Towards a Formalization of Responsibility

Tiago de Lima$^1$, Lambèr Royakkers$^1$, and Frank Dignum$^2$

$^1$ Eindhoven University of Technology, The Netherlands
$^2$ Utrecht University, The Netherlands

Abstract. In this paper we aim at formalizing two distinct notions of responsibility: backward-looking responsibility and forward-looking responsibility. The first notion implies blameworthiness while the second one implies obligation. They are related to each other though, since, under certain conditions, forward-looking implies backward-looking. We claim here that an ideal framework of cooperative agency should be able to express both notions and also capture this relation. We provide such a framework by using an extension of alternating-time temporal logic called CATL. It turns out that the possibility of explicitly use strategies in the object language of CATL provides a useful tool to attend our aim.

1 Introduction

When a group of agents joins forces to achieve a common goal but the expected outcome is not obtained, a natural reaction of the group is to try to determine who is “responsible” for the failure. Sometimes, it turns out to be a difficult task, since the meaning of the term ‘responsibility’ is often vague. We believe that a formalization of the concept of responsibility can be helpful in such situations. For instance, it can contribute to this discussion by giving some insights, clarifying and classifying meanings of responsibility. It can also provide a framework where criteria for holding agents responsible for failures can be stated, and justifications can be made clear.

The term ‘responsibility’ is used to designate several different notions [1]. Here we focus the attention in only two of them. The first, and perhaps the most common use of this term, is a notion that implies guilty, or blameworthiness. It is sometimes refereed as ‘moral responsibility’, but we prefer to call it ‘backward-looking responsibility’. As a practical example, suppose that a vehicle produced by a certain automobile company is defective. Also suppose that this vehicle has been sold to a customer who has detected the defect and is now asking for refund. A natural reaction of the company is to try to determine who is “responsible” for the defective product in order to avoid that such undesirable situation happens again. Following in part the definition proposed by [2] (see also [3]), an employee (or a group of employees) is backward-looking responsible for the occurrence of the defect in the product if and only if he/she (or they) freely, knowingly and intentionally behaves in such a way that is necessary for the occurrence of the defect.
The second notion designated by the term ‘responsibility’ is a distinct notion, that implies duty, or obligation. It is sometimes refereed as ‘task-based responsibility’, but we prefer to call it ‘forward-looking responsibility’. As a practical example, suppose that during a meeting, the president of the automobile company mentioned above turns to one of the employees and says: “— You are responsible for refunding our customers...”. This is a direct order. There is no guilty or blameworthiness in what is meant by the term ‘responsible’ in this case. Here instead, responsibility implies obligation: the president mean that the employee must perform such a task.

There is an interesting issue involving forward-looking responsibility that we will also address here. This notion inherits from obligations the distinction between ought-to-do and ought-to-be statements. That is, forward-looking responsibility can either be seen as regarding events or as regarding states of affairs. To see this distinction, compare the above order with the following one: “...and you are responsible for ensuring that the next product will be functioning well!”. In the latter order the particular task to be performed is not defined. The employee must figure it out by himself. Whatever decision is taken by the employee, the only fixed aim is that certain result must eventually be obtained.

Therefore, we have at least three different kinds of responsibility: backward-looking responsibility, forward-looking responsibility regarding state of affairs and forward-looking responsibility regarding events. In addition, we claim here that all of them are somehow related to each other, and thus an ideal formalization of responsibility should be able to express this relation. We argue in favour of this claim by using yet another example, adapted from [4]: consider that an agent is held forward-looking responsible for maintaining the balance of certain bank account positive. In this example, once the balance is negative, the agent immediately gets the forward-looking responsibility of making a deposit. That is, the forward-looking responsibility regarding the state of affairs ‘the balance is positive’ eventually implies the responsibility regarding the action ‘make a deposit’.

To show the relation between forward-looking and backward-looking we consider again the example of the automobile company. Suppose that the employee is held forward-looking responsible for ensuring that the product is functioning well but, in the end of the fabrication process, the new product is defective. Then, unless something out of the employee’s control has happened, he/she is morally responsible for such state of affairs. That is, forward-looking responsibility implies backward-looking responsibility, provided that the agent is able to control the situation.

In the next section we briefly present some earlier approaches to responsibility found in the literature. After that, we build up a logical framework for responsibility based on an extension of alternating-time temporal logic, called CATL. The semantics and syntax of CATL is presented in Section 3. In the sequel (Section 4), we show a way of expressing obligations in CATL. Obligations are used in order to represent forward-looking responsibility. Also the distinction between responsibilities regarding states of affairs and regarding events is ad-
dressed in this section. Section 5 shows how attempts can be expressed in CATL. Attempts are used to represent backward-looking responsibility. Also the relation between the latter and forward-looking responsibility is addressed in this section. Finally, Section 6 draws conclusions and discuss a possible future work.

2 Some Formal Approaches to Responsibility

In Section 1 we inevitably mentioned some of the necessary ingredients for a formal account of responsibility. Namely, backward-looking responsibility is defined using the concepts of free will, knowledge and intentions, while the example used to illustrate forward-looking responsibility at least suggests the need for obligations and actions. Also note that backward-looking responsibility is defined in terms of some event occurred in the past. Therefore, we also need to deal with the passage of time. In this work, we simplify the scenario and suppose that the requirements of free will and knowledge are always met. That is, we assume that in every situation, the agents act freely and they are able to foresee all the consequences of their actions.

In the simplified scenario though, we still need to have actions, intentions, obligations and time. Earlier works on the formalization of responsibility often do not deal with all of these ingredients. For instance, [5] deal only with obligations and agents intentions. They propose that responsibility should be seen as ‘obligation to ensure’. Their formalization is done by using a logic wherein one can write formulas of the form OE_aϕ, which stand for ‘it is obligatory that a ensures ϕ’. The most interesting feature of this approach is the validity of the scheme E_aE_bϕ → E_aϕ. It expresses that ‘if agent a ensures that agent b ensures ϕ, then agent a ensures ϕ’, which is a useful feature for modelling indirect agency. However, Santos & Carmo’s logic abstracts from time passage. It means that we cannot express backward-looking responsibility in this logic, since this concept is defined regarding some event in the past (cf. example in Section 1).

The logic for agent organization (LAO) [6] incorporates temporal operators. The authors propose to formalize responsibility in their logic by defining it in terms of ‘agents attempts’. It seems to be a better alternative than ‘obligation to ensure’ because the former avoids considering that the agent violates its responsibility in the case he is not, in fact, able to control the situation. However, LAO does not have actions in its language. Therefore, one cannot express forward-looking responsibility regarding events.

Grossi et al. [7] propose a logic with explicit reference to actions in its language. It is based on a combination of dynamic logic [8] and epistemic logic [9]. Its main feature is the possibility of expressing agent’s knowledge, that is one of the conditions for backward-looking responsibility. However, this logic does not permit to express that ‘agent a is able to bring about (or to avoid) ϕ’. This feature is important in order to define backward-looking responsibility adequately, since it is not “fair” to hold and agent responsible for an inevitable event, or even for any event that is beyond the agent’s control.
3 Alternating-Time Temporal Logic with Strategies

To build up our framework we use the logic CATL proposed by [10]. This is an extension of the alternating-time temporal logic (ATL), originally proposed by [11, 12]. The latter logic is intended to model multi-agent systems and allows reasoning about agents abilities and coalitions. CATL extends ATL by incorporating strategies. In CATL one can write formulas of the form $[s_C]\varphi$, which stand for ‘if the coalition $C$ follows the strategy $s$, then $\varphi$ holds’. This kind of statement is useful in the formalization of backward-looking responsibility, since one can ask whether a group of agents is able to contribute to bring about, or avoid, certain state of affairs. Strategies are also useful in the formalization of forward-looking responsibility, since they can also be seen as dynamic operators in the sense of the modal operators of dynamic logic, thus allowing a kind of reasoning about agent actions.

3.1 Models

The semantics of CATL is based on action-based alternating-time transition systems, hereafter simply called models. Assume a countable non-empty set $P$ of proposition letters, a finite non-empty set $Ag$ of agents, a countable non-empty set $Ac$ of actions, and a countable non-empty set $S$ of strategy terms. Moreover, let $J_{Ag}$ be the set of all joint actions available for $Ag$, i.e., the set of all total functions $\alpha : Ag \rightarrow Ac$, and let $\Sigma$ be the set of all available joint strategies, i.e., the set of all total functions $\sigma : W \rightarrow J_{Ag}$ such that for each $\sigma(w)$ there is $w' \in W$ such that $T(w, \sigma(w)) = w'$. Models are structures of the form $\langle W, T, I, V \rangle$ where:

- $W$ is a non-empty set of states (or possible worlds);
- $T : (W \times J_{Ag}) \rightarrow W$, is a partial transition function that for each pair $(w, \alpha)$ defines the state resulting from the performance of the joint action $\alpha$ in the possible world $w$;
- $I : S \rightarrow \Sigma$, is a function that defines the interpretation of strategy terms in the model; and
- $V : P \rightarrow \mathcal{P}(W)$, is a valuation function that defines the interpretation of each proposition letter in the model.

**Example 1.** Consider a scenario where two agents can control the state of a light bulb by toggling a switch. Let $P = \{\text{light}\}$, with light representing that the light is on and $\neg$light representing that the light is off. Moreover, let $Ag = \{a, b\}$, $Ac = \{\text{skip, toggle}\}$, with skip meaning ‘skip’ (do nothing) and toggle meaning ‘toggles the switch’, and $S = \{s\}$. This scenario can be modelled by the model $M = (W, T, I, V)$ where $W = \{w_0, w_1\}$, $I = \{(s, \sigma)\}$, for some $\sigma$ such that $\sigma(w_0)(a) = \text{skip}$, $V = \{(\text{light}, \{w_0\})\}$, and each line of Table 1 represents one element of $T$, e.g., from the first line, $T(w_0, \{(a, \text{skip}), (b, \text{skip})\}) = w_0$. The model $M$ is also graphically represented in Figure 1.

Assuming that we are currently in $w_0$, the next state depends on which actions $a$ and $b$ decide to undertake. If both decide to skip, the light will be on...
Table 1. Transition function used in Example 1.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>w₀</td>
<td>skip</td>
<td>skip w₀</td>
</tr>
<tr>
<td>w₀</td>
<td>skip</td>
<td>toggle w₁</td>
</tr>
<tr>
<td>w₀</td>
<td>toggle</td>
<td>skip w₁</td>
</tr>
<tr>
<td>w₀</td>
<td>toggle</td>
<td>toggle w₁</td>
</tr>
<tr>
<td>w₁</td>
<td>skip</td>
<td>skip w₁</td>
</tr>
<tr>
<td>w₁</td>
<td>skip</td>
<td>toggle w₀</td>
</tr>
<tr>
<td>w₁</td>
<td>toggle</td>
<td>skip w₀</td>
</tr>
<tr>
<td>w₁</td>
<td>toggle</td>
<td>toggle w₀</td>
</tr>
</tbody>
</table>

in the resulting state, but if at least one of them decides to toggle the switch, the light will be off in the next state.

This kind of model can be used to verify how agents can influence the state of the system. By assuming that the agents have complete knowledge about the system and all action are perceived by all agents, one can also use this kind of model to verify “intentions” of agents. The latter can be done by verifying ‘what would be true in the next state if the agent had decided to act in a different way’. Or, reformulating in other words, one can verify ‘what the agent has decided to bring about’. And this is the tool that we use in this paper to model responsibility. First though, we need a language wherein these statements can be expressed.

### 3.2 Language

The set $\mathcal{L}_s$ of state formulas and the set $\mathcal{L}_p$ of path formulas are the smallest sets that satisfy the following clauses:

- if $p \in P$, then $p \in \mathcal{L}_s$;
- if $\varphi \in \mathcal{L}_s$, then $\neg \varphi \in \mathcal{L}_s$;
- if $\varphi_1, \varphi_2 \in \mathcal{L}_s$, then $\varphi_1 \lor \varphi_2 \in \mathcal{L}_s$;
- if $s \in S$, $C \subseteq Ag$ and $\psi \in \mathcal{L}_p$, then $[SC] \psi \in \mathcal{L}_s$;
- if $C \subseteq Ag$ and $\psi \in \mathcal{L}_p$, then $\langle \langle C \rangle \rangle \psi \in \mathcal{L}_s$;
- if $\varphi \in \mathcal{L}_s$, then $X\varphi, G\varphi \in \mathcal{L}_p$;
- if $\varphi_1, \varphi_2 \in \mathcal{L}_s$, then $\varphi_1 U \varphi_2 \in \mathcal{L}_p$. 

![Fig. 1. Model used in Example 1.](image-url)
That is, state formulas are propositional formulas and those under the scope of one of the operators \([s_C]\) and \(\langle C \rangle\). Path formulas are those under the scope of one of the operators X, G and U. For instance, \(\langle C \rangle Xp\) and \(\langle C \rangle X[s_C](p_1 U p_2)\) are well formed state formulas, while \(Xp\) and \(p_1 U(p_2 \lor [s_C]p_1)\) are well formed path formulas. Note that in well formed formulas the operators \([\cdot]\) and \(\langle \cdot \rangle\) precede one of X, G or U, and the latter cannot precede each other. That is, \(XXp\) is not a formula, nor is \(\langle C \rangle \langle C \rangle p\).

Formulas of the form \([s_C]\psi\) stand for ‘if the coalition \(C\) follows the strategy \(s\) then \(\psi\) holds’, \(\langle C \rangle \psi\) stand for ‘the coalition \(C\) has the power of bringing about \(\psi\’\), \(X\psi\) stand for ‘\(\varphi\) holds in the next state’, \(G\psi\) stand for ‘\(\varphi\) holds from the current state on’, and \(\varphi_1 U \varphi_2\) stand for ‘\(\varphi_1\) holds from the current state on until \(\varphi_2\) holds’.

In what follows the common abbreviations for the operators \(\land\), \(\rightarrow\) and \(\iff\) are also used. The symbols \(\top\) and \(\bot\) abbreviate respectively \(p \lor \neg p\) and \(\neg(p \lor \neg p)\) for some \(p \in P\). In addition, for convenience we omit the brackets for coalitions inside the operator \(\langle \cdot \rangle\), i.e., we write \(\langle a_1, \ldots, a_n \rangle\) rather than \(\langle \{a_1, \ldots, a_n\} \rangle\).

### 3.3 Satisfaction Relation and Validity

Satisfaction of state formulas is defined with respect to a state in the model, while satisfaction of path formulas is defined with respect to a run of the system. In CATL jargon, a run of the system is called computation. A computation in a model \(M\) is an infinite sequence \(w_0, \alpha_0, w_1, \alpha_1, \ldots\) where each \(w_i \in W\), each \(\alpha_i \in J_{Ag}\), and for each pair \((w_i, \alpha_i)\) we have \(T(w_i, \alpha_i) = w_{i+1}\). \(\text{comp}(w)\) denotes the set of all possible computations starting in \(w\), and \(\text{comp}(\alpha_C, w)\) denotes the set of all possible computations starting in \(w\) where the coalition \(C\) follows the strategy \(\sigma\). In other words, \(\text{comp}(\alpha_C, w)\) denotes the set of computations such that for each \(a \in C\) and for each pair \((w_i, \alpha_i)\) in the sequence we have \((\sigma(w_i))(a) = \alpha_i(a)\). Moreover, let \(\lambda \in \text{comp}(\alpha_C, w)\), we denote \(\lambda[0]\) the first state of the sequence, \(\lambda[1]\) the second state, and so on. Note that for every \(\sigma\), \(\text{comp}(\sigma_0, w) = \text{comp}(w)\) and \(\text{comp}(\sigma_{Ag}, w)\) is a singleton.

The satisfaction relation \(\models\) between pairs of the form \(\langle M, w \rangle\), where \(M\) is a model and \(w\) is a state of \(M\), and state formulas \(\varphi \in L_s\) is inductively defined as follows:

\[
\begin{align*}
\langle M, w \rangle \models p & \iff w \in V(p); \\
\langle M, w \rangle \models \neg \varphi & \iff \text{not } \langle M, w \rangle \models \varphi; \\
\langle M, w \rangle \models \varphi_1 \lor \varphi_2 & \iff \langle M, w \rangle \models \varphi_1 \text{ or } \langle M, w \rangle \models \varphi_2; \\
\langle M, w \rangle \models [s_C] \varphi & \iff \text{for all } \lambda \in \text{comp}((I(s))(C), w) \text{ we have } \langle M, \lambda \rangle \models \psi; \\
\langle M, w \rangle \models \langle C \rangle \psi & \iff \text{for all } \sigma \in \Sigma \text{ such that } \\
& \quad \text{for all } \lambda \in \text{comp}(\sigma_C, w) \text{ we have } \langle M, \lambda \rangle \models \psi;
\end{align*}
\]

\(^3\)This definition does not use the update semantics as in [10], but it is easy to see that both are equivalent for the fragment of the language of CATL defined here.
and the satisfaction relation $|=\$ between pairs of the form $(M, \lambda)$, where $\lambda$ is a computation in $M$, and path formulas $\psi \in L_p$ is inductively defined as follows:

\[
\begin{align*}
(M, \lambda) & \models X \psi \quad \text{iff} \quad (M, \lambda[1]) \models \psi; \\
(M, \lambda) & \models G \psi \quad \text{iff} \quad \text{for all } i \in \mathbb{N} \text{ we have } (M, \lambda[i]) \models \psi; \\
(M, \lambda) & \models \varphi_1 U \varphi_2 \quad \text{iff} \quad \text{there is } i \in \mathbb{N} \text{ such that } (M, \lambda[i]) \models \varphi_2 \text{ and for all } k \in \mathbb{N}, \text{ if } 0 \leq k < i \text{ then } (M, \lambda[k]) \models \varphi_1.
\end{align*}
\]

For instance, $(M, w_0)$ of Example 1 satisfies $\langle a \rangle X \neg \text{light}$, but does not satisfy $\langle a \rangle X \text{light}$. Note that $\langle \theta \rangle$ is an universal quantifier over computations, while $\langle Ag \rangle$ is an existential quantifier over computations. Thus, $(M, w_0)$ also satisfies $\langle Ag \rangle X \text{light}$, but does not satisfy $\langle \theta \rangle X \text{light}$. Moreover, $(M, w_0)$ satisfies $\neg \langle s_{\{a\}} \rangle X \neg \text{light}$, but it does not satisfy $\langle s_{\{a\}} \rangle X \text{light}$.

**Remark 1.** By combining the operators $[\cdot]$ and $X$ one obtains a sort of dynamic operator. However, it differs from the operator $[\cdot]$ of propositional dynamic logic (PDL) [8]. For instance, $[s_C] [s_C] X \varphi$ is equivalent to $[s_C] X \langle \emptyset \rangle X \varphi$ in CATL. Since a strategy embeds all decisions made by the agents in every state of the model, the operator $[\cdot]$ of CATL works more or less like the star operator of PDL, i.e., as if the execution of the actions in $s$ were repeated ad infinitum. The interested reader can find more on the relation between CATL and PDL in [13].

As usual, a state formula $\varphi \in \mathcal{L}_s$ is valid (notation: $|= \varphi$) if and only if every pair $(M, w)$ satisfies $\varphi$. In the sequel, some relevant valid schemas of CATL are listed. Their validity follows directly from the axiomatization of ATL presented in [14].

1. $[s_0] \psi \leftrightarrow \langle \emptyset \rangle \psi$.
2. $[s_C] (\varphi_1 \land \varphi_2) \rightarrow ([s_C] \varphi_1 \land [s_C] \varphi_2)$.
3. $\langle C \rangle X \top$.
4. $\neg \langle C \rangle X \bot$.
5. $\neg \langle \emptyset \rangle X \neg \varphi \rightarrow \langle Ag \rangle X \varphi$.
6. $\langle C \rangle X (\varphi_1 \land \varphi_2) \rightarrow (\langle C \rangle X \varphi_1 \land \langle C \rangle X \varphi_2)$.
7. $\langle C \rangle G \varphi \leftrightarrow (\varphi \land \langle C \rangle X \langle C \rangle G \varphi)$.
8. $\langle C \rangle (\varphi_1 U \varphi_2) \leftrightarrow (\varphi_2 \lor (\varphi_1 \land \langle C \rangle X \langle C \rangle (\varphi_1 U \varphi_2)))$.

The model checking problem for CATL is the problem of determining, given a formula $\varphi$ and a pair $(M, w)$, whether or not $(M, w) \models \varphi$. Model checking over finite models can be solved in linear time by using a variation of the algorithm proposed by [12] for model checking in ATL [10].

The satisfiability problem for CATL is the problem of determining, given a formula $\varphi$, whether or not there exists a pair $(M, w)$ that satisfies $\varphi$. This problem has been shown to be in EXPTIME [15].

## 4 Obligations

In this paper we follow the idea of [5] and use the following definition for forward-looking responsibility:
The coalition $C$ is forward-looking responsible for the consequence $\varphi$ if and only if it is obligatory for $C$ to bring about $\varphi$.

Therefore, we need to introduce obligations in the framework of CATL. We do so by adapting the simple, yet effective, idea of [4]. It proposes to add special proposition letters to the language that stand for violations.

From now on we assume that the set of proposition letters is $P \cup P_v$, where $P_v = \{v_C \mid C \subseteq Ag \text{ and } C \neq \emptyset\}$. That is, for each coalition $C$, we introduce a new proposition letter $v_C$ that has the special meaning: ‘coalition $C$ is in violation’. In addition, models are now tuples of the form $\langle W, T, I, V \rangle$ where $W$, $T$ and $I$ are as before and the domain of $V$ is extended to the new set of proposition letters, i.e., $V : P \cup P_v \to \mathcal{P}(W)$. In addition, we require that models satisfy the scheme:

$$\neg \langle \emptyset \rangle X v_C.$$  

(1)

which is equivalent to the following model constraint:

for all $C \subseteq Ag$ and all $w \in W$
there is $\alpha \in J_{Ag}$ and $w' \in W$ such that $T(w, \alpha) = w'$ and $w' \notin V(v_C)$.

Formulas of the form $O_C \varphi$ stand for ‘it is obligatory for the coalition $C$ that $\varphi$ holds’. The idea given in [4] is to define obligations by the abbreviation $O_C \varphi \overset{\text{def}}{=} \Box (\neg \varphi \to v_C)$, where $\Box$ stand for some modal operator. That is, $O_C \varphi$ should be paraphrased by ‘it is necessary that if $\neg \varphi$ holds then coalition $C$ is in violation’. In the framework of CATL we have several candidates for $\Box$. For instance, if we use $\langle \emptyset \rangle \mathcal{G}$, we have the following reading for obligation: ‘no coalition can avoid that, from the current state on, $\neg \varphi$ implies violation’. Since other possibilities are equally interesting, we decided to define three different obligation operators:

$$O_{X_C} \varphi \overset{\text{def}}{=} \langle \emptyset \rangle X (\neg \varphi \to v_C) ,$$

$$O_{G_C} \varphi \overset{\text{def}}{=} \langle \emptyset \rangle \mathcal{G} (\neg \varphi \to v_C) ,$$

$$\varphi_1 \mathcal{O}_{U_C} \varphi_2 \overset{\text{def}}{=} \langle \emptyset \rangle ((\neg \varphi_1 \to v_C) \mathcal{U} \varphi_2) ,$$

where $\varphi$, $\varphi_1$ and $\varphi_2$ are state formulas.

A formula of the form $O_{X_C} \varphi$ is read ‘it is obligatory for $C$ that $\varphi$ holds in the next state’, $O_{G_C} \varphi$ is read ‘it is obligatory for coalition $C$ that $\varphi$ holds from the current state on’, and $\varphi_1 \mathcal{O}_{U_C} \varphi_2$ is read ‘it is obligatory for coalition $C$ that $\varphi_1$ holds from the current state on until $\varphi_2$ holds’.

The proposition below follows straightforwardly from the valid schemas listed on page 7 Section 3.3 and the scheme (1).

**Proposition 1.** The following schemas are valid in CATL with obligations:

1. $O_{X_C} \top$. 


2. \(-OX_C \bot\).
3. \(OX_C (\varphi_1 \land \varphi_2) \leftrightarrow (OX_C \varphi_1 \land OX_C \varphi_2)\).
4. \(OG_C \top\).
5. \(-OG_C \bot\).
6. \(OG_C (\varphi_1 \land \varphi_2) \leftrightarrow (OG_C \varphi_1 \land OG_C \varphi_2)\).
7. \(OG_C \varphi \leftrightarrow ((\varphi \rightarrow v_C) \land \langle \emptyset \rangle X OG_C \varphi)\).
8. \(\varphi OU_C \top\).
9. \(-((\varphi OU_C \bot))\).
10. \((\varphi_1 OU_C \varphi_2) \leftrightarrow (\varphi_2 \lor ((\neg \varphi_1 \rightarrow v_C) \land \langle \emptyset \rangle X (\varphi_1 OU_C \varphi_2)))\).

Note that Proposition 1.2 is equivalent to \(OX_C \varphi \rightarrow \neg OX_C \neg \varphi\). Analogously, Proposition 1.5 is equivalent to \(OG_C \varphi \rightarrow \neg OG_C \neg \varphi\). If we interpret permission as dual of obligation, then it means that in our framework everything that is obligatory is permitted.

In what follows, we want to show that forward-looking responsibility regarding state of affairs implies forward-looking responsibility regarding events. It means that we need to define obligation over actions in our framework. Following the idea of [16], we do so by using the dynamic operator available in our language, i.e., the operator \([\cdot]\). We want that formulas of the form \(O_C(s)\) stand for ‘it is obligatory for the coalition \(C\) to follow the strategy \(s\)’. We therefore define dynamic obligations in our framework as follows:

\[
OX_C(s) \overset{\text{def}}{=} [\overline{s}C]Xv_C, \\
OG_C(s) \overset{\text{def}}{=} [\overline{s}C]Gv_C,
\]

where \(\overline{s}\) stand for ‘everything else but \(s\)’, or not-\(s\).

Jamroga et al. [13] seem to suggest that it would be possible to define a strategy language containing operators such as union, sequence, etc., in a similar way as done for actions in PDL. This could allow for an account of ought-to-follow, in a similar way as done for ought-to-do, in the case of actions, exactly as in [16].

For the purpose of this paper though, the only requirement is the reasonable assumption that \(s\) is equivalent to \(\overline{s}\). That is, not-not-\(s\) is the same as \(s\). Therefore, we immediately have \([sC]\psi \leftrightarrow [\overline{s}C]\psi\) valid.

We show that the expected relation between the two kinds of forward-looking responsibilities follows from these definitions. For instance, in Example 1, let \(\sigma\) be a strategy such that \((\sigma(a))(w_0) = \text{toggle}\). Then we have that:

\(\langle M, w_0 \rangle \models OX_a \text{light} \rightarrow OX_a(\overline{s})\).

That is, the obligation for \(a\) that \(\text{light}\) holds in the next state implies that it is obligatory for \(a\) to follow not-\(s\). In other words, it is forbidden for \(a\) to follow \(s\).

To see that it indeed holds, suppose that \(\langle M, w_0 \rangle \models \langle \emptyset \rangle X(\neg \text{light} \rightarrow v_a)\). It is the case if and only if for all \(\lambda \in \text{comp}(w_0)\) we have \(\langle M, \lambda[1] \rangle \models \neg \text{light} \rightarrow v_a\). Now, note that for all \(\lambda \in \text{comp}(\sigma_a, w_0)\) we have \(\langle M, \lambda[1] \rangle \models \neg \text{light}\). Therefore, for all \(\lambda \in \text{comp}(\sigma_a, w_0)\) we have \(\langle M, \lambda[1] \rangle \models v_a\), if and only if \(\langle M, w_0 \rangle \models [s_a]Xv_a\), if and only if \(\langle M, w_0 \rangle \models OX_a(\overline{s})\).

This can easily be generalized as follows:
Proposition 2. Let a tuple \((M,w_0)\) be given, where \(M\) is a model as defined above and \(w_0 \in W\). Also let \((M,w_0) \models [s_C]X\neg \varphi\), where \(\lambda = w_0, \alpha_0, \omega_1, \ldots\) is a computation in \(M\) and \(I(s)\) is a joint strategy that such that \((I(s))(w_0) = \alpha_0\). Then we have that:

\[(M,w_0) \models OX_C \varphi \rightarrow OX_C (\varphi)\].

5 Attempts

In this paper we propose the following definition for backward-looking responsibility, based on [2]:

The coalition \(C\) is backward-looking responsible for \(\varphi\) if and only if \(C\) attempted to bring about \(\varphi\).

Therefore, we want to add to the language the operator \(\text{Att}\) that stands for attempting. The first question is whether this operator should define a state formula or a path formula. Note that to be able to evaluate agent attempts, it is necessary to look at the action that the agent has decided to undertake. Then it seems that this kind of responsibility asserts something about the past. That is the reason why we use the term ‘backward-looking’. This makes us conclude that attempts should be evaluated with respect to a run of the system, i.e., with respect to a computation. Therefore, we propose the operator \(\text{Att}\) as a path quantifier. Thus \(\text{Att}_C \varphi\) is a path formula that stands for ‘the coalition \(C\) attempts to bring about \(\varphi\)’.

The second question is what properties \(\text{Att}\) should satisfy. We propose that every tuple of the form \((M,\lambda)\) satisfies both \(\neg \text{Att}_C \bot\) and \(\neg \text{Att}_C \top\). With the former we avoid allowing coalitions to attempt to bring about something that is impossible, since it would imply that coalitions can attempt to bring about everything. The latter is more controversial, though reasonable. Remember that we want to base the definition of responsibility on attempts. If \(\text{Att}_C \top\) is satisfiable, then we allow coalitions to attempt to bring about something that is inevitable. It follows that coalitions can be held responsible for inevitable events, which is clearly not desirable.

Now note that we want to define this operator on the basis of the actions that the agents can decide to undertake. Therefore, we prefer to look for a semantical definition of an operator that satisfies the properties above and, at the same time, has a natural reading in terms of \(\text{CATL}\) models. The definition that we
propose is the following:

\[
\langle M, \lambda \rangle \models \text{Att}_C \varphi \quad \text{iff} \quad \text{for all } \lambda' \in \text{comp}(\lambda[0], \sigma^C) \text{ we have } \\
\langle M, \lambda'[1] \rangle \models \varphi \\
\text{or} \\
\text{there is } \lambda'' \in \text{comp}(\lambda[0], \sigma^C) \text{ such that } \\
\langle M, \lambda''[1] \rangle \models \neg \varphi,
\]

where \( \sigma^\lambda \) is the strategy followed in the computation \( \lambda \). That is, for every \( w_i \in \lambda \) we have \( \sigma^\lambda(w_i) = \alpha_i \).

This is a complex definition indeed. It may be helpful to look at an equivalent syntactical definition (remember that \( \langle \langle Ag \rangle \rangle \) is an existential quantifier):

\[
\langle M, \lambda \rangle \models \text{Att}_C \varphi \quad \text{iff} \quad \langle M, \lambda[0] \rangle \models ([s_C]X \varphi \land \langle Ag \rangle X \neg \varphi) \lor \\
(\neg [s_C]X \neg \varphi \land \langle C \rangle X \neg \varphi)
\]

where \( I(s) = \sigma^\lambda \).

The first clause means that the coalition \( C \) has decided to undertake an action that necessarily leads to \( \varphi \) in the next state, even though \( C \) could have decided to allow \( \neg \varphi \) in the next state. The second clause means that \( C \) decides to allow \( \varphi \) in the next state, even though \( C \) could have decided to bring about \( \neg \varphi \) in the next state.

Note that the operator \( \text{Att} \) actually stands for ‘attempted to bring about in the subsequent state’. The latter definition can easily be generalized to other attempt operators standing for ‘attempted to bring about from that state on’ and ‘attempted to bring about from that state on until’ in a similar way as done for obligations. The results that follow can also be generalized to these two other notions of attempt.

**Proposition 3.** The following formulas are valid in \( \text{CATL} \) with attempts:

1. \( \neg \text{Att}_C \bot \).
2. \( \neg \text{Att}_C \top \).

**Proof.**

1. Straightforward, since \( \neg [s_C]X \bot \) and \( [s_C]X \top \) are valid for every coalition \( C \).
2. Also straightforward, since \( \langle C \rangle X \top \) is valid for every coalition \( C \).

\[ \Box \]

We use Example 1 again to illustrate this new operator. Let \( \lambda = w_0, \alpha_0, w_1, \ldots \) be a computation where \( \alpha_0(w_0, a) = \text{skip} \) and \( \alpha_0(w_0, b) = \text{toggle} \), then \( \langle M, \lambda \rangle \) satisfies the formulas \( \text{Att}_{a} \text{light}, \text{Att}_{b} \neg \text{light} \) and \( \text{Att}_{\{a,b\}} \neg \text{light} \), but does not satisfy the formula \( \text{Att}_{\{a,b\}} \text{light} \).
Note however, that it is possible that a coalition attempts to bring about \( \varphi \) and that no individual in the coalition attempts to bring about \( \varphi \). To see this, consider an alternative setting with three agents and where the state of the light changes only if at least two of them toggle the switch. That is, consider \( M' = \langle W, T', I, V \rangle \) such that \( P, A, S, W, I \) and \( V \) are as in Example 1, \( Arg = \{ a, b, c \} \) and each line of Table 2 represents an element of \( T' \).

<table>
<thead>
<tr>
<th>( a )</th>
<th>( b )</th>
<th>( c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_0 )</td>
<td>skip</td>
<td>skip</td>
</tr>
<tr>
<td>( w_0 )</td>
<td>skip</td>
<td>skip</td>
</tr>
<tr>
<td>( w_0 )</td>
<td>skip</td>
<td>toggle</td>
</tr>
<tr>
<td>( w_0 )</td>
<td>toggle</td>
<td>skip</td>
</tr>
<tr>
<td>( w_0 )</td>
<td>toggle</td>
<td>skip</td>
</tr>
<tr>
<td>( w_0 )</td>
<td>toggle</td>
<td>toggle</td>
</tr>
<tr>
<td>( w_1 )</td>
<td>skip</td>
<td>skip</td>
</tr>
<tr>
<td>( w_1 )</td>
<td>skip</td>
<td>skip</td>
</tr>
<tr>
<td>( w_1 )</td>
<td>skip</td>
<td>toggle</td>
</tr>
<tr>
<td>( w_1 )</td>
<td>skip</td>
<td>toggle</td>
</tr>
<tr>
<td>( w_1 )</td>
<td>toggle</td>
<td>skip</td>
</tr>
<tr>
<td>( w_1 )</td>
<td>toggle</td>
<td>skip</td>
</tr>
<tr>
<td>( w_1 )</td>
<td>toggle</td>
<td>toggle</td>
</tr>
<tr>
<td>( w_1 )</td>
<td>toggle</td>
<td>toggle</td>
</tr>
</tbody>
</table>

Table 2. Transition function used in example with three agents.

Now, let \( \lambda = w_0, \alpha_0, w_1, \ldots \) be a computation in \( M' \) where \( \alpha_0(w_0, a) = \) skip, \( \alpha_0(w_0, b) = \) toggle and \( \alpha_0(w_0, c) = \) toggle. Then \( \langle M', \lambda \rangle \) satisfies the formula \( \text{Att}_{Ag} \neg \text{light} \), but does not satisfy the formulas \( \text{Att}_a \neg \text{light} \), \( \text{Att}_b \neg \text{light} \) or \( \text{Att}_c \neg \text{light} \).

In the sequel we present a key feature of this framework. We show that the relation mentioned in the introduction between forward-looking and backward-looking responsibility is captured in the formalism. That is, we show that if a coalition \( C \) is held forward-looking responsible for a certain goal \( \varphi \) and \( C \) has the power to succeed (i.e., it can avoid violation), then \( C \) is held backward-looking responsible for an eventual failure.

**Proposition 4.** Let \( M \) be a CATL model and let \( \lambda \) be a computation in \( M \). Then the following holds:

\[
\langle M, \lambda[0] \rangle \models \text{OX}_{C \varphi} \land \langle C \rangle X \neg v_C \text{ and } \langle M, \lambda \rangle \models X \neg \varphi \text{ implies } \langle M, \lambda \rangle \models \text{Att}_C \neg v_C .
\]

**Proof.** Assume \( I(s) = \sigma^\lambda \). First, we recall that \( \langle M, \lambda[0] \rangle \) satisfies \( \text{OX}_{C \varphi} \) if and only if \( \langle M, \lambda[0] \rangle \models \langle \emptyset \rangle X (\neg \varphi \rightarrow v_C) \). Therefore \( \langle M, \lambda[1] \rangle \models (\neg \varphi \rightarrow v_C) \). Now note that \( \langle M, \lambda \rangle \models X \neg \varphi \) implies \( \langle M, \lambda[1] \rangle \models \neg \varphi \). Then \( \langle M, \lambda[1] \rangle \models v_C \). Then
⟨M, λ[0]⟩ |= [sC]X¬vC. The latter, together with ⟨M, λ[0]⟩ |= ⟨⟨C⟩⟩X¬vC, immediately implies ⟨M, λ[0]⟩ |= AttCV C.

6 Conclusion and Future Work

In this paper we formalize two different notions of responsibility in an extension of CA TL. It is shown that the ability of expressing statements such as ‘if the coalition C follows the strategy s, then φ holds’ allows us to define dynamic obligations, and also an operator expressing attempts. The former are used to express forward-looking responsibility and the latter are used to express backward-looking responsibility. It is also shown that the framework captures the relation between the three kinds of responsibility, as previously required.

Possible future works include a couple of improvements. We consider that the most important one is to try to relax the assumption of complete information, i.e., the assumption that agents can foresee all the consequences of their actions. Note that this assumption is, in fact, unrealistic. For it is very common that agents do not have enough information to know whether the performance of a certain action will lead to an unwanted outcome. In such cases, the agent should not be considered responsible for the outcome, since it is not possible to accuse him/her of having attempted it. Van der Hoek and Wooldridge [17] propose another extension of ATL (without strategies) wherein one can express that ‘the agent a believes that he/she has the power of bringing about φ’. Augmented with strategies, this framework should be able to express that ‘the agent a believes that if he/she follows the strategy s, then φ holds’. In future works, we intend to use the latter idea to better approximate forward-looking and backward-looking responsibility in scenarios where agents have incomplete information about the system.

References