Towards a Formalization of Responsibility

Tiago de Lima[1] Lambèr Royakkers[1] Frank Dignum[2]

[1]Eindhoven University of Technology, the Netherlands

[2]Utrecht University, the Netherlands

3rd International Workshop on Normative Multiagent Systems
Luxembourg, 15 July 2008

Responsibility

Responsibility

Example. Alice, an employee of the financial department, has access to several different bank accounts of the company. From time to time the director of the department asks her to make money transfers between these accounts. But last Monday she heard from the director: "— From now on you will decide when and how to make the transfers. I am making you *responsible* for maintaining the balance of all accounts positive."

Responsibility

Example. Alice, an employee of the financial department, has access to several different bank accounts of the company. From time to time the director of the department asks her to make money transfers between these accounts. But last Monday she heard from the director: "— From now on you will decide when and how to make the transfers. I am making you *responsible* for maintaining the balance of all accounts positive."

The meaning of the term 'responsibility' in this example implies the duty, or the obligation, to ensure that each account balance will be positive.

Responsibility

Example. Alice, an employee of the financial department, has access to several different bank accounts of the company. From time to time the director of the department asks her to make money transfers between these accounts. But last Monday she heard from the director: "— From now on you will decide when and how to make the transfers. I am making you *responsible* for maintaining the balance of all accounts positive."

The meaning of the term 'responsibility' in this example implies the duty, or the obligation, to ensure that each account balance will be positive.

This is compatible with the following definition, suggested by [Santos and Carmo, 1996].

Definition (Notion 1). Agent a is responsible for φ if and only if a is obliged to ensure that φ.

Responsibility

Example (continuation). On Tuesday the balance of account 1 is 10,000 Euro, while the balance of account 2 is only 50 Euro! Moreover, the company will spend 5,000 Euro from account 2 either on Tuesday or Wednesday.

Responsibility

Example (continuation). On Tuesday the balance of account 1 is 10,000 Euro, while the balance of account 2 is only 50 Euro! Moreover, the company will spend 5,000 Euro from account 2 either on Tuesday or Wednesday.

So, Alice must make a decision. In particular, she has the choice between making a transfer from account 1 to account 2 on Tuesday or wait until Wednesday.

Responsibility

Example (continuation). Alice decides leave the transfer to Wednesday. However, the company spends the money on Tuesday, and therefore she hears from the director: “— You are *responsible* for the balance of account 2 is negative!”

Responsibility

Example (continuation). Alice decides leave the transfer to Wednesday. However, the company spends the money on Tuesday, and therefore she hears from the director: "— You are *responsible* for the balance of account 2 is negative!"

The meaning of the term 'responsibility' in this case implies blameworthiness, or the guilty of the negative balance.

Responsibility

Example (continuation). Alice decides leave the transfer to Wednesday. However, the company spends the money on Tuesday, and therefore she hears from the director: "— You are *responsible* for the balance of account 2 is negative!"

The meaning of the term 'responsibility' in this case implies blameworthiness, or the guilty of the negative balance.

The latter is compatible with the following definition, based on [Kein, 1993] and [Heinaman, 1993].

Definition (Notion 2). Agent a is responsible for φ if and only if a freely, knowingly and intentionally behaves in such a way that is necessary for the occurrence of a "wrong" consequence φ.

Responsibility

We therefore can distinguish (at least) two different uses for the term ‘responsibility’.

Responsibility

We therefore can distinguish (at least) two different uses for the term ‘responsibility’.

Notion 1 is called **forward-looking responsibility**.

Notion 2 is called **backward-looking responsibility**.

Responsibility

We therefore can distinguish (at least) two different uses for the term ‘responsibility’.

Notion 1 is called **forward-looking responsibility**.

Notion 2 is called **backward-looking responsibility**.

Note that these two notions are somehow related. For instance, in the example Alice is considered backward-looking responsible for the negative balance because she was firstly held forward-looking responsible for maintaining the balance positive.

Responsibility

We therefore can distinguish (at least) two different uses for the term ‘responsibility’.

Notion 1 is called **forward-looking responsibility.**

Notion 2 is called **backward-looking responsibility.**

Note that these two notions are somehow related. For instance, in the example Alice is considered backward-looking responsible for the negative balance because she was firstly held forward-looking responsible for maintaining the balance positive.

In this work we try to build a framework wherein one can formalize these two notions and capture the relation between them.

Action-Based Alternating-Time Transition Systems

Action-Based Alternating-Time Transition Systems

Assume:

- a set of actions Act,
- a set of atoms Atm,
- a finite set of agents Agt.

Action-Based Alternating-Time Transition Systems

Assume:

- a set of actions Act,
- a set of atoms Atm,
- a finite set of agents Agt.

$Jact$ denotes the set of all functions $\alpha : Agt \to Act$
(i.e., the set of joint actions available for Agt).

Action-Based Alternating-Time Transition Systems

Assume:

- a set of actions Act,
- a set of atoms Atm,
- a finite set of agents Agt.

$Jact$ denotes the set of all functions $\alpha : Agt \to Act$
(i.e., the set of joint actions available for Agt).

Models are tuples $\langle W, T, \Sigma, V \rangle$ where:

- W is the set of states (or possible worlds),

Action-Based Alternating-Time Transition Systems

Assume:

- a set of actions Act,
- a set of atoms Atm,
- a finite set of agents Agt.

$Jact$ denotes the set of all functions $\alpha : Agt \rightarrow Act$
(i.e., the set of joint actions available for Agt).

Models are tuples $\langle W, T, \Sigma, V \rangle$ where:

- W is the set of states (or possible worlds),
- $T : (W \times Jact) \rightarrow W$ is a partial transition function,

Action-Based Alternating-Time Transition Systems

Assume:

- a set of actions Act,
- a set of atoms Atm,
- a finite set of agents Agt.

$Jact$ denotes the set of all functions $\alpha : Agt \to Act$
(i.e., the set of joint actions available for Agt).

Models are tuples $\langle W, T, \Sigma, V \rangle$ where:

- W is the set of states (or possible worlds),
- $T : (W \times Jact) \to W$ is a partial transition function,
- Σ is the set of all functions $\sigma : W \to Jact$ such that for each $\sigma(w)$ there is $w' \in W$ such that $T(w, \sigma(w)) = w'$ (i.e., the set of joint strategies available for Agt),

Action-Based Alternating-Time Transition Systems

Assume:

- a set of actions Act,
- a set of atoms Atm,
- a finite set of agents Agt.

$Jact$ denotes the set of all functions $\alpha : Agt \to Act$
(i.e., the set of joint actions available for Agt).

Models are tuples $\langle W, T, \Sigma, V \rangle$ where:

- W is the set of states (or possible worlds),
- $T : (W \times Jact) \to W$ is a partial transition function,
- Σ is the set of all functions $\sigma : W \to Jact$ such that for each $\sigma(w)$ there is $w' \in W$ such that $T(w, \sigma(w)) = w'$ (i.e., the set of joint strategies available for Agt),
- $V : Atm \to 2^W$ is the interpretation of atoms.

Action-Based Alternating-Time Transition Systems

For example:
$Act = \{skip, spend, transf\}$, $Atm = \{p\}$, and $Agt = \{a, c\}$.

Action-Based Alternating-Time Transition Systems

For example:

$Act = \{skip, spend, transf\}$, $Atm = \{p\}$, and $Agt = \{a, c\}$.

$Jact$ contains:

$\alpha_0 = \{(a, skip), (c, spend)\}$, $\alpha_1 = \{(a, transf), (c, skip)\}$, ...

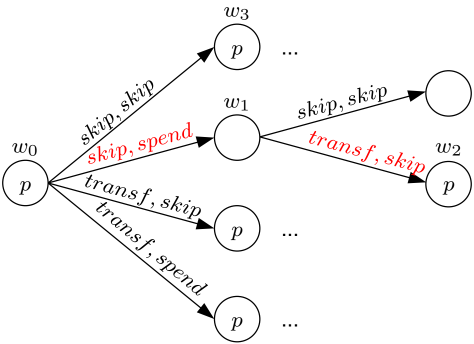
Action-Based Alternating-Time Transition Systems

For example:

$Act = \{skip, spend, transf\}$, $Atm = \{p\}$, and $Agt = \{a, c\}$.

$Jact$ contains:

$\alpha_0 = \{(a, skip), (c, spend)\}$, $\alpha_1 = \{(a, transf), (c, skip)\}$, $\ldots$



Σ contains: $\sigma = \{(w_0, \alpha_0), (w_1, \alpha_1), \ldots\}$.

Language

Language

$\mathcal{L}_s$ is the set of state formulas.

$\mathcal{L}_p$ is the set of path formulas.

Language

$\mathcal{L}_s$ is the set of state formulas.

$\mathcal{L}_p$ is the set of path formulas.

$\mathcal{L}_s$ is defined by:

- if $p \in Atm$ 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 $\varphi_1 \vee \varphi_2 \in \mathcal{L}_s$;
- if $\sigma \in \Sigma$, $C \subseteq Agt$ and $\psi \in \mathcal{L}_p$ then $[C{:}\sigma]\psi \in \mathcal{L}_s$;
- if $C \subseteq Agt$ and $\psi \in \mathcal{L}_p$ then $\langle\!\langle C \rangle\!\rangle\psi \in \mathcal{L}_s$;

Language

$\mathcal{L}_s$ is the set of state formulas.

$\mathcal{L}_p$ is the set of path formulas.

$\mathcal{L}_s$ is defined by:

- if $p \in Atm$ 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 $\varphi_1 \vee \varphi_2 \in \mathcal{L}_s$;
- if $\sigma \in \Sigma$, $C \subseteq Agt$ and $\psi \in \mathcal{L}_p$ then $[C{:}\sigma]\psi \in \mathcal{L}_s$;
- if $C \subseteq Agt$ and $\psi \in \mathcal{L}_p$ then $\langle\!\langle C \rangle\!\rangle\psi \in \mathcal{L}_s$;

and $\mathcal{L}_p$ is defined by:

- if $\varphi \in \mathcal{L}_s$ then $\mathrm{X}\varphi, \mathrm{G}\varphi \in \mathcal{L}_p$;
- if $\varphi_1, \varphi_2 \in \mathcal{L}_s$ then $\varphi_1 \mathrm{U} \varphi_2 \in \mathcal{L}_p$.

Language

For example:

State formula: $p \vee \neg p$

Language

For example:

State formula: $p \vee \neg p$

Path formula: $\mathrm{X}(p \vee \neg p)$

Language

For example:

State formula: $p \vee \neg p$

Path formula: $\mathrm{X}(p \vee \neg p)$

State formula: $[C{:}\sigma]\mathrm{X}(p \vee p)$

Language

For example:

State formula: $p \vee \neg p$

Path formula: $\mathrm{X}(p \vee \neg p)$

State formula: $[C{:}\sigma]\mathrm{X}(p \vee p)$

Path formula: $p\mathrm{U}([C{:}\sigma]\mathrm{X}(p \vee p))$

Language

For example:

State formula: $p \vee \neg p$

Path formula: $\mathrm{X}(p \vee \neg p)$

State formula: $[C{:}\sigma]\mathrm{X}(p \vee p)$

Path formula: $p\mathrm{U}([C{:}\sigma]\mathrm{X}(p \vee p))$

State formula: $\langle\langle C \rangle\rangle(p\mathrm{U}([C{:}\sigma]\mathrm{X}(p \vee p)))$

Language

For example:

State formula: $p \vee \neg p$

Path formula: $\mathrm{X}(p \vee \neg p)$

State formula: $[C{:}\sigma]\mathrm{X}(p \vee p)$

Path formula: $p\mathrm{U}([C{:}\sigma]\mathrm{X}(p \vee p))$

State formula: $\langle\!\langle C \rangle\!\rangle(p\mathrm{U}([C{:}\sigma]\mathrm{X}(p \vee p)))$

These are not well-formed formulas:

$\mathrm{GX}p$

$[C{:}\sigma][C{:}\sigma]p$

$\langle\!\langle C \rangle\!\rangle[C{:}\sigma]p$

$[C{:}\sigma]\langle\!\langle C \rangle\!\rangle p$

Language

Intended meanings:

$\mathrm{X}\varphi$: 'φ is true in the next state'.

$\mathrm{G}\varphi$: 'φ is true from the current state on'.

$\varphi_1 \mathrm{U} \varphi_2$: '$\varphi_1$ is true from the current state on until φ_2 is true'.

$\langle\langle C \rangle\rangle \psi$: 'coalition C has the power of bringing about ψ'.

$[C{:}\sigma]\psi$: 'if coalition C follows strategy σ then ψ is true'.

Semantics

Semantics

A computation is an infinite sequence $w_0, \alpha_0, w_1, \alpha_1, w_0, \ldots$ such that for each pair (w_i, α_i) we have $T(w_i, \alpha_i) = w_{i+1}$ (i.e., it is a path in the model).

Semantics

A computation is an infinite sequence $w_0, \alpha_0, w_1, \alpha_1, w_0, \ldots$ such that for each pair (w_i, α_i) we have $T(w_i, \alpha_i) = w_{i+1}$ (i.e., it is a path in the model).

$\Lambda(w)$ denotes the set of all computations starting at w.

Semantics

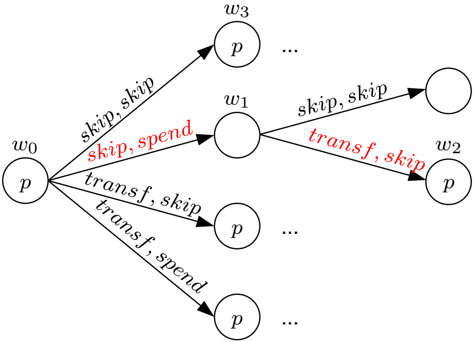
A computation is an infinite sequence $w_0, \alpha_0, w_1, \alpha_1, w_0, \ldots$ such that for each pair (w_i, α_i) we have $T(w_i, \alpha_i) = w_{i+1}$ (i.e., it is a path in the model).

$\Lambda(w)$ denotes the set of all computations starting at w.

$\Lambda(w, C{:}\sigma)$ denotes the set of all computations such that for each $a \in C$ and each pair (w_i, α_i) in the sequence we have $(\sigma(w_i))(\alpha_i) = \alpha_i(a)$
(i.e., it denotes the set of all computations starting at w such that coalition C follows strategy σ).

Semantics

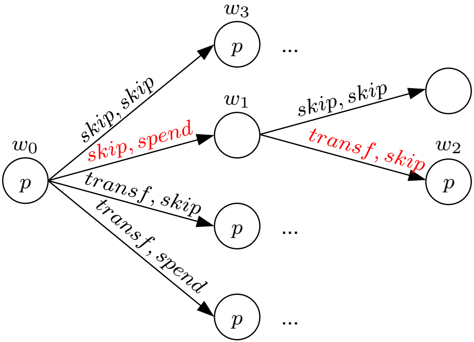
For example:



$$\sigma = \{(w_0, \alpha_0), (w_1, \alpha_1), \ldots\}$$

Semantics

For example:

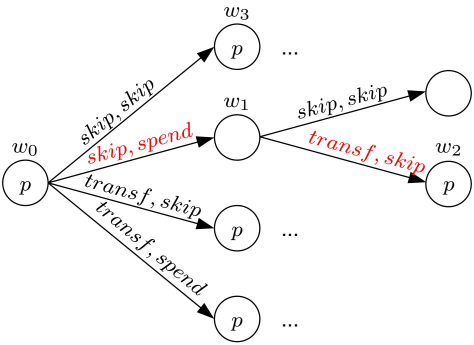


$\sigma = \{(w_0, \alpha_0), (w_1, \alpha_1), \ldots\}$

$\Lambda(w_0)$ is the whole tree.

Semantics

For example:

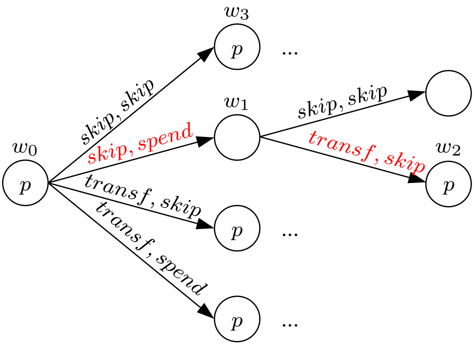


$\sigma = \{(w_0, \alpha_0), (w_1, \alpha_1), \ldots\}$

$\Lambda(w_0)$ is the whole tree.
$\Lambda(w_0, \{a\}{:}\sigma)$ contains computations "passing" by w_1, w_2 and w_3.

Semantics

For example:



$\sigma = \{(w_0, \alpha_0), (w_1, \alpha_1), \ldots\}$

$\Lambda(w_0)$ is the whole tree.

$\Lambda(w_0, \{a\}{:}\sigma)$ contains computations "passing" by w_1, w_2 and w_3.

Note that $\Lambda(w, \emptyset{:}\sigma) = \Lambda(w)$ and $\Lambda(w, Agt{:}\sigma)$ is a singleton.

Semantics

$\mathcal{M}, w \models [C{:}\sigma]\psi$	iff	for all $\lambda \in \Lambda(w, C{:}\sigma)$ we have $\mathcal{M}, \lambda \models \psi$
$\mathcal{M}, w \models \langle\langle C \rangle\rangle\psi$	iff	there is $\sigma \in \Sigma$ such that
		for all $\lambda \in \Lambda(w, C{:}\sigma)$ we have $\mathcal{M}, \lambda \models \psi$

Semantics

$\mathcal{M}, w \models [C{:}\sigma]\psi$	iff	for all $\lambda \in \Lambda(w, C{:}\sigma)$ we have $\mathcal{M}, \lambda \models \psi$
$\mathcal{M}, w \models \langle\langle C \rangle\rangle\psi$	iff	there is $\sigma \in \Sigma$ such that
		for all $\lambda \in \Lambda(w, C{:}\sigma)$ we have $\mathcal{M}, \lambda \models \psi$

Let $\lambda = w_0, \alpha_0, w_1, \alpha_1, \ldots$:

$\mathcal{M}, \lambda \models \mathrm{X}\varphi$	iff	$\mathcal{M}, w_1 \models \varphi$
$\mathcal{M}, \lambda \models \mathrm{G}\varphi$	iff	for all $i \in \mathbb{N}$ we have $\mathcal{M}, w_i \models \varphi$
$\mathcal{M}, \lambda \models \varphi_1 \mathrm{U} \varphi_2$	iff	there is $i \in \mathbb{N}$ such that $\mathcal{M}, w_i \models \varphi_2$ and
		for all $k \in \mathbb{N}$ if $0 \leq k < i$ then $\mathcal{M}, w_k \models \varphi_1$

CATL

CATL model checking is in PTIME [van der Hoek et al., 2005].

CATL satisfiability checking is in EXPTIME [Walther et al., 2007].

Adding Obligations

Adding Obligations

We adapt the idea of [d'Altan et al., 1996].

Adding Obligations

We adapt the idea of [d'Altan et al., 1996].

First, the set of atoms is now $Atm \cup Atm_v$ where:

$$Atm_v = \{v_C : C \subseteq Agt \text{ and } C \neq \emptyset\}$$

Adding Obligations

We adapt the idea of [d'Altan et al., 1996].

First, the set of atoms is now $Atm \cup Atm_v$ where:

$$Atm_v = \{v_C : C \subseteq Agt \text{ and } C \neq \emptyset\}$$

Second, models are as before, but:

- $V : Atm \cup Atm_v \to 2^W$,
- for all $C \subseteq Agt$ and all $w \in W$
  there is $\alpha \in Jact$ and $w' \in W$ such that
  $T(w, \alpha) = w'$ and $w' \notin V(v_C)$.

(The latter is equivalent to the axiom scheme $\neg\langle\langle\emptyset\rangle\rangle \mathrm{X} v_C$.)

Adding Obligations

We adapt the idea of [d'Altan et al., 1996].

First, the set of atoms is now $Atm \cup Atm_v$ where:

$$Atm_v = \{v_C : C \subseteq Agt \text{ and } C \neq \emptyset\}$$

Second, models are as before, but:

- $V : Atm \cup Atm_v \to 2^W$,
- for all $C \subseteq Agt$ and all $w \in W$ there is $\alpha \in Jact$ and $w' \in W$ such that $T(w, \alpha) = w'$ and $w' \notin V(v_C)$.

(The latter is equivalent to the axiom scheme $\neg\langle\langle\emptyset\rangle\rangle \mathrm{X} v_C$.)

Then obligations can be defined as abbreviations:

$$\mathrm{OX}_C\varphi \overset{\text{def}}{=} \langle\langle\emptyset\rangle\rangle\mathrm{X}(\neg\varphi \to v_C)$$

$$\mathrm{OG}_C\varphi \overset{\text{def}}{=} \langle\langle\emptyset\rangle\rangle\mathrm{G}(\neg\varphi \to v_C)$$

where φ is a state formula.

Adding Intentional Behaviour

Adding Intentional Behaviour

$\mathrm{A}_C\psi$ should be read: ‘coalition C attempts to bring about ψ’.

Adding Intentional Behaviour

$\mathrm{A}_C\psi$ should be read: 'coalition C attempts to bring about ψ'.

We propose that coalition C attempts to bring about ψ whenever C behaves in such a way that will "probably" lead to a state satisfying ψ.

Adding Intentional Behaviour

$\mathrm{A}_C\psi$ should be read: 'coalition C attempts to bring about ψ'.

We propose that coalition C attempts to bring about ψ whenever C behaves in such a way that will "probably" lead to a state satisfying ψ.

In our framework it can be defined by:

Coalition C attempts to bring about ψ if and only if either
C brings about ψ even though C could allow for $\neg\psi$, or
C allows for ψ even though C could bring about $\neg\psi$.

Adding Intentional Behaviour

$\mathrm{A}_C\psi$ should be read: ‘coalition C attempts to bring about ψ’.

We propose that coalition C attempts to bring about ψ whenever C behaves in such a way that will “probably” lead to a state satisfying ψ.

In our framework it can be defined by:

Coalition C attempts to bring about ψ if and only if either
C brings about ψ even though C could allow for $\neg\psi$, or
C allows for ψ even though C could bring about $\neg\psi$.

Note that $\mathrm{A}_C\psi$ must be a path formula.

For example, if we want to check whether coalition C attempts to bring about $\mathrm{X}\varphi$, it is necessary to look at the joint action that C will execute in the current state. Therefore, $\mathrm{A}_C\mathrm{X}\varphi$ cannot be evaluated in a single state of the system. Rather, it should be evaluated in a “run of the system”. In CATL terms: it should be evaluated in a computation.

Adding Intentional Behaviour

Formally:

Let $\lambda = w_0, \alpha_0, w_1, \alpha_1, \ldots$, and
let $\sigma = \{(w_0, \alpha_0), (w_1, \alpha_1), \ldots\}$.

$\mathcal{M}, \lambda \models \mathrm{A}_C \psi$ if and only if

for all $\lambda' \in \Lambda(w_0, C{:}\sigma)$ we have $\mathcal{M}, \lambda' \models \psi$ and
there is $\sigma' \in \Sigma$ and $\lambda'' \in \Lambda(w_0, C{:}\sigma')$ such that $\mathcal{M}, \lambda'' \not\models \psi$

or

there is $\lambda' \in \Lambda(w_0, C{:}\sigma)$ such that $\mathcal{M}, \lambda' \models \psi$ and
there is $\sigma' \in \Sigma$ s.t. for all $\lambda'' \in \Lambda(w_0, C{:}\sigma')$, $\mathcal{M}, \lambda'' \not\models \psi$

Adding Intentional Behaviour

Unfortunately, the logic is not expressive enough to accommodate the operator A as an abbreviation.

Adding Intentional Behaviour

Unfortunately, the logic is not expressive enough to accommodate the operator A as an abbreviation.

Instead of proposing a new logic we prefer to keep our formalism as simple as possible. We therefore decided to express a notion that is close to the latter.

Adding Intentional Behaviour

Unfortunately, the logic is not expressive enough to accommodate the operator A as an abbreviation.

Instead of proposing a new logic we prefer to keep our formalism as simple as possible. We therefore decided to express a notion that is close to the latter.

We express that
'If C follows σ then C attempts to bring about φ in the next state', by:

$$\mathrm{AX}_{C:\sigma}\varphi \stackrel{\text{def}}{=} ([C{:}\sigma]\mathrm{X}\varphi \wedge \langle\!\langle Agt \rangle\!\rangle \mathrm{X}\neg\varphi) \vee (\neg[C{:}\sigma]\mathrm{X}\neg\varphi \wedge \langle\!\langle C \rangle\!\rangle \mathrm{X}\neg\varphi)$$

and we express that
'If C follows σ then C attempts to bring about φ from now on', by:

$$\mathrm{AG}_{C:\sigma}\varphi \stackrel{\text{def}}{=} ([C{:}\sigma]\mathrm{G}\varphi \wedge \langle\!\langle Agt \rangle\!\rangle (\top \mathrm{U} \neg\varphi)) \vee (\neg[C{:}\sigma](\top \mathrm{U} \neg\varphi) \wedge \langle\!\langle C \rangle\!\rangle (\top \mathrm{U} \neg\varphi))$$

Forward-Looking vs. Backward-Looking

Forward-Looking vs. Backward-Looking

Forward-looking responsibility is defined by:

$$\mathrm{FRX}_C\varphi \stackrel{\text{def}}{=} \mathrm{OX}_C\varphi \wedge \langle\!\langle C \rangle\!\rangle \mathrm{X}\varphi$$

Forward-Looking vs. Backward-Looking

Forward-looking responsibility is defined by:

$$\mathrm{FRX}_C\varphi \stackrel{\text{def}}{=} \mathrm{OX}_C\varphi \wedge \langle\!\langle C \rangle\!\rangle \mathrm{X}\varphi$$

Backward-looking responsibility is defined by:

$$\mathrm{BRX}_{C:\sigma}v_C \stackrel{\text{def}}{=} [C{:}\sigma]\mathrm{X}v_C \wedge \mathrm{AX}_{C:\sigma}v_C$$

Forward-Looking vs. Backward-Looking

Forward-looking responsibility is defined by:

$$\mathrm{FRX}_C\varphi \stackrel{\text{def}}{=} \mathrm{OX}_C\varphi \wedge \langle\!\langle C \rangle\!\rangle \mathrm{X}\varphi$$

Backward-looking responsibility is defined by:

$$\mathrm{BRX}_{C:\sigma}v_C \stackrel{\text{def}}{=} [C{:}\sigma]\mathrm{X}v_C \wedge \mathrm{AX}_{C:\sigma}v_C$$

$\mathrm{BRX}_{C:\sigma}v_C$ is read: ‘if C follows σ then C is backward-looking responsible for v_C in the next state’.

Forward-Looking vs. Backward-Looking

Forward-looking responsibility is defined by:

$$\mathrm{FRX}_C\varphi \overset{\text{def}}{=} \mathrm{OX}_C\varphi \wedge \langle\!\langle C \rangle\!\rangle \mathrm{X}\varphi$$

Backward-looking responsibility is defined by:

$$\mathrm{BRX}_{C:\sigma} v_C \overset{\text{def}}{=} [C{:}\sigma]\mathrm{X} v_C \wedge \mathrm{AX}_{C:\sigma} v_C$$

$\mathrm{BRX}_{C:\sigma} v_C$ is read: 'if C follows σ then C is backward-looking responsible for v_C in the next state'.

We define it only for violations because of the "wrong-doing" condition.

Forward-Looking vs. Backward-Looking

The following formula is valid:

$$(\mathrm{FRX}_C\varphi \land [C{:}\sigma]\mathrm{X}\neg\varphi) \rightarrow \mathrm{BRX}_{C:\sigma}v_C)$$

If C is held forward-looking responsible for φ and C follows a strategy that leads to a failure then C is backward-looking responsible for it.

Forward-Looking vs. Backward-Looking

The following formula is valid:

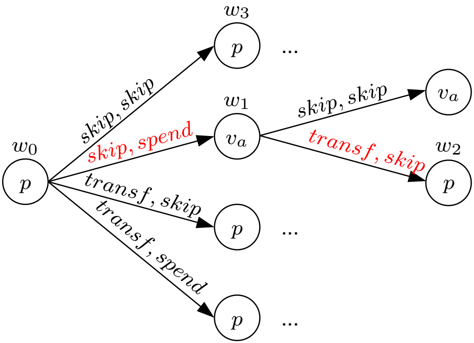
$$(\mathrm{FRX}_C\varphi \wedge [C{:}\sigma]\mathrm{X}\neg\varphi) \rightarrow \mathrm{BRX}_{C:\sigma}v_C)$$

If C is held forward-looking responsible for φ and C follows a strategy that leads to a failure then C is backward-looking responsible for it.

Proof. Indeed since:
$\mathcal{M}, w \models \mathrm{OX}_C\varphi$ iff $\mathcal{M}, w \models \langle\langle\emptyset\rangle\rangle\mathrm{X}(\neg\varphi \rightarrow v_C)$.
Then $\mathcal{M}, w \models \mathrm{OX}_C\varphi \wedge [C{:}\sigma]\mathrm{X}\neg\varphi$ implies $\mathcal{M}, w \models [C{:}\sigma]\mathrm{X}v_C$.
Moreover, remember that $\mathcal{M}, w \models \neg\langle\langle\emptyset\rangle\rangle\mathrm{X}v_C$,
which implies $\mathcal{M}, w \models \langle\langle Agt\rangle\rangle\mathrm{X}\neg v_C$.
Therefore, $\mathcal{M}, w \models [C{:}\sigma]\mathrm{X}v_C \wedge \langle\langle Agt\rangle\rangle\mathrm{X}\neg v_C$,
which immediately implies $\mathcal{M}, w \models \mathrm{BRX}_{C:\sigma}v_C$. □

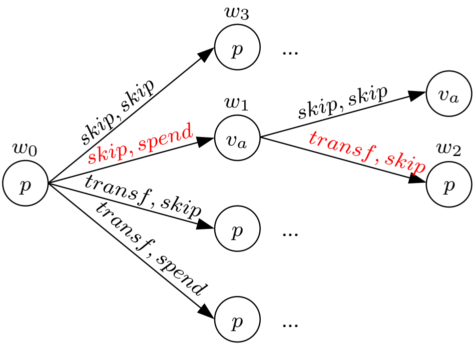
Forward-Looking vs. Backward-Looking

For example,



Forward-Looking vs. Backward-Looking

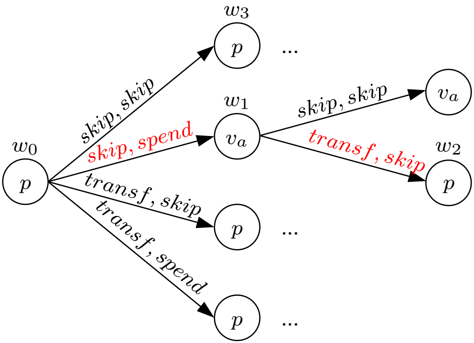
For example,



We have $\mathcal{M}, w_0 \models \langle\langle \emptyset \rangle\rangle \mathrm{X}(\neg p \to v_a) \wedge \langle\langle a \rangle\rangle \mathrm{X} p$.

Forward-Looking vs. Backward-Looking

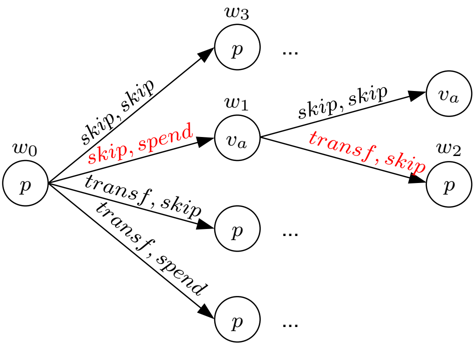
For example,



We have $\mathcal{M}, w_0 \models \langle\langle\emptyset\rangle\rangle \mathrm{X}(\neg p \to v_a) \wedge \langle\langle a\rangle\rangle \mathrm{X}p$.
If and only if $\mathcal{M}, w_0 \models \mathrm{OX}_a p \wedge \langle\langle a\rangle\rangle \mathrm{X}p$.

Forward-Looking vs. Backward-Looking

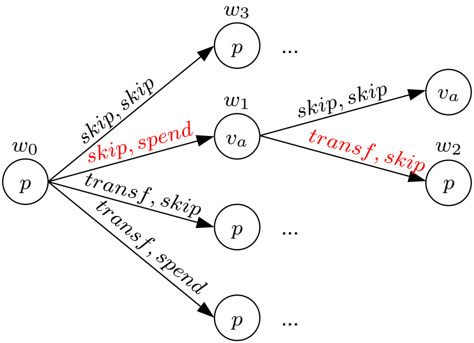
For example,



We have $\mathcal{M}, w_0 \models \langle\langle\emptyset\rangle\rangle \mathrm{X}(\neg p \to v_a) \wedge \langle\langle a\rangle\rangle \mathrm{X}p$.
If and only if $\mathcal{M}, w_0 \models \mathrm{OX}_a p \wedge \langle\langle a\rangle\rangle \mathrm{X}p$.
Then we have $\mathcal{M}, w_0 \models \mathrm{FRX}_a p$.

Forward-Looking vs. Backward-Looking

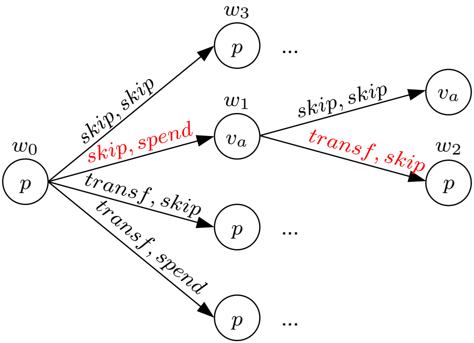
For example,



We have $\mathcal{M}, w_0 \models \langle\langle\emptyset\rangle\rangle \mathrm{X}(\neg p \rightarrow v_a) \wedge \langle\langle a\rangle\rangle \mathrm{X}p$.
If and only if $\mathcal{M}, w_0 \models \mathrm{OX}_a p \wedge \langle\langle a\rangle\rangle \mathrm{X}p$.
Then we have $\mathcal{M}, w_0 \models \mathrm{FRX}_a p$.
We also have $\mathcal{M}, w_0 \models [a{:}\sigma]\mathrm{X}v_a \wedge \langle\langle Agt\rangle\rangle \mathrm{X}\neg v_a$.

Forward-Looking vs. Backward-Looking

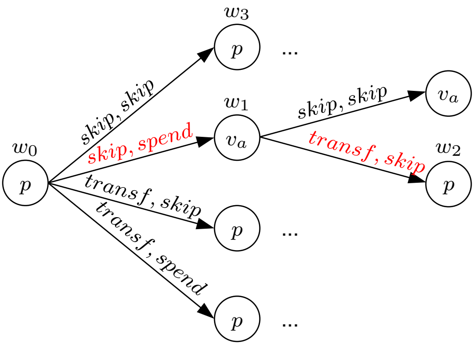
For example,



We have $\mathcal{M}, w_0 \models \langle\!\langle \emptyset \rangle\!\rangle \mathrm{X}(\neg p \rightarrow v_a) \wedge \langle\!\langle a \rangle\!\rangle \mathrm{X} p$.
If and only if $\mathcal{M}, w_0 \models \mathrm{OX}_a p \wedge \langle\!\langle a \rangle\!\rangle \mathrm{X} p$.
Then we have $\mathcal{M}, w_0 \models \mathrm{FRX}_a p$.
We also have $\mathcal{M}, w_0 \models [a{:}\sigma]\mathrm{X} v_a \wedge \langle\!\langle Agt \rangle\!\rangle \mathrm{X} \neg v_a$.
Then we have $\mathcal{M}, w_0 \models [a{:}\sigma] v_a \wedge \mathrm{AX}_{a:\sigma} v_a$.

Forward-Looking vs. Backward-Looking

For example,



We have $\mathcal{M}, w_0 \models \langle\langle \emptyset \rangle\rangle \mathrm{X}(\neg p \rightarrow v_a) \wedge \langle\langle a \rangle\rangle \mathrm{X} p$.
If and only if $\mathcal{M}, w_0 \models \mathrm{OX}_a p \wedge \langle\langle a \rangle\rangle \mathrm{X} p$.
Then we have $\mathcal{M}, w_0 \models \mathrm{FRX}_a p$.
We also have $\mathcal{M}, w_0 \models [a{:}\sigma]\mathrm{X} v_a \wedge \langle\langle Agt \rangle\rangle \mathrm{X} \neg v_a$.
Then we have $\mathcal{M}, w_0 \models [a{:}\sigma] v_a \wedge \mathrm{AX}_{a:\sigma} v_a$.
Therefore, $\mathcal{M}, w_0 \models \mathrm{BRX}_{a:\sigma} v_a$.

Future Works

States of Affairs vs. Actions

Future Works

States of Affairs vs. Actions

Forward-looking responsibility inherits from obligations the distinction between ought-to-do and ought-to-be statements.

Future Works

States of Affairs vs. Actions

Forward-looking responsibility inherits from obligations the distinction between ought-to-do and ought-to-be statements.

Consider the example of the bank account again. Note that the responsibility of maintaining the balances positive together with the fact that account 2 will be negative implies the responsibility for making a transfer to account 2.

Future Works

States of Affairs vs. Actions

Forward-looking responsibility inherits from obligations the distinction between ought-to-do and ought-to-be statements.

Consider the example of the bank account again. Note that the responsibility of maintaining the balances positive together with the fact that account 2 will be negative implies the responsibility for making a transfer to account 2.

That is, Alice is responsible for executing an action (or, following a strategy).

Future Works

States of Affairs vs. Actions

Forward-looking responsibility inherits from obligations the distinction between ought-to-do and ought-to-be statements.

Consider the example of the bank account again. Note that the responsibility of maintaining the balances positive together with the fact that account 2 will be negative implies the responsibility for making a transfer to account 2.

That is, Alice is responsible for executing an action (or, following a strategy).

We can also define dynamic obligations as abbreviations:

$$\mathrm{OX}_C(\sigma) \stackrel{\text{def}}{=} [C{:}\overline{\sigma}]\mathrm{X}v_C$$

$$\mathrm{OG}_C(\sigma) \stackrel{\text{def}}{=} [C{:}\overline{\sigma}]\mathrm{G}v_C$$

where $\overline{\sigma}$ is stand for not-σ.

Future Works

States of Affairs vs. Actions

Forward-looking responsibility inherits from obligations the distinction between ought-to-do and ought-to-be statements.

Consider the example of the bank account again. Note that the responsibility of maintaining the balances positive together with the fact that account 2 will be negative implies the responsibility for making a transfer to account 2.

That is, Alice is responsible for executing an action (or, following a strategy).

We can also define dynamic obligations as abbreviations:

$$\mathrm{OX}_C(\sigma) \stackrel{\text{def}}{=} [C{:}\overline{\sigma}]\mathrm{X}v_C$$

$$\mathrm{OG}_C(\sigma) \stackrel{\text{def}}{=} [C{:}\overline{\sigma}]\mathrm{G}v_C$$

where $\overline{\sigma}$ is stand for not-σ.

We need to define operations over strategies, similar to PDL.

Future Works

Complete Knowledge Assumption

Future Works

Complete Knowledge Assumption

Consider once more the example with Alice. Note that we made the implicit assumption that she knows the result of action $spend$.

Future Works

Complete Knowledge Assumption

Consider once more the example with Alice. Note that we made the implicit assumption that she knows the result of action $spend$.

What would happen if Alice did not know it? We probably would not consider her backward-looking responsible for the balance is negative on Wednesday.

Future Works

Complete Knowledge Assumption

Consider once more the example with Alice. Note that we made the implicit assumption that she knows the result of action $spend$.

What would happen if Alice did not know it? We probably would not consider her backward-looking responsible for the balance is negative on Wednesday.

The addition of an operator K for knowledge in ATL was already proposed by [van der Hoek and Wooldridge, 2003].

However in our framework there are some technical problems to be solved. For instance its interaction with obligations.

Thank you!