Accountable Protocols in Abductive Logic Programming

published in
ACM Transactions on Internet Technology 18(4), 46, 2018

Marco Gavanelli\textsuperscript{1}  Marco Alberti\textsuperscript{2}  Evelina Lamma\textsuperscript{1}

\textsuperscript{1}Department of Engineering
\textsuperscript{2}Department of Mathematics and Computer Science
Contents

1 Motivation
   - Accountability

2 SCIFF
   - SCIFF Recap
   - SCIFF Violation Answer

3 Defining Accountability in SCIFF
   - Accountable agent
   - Reply to an account
   - Accountable protocol

4 Ensuring Accountability
   - Syntactic features to ensure accountability
   - Checking accountability
The problem of many hands ...

1984, Union Carbide, Bhopal (India): release of methyl isocyanate. 3,787 deaths, 40,000 individuals permanently disabled. The CEO refused to answer to homicide charges. The U.S. denied several extradition requests.

Bovens (1998):

My point of departure is what Dennis Thompson (1980) called the problem of many hands: because in complex organisations many different functionaries [...] often contribute to the policy and decisions of the organisation, it is often extraordinarily difficult to determine who is responsible for the organisation’s conduct
Accountability in the literature of moral responsibility

Van de Poel et al. [2015]: five moral meanings of responsibility: accountability, blameworthiness, liability, obligation and virtue.

**Accountability (Van de Poel et al. [2015])**

[Agent] $i$ is responsible-as-accountable for [action or state of affairs] $\varphi$ implies that $i$ should account for (the occurrence of) $\varphi$, in particular for $i$’s role in doing, or bringing about $\varphi$, or for $i$’s role in failing to prevent $\varphi$ from happening.
Van de Poel et al. [2015]

Accountability implies blameworthiness unless the accountable agent can show that a reasonable excuse applies that frees her from blameworthiness. So holding an agent i accountable shifts the burden of proof for showing that i is not blameworthy to the agent i: the agent is now to show – by giving an account – that she is not blameworthy.

Account for the occurrence of \( \varphi \)

I can prove I’m not blameworthy
Abductive Logic Programming (ALP)

- An ALP program $\mathcal{P}$ is a triple:
  \[ \langle KB, A, IC \rangle \]

- $KB$, set of LP clauses having (in their body) some distinguished atoms built from the set of abducibles predicates $A$

- $IC$, set of logic formulae, called *Integrity Constraints*

- Given a goal $G$, find a set of abducible atoms $\Delta$, such that:
  \[ KB \cup \Delta \models G \]

Also, $\Delta$ and $KB$ should also satisfy $IC$:

\[ KB \cup \Delta \models IC \]
Motivation

SCIFF

Defining Accountability in SCIFF

Ensuring Accountability

SCIFF Recap

SCIFF Violation Answer

SCIFF program: $\langle KB, IC \rangle$

$KB$, a logic program to express domain specific knowledge

$IC$, a set of (forward) implications ($Body \rightarrow Head$) to constrain the expected behaviour of the interacting parties

Happened events, $HAP$: a set of (ground) atoms of kind $H(do(Sender, Receiver, Content))$, representing the actual behaviour

Abducibles, named expectations, represent expected behaviour:

- $E(do(Sender, Receiver, Content))$: positive expectations, with a deontic reading of obligation ($\exists$, if variables in arguments)
- $EN(do(Sender, Receiver, Content))$: negative expectations, read as prohibition ($\forall$, if variables in arguments)
- $\neg E(do(Sender, Receiver, Content))$: negation of positive expectation, or explicit absence of obligation
- $\neg EN(do(Sender, Receiver, Content))$: negation of negative expectation, or explicit permission
Motivation

SCIFF

Defining Accountability in SCIFF
Ensuring Accountability

SCIFF Recap

SCIFF Violation Answer

SCIFF answer

An abductive answer is a SCIFF answer if it satisfies

\[ KB \cup HAP \cup \Delta \models G \]
\[ KB \cup HAP \cup \Delta \models IC \]

\[ KB \cup HAP \cup \Delta \models E(X) \land \neg E(X) \rightarrow false \]
\[ KB \cup HAP \cup \Delta \models EN(X) \land \neg EN(X) \rightarrow false \] (explicit negation)

positive and negative expectations cannot be contradicting

\[ KB \cup HAP \cup \Delta \models E(X) \land EN(X) \rightarrow false \]

and the set of expectations should be fulfilled by the actual history

\[ KB \cup HAP \cup \Delta \models E(X) \rightarrow H(X) \]
\[ KB \cup HAP \cup \Delta \models EN(X) \land H(X) \rightarrow false \]
An abductive answer is a SCIFF violation answer if it satisfies

\[ KB \cup HAP \cup \Delta \models G \]  
\[ KB \cup HAP \cup \Delta \models IC \]  

(Abduction)

\[ KB \cup HAP \cup \Delta \models E(X) \land \lnot E(X) \rightarrow false \]  
\[ KB \cup HAP \cup \Delta \models EN(X) \land \lnot EN(X) \rightarrow false \]  

(explicit negation)

positive and negative expectations cannot be contradicting

\[ KB \cup HAP \cup \Delta \models E(X) \land EN(X) \rightarrow false \]

and it does not satisfy at least one of the following

\[ KB \cup HAP \cup \Delta \models E(X) \rightarrow H(X) \]  
\[ KB \cup HAP \cup \Delta \models EN(X) \land H(X) \rightarrow false \]
**Accountability in SCiFF**

**Definition (Accountable agent)**

An agent $A$ is **Accountable** if there exists no SCiFF Answer, and there exists a Violation Answer $\Delta^\vee$ such that $E(\text{do}(A, _, _)) \in \Delta^\vee$ or $EN(\text{do}(A, _, _)) \in \Delta^\vee$.

$$H(\text{do}(alice, bob, a_1)) \rightarrow E(\text{do}(bob, alice, a_2)) \land EN(\text{do}(Z, bob, forbid)) \lor E(\text{do}(bob, alice, a_3))$$

[Diagram of the accountability process involving Alice, Bob, and John with actions $a_1$, $a_2$, $a_3$, and a 'forbid!' action.]
Motivation
Defining Accountability in SCIFF
Ensuring Accountability

Accountability in SCIFF

Definition (Accountable agent)

An agent \( A \) is **Accountable** if there exists no SCIFF Answer, and there exists a Violation Answer \( \Delta^\forall \) such that \( E(do(A, _, _)) \in \Delta^\forall \) or \( EN(do(A, _, _)) \in \Delta^\forall \).

\[
H(do(alice, bob, a_1)) \rightarrow E(do(bob, alice, a_2)) \land EN(do(Z, bob, forbid))
\land E(do(bob, alice, a_3))
\]

Alice
\( a_1 \)
Bob
forbid!
John
Accountability in SCIFF

**Definition (Accountable agent)**

An agent $A$ is *Accountable* if there exists no SCIFF Answer, and there exists a Violation Answer $\Delta^\forall$ such that $E(\text{do}(A,\_\_,\_\_)) \in \Delta^\forall$ or $\text{EN}(\text{do}(A,\_\_,\_\_)) \in \Delta^\forall$.

$$H(\text{do}(alice, bob, a_1)) \rightarrow E(\text{do}(bob, alice, a_2)) \land \text{EN}(\text{do}(Z, bob, forbid)) \lor E(\text{do}(bob, alice, a_3))$$
Accounting the two agents

\[ \text{H}(\text{do}(\text{alice, bob, } a_1)) \rightarrow \text{E}(\text{do}(\text{bob, alice, } a_2)) \land \text{EN}(\text{do}(Z, \text{bob, forbid})) \lor \text{E}(\text{do}(\text{bob, alice, } a_3)) \]

Alice \quad a_1 \quad Bob

forbid!

Account: Why did you say forbid?

Bob \quad a_2a_3
Accounting the two agents

$$H(\text{do}(alice, bob, a_1)) \rightarrow E(\text{do}(bob, alice, a_2)) \land EN(\text{do}(Z, bob, forbid)) \lor E(\text{do}(bob, alice, a_3))$$
Accounting the two agents

\[ H(\text{do}(alice, bob, a_1)) \rightarrow E(\text{do}(bob, alice, a_2)) \land EN(\text{do}(Z, bob, forbid)) \lor E(\text{do}(bob, alice, a_3)) \]

Alice

Bob

John

Account: Why did you say forbid?

Bob should have said \( a_3 \)

ok

John

Account: Why didn’t you say \( a_2 \) nor \( a_3 \)?
Accounting the two agents

\[ H(\text{do}(alice, bob, a_1)) \rightarrow E(\text{do}(bob, alice, a_2)) \land EN(\text{do}(Z, bob, forbid)) \lor E(\text{do}(bob, alice, a_3)) \]

Alice

Bob

John

Account: Why did you say forbid?

ok

Bob should have said \( a_3 \)

Account: Why didn’t you say \( a_2 \) nor \( a_3 \)?

???

Gavanelli, Alberti and Lamma
**Motivation**

SCIFF

**Defining Accountability in SCIFF**

**Ensuring Accountability**

**Accountable agent**

**Reply to an account**

**Accountable protocol**

---

## Accounting

\[ H(\text{do}(alice, bob, a_1)) \rightarrow E(\text{do}(bob, alice, a_2)) \land EN(\text{do}(Z, bob, forbid)) \lor E(\text{do}(bob, alice, a_3)) \]

### Definition (Reply to an account)

If there exists no SCIFF answer for a given protocol, an accounted agent \( A \) can account for its behavior if there exists a violation answer \( \Delta^v \) (called *Reply R to an account*) such that

\[ \not\exists E(\text{do}(A, _, _)) \in \Delta^v \text{ and } \not\exists EN(\text{do}(A, _, _)) \in \Delta^v. \]

If there exists no reply to an account, the accounted agent \( A \) is indicted.

---

**Why did you say forbid?**

**If Bob had said a\(_3\) there would have been no violation**

**ok**

---

Gavanelli, Alberti and Lamma
Definition (Reply to an account)

If there exists no SCI\(\text{IFF}\) answer for a given protocol, an accounted agent \(A\) can account for its behavior if there exists a violation answer \(\Delta^v\) (called \textit{Reply R to an account}) such that

\[
\not\exists \mathbf{E}(\text{do}(A, _, _)) \in \Delta^v \quad \text{and} \quad \not\exists \mathbf{EN}(\text{do}(A, _, _)) \in \Delta^v.
\]

If there exists no reply to an account, the accounted agent \(A\) is \textit{indicted}.

Why didn’t you say \(a_2\) nor \(a_3\)?
Another example

\[ H(\text{do}(alice, bob, a_1)) \rightarrow E(\text{do}(bob, alice, a_2)) \land EN(\text{do}(Z, bob, forbid)) \lor E(\text{do}(bob, alice, a_3)) \]
Another example

\[ H(do(alice, bob, a_1)) \rightarrow E(do(bob, alice, a_2)) \land EN(do(Z, bob, forbid)) \lor E(do(bob, alice, a_3)) \]
Another example

$$H(do(alice, bob, a_1)) \rightarrow E(do(bob, alice, a_2)) \land EN(do(Z, bob, forbid)) \lor E(do(bob, alice, a_3))$$
Another example

\[ H(do(alice, bob, a_1)) \rightarrow E(do(bob, alice, a_2)) \land \text{EN}(do(Z, bob, forbid)) \lor E(do(bob, alice, a_3)) \]

No one is indicted, but still there is a violation. This is a badly designed protocol.
Accountable protocol

Definition (Accountable protocol)

A protocol $\mathcal{P}$ is accountable if for each history $\mathcal{HAP}$ such that $\mathcal{P}_{\mathcal{HAP}} \not\models G$ the set of indicted agents is not empty.

How can we ensure accountability of a protocol?

- Restrict syntax to ensure accountability
- check a posteriori if a protocol is (not) accountable
Accountable protocol

Definition (Accountable protocol)

A protocol \( \mathcal{P} \) is accountable if for each history \( \text{HAP} \) such that \( \mathcal{P}_{\text{HAP}} \not\models G \) the set of indicted agents is not empty.

How can we ensure accountability of a protocol?

- Restrict syntax to ensure accountability
- Check a posteriori if a protocol is (not) accountable
Definition (Consequential protocol)

A protocol is *consequential* if it consists of a set of integrity constraints

\[
\neg H(do(X_1^b, Y_1^b, A_1^b, T_1^b)) \land \ldots \land \neg H(do(X_n^b, Y_n^b, A_n^b, T_n^b)) \land \\
c_1^b \land c_2^b \land \ldots \land c_p^b \rightarrow c_1^h \land c_2^h \land \ldots \land c_q^h \land \\
\neg E/EN(do(C, Y_1^h, A_1^h, T_1^h)) \lor \ldots \lor \neg E/EN(do(C, Y_m^h, A_m^h, T_m^h))
\]

where \( C \) occurs in at least one \( H \) atom in the body, and where \( c_i^h \) and \( c_i^b \) can be either CLP constraints or predicates that are defined in the \( KB \) and that do not depend on abducible predicates.

Intuitively:

- **Body** does not contain expectations
- all expectations in the **Head** are about the same agent
- the name of that agent is ground when the IC is triggered
Theorem: consequential $\Rightarrow$ accountable

**Theorem**

A consequential protocol is accountable.

**Proof.**

See [Gavanelli et al., ACM Transactions on Internet Technologies, 2018].
Protocols published in previous works

<table>
<thead>
<tr>
<th>Protocol name</th>
<th>Consequential</th>
<th>Accountable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication primitive semantics</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Negotiation (stages 1&amp;2)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>FIPA Query-Ref</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>E-commerce choreography</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>First-Price Sealed Bid Auction</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>English Auction</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Needham-Schroeder</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>NetBill</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Software license agreement</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>E-commerce protocol</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>E-commerce protocol</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Checking accountability

Data: A protocol
Result: accountable

accountable ← true;
while accountable do
    generate a history HAP;
    run SCIFF;
    if not obtained SCIFF answer then
        compute all SCIFF violation answers;
        $S \leftarrow \bigcap \{\text{indicted agents of violation answers}\}$;
        if $S = \emptyset$ then
            accountable ← false;
        end
    end
end

Similar to Bounded Model Checking. Can identify non-accountability. Loops in case the protocol is accountable.
Results

The previous algorithm was able to detect non-accountability in a First-Price Sealed Bid protocol, containing the IC:

\[
\begin{align*}
H(do(Auctioneer, Bidders, opauc(Item, T_{\text{dead}}, T_{\text{notify}}, fpsb)), T_{\text{open}}),
\quad &
H(do(Bidder_1, Auctioneer, bid(Item, Q_1)), T_1), T_1 < T_{\text{dead}} \\
\rightarrow &
E(do(Bidder_2, Auctioneer, bid(Item, Q_2)), T_2),
\quad Q_2 > Q_1, T_2 < T_{\text{dead}} \\
\lor &
E(do(Auctioneer, Bidder_1, answ(win, Item, Q_1)), T_{\text{win}}),
\quad T_{\text{win}} < T_{\text{dead}} + T_{\text{notify}}
\end{align*}
\]
## Results

<table>
<thead>
<tr>
<th>History depth</th>
<th>FPSB Auction (Original)</th>
<th>FPSB Auction (Revised)</th>
<th>English Auction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>✓ (2ms)</td>
<td>✓ (1ms)</td>
<td>✓ (1ms)</td>
</tr>
<tr>
<td>1</td>
<td>✓ (33ms)</td>
<td>✓ (23ms)</td>
<td>✓ (44ms)</td>
</tr>
<tr>
<td>2</td>
<td>✓ (48ms)</td>
<td>✓ (249ms)</td>
<td>✓ (103ms)</td>
</tr>
<tr>
<td>3</td>
<td>✓ (3,727ms)</td>
<td>✓ (3,727ms)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>✓ (335,073ms)</td>
<td>✓ (335,073ms)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>timeout (1,000,000ms)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

✓ = accountability not disproved  
✗ = accountability disproved
Conclusions

- Accountability important in human societies, will be even more in software agent societies (self-driving cars, etc.)
- Notion of accountability taken from the field of ethics and philosophy
- formally defined within SCIIF
- syntactic features that ensure accountability
- software tool to check accountability
Thank you for your attention
Thank you for your attention