Reasoning about Normative Systems

Wiebe van der Hoek

JURIX, 16/12/2010

1Work with Thomas Ågotnes, Mark Roberts, & Michael Wooldridge
Overview

1. Introduction and Background
   - Computational Modeling of Normative Systems
   - Logic
   - Alternating-time Temporal Logic (ATL)

2. Robustness
   - Introduction
   - Necessary and Sufficient Coalitions
   - Quantifying Robustness

3. Outlook
   - Power in Normative Systems
   - Tractable Instances

4. Conclusion
Contents

1. Introduction and Background
   - Computational Modeling of Normative Systems
   - Logic
   - Alternating-time Temporal Logic (ATL)

2. Robustness

3. Outlook

4. Conclusion
Social software
- an interdisciplinary research program
- mathematical tools and techniques from game theory and computer science
- to analyze and design social procedures.
- goals: modeling social situations, developing theories of correctness, and designing social procedures.

Key research field: multi-agent systems
- Concerned with systems of interacting, autonomous (more or less) rational, agents
- *Coordination* is the overall issue
A normative system (social law) is a set of constraints on agent behaviour.

A norm set of rules imposed upon a multi-agent system with the goal of ensuring that some desirable behaviour will result.

Work by prohibiting the performance of certain actions in certain states.
A normative system (social law) is a set of constraints on agent behaviour.

Norm

A set of rules imposed upon a multi-agent system with the goal of ensuring that some desirable behaviour will result.

- Work by prohibiting the performance of certain actions in certain states.
- This is not social sciences nor is it law....
A normative system (social law) is a set of constraints on agent behaviour.

**Norm**

A set of rules imposed upon a multi-agent system with the goal of ensuring that some desirable behaviour will result

- Work by prohibiting the performance of certain actions in certain states.
- This is not social sciences nor is it law.... it is computer science!
Example

**Norms:**

- **nc:** controller only sets east to green if E is waiting and W is not waiting, similar for west.
- **nE:** Train E only enters on green.
- **nW:** Train W only enters on green.
There are two ways in which social laws can come to exist in a system:

1. **Offline design**
   Mechanisms are engineered at *design time*

2. **Emergence at run-time**
   Agents develop the social laws at run-time; typically by co-learning, copying, ...

We are here interested in **offline design**.
Starting point

- State-transition models
- Quite common abstraction
  - Ex.: UML
From state-transition model to tree

Every path in the tree represents a possible execution of the system
We can use **Computation Tree Logic** (CTL) to express properties of the system.

- **E ∃ φ** on some path, φ is true next
- **E(φ U ψ)** on some path, φ until ψ
- **E ◻ φ** on some path, eventually φ
- **E □ φ** on some path, always φ
- **A ∃ φ** on all paths, φ is true next
- **A(φ U ψ)** on all paths, φ until ψ
- **A ◻ φ** on all paths, eventually φ
- **A □ φ** on all paths, always φ
E\(\bigcirc \varphi\): on some path, \(\varphi\) is true next
A\(\bigcirc\varphi\): on all paths, \(\varphi\) is true next
E\(\Diamond \varphi\): on some path, eventually \(\varphi\)
A♦ϕ: on all paths, eventually ϕ
$$E \varphi: \text{on some path, always } \varphi$$
A $\varphi$: on all paths, always $\varphi$
$A(\varphi \mathcal{U} \psi)$: on all paths, $\varphi$ until $\psi$
Model Checking

- is an (automated) \textit{verification} technique
- given a finite set of states $S$
  - an elevator, a railway (cross-road), a communication protocol, \ldots
- and a \textbf{temporal property} $\varphi$
  - safety: ‘\textbf{bad things will never happen}’ $A \Box \neg \psi$
  - liveness: ‘something \textbf{good will eventually occur}’ $A \diamond \psi$
- \textbf{exhaustively check} whether $\varphi$ is true of $S$
Model Checking as Planning

exhaustively check whether $\varphi$ is true of $S$

- if YES: that’s fine
- if NO: read off trace of states
  - if property was good: improve the system
    “model checking as diagnosis”
    cf. $S \not\models A\square \text{good}$
  - if property was bad: use the trace as a plan
    “model checking as planning”
    cf. $S \not\models A\square \text{bad}$
Computational Properties of CTL

- **Satisfiability problem for CTL:**
  Given CTL formula \( \varphi \) is there some model that satisfies \( \varphi \)?
  Time complexity: \( \text{EXPTIME}-\text{complete} \).
  (So directly proving properties of systems using CTL looks to be hard.)

- **Model checking problem for CTL:**
  Given model \( M = \langle S, R, \pi \rangle \), state \( s_0 \in S \), and formula \( \varphi \), is \( \varphi \) is true at state \( s_0 \) in \( M \)?
  Time complexity: \( O(|M|.|\varphi|) \).
In 1997, Alur, Henzinger & Kupferman proposed a natural variation of CTL known as **Alternating-time Temporal Logic (ATL)**.

Branching used to model evolution of a system controlled by a set of **agents**, which can affect the future by making **choices**.

The particular future that will emerge depends on **combination** of choices that agents make.

Thus: a temporal logic built on the notion of **agency**.
Cooperation Modalities

- Path quantifiers $A$, $E$ are replaced by cooperation modalities:

$$
\langle\langle G \rangle\rangle \varphi
$$

means

“group $G$ can cooperate to ensure that $\varphi$”

or, equivalently:

“$G$ have a collective strategy to force $\varphi$”

- ATL generalises CTL, since

$$
\langle\langle \emptyset \rangle\rangle \text{ is same as } A
$$

$$
\langle\langle \Sigma \rangle\rangle \text{ is same as } E
$$
Computational Problems

- **Satisfiability** problem for ATL: EXPTIME-complete (van Drimmelen, 2003). Hence no worse that CTL.
- **Model-checking** problem: PTIME-complete. (Efficient model checkers have been implemented: MOCHA.)
Example ATL Formulae

\(\langle wvdh \rangle \Diamond \text{bored\_audience} \)

wvdh has a strategy for ensuring that the audience is eventually bored

\(\neg \langle \text{logicians} \rangle \Box \text{excited} \)

logicians have no strategy for ensuring that the audience is always excited

\(\langle \text{law\_people, computer\_scientists} \rangle (\text{bored} \cup \text{excited}) \)

people from law and computer scientists, have a strategy for ensuring that, even if the audience were now bored, in the end they will all be excited (!)
Controller Example

Introduction and Background

Robustness

Outlook

Conclusion

Controller Example

\(s\)

\(x=0\)

\(<\text{rej},\text{set}(0)>\)

\(<\text{rej},\text{set}(1)>\)

\(<\text{acc},\text{set}(0)>\)

\(x=1\)

\(<\text{rej},\text{set}(0)>\)

\(<\text{rej},\text{set}(1)>\)

\(<\text{acc},\text{set}(1)>\)

\(<\text{acc},\text{set}(1)>\)

\(<\text{acc},\text{set}(0)>\)
Controller Example

\[\begin{align*}
&x=0 \rightarrow \langle \langle s \rangle \rangle \\
&\land \\
&x=1 \\
&\rightarrow \langle \langle s \rangle \rangle
\end{align*}\]
Controller Example

- $(x = 0 \rightarrow \langle\langle s \rangle\rangle \mathcal{O} x = 0) \land (x = 1 \rightarrow \langle\langle s \rangle\rangle \mathcal{O} x = 1)$
- $s$ can enforce the value of $x$ to remain the same in the next step
Controller Example

\[\begin{aligned}
\text{x} = 0 &\rightarrow \langle \langle s \rangle \rangle \bigcirc \text{x} = 0 \land \langle \langle s \rangle \rangle \bigcirc \text{x} = 1 \\
\text{x} = 0 &\rightarrow \neg \langle \langle c \rangle \rangle \lozenge \text{x} = 1
\end{aligned}\]

\text{c cannot change the value from 0 to 1, even in multiple steps}
Controller Example

- \((x = 0 \rightarrow \langle s \rangle \Box x = 0) \land (x = 1 \rightarrow \langle s \rangle \Box x = 1)\)
- \(x = 0 \rightarrow \neg\langle c \rangle \Diamond x = 1\)
- \(x = 0 \rightarrow \neg\langle s \rangle \Diamond x = 1\)

\(s\) cannot change the value on his own either
Controller Example

- $(x = 0 \rightarrow \llbracket s \rrbracket \bigcirc x = 0) \land (x = 1 \rightarrow \llbracket s \rrbracket \bigcirc x = 1)$
- $x = 0 \rightarrow \neg \llbracket c \rrbracket \Diamond x = 1$
- $x = 0 \rightarrow \neg \llbracket s \rrbracket \Diamond x = 1$
- $x = 0 \rightarrow \llbracket s, c \rrbracket \Diamond x = 1$

$s$ and $c$ can cooperate to change the value effectively
Social Laws Framework

In our framework, a social law consists of two parts:

1. An **objective**
   What we **want to achieve** with this social law.

2. A **behavioural constraint**
   The mechanism by which we will achieve it.
A social law is a pair

\[(\varphi, \eta)\]

where:

- \(\varphi\) is an ATL formula called the **objective** of the law
- \(\eta : (Ac_1 \cup \cdots \cup Ag_n) \rightarrow 2^Q\) is a **behavioural constraint**.
Behavoural constraints are required to be “reasonable” (every agent must be able to do something).

- Implement $\eta$ in AATS $S = \text{eliminate from } S \text{ all transitions forbidden by } \eta$
- Implementation of $\eta$ is an update on AATSS, resulting in a new AATS
- AATSS obtained from $S$ by implementing $\eta$ denoted by $S + \eta$
Consider *universal* and *existential* sublanguages of ATL, denoted $\mathcal{L}^u$ and $\mathcal{L}^e$:

$$\mathcal{L}^u \quad v ::= p \mid \neg p \mid v \ conjunctions v \mid \langle\langle\rangle\rangle \Diamond v \mid \langle\langle\rangle\rangle \Box v \mid \langle\langle\rangle\rangle v \cup \langle\langle\rangle\rangle v$$

$$\mathcal{L}^e \quad e ::= p \mid \neg p \mid e \ conjunctions e \mid \langle\langle Ag\rangle\rangle \Diamond e \mid \langle\langle Ag\rangle\rangle e \mid \langle\langle Ag\rangle\rangle \Box e \mid \langle\langle Ag\rangle\rangle$$

where $p \in \Phi$. 
Suppose we have AATS $S$, a behavioural constraint $\eta$, a state $q$ in $S$, and formulae $\nu \in \mathcal{L}^u$, $\epsilon \in \mathcal{L}^e$. Then:

1. **Implementation preserves universal properties:**
   If $S, q \models \nu$ then $S \dagger \eta, q \models \nu$.

2. **Existential properties of updated systems are there in the original system:**
   If $S \dagger \eta, q \models \epsilon$ then $S, q \models \epsilon$. 
A social law $(\varphi, \eta)$ is effective in $S$ if

$$S \vdash \eta, q_0 \models \真相 \varphi$$

That is, if after implementing it, the objective is guaranteed to hold.

Thus, the Effectiveness Problem is:

*Given $S$ and $(\varphi, \eta)$ over $S$, determine whether $(\varphi, \eta)$ is effective in $S$.*

Implies effectiveness problem may be solved in time polynomial in the size of $S$ and $\varphi$. 
An Example System
Example Social Laws

- Obvious requirement: the trains don’t crash:

\[ O_1 = \neg (in_E \land in_W) \]

- Consider the behavioural constraint \( \eta_1 \) such that:
  - when both agents are waiting to enter the tunnel, the eastbound train is prevented from moving;
  - when the westbound train is already in the tunnel and the eastbound train is waiting to enter the tunnel, then the eastbound train is prevented from moving; and
  - when the eastbound train is already in the tunnel and the westbound train is waiting to enter the tunnel, then the westbound train is prevented from moving.

\( (O_1, \eta_1) \) is an effective social law in the trains system.
More Examples

- But consider the social law $\eta_2$ that simply prevents both trains from moving: $(O_1, \eta_2)$ is also effective!

- Refine our original objective:

$$O_2 = O_1 \land \bigwedge_{i \in \{E,W\}} \left( \begin{array}{l} (\text{away}_i \rightarrow \langle \langle i \rangle \rangle \Diamond \text{waiting}_i) \land \\ (\text{waiting}_i \rightarrow \langle \langle i \rangle \rangle \Diamond (\text{in}_i \land O_1)) \land \\ (\text{in}_i \rightarrow \langle \langle i \rangle \rangle \bigcirc \text{away}_i) \end{array} \right)$$

- $\eta_3$ forbids trains from lingering in tunnel, but is otherwise the same as $\eta_1$: $(O_2, \eta_3)$ is effective.
Given $S$ and a formula $\varphi$ of $\text{ATL}$ representing an objective, does there exist a $\eta$ such that $(\varphi, \eta)$ is an effective social law in $S$.

- This problem is...  
  \text{NP}-complete for arbitrary $\text{ATL}$ objectives formulae...  
  \text{NP}-complete for $\text{CTL}$-objectives...  
  ... and even $\text{NP}$-complete for $\mathcal{L}^u$ objectives!

- But it is polynomial for propositional logic objectives.
Given $S$ and a formula $\varphi$ of $\text{ATL}$ representing an objective, exhibit a behavioural constraint $\eta$ such that $(\varphi, \eta)$ is an effective social law in $S$ if such a constraint exists, otherwise answer “no”.

In general, requires solving $\text{NP}$-hard optimisation problem...
There is a link between model checking and feasibility.

Intuition: An objective is feasible if the agents could cooperate to make it work.

Suppose $\varphi$ is a propositional logic formula (representing an objective), and $S$ is an aats. Then:

$$S \models \langle Ag \rangle \square \varphi \iff \varphi \text{ is feasible in } S.$$  

So, we get synthesis here as a “side effect” of model checking.
This result does not hold for arbitrary formulae.
Consider the following objective:

\[ p \land \langle i \rangle \Diamond \neg p \]

On the one hand want to delete all \( \neg p \) states (to ensure \( \square p \))
But on the other hand, we need them to ensure \( \square \langle i \rangle \Diamond \neg p \).
For arbitrary ATL objectives, we have:

Suppose \( \nu \in L^u \) is a universal ATL formula
(representing an objective), and \( S \) is an AATS. Then:

\[ S \models \langle Ag \rangle \Box \nu \text{ implies } \nu \text{ is feasible in } S. \]
A simplification

1. We label each transition with the name of an agent.
   - This assumes asynchronous action.
(2) We label some of the transitions as **undesirable** or illegal

- Such a labelling is called a **behavioural constraint**
- It is typically the case that if none of the illegal transitions are used, the system will behave in a desirable way
- Fundamental assumption: agents choose whether or not to comply
Example

A system with a single non-sharable resource, which is desired by two agents.

We have two states, $s$ and $t$, and two corresponding Boolean variables $p_1$ and $p_2$, which are mutually exclusive.

$p_i$ means “agent $i$ has control”

Each agent has two possible actions, when in possession of the resource: either give it away, or keep it.
**Example**

Initial states: $S_0 = \{s, t\}$.

- $K \models \mathsf{E} \mathsf{O} p_2$
- $K \models \neg \mathsf{A} \mathsf{O} p_2$
- $K \models \mathsf{E} \mathsf{O} \mathsf{E} \mathsf{□} p_2$
Example

\[ K: \]

\[ \eta_1 = \{(s, s)\} \]

\[ K^\dagger \eta_1 \]

\[ \begin{align*} &p_1 & &p_2 \\ &s & &t \end{align*} \]
Example

$K$: 

Let $\eta_1 = \{(s, s)\}$. $K \upharpoonright \eta_1$: 

---

Let $\eta_1 = \{(s, s)\}$. $K \upharpoonright \eta_1$: 

---
Compliance

- What happens if not all the agents comply with the norm?
- Largely overlooked aspect
- Many reasons for non-compliance
  - deliberate and rational
  - deliberate and irrational
  - accidental
Define operators on normative systems which correspond to non-compliance of groups of agents

$$\eta \uparrow C$$

is the normative system that is the same as $\eta$ except that it only contains the arcs of $\eta$ that correspond to the actions of agents in $C$.

$$\eta \downarrow C$$

denotes the normative system that is the same as $\eta$ except that it only contains the arcs of $\eta$ that do not correspond to actions of agents in $C$. 
Example

\[ \varphi = A \diamond p \]

\[ K \not \models A \diamond p \]
Example

\[ \varphi = A \diamond p \]

\[ K \not\models A \diamond p \]

\[ K \vdash \eta \models A \diamond p \]
Not all norms are created equal

- There might be several effective normative systems
- Which one is better?

\[ K \vdash \eta \models A \diamond p \]
\[ K \vdash \eta' \models A \diamond p \]
\[ K \vdash \eta'' \models A \diamond p \]
A normative system is robust to the extent to which it remains effective (i.e., the system goal is still satisfied) in the event of non-compliance by some agents.
Example

The system will not overheat as long as at least one sensor works as it should and either one of the relief valves is working as it should or the automatic shutdown is working as it should.
We formalise three approaches to characterising robustness:

1. Identify coalitions whose compliance is necessary or sufficient
2. Find the number of agents that we can tolerate non-compliance from
3. Logical characterisations
Let a context $K$, a norm $\eta$ and an objective $\varphi$ be given. We say that $C \subseteq Ag$ are sufficient for $\eta$ if the compliance of $C$ with $\eta$ is effective, i.e., iff:

$$\forall C' \subseteq Ag : (C \subseteq C') \implies [K \vdash (\eta \restriction C') \models \varphi].$$
Let a context $K$, a norm $\eta$ and an objective $\varphi$ be given. We say that $C \subseteq Ag$ are sufficient for $\eta$ if the compliance of $C$ with $\eta$ is effective, i.e., iff:

$$\forall C' \subseteq Ag : (C \subseteq C') \implies [K \vdash (\eta \upharpoonright C') \models \varphi].$$

$$\varphi = A \square p$$
Let a context $K$, a norm $\eta$ and an objective $\varphi$ be given. We say that $C \subseteq Ag$ are sufficient for $\eta$ if the compliance of $C$ with $\eta$ is effective, i.e., iff:

$$\forall C' \subseteq Ag : (C \subseteq C') \Rightarrow [K \vdash (\eta \upharpoonright C') \models \varphi].$$

$$\varphi = A \square p$$

Sufficient: $\{c, a\}$
Let a context $K$, a norm $\eta$ and an objective $\varphi$ be given. We say that $C \subseteq Ag$ are sufficient for $\eta$ if the compliance of $C$ with $\eta$ is effective, i.e., iff:

$$\forall C' \subseteq Ag : (C \subseteq C') \Rightarrow [K \vdash (\eta \upharpoonright C') \models \varphi].$$

$$\varphi = A \square p$$

Sufficient: $\{c, a\} \quad \{c, b\}$
Let a context $K$, a norm $\eta$ and an objective $\varphi$ be given. We say that $C \subseteq Ag$ are sufficient for $\eta$ if the compliance of $C$ with $\eta$ is effective, i.e., iff:

$$\forall C' \subseteq Ag : (C \subseteq C') \Rightarrow [K \vdash (\eta \restriction C') \models \varphi].$$
Let a context $K$, a norm $\eta$ and an objective $\varphi$ be given. We say that $C \subseteq Ag$ are sufficient for $\eta$ if the compliance of $C$ with $\eta$ is effective, i.e., iff:

$$\forall C' \subseteq Ag : (C \subseteq C') \implies [K \vdash (\eta \upharpoonright C') \models \varphi].$$

$C$ being sufficient $\implies C \cup \{d\}$ is sufficient, e.g.,

$$K \vdash (\eta \upharpoonright C) \models E\Diamond \text{happy}(d) \& K \vdash (\eta \upharpoonright C \cup \{d\}) \not\models E\Diamond \text{happy}(d)$$
Take the system above, with $\varphi = E \circ A \circ p$.

1. $\{a\}$ is sufficient
2. $K \vdash \eta \upharpoonright \{b\} \models \varphi$;
3. none of $\{b\}$, $\{c\}$ or $\{b,c\}$ is sufficient
We say that $C$ are **necessary** for $\eta$ iff $C$ must comply with $\eta$ in order for it to be effective, i.e., iff:

$$\forall C' \subseteq A : [K + (\eta \upharpoonright C') \models \varphi] \implies (C \subseteq C').$$
We say that $C$ are necessary for $\eta$ iff $C$ must comply with $\eta$ in order for it to be effective, i.e., iff:

$$\forall C' \subseteq A : [K \vdash (\eta \upharpoonright C') \models \varphi] \Rightarrow (C \subseteq C').$$

$$\varphi = A \Box p$$
We say that $C$ are necessary for $\eta$ iff $C$ must comply with $\eta$ in order for it to be effective, i.e., iff:

$$\forall C' \subseteq A : [K + (\eta \upharpoonright C') \models \varphi] \implies (C \subseteq C').$$

$$\varphi = A \Box p$$

Necessary: $\{c\}$
Example: Use of Resources

Four agents $a, b, c, d$ attend a conference which’ facilities offer Resources $R_j$, where certain combinations are Useful ($ufl(k)$) for $k$, who may own a resource of type $j$: $k^o_j$.

$$R_1 = \{printer_1\}, R_2 = \{scanner_1, scanner_2\}, R_3 = \{pc_1, pc_2, pc_3\}$$

$$ufl(a) = \{R_1, R_3\}, ufl(b) = \{R_1, R_2\}, ufl(c) = \{R_2, R_3\}, ufl(d) = \{R_2\}$$

state: $s = \langle O_a, O_b, O_c, O_d, i \rangle$ and $s_0 = \langle \emptyset, \emptyset, \emptyset, \emptyset, a \rangle$

- assumption: agents take turns: $a, b, c, d, a, b, c, d, a, \ldots$
- call this system $K_0$
Example (ctd)

Let $\eta_0$ be the norm on $K_0$ that imposes that no agent

1. owns two resources of the same type at the same time
2. takes possession of a resource that he does not need,
3. takes possession of two new resources simultaneously
4. fails to take possession of a useful resource when available at his turn

Let $K_1 = K_0 \dagger \eta_0$ and $happy(k) = \bigwedge_{R_j \in ufl(k)} k^o_j$
Example: Sufficient Coalitions

\[ ufl(a) = \{R_1, R_3\}, \ ufl(b) = \{R_1, R_2\}, \ ufl(c) = \{R_2, R_3\}, \ ufl(d) = \{R_2\} \]
Example: Sufficient Coalitions

\[ ufl(a) = \{R_1, R_3\}, ufl(b) = \{R_1, R_2\}, ufl(c) = \{R_2, R_3\}, ufl(d) = \{R_2\} \]

\[ \varphi_1 = \bigwedge_{i \in A} A \Diamond \text{happy}(i) \]

\[ \eta_1 = \{(s, s') | \text{turn}(s) = i \& O_i = ufl(i) \& O'_i \neq \emptyset\} \]
Example: Sufficient Coalitions

\( u f l(a) = \{R_1, R_3\}, u f l(b) = \{R_1, R_2\}, u f l(c) = \{R_2, R_3\}, u f l(d) = \{R_2\} \)

\[
\varphi_1 = A \square \bigwedge_{i \in A} A \Diamond \text{happy}(i)
\]

\[
\eta_1 = \{(s, s') | \text{turn}(s) = i \& O_i = u f l(i) \& O_i' \neq \emptyset\}
\]

- if \( a \) does not comply, he can keep the printer forever, making \( b \) unhappy
- same holds for \( b \) wrt \( a \)
- so \( \{a, b\} \) is a necessary coalition
Example: Sufficient Coalitions

\[ u_{fl}(a) = \{R_1, R_3\} \]
\[ u_{fl}(b) = \{R_1, R_2\} \]
\[ u_{fl}(c) = \{R_2, R_3\} \]
\[ u_{fl}(d) = \{R_2\} \]

\[ \varphi_1 = A \bigwedge_{i \in A} A \diamond \text{happy}(i) \]
\[ \eta_1 = \{(s, s') \mid \text{turn}(s) = i \land O_i = u_{fl}(i) \land O'_i \neq \emptyset\} \]

- if \( a \) does not comply, he can keep the printer forever, making \( b \) unhappy
- same holds for \( b \) wrt \( a \)
- so \( \{a, b\} \) is a necessary coalition
- still, \( \{a, b\} \) is not sufficient: (\( c \) and \( d \) can keep the scanner)
  So, \( c \) or \( d \) needs to comply to free a scanner:
  \[ K_1 \vdash (\eta_1 \upharpoonright \{a, b, c\}) \models \varphi_1 \]
- Sufficient coalitions: \( \{a, b, c\} \), \( \{a, b, d\} \) and \( \{a, b, c, d\} \).
Example: Sufficient Coalitions

\[ ufl(a) = \{R_1, R_3\}, ufl(b) = \{R_1, R_2\}, ufl(c) = \{R_2, R_3\}, ufl(d) = \{R_2\} \]

\[ \varphi_1 = A \Box \bigwedge_{i \in A} A \Diamond \text{happy}(i) \]

\[ \eta_1 = \{(s, s') \mid \text{turn}(s) = i \land O_i = ufl(i) \land O'_i \neq \emptyset\} \]
Example: Sufficient Coalitions

\[\text{ufl}(a) = \{R_1, R_3\}, \text{ufl}(b) = \{R_1, R_2\}, \text{ufl}(c) = \{R_2, R_3\}, \text{ufl}(d) = \{R_2\}\]

\[\varphi_2 = \text{E} \square \neg \text{happy}(b)\]
Example: Sufficient Coalitions

\[ ufl(a) = \{R_1, R_3\}, ufl(b) = \{R_1, R_2\}, ufl(c) = \{R_2, R_3\}, ufl(d) = \{R_2\} \]

\[ \varphi_2 = E \Box \neg \text{happy}(b) \]

Observation:

\[ K_1 \vdash (\eta_1 \upharpoonright \{b\}) \models \varphi_2, \text{ but } \]
\[ K_1 \vdash (\eta_1 \upharpoonright \{a, b, c\}) \models \neg \varphi_2 ! \]
Some Results

**General C-sufficiency**

Deciding C-sufficiency is co-NP-complete
Some Results

General $C$-sufficiency
Deciding $C$-sufficiency is co-$NP$-complete

Universal $C$-sufficiency
Deciding $C$-sufficiency for universal objectives is polynomial time decidable
Some properties

- There might be no sufficient coalitions.
- There is always a necessary coalition: the empty coalition.
- There might be disjoint sufficient coalitions.
- There might be no non-empty necessary coalitions.
- If $C$ is necessary and $C'$ sufficient, then $C \subseteq C'$.
- ...
Feasibility of Robust Systems

Given a goal $\varphi$, and a ‘reliable’ coalition $C$:

$C$-sufficient feasibility

$$\exists \eta : (K \vdash \eta \models \varphi) \quad \& \quad \forall C' \subseteq Ag : (C \subseteq C') \Rightarrow [K \vdash (\eta \upharpoonright C') \models \varphi].$$
Feasibility of Robust Systems

Given a goal $\varphi$, and a ‘reliable’ coalition $C$:

**C-sufficient feasibility**

$$
\exists \eta : (K \vdash \eta \models \varphi) \quad \& \\
\forall C' \subseteq Ag : (C \subseteq C') \Rightarrow [K \vdash (\eta \upharpoonright C') \models \varphi].
$$

**Theorem**

Deciding $C$-sufficient feasibility is $\Sigma^p_2$-complete.
$k$-sufficiency

Let $K$ and $\varphi$ be given.

**Definition**

Where $k \geq 1$, we say a normative system $\eta$ is $k$-sufficient if the compliance of any arbitrary $k$ agents is sufficient to ensure that the normative system is effective with respect to $\varphi$. Formally, this involves checking that:

$$\forall C \subseteq A : (|C| \geq k) \implies (K \vdash (\eta \upharpoonright C)) \models \varphi.$$
Example (thanks to Dov Gabbay)

A senate with $n$ members. Normative system: follow the party line. The normative system is robust in the sense that we can tolerate $k$ rebels and still function towards our goals.
Let $K$ and $\varphi$ be given.

**Definition**

$\eta$ is $k$-necessary (w.r.t. $K$, $\varphi$) iff:

$$\forall C \subseteq A : (K \vdash (\eta \upharpoonright C)) \models \varphi \quad \Rightarrow \quad (|C| \geq k).$$
We define the *resilience* of a normative system $\eta$ (w.r.t. $K$, $\varphi$) as the largest number of non-compliant agents the system can tolerate.

**Definition**

the resilience is the largest number $k$, $k \leq n$, such that

$$\forall C \subseteq A : (|C| \leq k) \quad \Rightarrow \quad (K \vdash (\eta \upharpoonright A \setminus C)) \models \varphi.$$

where $n$ is the number of agents.
Theorems

Deciding $k$-sufficiency, $k$-necessity and resilience is co-NP-complete.
Contents

1. Introduction and Background
2. Robustness
3. Outlook
   - Power in Normative Systems
   - Tractable Instances
4. Conclusion
The Importance of Individual Agents

- Both sufficient and necessary agents are *important*. However, agents who (for example) are neither sufficient or necessary might have a very different degree of importance.
  - For example, it might be that one agent ensures the goal when he joins *almost all* coalitions, but not *all* (hence, she is not sufficient).
- Given this, it makes sense to consider in more detail *how important* agents are to success/failure of the normative system.
- Our idea: use *power indices* developed in cooperative game theory/voting theory for this purpose.
Power in Normative Systems

- Idea: measure the power of agents by complying/not complying with a norm.
- Given $K, \eta, \varphi$, we obtain a simple cooperative game $\langle A, \nu_S \rangle$, where $A$ is as in $K$, and $\nu_S$ is:

$$\nu_S(C) = \begin{cases} 
1 & \text{if } K \vdash (\eta \upharpoonright C) \models \varphi \\
0 & \text{otherwise.}
\end{cases}$$

In other words, a coalition “wins” if their compliance to $\eta$ (and the others not complying) will make $\varphi$ hold.
Power Indices

- **Power indices** characterise the *influence* that an agent has, by measuring how effective the agent is at turning a losing coalition into a winning coalition.

- Agent $i$ is said to be a *swing player* for $C \subseteq A$ if $C$ is not winning but $C \cup \{i\}$ is.

Define a function $\text{swing}(C, i)$ (where $i \notin C$) so that this function returns 1 if $i$ is a swing player for $C$, and 0 otherwise, i.e.,

$$\text{swing}(C, i) = \begin{cases} 1 & \text{if } \nu(C) = 0 \text{ and } \nu(C \cup \{i\}) = 1 \\ 0 & \text{otherwise.} \end{cases}$$
The Banzhaf Score

The Banzhaf score for $i$, $\sigma_i$, is number of coalitions for which $i$ is a swing player:

$$\sigma_i = \sum_{C \subseteq A \setminus \{i\}}^{\text{swing}(C, i)}$$  \hspace{1cm} (1)
The Banzhaf Measure

The Banzhaf measure, denoted $\mu_i$, is the probability that $i$ would be a swing player for a coalition chosen at random from $2^{A \setminus \{i\}}$:

$$\mu_i = \frac{\sigma_i}{2^{n-1}} \quad (2)$$
The Banzhaf Index

- **Banzhaf index** for \( i \in A \), denoted by \( \beta_i \), is the proportion of coalitions for which \( i \) is a swing to the total number of swings in the game:

\[
\beta_i = \frac{\sigma_i}{\sum_{j \in A} \sigma_j}
\]  

(3)
Complexity of the Banzhaf Score

Theorem

Given a social system $S = \langle K, \varphi, \eta \rangle$ and agent $i$ in $K$, computing the Banzhaf score $\sigma_i$ for $i$ in the corresponding coalitional game $G(S)$ is $\#P$-complete.

This is a very negative result: worse than NP hardness.
Complexity of Computing Power

**Theorem**

*Given a social system* $S = \langle K, \varphi, \eta \rangle$ *and agent* $i$ *in* $K$, *the following problems are* #P*-equivalent: computing the Banzhaf index* $\beta_i$; *and computing the Banzhaf measure* $\mu_i$. 
Dummies and Dictators

We say that a player $i$ is a dictator in a social system if $\mu_i = 1$, and a dummy if $\mu_i = 0$.

Theorem

Given a social system $S = \langle K, \varphi, \eta \rangle$ and agent $i$ in $K$, the following problems are co-NP-complete: checking whether $\sigma_i = 0$; checking whether $\mu_i = 0$; checking whether $\mu_i = 1$; checking whether $\beta_i = 0$; checking whether $\beta_i = 1$; checking whether $\varsigma_i = 0$; and checking whether $\varsigma_i = 1$. 
A normative system is *minimal* if no transitions can be eliminated without the norm failing.

**Theorem**

If $S = \langle K, \varphi, \eta \rangle$ is a minimal social system, then for each $i \in A(\eta)$, the values $\sigma_i$, $\mu_i$, $\beta_i$, and $\varsigma_i$ are polynomial time computable.
Suppose your objective is to keep $p$ true.
A *bridge* normative system is an easily identified type of minimal system: where we have a single transition (the bridge) leading to a “bad region” in which the objective is never satisfied.

Bridge norms are minimal, and can be easily identified.

Certain *tree-like* systems can also be seen to have minimal normative systems, and easily computable power indices.
Contents

1. Introduction and Background
2. Robustness
3. Outlook
4. Conclusion
Normative systems and Computer Science meet at *transition systems*

*Computationally* some negative results, but also identified positive cases

I have not emphasized the *reasoning* aspect at the object language level

*Extensions*: goals, utilities, rationality, knowledge of norms

Ability to reason about *several norms* simultaneously