Norms in Agent Systems

Mehdi Dastani

Utrecht University
Norms
Norm Types

Norms (e.g., legal, social, moral) are a popular candidate for coordination. Norms are standards of behaviour which prescribe certain behavior.

- Norms can be about **states**, **actions**, or **behaviors**, e.g.,
  - state norms: *The maximum length of papers is 15 pages.*
  - action norms: *PC members cannot review own papers.*
  - behavior norms: *Reviews should be delivered in time.*

- Norm can be **enforced** by means of rewards/sanctions or regimented.
  - Enforcement: *sanction page limit violation by additional fees.*
  - Regimentation: *prevent reviewing conflicting papers.*
  - Enforcement: *blacklist PC members with late reviews.*
Norm Perspectives

Control and coordination perspectives on norms in agent systems:

- Norms and Decision
- Norms and Interaction
- Computational Models for Norm-based Coordination
- Norm-based Organisation
An agent decision depends on its internals and norms

- **Norms awareness** Norm internalisation, Knowledge about (explicit/implicit) norms and consequence of their violations, Knowledge about enforcement mechanisms

- **Normative deliberation** Norms and cognition, Balancing between objectives/preferences and norms

- **Norms and Emotions** Moral Norms, Norms as coping mechanism
Norms as specifications of agents’ interaction

- **Norms and Equilibria** Equilibria represent (social) norms, norms as constraints on players’ actions or strategies.

- **Norms and Mechanism Design** Norms as specification of game form, if/which norms implements a social choice function in an equilibrium.

- **Norms and Evolutionary Game Theory (limited rationality)** Modelling the development of (social) norms, norm emergence, norms are successful strategies (higher payoffs), norm stability (robust against mutations), norm dynamics.

- **Norms and Coalitions** Norms operate in coalition formation, e.g., parity (rewards proportional to resources of members), equality (rewards shared equally), effect of norms on coalitions.
Computational Models for Norm-based Coordination

Norms can be used to build coordination systems

- Norm Expressivity versus Norm Computation: State- and action-based norms, norms with deadlines and sanctions, abstract and concrete norms

- Norm Dynamics: Operational semantics for norm revision, norm life-cycles, norm revision, abrogation and annulment

- Enforcement = Monitoring + Regulation: Exogenous versus endogenous implementation, norm regimentation, enforcement strategies, collective enforcement, run-time norm enforcement
Norm-based Organisation

Organisation can be engineered in terms of norms

- Normative organisations Electronic institution, Virtual organisation, Normative environment

- Norms in distributed organisations Assignment of norms to distributed organisations with different monitoring and enforcement capabilities, Distributed norm monitoring and enforcement

- Norm conflict and norm revision in organisations Detection of norm conflict, Resolving norm conflict, Norm change, Norm authorization
Application of Norms
Applications of Norms in Agent Systems

The following applications can be modelled as norm-based agent systems.

- business process management
- traffic management
- transportation systems
- financial systems
- healthcare
Companies, operating in dynamic and competitive environments, want to model and change their business processes rapidly and at lower cost while remaining compliant with business rules, companies’ policies and legislations, and contracts.

Monitoring processes provide statistics on the performance of processes, e.g., tracking the state of a customer order (e.g. order arrived, awaiting delivery, invoice paid) to detect and correct operational problems.
Business Rules (at Process, Task and Data levels)

- If a mortgage is accepted, then mortgage should be offered within two days after the request was received (Process)
- If LTV $\geq 85\%$ then interest deviation $= 0.4\%$ (Task)
- Rejected mortgages must have a reason for rejection (Data)
General architecture of an IT solution
The aim is to monitor and assess business rules at runtime, and to generate runtime responses when processes are not complaint with business rules.

Technical Challenges

- Representing business rules, laws and regulations
- Runtime monitoring and assessment
- Generating responses to non-compliant processes
Road traffic needs to be improved for the aspects of safety and the network performance by cooperative vehicle-roadside systems using car-to-car and car-to-road interactions.
The aim is to model decentralized, collaborative and scalable traffic management system.

Local control units assess, manage, and coordinate local traffic on the basis of available information, ranging from local to regional sensing and legal information.
Smart Road Project (UU & TNO & TUD)

The aim is to model decentralized, collaborative and scalable traffic management system.

Technical Challenges:

- Representing and reasoning with traffic rules.
- Distributed and imperfect monitoring and control.
- Simulating distributed traffic management systems.
Traffic rules are **instantiated** and communicated to cars. E.g., it is prohibited to exceed 50mph (20mph) in a ring road (town center).

Cars decide to **obey** or **violate** traffic rules. E.g., Cars violate obligations if they have more urgent goals.
Traffic rules are **decided** and communicated to cars. E.g., Normally, G-cars are obliged to slow down and give priority to simultaneously arriving Y-cars, unless there is heavy traffic on ramp which obliges Y-cars to either slow down or move to left lane and give priority to G-cars.
Engineering Norm-based Agent Systems
The development of normative multi-agent systems requires a programming language that support the implementation of norms and rewards/sanctions.

- **Brute facts** model the domain specific (environment) state, including action specifications;
- Agents modify brute facts by performing **actions**;
- **Control cycle** monitors agents actions and realizes their effects.
Coordination Component with Norms

- Brute facts model the domain specific (environment) state, including action specifications;
- Agents modify brute facts by performing actions;
- Ideal brute state described by norm;
- Violations and role enactment stored by institutional facts;
- Norm obedience/violation might lead to rewards/sanctions;
- Control cycle monitors agents actions and realizes their effects in the context of the norms and rewards/sanctions.
State-based norms (JLC 2011):

\[ F(\phi) \quad \text{Prohibited to achieve } \phi \text{ states} \]
\[ O(\phi) \quad \text{Obliged to achieve } \phi \text{ states} \]

Action-based norms (IJCAI 2011):

\[ F(\phi, \alpha) \quad \text{Prohibited to perform action } \alpha \text{ in } \phi \text{ states} \]
\[ O(\phi, \alpha) \quad \text{Obliged to perform action } \alpha \text{ in } \phi \text{ states} \]

Behaviour-based norms (IJCAI 2013):

\[ (\text{cond}, F(\phi), d) \quad \text{if } \text{cond}, \text{ then prohibited to achieve } \phi \text{ before } d \]
\[ (\text{cond}, O(\phi), d) \quad \text{if } \text{cond}, \text{ then obliged to achieve } \phi \text{ before } d \]
Examples of Behaviour-based Norms

Norms:

reviewdue(R):
  < phase(review) and assigned(R,P)
  , O(review(R,P))
  , phase(collect)>

minreviews(P):
  < phase(submission) and paper(P)
  , O( nrReviews(P) >= 2 )
  , phase(collect)>

pagelimit(P):
  < phase(submission) and paper(P),
  , F(pages(P) > 15)
  , phase(review)>
Recall norm:

\text{reviewdue} (R):
\langle \text{phase}\left(\text{review}\right) \text{ and } \text{assigned} (R,P), \text{O} (\text{review}(R,P)), \text{phase}\left(\text{collect}\right) \rangle

\text{brute facts} : \{ \text{phase}(\text{review}), \text{paper}(\text{p5}) \}

\text{inst. facts} : \{ \text{rea}(\text{john},\text{chair}) \}
Recall norm:

\[ \text{reviewdue}(R): \]
\[ < \text{phase}(	ext{review}) \text{ and } \text{assigned}(R,P), O(\text{review}(R,P)), \text{phase}(\text{collect}) > \]

assign(john, rose, p5)

brute facts : \{ phase(review), paper(p5), assigned(rose,p5) \}

inst. facts : \{ rea(john,chair) , 

(\text{reviewdue}(rose), O(\text{review}(rose,p5)), \text{phase}(\text{collect})) \}
Recall norm:

\[
\text{reviewdue}(R): \\
< \text{phase(review)} \text{ and } \text{assigned}(R,P), \ O(\text{review}(R,P)), \ \text{phase(collect)} >
\]
Evolution of Obligations

Recall norm:

\[ \text{reviewdue}(R): \]
\[ < \text{phase}(\text{review}) \text{ and } \text{assigned}(R,P), \ O(\text{review}(R,P)), \ \text{phase}(\text{collect}) > \]

assign(john,rose,p5) \quad \Rightarrow \quad uploadReview(rose,p5)

brute facts :  \{ \text{phase}(\text{review}), \ \text{paper}(p5), \ \text{assigned}(\text{rose},p5), \ \text{review}(\text{rose},p5) \} \\
inst. facts  :  \{ \text{rea}(\text{john},\text{chair}), \ \text{obey}(\text{reviewdue}(\text{rose})) \}
Recall norm:

\[
\text{reviewdue}(R) : \langle \text{phase(review)} \land \text{assigned}(R,P), O(\text{review}(R,P)), \text{phase(collect)} \rangle
\]

\[
\begin{align*}
\text{assign(john,rose,p5)} \\
\text{brute facts} & : \{ \text{phase(review)}, \text{paper(p5)}, \text{assigned(rose,p5)} \} \\
\text{inst. facts} & : \{ \text{rea(john,chair)} \}, \\
& \quad (\text{reviewdue(rose)}, O(\text{review(rose,p5)}), \text{phase(collect)})
\end{align*}
\]
Recall norm:

\[ \text{reviewdue}(R): \]
\[ \langle \text{phase(review)} \text{ and } \text{assigned}(R,P), \text{ O(review}(R,P)), \text{ phase(collect)} \rangle \]

\[ \text{assign(john,rose,p5)} \]
\[ \text{start(john,collect)} \]

\text{brute facts} : \{ \text{phase(collect), paper(p5), assigned(rose,p5)} \}

\text{inst. facts} : \{ \text{rea(john,chair), viol(reviewdue(rose))} \}
Behavior of an Obligation Summarized

\[ \neg \phi_x, \neg \phi_d, \neg \phi_l, \neg (O \phi_x, \phi_d) \]

- **obey**($\phi_l$)
- **viol**($\phi_l$)
- **obey**($\phi_l$)
Behavior of a Prohibition Summarized

\[ (\phi_l, F\varphi_x, \varphi_d) \]

\[ \neg\varphi_x, \neg\varphi_d \] \rightarrow \[ \neg\varphi_x, \neg\varphi_d \] \rightarrow \[ \neg\varphi_x, \varphi_d \] \rightarrow \[ \varphi_x, \varphi_d \] \rightarrow \[ \neg\varphi_x, \varphi_d \]

\[ \text{viol}(\phi_l) \] \rightarrow \[ \text{obey}(\phi_l) \]

\[ (\phi_l, F\varphi_x, \varphi_d) \] \rightarrow \[ \varphi_x, \neg\varphi_d \] \rightarrow \[ \text{viol}(\phi_l) \]

\[ (\phi_l, F\varphi_x, \varphi_d) \] \rightarrow \[ \varphi_x, \neg\varphi_d \] \rightarrow \[ \text{viol}(\phi_l) \]
Definition (Configuration of Coordination Component)

The state of a coordination component is a tuple $\langle \sigma_b, \sigma_i, \Delta \rangle$ with:

- $\sigma_b$ a set of ground first-order atoms, the brute state;
- $\sigma_i$ a set of ground first-order atoms, the institutional state;
- $\Delta$ a set of norms (Static);

For simplicity we ignore static $\Delta$ and present a configuration as:

$\langle \sigma_b, \sigma_i \rangle$
Triggering Norms

Transition Rule
Given ground substitution $\theta$, the rule for triggering of norms is defined as:

$$ns = (\phi_l : \langle \varphi_c, P(\varphi_x), \varphi_d \rangle) \in \Delta \quad \sigma_b \models \varphi_c \theta \quad ni = \text{inst}(ns, \theta)$$

$$\langle \sigma_b, \sigma_i \rangle \rightarrow \langle \sigma_b, \sigma_i \cup \{ni\} \rangle$$
Triggering Norms

Transition Rule

Given ground substitution $\theta$, the rule for triggering of norms is defined as:

$$ns = (\phi_l : \langle \varphi_c, \mathbb{P}(\varphi_x), \varphi_d \rangle) \in \Delta \quad \sigma_b \models \varphi_c \theta \quad ni = \text{inst}(ns, \theta)$$

$$\langle \sigma_b, \sigma_i \rangle \rightarrow \langle \sigma_b, \sigma_i \cup \{ni\} \rangle$$

- When the condition of a norm is satisfied
Transition Rule

Given ground substitution $\theta$, the rule for triggering of norms is defined as:

$$\text{ns} = (\phi_l : \langle \varphi_c, P(\varphi_x), \varphi_d \rangle) \in \Delta \quad \sigma_b \models \varphi_c \theta \quad ni = \text{inst}(\text{ns}, \theta)$$

$$\langle \sigma_b, \sigma_i \rangle \rightarrow \langle \sigma_b, \sigma_i \cup \{ni\} \rangle$$

- We instantiate it and add it to the institutional facts
Transition Rule
The rule for violation of an obligation is defined as follows:

\[
\begin{align*}
(\phi_I, O(\varphi_x), \varphi_d) \in \sigma_i & \quad \sigma_b \not\models \varphi_x \quad \sigma_b \models \varphi_d \\
\langle \sigma_b, \sigma_i \rangle & \longrightarrow \langle \sigma_b, (\sigma_i \setminus \{ni\}) \cup \{\text{viol}(\phi_I)\} \rangle
\end{align*}
\]
Transition Rule
The rule for violation of an obligation is defined as follows:

\[(\phi_l, O(\varphi_x), \varphi_d) \in \sigma_i \quad \sigma_b \not\models \varphi_x \quad \sigma_b \models \varphi_d\]

\[\langle \sigma_b, \sigma_i \rangle \rightarrow \langle \sigma_b, (\sigma_i \setminus \{ni\}) \cup \{viol(\phi_l)\} \rangle\]

- When an obligation is violated
Monitoring Obligations

Transition Rule
The rule for violation of an obligation is defined as follows:

\[
(\phi_l, O(\varphi_x), \varphi_d) \in \sigma_i \quad \sigma_b \not\models \varphi_x \quad \sigma_b \models \varphi_d
\]

\[
\langle \sigma_b, \sigma_i \rangle \longrightarrow \langle \sigma_b, (\sigma_i \setminus \{ni\}) \cup \{viol(\phi_l)\} \rangle
\]

- We remove it from the institutional facts
Transition Rule

The rule for violation of an obligation is defined as follows:

\[
(\phi_l, O(\varphi_x), \varphi_d) \in \sigma_i \quad \sigma_b \not\models \varphi_x \quad \sigma_b \models \varphi_d
\]

\[
\langle \sigma_b, \sigma_i \rangle \longrightarrow \langle \sigma_b, (\sigma_i \setminus \{ni\}) \cup \{\text{viol}(\phi_l)\} \rangle
\]

- We record its violation to the institutional facts
Operational Semantics: Properties

- Detached obligations remain in force until obeyed or violated.
- Violation of detached obligations are recorded in deadline states.
- Violation is inevitable in case of conflicting norms.
Business Process Management: Accepted mortgages should be offered within two days after it was received.

mortgageAccepted: } label
  < mortgage(M) & accepted(M) } condition
    , O(offer(M)) } obligation
    , currentDate=date(M)+2 } deadline

Smart Roads: It is prohibited to exceed 50mph (20mph) in a ring road (town center)

speedNorm: } label
  < entervillage(C) } condition
    , F(inVillage(C) & ex20mph(C)) } prohibition
    , leaveVillage(C) } deadline
Norm Programming using AspectJ: Pointcut

```java
public aspect SpeedNorm {
    private ArrayList<SNDetachment> detachments;

    pointcut condition(Car c, String type) :
        call(public * OrganisationInterface.entervillage(..)) &&
        args(c,type);

    pointcut prohibition(Car c, Sensor s) :
        call(* OrganisationInterface.inVillage(..)) &&
        args(c, s);

    pointcut deadline(Car c, String type) :
        call(public * OrganisationInterface.leaveVillage(..)) &&
        args(c,type);
}
```

```
speedNorm: < entervillage(C),
    F( inVillage(C) & ex20mph(C) ),
    leaveVillage(C) >
```
after() returning(Car c, String type) : condition(c, type){
    if(type.equals("urban")){
        detachments.add(new SNDetachment("urban",c, 50, 50));
        c.getDeonticProxy().addSpeedNorm(50);
    }
}

after(Car c, Sensor s): prohibition(c, s){
    for(SNDetachment d : detachments){
        if(d.getCar().equals(c)){
            if(s.getVelocity(c)>d.getLimit()){  
                organisation.makeFine(c,d.getFine());
            }
        }
    }
}

after(Car c, String type): deadline(c, type){
    if( d.getCar().equals(c )){
        detachments.remove(d);
    }
}
Norm Enforcement in Agent Systems
What is Norm Enforcement? (JANCL 2012, IJCAI 2013)

- Enforcing norms on system behaviours is updating the system with the norms.

- The questions we are interested in are:
  - How do norms change system behaviours?
  - How to check the properties of the norm enforcement?

- Some related work:
  - Ågotnes, van der Hoek, Wooldridge; *Robust normative systems* (IGPL 2010)
  - Dastani, Grossi, Meyer; *A Logic for Normative Multi-Agent Programs* (JLC 2011)
  - Knobbout and Dastani; *Reasoning under Compliance Assumptions in Normative Multiagent Systems* (AAMAS 2012)
General Architecture

- rewards/sanctions
- norms
- brute facts
- institutional facts

Control cycle
Conditional Norms

- Assume we have disjoint sets of **brute facts** propositional atoms $\Pi_b$ and **institutional facts** propositional atoms $\Pi_s$
- Let $cond$, $\phi$, $d$ be boolean combinations of propositional variables from $\Pi_b$ and $san \in \Pi_s$
- A **conditional obligation** is represented by the tuple

  $$(cond, O(\phi), d, san)$$

- A **conditional prohibition** is represented by the tuple

  $$(cond, F(\phi), d, san)$$

- A **norm set** $N$ is a set of conditional obligations and conditional prohibitions
State Violating Norms

- A state $\rho[i]$ violates a conditional obligation $(cond, O(\phi), d, san)$ on run $\rho$ iff

  $$\rho, i \models d \land \neg \phi \land ((\forall (\neg \phi \land \neg d)S(cond \land \neg \phi \land \neg d)) \lor cond)$$

  i.e., obligations are violated if $\phi$ does not become true in or before the deadline state.

- A state $\rho[i]$ violates a conditional prohibition $(cond, F(\phi), d, san)$ on run $\rho$ iff

  $$\rho, i \models \phi \land \neg d \land ((\forall (\neg \phi \land \neg d)S(cond \land \neg \phi \land \neg d)) \lor cond)$$

  i.e., prohibitions are violated in the first state where $\phi$ becomes true.
Let $M = (S, R, V)$ be a finite transition system with initial state $s_0$ and $N$ a finite set of conditional obligations and prohibitions.

A normative update of $M$ with $N$, $M^N = (S^N, R^N, V^N)$, is a tree unravelling $T(M)$ of $M$ where all norms from $N$ are enforced on all runs, i.e., in each tree node $s'$, $V^N(s')$ contains sanction atoms for all norms violated in $s'$. 
Example

Consider an obligation \((c, O(q), d, sanO)\) and a prohibition \((c', F(p), d', sanF)\)
Consider an obligation \((c, O(q), d, sanO)\) and a prohibition \((c', F(p), d', sanF)\)
Consider an obligation \((c, O(q), d, sanO)\) and a prohibition \((c', F(p), d', sanF)\).
Language of **CTLS**

- We propose a new logic, **CTLS**, for reasoning about normative updates of single-agent systems.

- **CTLS** is **CTL** with Sanction bounds.

- Path quantifiers have the form $E^{\leq Z}$, where $Z$ is a multiset of sanction bounds (where multiplicity of a sanction $san$ is infinity, we represent this as $\infty \ast san$).

- $E^{\leq Z}$ means 'there exists a path of sanction cost at most $Z$'.

\[
p \in \Pi_b \cup \Pi_s \mid \neg \phi \mid \phi \land \phi \mid E^{\leq Z} X \phi \mid E^{\leq Z} \phi \ Until \ \phi \mid E^{\leq Z} G \phi
\]
Semantics of **CTLS**

The truth of **CTLS** formulas is defined relative to a tree model $T$ (intuitively, a normative update) and a state $s \in T$: 

- $T, s \models E^{\leq Z} X \phi$ iff there exists a fullpath $\rho'$ with $\rho'[0] = s$, such that $T, \rho'[1] \models \phi$ and $\text{sanctions}(\rho') \leq Z$

- $T, s \models E^{\leq Z} \phi \text{ Until } \psi$ iff there exists a fullpath $\rho'$ with $\rho'[0] = s$, such that for some $n \geq 0$, $T, \rho'[n] \models \psi$ and for every $i, i < n$, $T, \rho'[i] \models \phi$ and $\text{sanctions}(\rho') \leq Z$

- $T, s \models E^{\leq Z} G \phi$ iff there exists a fullpath $\rho'$ with $\rho'[0] = s$, such that for every $i$, $T, \rho'[i] \models \phi$ and $\text{sanctions}(\rho') \leq Z$
The model-checking problem for a normative update in CTLS takes as inputs

- a finite transition system $M = (S, R, V)$,
- a state $s_0 \in S$,
- a finite set of conditional norms $N$, and
- a formula $\phi$ of CTLS

It returns true if $M^N, s_0 \models \phi$, and false otherwise.
The model-checking problem for a normative update in CTLS is in PSPACE (proof uses guessing and checking a polynomially representable path).

It is PSPACE-hard by reduction of QSAT problem (proof idea adapted from Bulling and Jamroga’s (IJCAI 2011) proof of PSPACE-hardness of $CTL^+$).
We also consider normative update of a multi-agent system (concurrent game structure)

Properties of a normative update of a MAS can be expressed in *ATLS* (*ATL* with Sanction bounds)

\[ p \in \Pi_b \cup \Pi_s \mid \neg \phi \mid \phi \land \psi \mid \langle\langle C\rangle\rangle \leq Z X \phi \mid \langle\langle C\rangle\rangle \leq Z G \phi \mid \langle\langle C\rangle\rangle \leq Z \phi U \psi \]

\[ \langle\langle C\rangle\rangle \leq Z \gamma \] means ‘the group of agents \( C \) has a strategy, all executions of which incur at most \( Z \) sanctions and satisfy the formula \( \gamma \), whatever the other agents in \( A \setminus C \) do’

\[ M^N, s \models \langle\langle C\rangle\rangle \leq Z \gamma \] iff there exists a strategy \( F_C \) of sanction cost at most \( Z \) in \( s \) such that for all \( \rho \in \text{out}(s, F_C) \), \( M^N, \rho \models \gamma \)
Complexity for *ATLS* normative update checking

- The model-checking problem for a normative update in *ATLS* is in PSPACE.
- It is PSPACE-hard from PSPACE-hardness of *CTLs*. 
Monitoring Norms in Agent Systems
Existing work on multi-agent systems typically assume perfect monitoring.

We want to develop a very general framework in order to ... 

▶ ... characterize monitors.
▶ ... reason about monitors.
▶ ... study the relation between monitors and behaviours.
Example: Traffic Regulations

When entering a village, it is prohibited to drive faster than 20 mph, until leaving the village.

\[ n = (enter\text{Village}, F(in\text{Village} \land 20\text{mph}), leave\text{Village}) \]

\[ n = (c, F(\phi), d) \]
Example: Traffic Monitoring

Possible behaviours of a car passing a village
Example: Traffic Monitoring

Possible behaviours of a car passing a village

$$\left( \text{entV} \ ; \ \text{inV} \land s_{20} \ ; \ \text{leaV} \right)^*$$
Example: Traffic Monitoring

Possible behaviours of a car passing a village

\[(\text{entV} \; ; \; \text{inv} \; \land \; s_{30} \; ; \; \text{leaV})^*\]
Example: Traffic Monitoring

Possible behaviours of a car passing a village

\[(\text{entV} ; \text{inV} \land s_{20} ; \text{leaV} ; \text{entV} ; \text{inV} \land s_{40} ; \text{leaV})^*\]
A monitor $m_\Phi$ is specified by a set of propositional state queries (perfectly observable state properties)
$\Phi = \{\phi_1, \ldots, \phi_n\}$

A monitor $m_\Phi$ define a partition of the set of states $S$ of a transition system. For all $s \in S$, we have

$$s \sim_\Phi s' \text{ iff } \forall \phi \in \Phi : s \models \phi \iff s' \models \phi$$

$\Phi = \{entV, inV, leaV, s_{20} \lor s_{30}\}$ defines partition

$\{\{s_1\}, \{s_4\}, \{s_2, s_3\}, \{s_5\}\}$
Path-based Imperfect Monitors

- A monitor $m_\Phi$ is specified by a set of propositional queries on states $\Phi = \{\phi_1, \ldots, \phi_n\}$

- Lifting a partition on states to a partition on paths in transition systems

$$\rho \sim_\Phi \rho' \iff \forall i: \rho[i] \sim_\Phi \rho'[i]$$

$$\left(s_1 s_2 s_5\right)^* \sim_\Phi \left(s_1 s_3 s_5\right)^*$$
Behaviour-based Imperfect Monitors

An imperfect monitor $m_\Phi$ defines a partitioning on the set of possible behaviours.

$$\Phi = \{entV, inV, leaV, s_{20} \lor s_{30}\}$$
An imperfect monitor $m_\Phi$ defines a partitioning on the set of possible behaviours.

$$\Phi = \{ \text{entV}, \text{inV}, \text{leaV}, s_{20} \lor s_{30} \}$$
Imperfect Monitors and Traffic Violations

Given a transition system $M$ and a regulation $n$, some equivalent classes defined by monitor $m_\Phi$ contain only violating behaviours, some only obeying behaviours, and some both.
Given a transition system $M$ and a regulation $n$, a set of violations of $n$ on $M$ is defined as:

$$\text{Viol}(M, n) = \{ \rho \mid \exists i \ M, \rho, i \models v(n) \}$$
Imperfect Monitors and Traffic Violations

Can we approximate regulation $n$ by a new one $n'$ for which the monitor $m_\Phi$ acts as a **perfect monitor**?

\[
\begin{array}{cccc}
\rho_1 \Phi & \rho_2 \Phi & \rho_3 \Phi & \rho_4 \Phi \\
\rho_5 \Phi & \rho_6 \Phi & \rho_7 \Phi & \rho_8 \Phi \\
\vdots & \vdots & \vdots & \vdots \\
\rho_1 \Phi & \rho_m \Phi & \rho_n \Phi & \cdots \\
\end{array}
\]

\[
\text{Viol}(M, n') \quad \text{Viol}(M, n)
\]
An exponential (in the set of observable properties $\Phi$) algorithm is designed that generates optimal approximation of norms (Re-Synthesizing Norms).

**Theorem (Complexity of Approximation Problem)**

*We prove that the approximation problem is at least as hard as the interpolation problem for propositional logic. Complexity of interpolation problem is open but is widely assumed to be at least EXPTIME.*
An alternative approach to the approximation problem is to combine monitors in order to perfectly monitor norms.

**Theorem (Complexity of Monitor Combinations)**

Given a transition system, a finite set of monitors, and a norm (LTL formulae). The problem whether some combination of the monitors can perfectly monitor the given norm is $\text{PSPACE}$-complete in the length of transition systems, set of monitors, and norm.