performed reasoning can be "saved", because it has used beliefs which have not been affected by the update. Conventional knowledge representation and reasoning systems assume that no parts of reasoning can be saved, by restarting the reasoning process anew.

In this paper, we show how belief updating and reasoning can be interleaved, so that reasoning with beliefs that are not affected by the updates can be saved and, if necessary, used after the updating. We assume that reasoning is performed by means of the abductive proof procedure IFF [3] and that beliefs are represented by means of abductive logic programs (ALPs) [6]. ALPs consist of logic programs and integrity constraints. We also assume that belief updating amounts to the addition or deletion of rules in the logic program component of the ALP, or to the addition or deletion of integrity constraints in the corresponding component of the ALP.

In this paper, we do not address how the updates are determined. In particular, we do not focus on "belief revision" (e.g. as understood in [1]) , but rather on how updates produced by such revision can be interleaved with the reasoning process.

In order to save any reasoning effort not affected by updates, we adopt a labeling technique that maintains dependencies. We define a new variant of the IFF procedure, called *LIFF*, with labels occurring in derivations. We formalise the soundness of LIFF, wrt the semantics underlying the IFF procedure. We concentrate on the propositional case only for the definition of LIFF and the results.

Our work is closely related to the work of [4, 5], which share our aims. However, we extend an abductive proof procedure (IFF) for beliefs in the form of ALPs whereas [4] extends conventional SLDNF, for ordinary logic programs. Our knowledge representation and reasoning choice paves the way to the use of our techniques for agent applications, following the abductive logic agent approach of [8, 13, 7]. Moreover, we use a labeling technique rather than the (arguably more space consuming) technique based on full data structures associated to atoms as in [4].

Amongst agent applications for which our approach is particularly useful are those where the agents are situated in dynamic environments and need to adjust quickly to the changes in that environment. For example, a situated pro-active agent can plan for its goals while it interleaves the planning with action execution and observations in the environment; such an agent has, for example been described in [10]. The observations that it makes and records from the environment and the success or failure of the actions that it attempts to execute can lead to the agent revising its beliefs. LIFF allows the agent to keep as much of its partial plan as possible and to replan only those parts that are affected by the changes in its beliefs.

Another agent application that would benefit from our work is information integration via a mediator, for example to realise the semantic web. In this application a mediator agent receives the user query specified in some user-oriented high-level language. It then constructs a query plan translating the user query into a form understandable by the sources of information it has at its disposal. It contacts the sources for the necessary information, integrates their answers, and translates the answer into the language of the user. Such an information integration approach based on abductive logic programming and, in particular the IFF proof procedure, has been described in [19]. Now, while the agent is constructing the query plan for the user query, it may receive information about content or availability changes to the information sources. LIFF would allow the agent to accommodate these changes within its reasoning without having to discard all the work it has done in constructing the query plan.

This paper is a revised and extended version of [12].

2 PRELIMINARIES

Abductive logic programming is a general-purpose knowledge representation and reasoning framework that can be used for a number of applications and tasks [6, 8, 11, 14, 13, 17, 18]. It relies upon a logic-based representation of beliefs, for any given domain of application, via *abductive logic programs* (ALPs), and the execution of logic-based reasoning engines, called *abductive proof procedures*,

[1] Department of Computing, Imperial College London, UK, email:{fs,ft}@doc.ic.ac.uk

for reasoning with such representations. In this paper we concentrate on the propositional case.

An ALP consists of

- A **logic program**, P, namely a set of if-rules of the form $Head \leftarrow Body$, where $Head$ is an atom and $Body$ is either $true$ (in which case we write the if-rule simply as $Head$) or a conjunction of literals. P is understood as a (possibly incomplete) set of beliefs.
- A set of **abducible atoms**, $\mathcal{A}$, which are assumed not to occur in the $Head$ of any if-rule in P. Abducible atoms can be used to "augment" the (beliefs held within the) logic program, subject to the satisfaction of the integrity constraints (see below).
- A set of **integrity constraints**, I, namely if-then-rules of the form $Condition \Rightarrow Conclusion$, where $Condition$ is either $true$ (in which case we write the if-then-rule simply as $Conclusion$) or a conjunction of literals, and $Conclusion$ is $false$ or an atom. The integrity constraints are understood as properties that must be "satisfied" by any "acceptable" extension of the logic program by means of abducible atoms.

In the sequel, $\overline{\mathcal{A}}$ will refer to the complement of the set $\mathcal{A}$ wrt the set of all atoms in the vocabulary of P, i.e. $\overline{\mathcal{A}}$ is the set of non-abducible atoms. Both IFF and LIFF rely upon the **completion** [2] of logic programs. The completion of an atom p defined, within a propositional logic program, by the if-rules $p \leftarrow D_1, \ldots, p \leftarrow D_k$ is the **iff-definition** $p \leftrightarrow D_1 \vee \ldots \vee D_k$. The completion of an atom p not defined in a given logic program is $p \leftrightarrow false$.The **selective completion** $comp_S(P)$ of a logic program P wrt a set of atoms S in the vocabulary of P is the union of the completions of all the atoms in S. Both IFF and LIFF use $comp_{\overline{\mathcal{A}}}(P)$, i.e. the selective completion of P wrt $\overline{\mathcal{A}}$.

Abductive proof procedures aim at computing "explanations" (or "abductive answers") for "queries", given an ALP representing knowledge about some underlying domain. A **query** is a (possibly empty) conjunction of literals. Given an ALP $\langle P, \mathcal{A}, I\rangle$, an **explanation** (or **abductive answer**) for a query Q is a (possibly empty) set E of abducible atoms such that $P \cup E$ *entails* Q and $P \cup E$ *satisfies* I. Various notions of entailment and satisfaction can be adopted, for example entailment and satisfaction could be entailment and consistency, respectively, in first-order, classical logic. In this paper, we adopt truth wrt the 3-valued completion semantics of [9] as the underlying notion of entailment and satisfaction, inherited from IFF [3].

IFF generates a **derivation** consisting of a **sequence of goals**, starting from an **initial goal**, which is the given query conjoined with the integrity constraints in I. IFF computes explanations (referred to as **extracted answers**) for the given query extracting them from disjuncts in a goal in a derivation from the initial goal, when no further inference rule can be applied to these disjuncts.

Goals in a derivation are obtained by applying **inference rules** (see section 5.1). In the simplest case, these goals 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 as integrity constraints , and the D_i are disjunctions of conjunctions of literals and implications. Implications are obtained by 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 $A \Rightarrow false$.

IFF assumes that the inference rules are applied in such a way that every goal in a derivation is a *disjunction of simple goals*. LIFF will make the same assumption.

We omit here the definition of the inference rules of IFF, as they can be re-constructed from the inference rules of LIFF, given in section 5, by dropping the labels. In section 4 we illustrate the behaviour of IFF and LIFF with an example. The example illustrates the main difference between IFF and LIFF, namely the fact that the latter associates labels to literals and implications in goals, to be exploited when the ALP is updated.

3 ASSIMILATING UPDATES IN THE ABDUCTIVE LOGIC PROGRAM

As mentioned in section 1, the decision of what to revise and how can be done in a number of different ways. However the belief revision is handled, at the end of the process, if-rules or integrity constraints will need to be added or deleted from the ALP. In this section we define how a given ALP $\langle P, \mathcal{A}, I\rangle$ is revised after (any number of) the following kinds of updates:

- the addition to P of if-rules,
- the deletion from P of if-rules,
- the addition to I of integrity constraints,
- the deletion from I of integrity constraints.

Note that *modification* of if-rules and integrity constraints can be achieved by suitable combinations of deletions and additions.

There are a number of different ways that we can accommodate addition and deletion of if-rules in an ALP. For example, we can decide whether or not an atom that is originally abducible will, in effect, remain abducible after any updates. Similarly we can decide whether or not an atom that is originally non-abducible will always be non-abducible, even if all its definitions are deleted from the logic program. LIFF is independent of these choices, and it can deal uniformly with any combinations or variations of these choices. Below, we arbitrarily make the concrete choice where, after any updates, non-abducible atoms remain non-abducible and abducible atoms remain "abducible", in the sense that they can always be made to hold by abduction, as we will see later. Below, $\langle P, \mathcal{A}, I\rangle$ denotes the ALP before the update.

3.1 Addition of if-rules

Let the update U be the addition of the if-rule $p \leftarrow B$, with $B \neq false$, [2] and with B possibly $true$. We will refer to $\mathcal{A}_B$ as the set of atoms occurring in B but not in the vocabulary of $\langle P, \mathcal{A}, I\rangle$.

We refer to the revision of $\langle P, \mathcal{A}, I\rangle$ by U as $U(\langle P, \mathcal{A}, I\rangle)$=$\langle P', \mathcal{A}', I\rangle$, obtained as follows:

1. if p is an abducible in $\mathcal{A}$, then
   - $P' = P \cup \{p \leftarrow B\} \cup \{p \leftarrow p^*\}$,
   - $\mathcal{A}' = ((\mathcal{A} \cup \{p^*\}) - \{p\}) \cup \mathcal{A}_B$,

   where p^* is an atom not in the vocabulary of $\langle P, \mathcal{A}, I\rangle$;
2. if p is in the vocabulary of $\langle P, \mathcal{A}, I\rangle$ but is not abducible or p is not in the vocabulary of $\langle P, \mathcal{A}, I\rangle$, then
   - $P' = P \cup \{p \leftarrow B\}$,
   - $\mathcal{A}' = \mathcal{A} \cup \mathcal{A}_B$.

Note that, in case 1, the definition of p in P' conforms to our policy that atoms that are abducible in the initial ALP always remain

[2] The addition of $p \leftarrow false$ is not allowed directly, but it can be achieved by deleting all if-rules with head p. We assume that p remains in the language of the ALP even after all if-rules for p are deleted from it.

"abducible", in the sense that they can be assumed to hold by abduction (of atoms p^*). Note also that we assume that any atom in the body of any added rule but not present in the vocabulary of the ALP prior to the update is treated as a new abducible (in $\mathcal{A}_B$). This choice sees new undefined atoms as "open". Finally, note that we do not allow the addition of new abducible atoms explicitly, from the outside. However, new abducible atoms are added as a by-product of adding definitions for abducible atoms (case 1 above) or definitions of atoms containing the new abducible atoms in the body, as atoms that did not occur before in the ALP (cases 1-2 above).

3.2 Deletion of if-rules

Let update U be the deletion of $p \leftarrow B \in P$ (B possibly $true$). We will assume that B is different from a p^* atom, namely an atom introduced in case 1 in section 3.1. Revising $\langle P, \mathcal{A}, I\rangle$ by U gives $U(\langle P, \mathcal{A}, I\rangle)$=$\langle P', \mathcal{A}, I\rangle$, obtained as follows: $P' = P - \{p \leftarrow B\}$.

3.3 Addition of integrity constraints

Let update U be the addition of $C \Rightarrow D$. Let $\mathcal{A}_U$ be the set of all atoms occurring in C, D but not in the vocabulary of $\langle P, \mathcal{A}, I\rangle$. Revising $\langle P, \mathcal{A}, I\rangle$ by U gives $U(\langle P, \mathcal{A}, I\rangle)$=$\langle P, \mathcal{A}', I'\rangle$, obtained as follows:

- $I' = I \cup \{C \Rightarrow D\}$;
- $\mathcal{A}' = \mathcal{A} \cup \mathcal{A}_U$.

3.4 Deletion of integrity constraints

Let update U be the deletion of $C \Rightarrow D \in I$. Revising $\langle P, \mathcal{A}, I\rangle$ by U gives $U(\langle P, \mathcal{A}, I\rangle)$=$\langle P, \mathcal{A}, I'\rangle$, obtained as follows: $I' = I - \{C \Rightarrow D\}$.

3.5 Multiple updates

Here we define the effect of multiple updates upon an initially given ALP $\langle P, \mathcal{A}, I\rangle$. Let the sequence of updates be $U_1, \ldots, U_n$, with $n > 1$. Then, $U_1, \ldots, U_n$ applied to $\langle P, \mathcal{A}, I\rangle$ gives

$U_n \circ U_{n-1} \circ \ldots \circ U_1(\langle P, \mathcal{A}, I\rangle) = U_n(U_{n-1}(\ldots(U_1(\langle P, \mathcal{A}, I\rangle))))$.

Later, in section 5.2, we will allow empty updates to occur in sequences of updates. An empty update stands for no update at all.

4 AN ILLUSTRATIVE EXAMPLE

In this section we illustrate the application of LIFF and its relationship to IFF via a concrete example. The example also illustrates the potential saving in re-computation, for example, compared to "backtracking" to an appropriate place in the search space.

Given the ALP $\langle P, \mathcal{A}, I\rangle$ with

P: $p \leftarrow r, \quad q \leftarrow s, \quad r \leftarrow t, \quad m \leftarrow \neg a$
$\mathcal{A}$: a, n, s, t
I: $s \wedge m \Rightarrow n$

and a query $p \wedge q$, IFF may derive the following sequence of goals (where, with an abuse of notation, I will stand for $s \wedge m \Rightarrow n$):

$p \wedge q \wedge I$
by unfolding p with $Comp_{\overline{\mathcal{A}}}(P)$:
$r \wedge q \wedge I$
by unfolding q with $Comp_{\overline{\mathcal{A}}}(P)$:
$r \wedge s \wedge I$
by unfolding r with $Comp_{\overline{\mathcal{A}}}(P)$:
$t \wedge s \wedge I$
by propagation with $s \wedge m \Rightarrow n \in I$:
$t \wedge s \wedge (m \Rightarrow n) \wedge I$
by unfolding m with $Comp_{\overline{\mathcal{A}}}(P)$:
$t \wedge s \wedge (\neg a \Rightarrow n) \wedge I$
by negation elimination and splitting:
$[t \wedge s \wedge a \wedge I] \vee [t \wedge s \wedge n \wedge I]$

From the final goal in this derivation, IFF would extract the explanations $\{t, s, a\}$ (from the first disjunct) and $\{t, s, n\}$ (from the second disjunct) for the given query.

Given the same $\langle P, \mathcal{A}, I\rangle$ and query, the analogous LIFF derivation would consist of the following sequence of goals with labels. Labels are formally given in definition 1. A label $\{\langle \mathtt{X_1}; \mathtt{Y_1}\rangle, \ldots \langle \mathtt{X_n}; \mathtt{Y_n}\rangle\}$ associated with a literal conjunct or an implication Z in a goal, intuitively indicates that "if $\mathtt{X_i}$ is updated, then Z should be replaced by $\mathtt{Y_i}$". In the simplest case, each $\mathtt{X_i}$ is an atom or an integrity constraint and each $\mathtt{Y_i}$ is an atom or an implication or $\emptyset$, standing for empty. For ease of reading where the label of a conjunct does not change from one goal to the next in the sequence we do not repeat the label.

$p : \{\} \wedge q : \{\} \wedge I : \{\langle \mathtt{I}; \emptyset\rangle\}$
by unfolding p with $Comp_{\overline{\mathcal{A}}}(P)$:
$r : \{\langle \mathtt{p}; \mathtt{p}\rangle\} \wedge q \wedge I$
by unfolding q with $Comp_{\overline{\mathcal{A}}}(P)$:
$r \wedge s : \{\langle \mathtt{q}; \mathtt{q}\rangle\} \wedge I$
by unfolding r with $Comp_{\overline{\mathcal{A}}}(P)$:
$t : \{\langle \mathtt{p}; \mathtt{p}\rangle, \langle \mathtt{r}; \mathtt{r} : \{\langle \mathtt{p}; \mathtt{p}\rangle\}\rangle\} \wedge s \wedge I$
by propagation (with $s \wedge m \Rightarrow n \in I$):
$t \wedge s \wedge (m \Rightarrow n) : \{\langle \mathtt{q}; \emptyset\rangle, \langle \mathtt{I}; \emptyset\rangle\} \wedge I$
by unfolding m with $Comp_{\overline{\mathcal{A}}}(P)$:
$t \wedge s \wedge (\neg a \Rightarrow n) : \mathtt{label} \wedge I$
where $\mathtt{label}$ is $\{\langle \mathtt{q}; \emptyset\rangle, \langle \mathtt{I}; \emptyset\rangle, \langle \mathtt{m}; (\mathtt{m} \Rightarrow \mathtt{n}) : \{\langle \mathtt{q}; \emptyset\rangle, \langle \mathtt{I}; \emptyset\rangle\}\rangle\}$
by negation elimination and splitting:
$[t \wedge s \wedge a : \mathtt{label} \wedge I] \vee [t \wedge s \wedge n : \mathtt{label} \wedge I]$

Suppose at this stage of the derivation there is an update (addition or deletion) to the definition of p. If we were to simply "backtrack" to where the definition of p was used the first time we would go back to the very first goal of the derivation, namely $p \wedge q \wedge I$ and, effectively, start all over again. But using the labels of LIFF we would simply rewrite the last goal of the derivation as $[p \wedge s \wedge a] \vee [p \wedge s \wedge n]$ (for ease of reading we have dropped the labels here) thus in effect, saving the work done in the application of 4 out of the 6 inferences.

5 THE LIFF PROOF PROCEDURE

Definition 1 A **label** (for an atom or implication) is a set
$$\{\langle \mathtt{X_1}; \mathtt{Y_1}\rangle, \ldots, \langle \mathtt{X_n}; \mathtt{Y_n}\rangle\}$$
$n \geq 0$, with each $\mathtt{X_i}$ an atom or an integrity constraint and each $\mathtt{Y_i}$ either

- empty: $\emptyset$, or
- an atom (possibly with a label), or
- an implication (possibly with a label),

and $\mathtt{X_i} \neq \mathtt{X_j}$ for all $\mathtt{i} \neq \mathtt{j}$.

Note that labels can be empty (i.e. if $n = 0$). Each $\langle \mathtt{X_i}; \mathtt{Y_i}\rangle$ in a label for Z indicates that Z should be replaced by $\mathtt{Y_i}$ if any update is made upon $\mathtt{X_i}$. In effect, this label identifies the components that have contributed to the derivation of Z. We impose that all $\mathtt{X_i}$ are different so that there is no ambiguity as to what Z has to be replaced with when $\mathtt{X_i}$ is updated.

We will refer to labeled atoms/implications simply as atoms/implications. Moreover, the terminology of goals and simple goals will carry through in the presence of labels.

5.1 LIFF inference rules

Given an ALP $\langle P, \mathcal{A}, I\rangle$ and an initial query Q, we will define *LIFF derivations* for Q in terms of sequences of goals, $G_1, \ldots, G_n$, such that $G_1 = Q : \{\} \wedge I^*$, where $(A_1 \wedge \ldots \wedge A_k) : \mathtt{l}$ $(k > 1)$ stands for $A_1 : \mathtt{l} \wedge \ldots \wedge A_k : \mathtt{l}$ [3], $I^* = \{i : \langle \mathtt{i}; \emptyset\rangle | i \in I\}$, and each G_{i+1} is obtained from the previous goal G_i either by enforcing an update (see section 5.2) or by application of one of the inference rules below.

Unfolding an atomic conjunct: given $p \leftrightarrow D_1 \vee \ldots \vee D_n$ in $comp_{\overline{\mathcal{A}}}(P)$ and an atom $p : \mathtt{l}$ which is a conjunct of a simple goal in G_i,
then G_{i+1} is G_i with $p : \mathtt{l}$ replaced by $(D_1 \vee \ldots \vee D_n) : \mathtt{l}'$, where $\mathtt{l}' = \mathtt{l}$ if $\langle \mathtt{p}; \mathtt{S}\rangle \in \mathtt{l}$, for some $\mathtt{S}$, and $\mathtt{l}' = \mathtt{l} \cup \{\langle \mathtt{p}; \mathtt{p} : \mathtt{l}\rangle\}$ otherwise.

Unfolding an atom in the conditions of an implication: given $p \leftrightarrow D_1 \vee \ldots \vee D_n$ in $comp_{\overline{\mathcal{A}}}(P)$ and an implication $[L_1 \wedge \ldots \wedge L_i \wedge \ldots \wedge L_m \Rightarrow A] : \mathtt{l}$ which is a conjunct of a simple goal of G_i with $L_i = p$,
then G_{i+1} is G_i with the implication replaced by the conjunction
$[L_1 \wedge \ldots \wedge D_1 \wedge \ldots \wedge L_m \Rightarrow A] : \mathtt{l}' \wedge \ldots \wedge$
$[L_1 \wedge \ldots \wedge D_n \wedge \ldots \wedge L_m \Rightarrow A] : \mathtt{l}'$,
where $\mathtt{l}' = \mathtt{l}$ if $\langle \mathtt{p}; \mathtt{S}\rangle \in \mathtt{l}$, for some $\mathtt{S}$, and $\mathtt{l}' = \mathtt{l} \cup \{\langle \mathtt{p}; \mathtt{L_1} \wedge \ldots \wedge \mathtt{L_m} \Rightarrow \mathtt{A} : \mathtt{l}\rangle\}$ otherwise.

Propagation: given an atom $p : \mathtt{l}$ and an implication $[L_1 \wedge \ldots \wedge L_i \wedge \ldots \wedge L_m \Rightarrow A] : \mathtt{l}'$ with $L_i = p$, both conjuncts of the same simple goal in G_i,
then, if the implication $[L_1 \wedge \ldots L_{i-1} \wedge L_{i+1} \wedge \ldots \wedge L_m \Rightarrow A] : \{\langle \mathtt{q}; \emptyset\rangle | \langle \mathtt{q}; \mathtt{r}\rangle \in \mathtt{l} \cup \mathtt{l}'\}$ 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 $not\, A_1 \wedge \ldots \wedge not\, A_m \Rightarrow A : \mathtt{l}$ 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 $[A \vee A_1 \vee \ldots \vee A_m] : \mathtt{l}$. [4]

Logical simplification replaces:

- $[B : \mathtt{l} \vee C : \mathtt{l}'] \wedge E : \mathtt{l}''$ by
  $[B : \mathtt{l} \wedge E : \mathtt{l}''] \vee [C : \mathtt{l}' \wedge E : \mathtt{l}'']$ (**splitting**)
- $B : \mathtt{l} \wedge B : \mathtt{l}$ by $B : \mathtt{l}$
- $B : \mathtt{l} \vee B : \mathtt{l}$ by $B : \mathtt{l}$
- $not\, A : \mathtt{l}$, where A is an atom, by $(A \Rightarrow false) : \mathtt{l}$

Note that we are not including some simplification steps present in IFF, e.g. $A \wedge false \equiv false$ and $A \vee false \equiv A$. Such simplification steps only affect the "efficiency" of IFF, rather than its correctness. We leave these steps out in LIFF because it simplifies the labels while allowing us to incorporate updates and revise the goals when necessary (e.g. revising $A \wedge false$, if what led to $false$ is updated). For similar reasons we also do not include a simplification step for "merging" identical conjuncts which have different labels.

Note that for goals in LIFF derivations to be guaranteed to be disjunctions of simple goals, it is sufficient that every step of unfolding and negation elimination is followed by a step of splitting. We will assume this is the case in all derivations.

[3] In the sequel, $(D_1 \vee \ldots \vee D_n) : \mathtt{l}$ will similarly stand for $D_1 : \mathtt{l} \vee \ldots \vee D_n : \mathtt{l}$.

[4] In [3], 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 (ignoring the labels) is equivalent to the one in [3].

5.2 LIFF derivations

In LIFF derivations, the application of inference rules (as given in section 5.1) is interleaved with the assimilation of updates within the ALP (as given in section 3) and the application of these updates to goals, defined as follows:

Definition 2 Given a goal G and an update U, the **updated goal wrt** G **and** U is a goal G' obtained as follows:

- if U is the addition or deletion of an if-rule $p \leftarrow B$, then G' is G where every conjunct with $\langle p; Q\rangle$ in its label is replaced by Q;
- if U is the deletion of an integrity constraint X then G' is G where every conjunct with label containing $\langle X; Q\rangle$ is replaced by Q (Q will be $\emptyset$ in all such cases);
- if U is the addition of an integrity constraint i then G' is G with an added conjunct $i : \{\langle i; \emptyset\rangle\}$.

In all cases, any resulting conjunct which is $\emptyset$ is deleted.

Definition 3 Given a query Q, an ALP $\langle P, \mathcal{A}, I\rangle$, and a sequence of updates $U_1, \ldots, U_n$, $n > 0$, a **LIFF derivation** is a sequence

$$(G_1, U_1, R_1, ALP_1), \ldots, (G_n, U_n, R_n, ALP_n)$$

where, for each $1 \leq i \leq n$,

- G_i is a goal
- U_i is an update (possibly empty)
- R_i is empty or (an instance of) an inference rule applied to a simple goal in G_i which does not have $false$ as a conjunct
- exactly one of U_i and R_i is non-empty
- ALP_i is an ALP

and $G_1 = Q : \{\} \wedge I^*$ (where I^* is defined in section 5.1), $ALP_1 = \langle P, \mathcal{A}, I\rangle$, and, for each $1 < i \leq n$,

- if R_{i-1} is non-empty then
  - G_i is obtained from G_{i-1} by applying R_{i-1}
  - $ALP_i = ALP_{i-1}$
- if U_{i-1} is non-empty then
  - G_i is the updated goal wrt G_{i-1} and U_{i-1}
  - $ALP_i = U_{i-1}(ALP_{i-1})$.

The **end goal** and **end ALP** in a LIFF derivation

$$(G_1, U_1, R_1, ALP_1), \ldots, (G_n, U_n, R_n, ALP_n)$$

are G and ALP, respectively, such that:

- if R_n is non-empty then
  - G is obtained from G_n by applying R_n
  - $ALP = ALP_n$
- if U_n is non-empty then
  - G is the updated goal wrt G_n and U_n
  - $ALP = U_n(ALP_n)$.

We will refer to a LIFF derivation whose updates are all empty as a **static LIFF derivation**.

5.3 Successful LIFF derivations and extracted answers

Here we formalise the notion of answer extraction for LIFF, by adapting the notion of answer extraction for IFF to take labels into account.

Given a LIFF derivation for a query Q, ALP $\langle P, \mathcal{A}, I\rangle$ and a given sequence of updates, let G be the end goal of the derivation. Let D be a disjunct of G:

- if no inference rule can be applied to D, then D is called **conclusive**;
- if $false : \mathtt{1}$ is a conjunct in D, then D is called **failed**;
- if D is conclusive and not failed then it is called **successful**.

Then, the derivation is **successful** iff there exists a successful disjunct D in G. An **answer extracted from** a successful LIFF-derivation wrt a query Q, an ALP and a sequence of updates is the set of all abducible atoms, without labels, in a successful disjunct D in the end goal of the derivation.

6 SOUNDNESS OF THE LIFF PROOF PROCEDURE

Theorem 1 (Soundness) *Let E be an answer extracted from a successful LIFF-derivation wrt Q, $\langle P, \mathcal{A}, I\rangle$and a sequence of updates. Let $\langle P', \mathcal{A}', I'\rangle$ be the ALP resulting after the sequence of updates. Then E is an explanation for Q, wrt $\langle P', \mathcal{A}', I'\rangle$.*

The proof of this theorem relies upon the following lemmas and the correctness of the IFF proof procedure.

Lemma 1 *Every non-static LIFF derivation wrt Q, $\langle P, \mathcal{A}, I\rangle$, $U_1, \ldots, U_m$, with end goal G and end ALP $\langle P', \mathcal{A}', I'\rangle$, can be mapped onto a static LIFF derivation wrt Q, $\langle P', \mathcal{A}', I'\rangle$, ending at (a simplified version of) G.*

Lemma 2 *Every static LIFF derivation wrt Q, $\langle P, \mathcal{A}, I\rangle$, can be mapped onto an IFF derivation wrt Q.*

7 CONCLUSIONS AND FUTURE WORK

In this paper we have proposed a dynamic abductive logic programming proof procedure, called LIFF, which modifies the existing proof procedure IFF by adding labels to goals. The labels keep track of dependencies amongst the atoms and implications in goals and between these and the logic program. By keeping track of these dependencies, after updating the ALP, LIFF can keep parts of the reasoning that have not been affected by the updates and determine how best to replace those parts that have been affected. It thus allows reasoning in dynamic environments and contexts without having to discard earlier reasoning when changes occur. We have considered updates consisting of addition and deletion of if-rules and integrity constraints.

LIFF is particularly good at saving earlier reasoning effort when the definitions of relatively lower level atoms are modified. This kind of update is particularly prominent in the case when we use abductive logic programming and agent-based techniques for information integration [19]. In such an application the higher level atoms are user-level and auxiliary atoms, and the lower level atoms are related to information sources. Changes in various aspects of the information sources, for example content or availability, amount to updating these lower level atoms.

Our work on LIFF shares the same objectives as that of [4]. However whereas we adapt IFF for abductive logic programming, they adapt conventional SLDNF for ordinary logic programs. We also share some of the objectives of [15, 16]. In their work on speculative computation they allow reasoning to proceed with default values for specific facts and then accommodate updates which replace the default values, attempting to keep some of the earlier computation. They, however, use SLD derivation for Horn theories (no negation in rules and no integrity constraints). They also keep a form of dependency, but it is at a much coarser level compared to ours, as the dependency information is for the whole goal, rather than for the individual conjuncts in goals, thus allowing fewer savings in computations.

We have described LIFF and a soundness result for propositional abductive logic programs. Work is currently in progress for the predicate case, catering for the additional inference rules of factoring, case analysis and equality rewriting. Completeness results are subject of future work. It would also be worth exploring how this work scales up to substantial applications that would require large knowledge bases and frequent updates.

REFERENCES

[1] C. E. Alchourron, P. Gardenfors, and D. Makinson, 'On the logic of theory change: Partial meet functions for contraction and revision', *Journal of Symbolic Logic*, **50**, 510–530, (1985).

[2] K. L. Clark, 'Negation as Failure', in *Logic and Data Bases*, 293–322, Plenum Press, (1978).

[3] T. H. Fung and R. A. Kowalski, 'The IFF proof procedure for abductive logic programming', *Journal of Logic Programming*, **33**(2), 151–165, (1997).

[4] H. Hayashi, 'Replanning in robotics by dynamic SLDNF', in *Proc. Workshop on Scheduling and Planning meet Real-time Monitoring in a Dynamic and Uncertain World*, (1999).

[5] H. Hayashi, K. Cho, and A. Ohsuga, 'Integrating planning, action execution, knowledge updates and plan modifications via logic programming', in *Proc. ICLP*, volume 2401 of *LNCS*. Springer-Verlag, (2002).

[6] A. C. Kakas, R. A. Kowalski, and F. Toni, 'The role of abduction in logic programming', in *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 5, 235–324, Oxford University Press, (1998).

[7] A.C. Kakas, P. Mancarella, F. Sadri, K. Stathis, and F. Toni, 'The KGP model of agency', in *Proc. ECAI*, pp. 33–37. IOS Press, (2004).

[8] R. A. Kowalski and F. Sadri, 'From logic programming towards multi-agent systems', *Annals of Mathematics and Artificial Intelligence*, **25**(3/4), 391–419, (1999).

[9] K. Kunen, 'Negation in logic programming', in *Journal of Logic Programming*, volume 4, pp. 289–308, (1987).

[10] P. Mancarella, F. Sadri, G. Terreni, and F. Toni, 'Planning partially for situated agents', in *CLIMA V*, pp. 230–248, (2004).

[11] F. Sadri and F. Toni, 'Abduction with negation as failure for active and reactive rules', in *Proc. AI*IA*, number 1792 in LNAI, pp. 49–60. Springer-Verlag, (2000).

[12] F. Sadri and F. Toni, 'Interleaving belief revision and reasoning: preliminary report', in *Proc. CILC*, (2005).

[13] F. Sadri, F. Toni, and P. Torroni, 'An abductive logic programming architecture for negotiating agents', in *Proc. JELIA*, volume 2424 of *LNCS*, pp. 419–431. Springer-Verlag, (2002).

[14] F. Sadri, F. Toni, and P. Torroni, 'Dialogues for negotiation: agent varieties and dialogue sequences', in *Proc. ATAL*, volume 2333 of *LNAI*, pp. 405–421. Springer-Verlag, (2002).

[15] K. Satoh, K. Inoue, K. Iwanuma, and C Sakama, 'Speculative computation by abduction under incomplete communication environments', in *Proc. ICMAS*, pp. 263–270, (2000).

[16] K. Satoh and K. Yamamoto, 'Speculative computation with multi-agent belief revision', in *Proc. AAMAS*, pp. 897–904, (2002).

[17] F. Toni, 'Automated information management via abductive logic agents', *Journal of Telematics and Informatics*, **18**(1), 89–104, (2001).

[18] F. Toni and K. Stathis, 'Access-as-you-need: a computational logic framework for flexible resource access in artificial societies', in *Proc. ESAW*, LNAI, pp. 126–140. Springer-Verlag, (2002).

[19] I. Xanthakos, *Semantic Integration of Information by Abduction*, Ph.D. dissertation, Imperial College London, 2003.