SOCIAL LAWS IN ALTERNATING TIME

Wiebe van der Hoek, Mark Roberts, Michael Wooldridge

University of Liverpool, UK

Overview

- In this talk, I will:

  Introduce a "social laws" framework for multiagent systems, based around the *alternating-time temporal logic* of Alur et al, and in particular, three key problems: *effectiveness*, *feasibility*, and *synthesis*.

- Structure of the talk:
  - introduce ATL;
  - introduce social laws;
  - introduce our social laws framework;
  - point to the future.

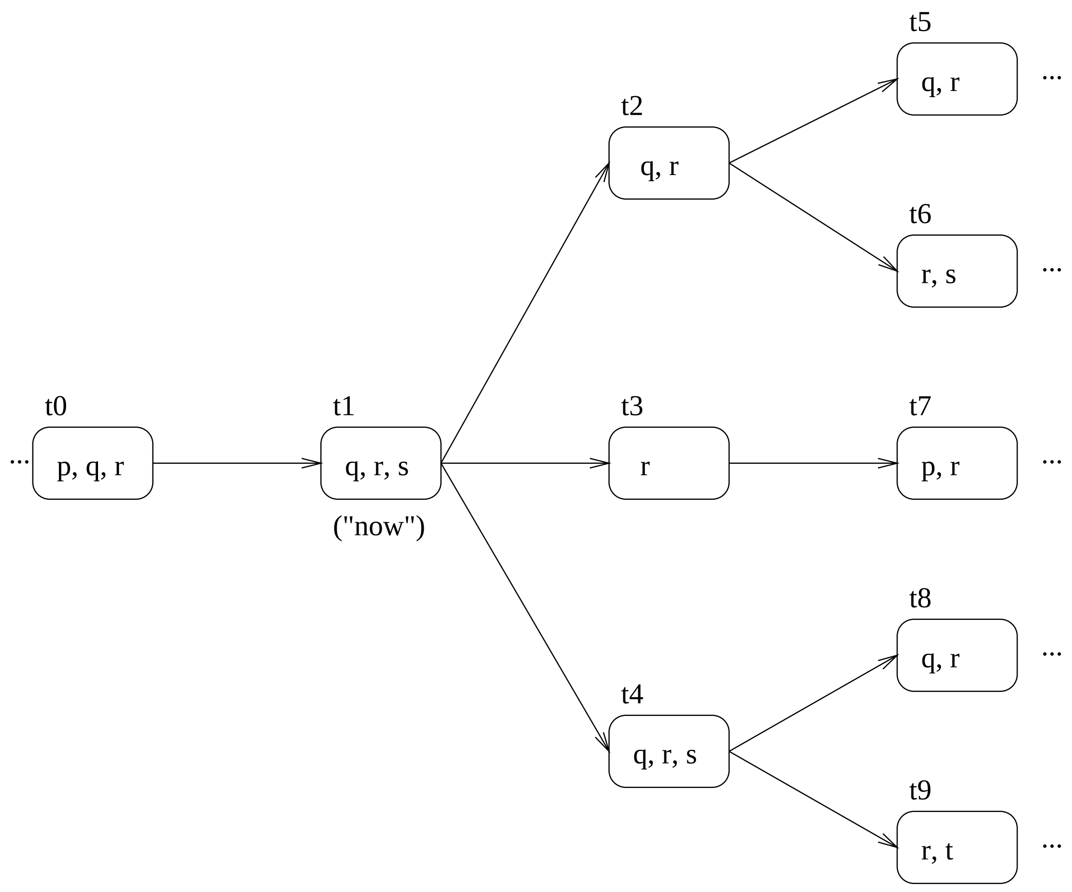
ATL: A Logic for Multiagent Systems

- *Alternating-time Temporal Logic* (“ATL”) was introduced in 1997 as a logic for reasoning about *game-like distributed systems*: i.e., *multi-agent systems*.
- Main item of novelty: allows us to talk about *powers* of system components.
- Main reason why it’s exciting: it *generalises* CTL, the “generic” branching time logic, without appearing to be more computationally complex.

Branching-time Temporal Logics

- Natural to view the possible computations of a system as a *tree* linear in the past, branching into the future.
- Branching corresponds to different ways in which non-determinism can be resolved.

A Branching time model



Computation Tree Logic: CTL

- Extends propositional logic with
  - *path quantifiers* A, E
  - *tense modalities* $\bigcirc, \Diamond, \Box, \mathcal{U}$
- Possible combinations of these are restricted as follows:

$A\bigcirc\varphi$	“on all paths, φ is true next
$A\Diamond\varphi$	“on all paths, φ is eventually true
$A\Box\varphi$	“on all paths, φ is always true
$A\varphi\mathcal{U}\psi$	“on all paths, φ is true until ψ
$E\bigcirc\varphi$	“on some path, φ is true next
$E\Diamond\varphi$	“on some path, φ is eventually true
$E\Box\varphi$	“on some path, φ is always true
$E\varphi\mathcal{U}\psi$	“on some path, φ is true until ψ

Models for CTL

- Models for CTL are *Kripke structures*:

$$\langle S, R, \pi \rangle$$

  where

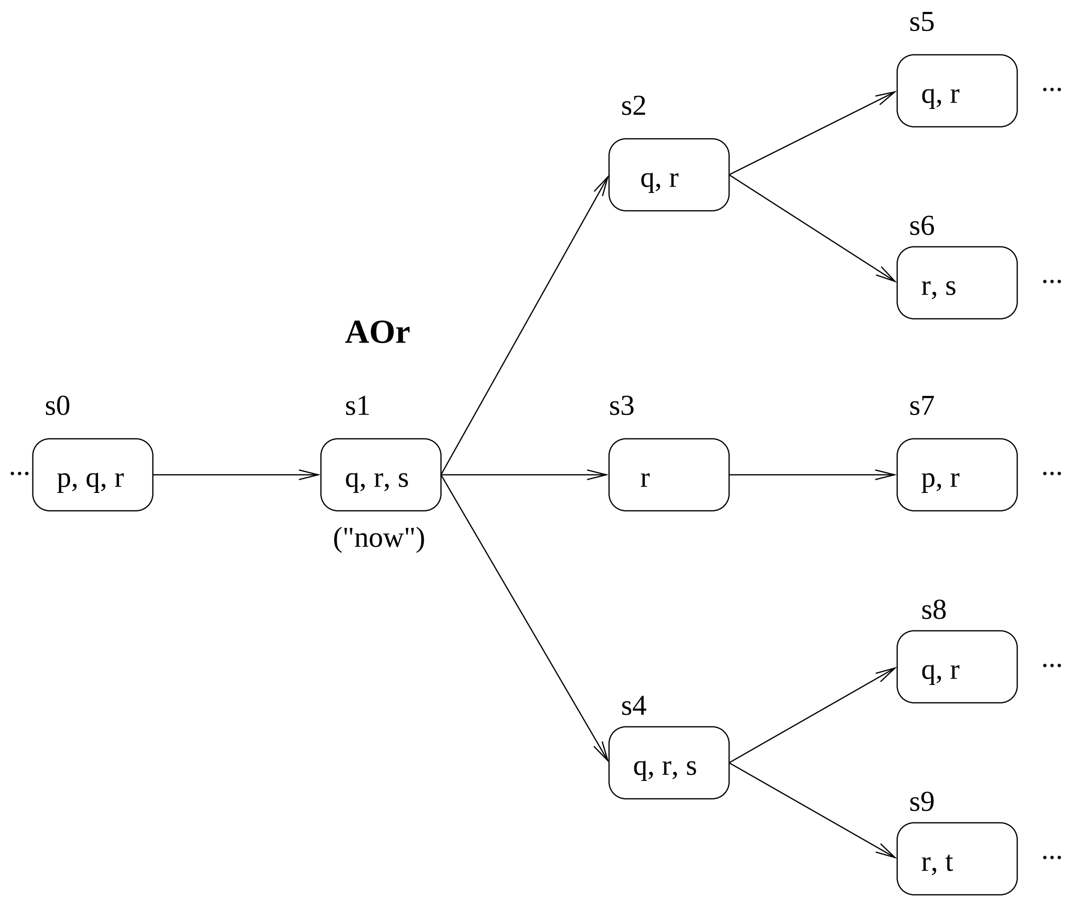
  – S is the set of possible system states

  – $R \subseteq S \times S$ is a total binary *next state* relation on S

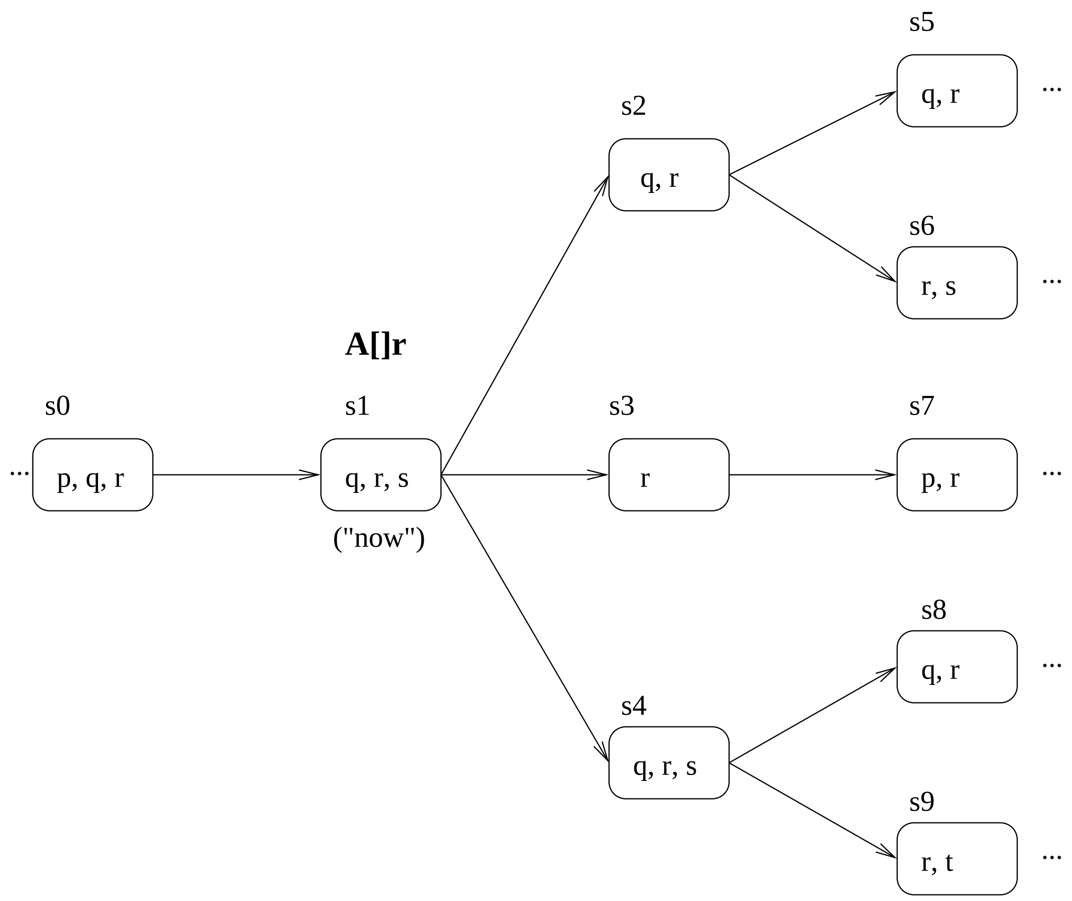
  – $\pi : S \rightarrow 2^{\Pi}$ says which propositions are true in each state.

- The branches are obtained by *unwinding* this relation, giving *paths* through the structure.

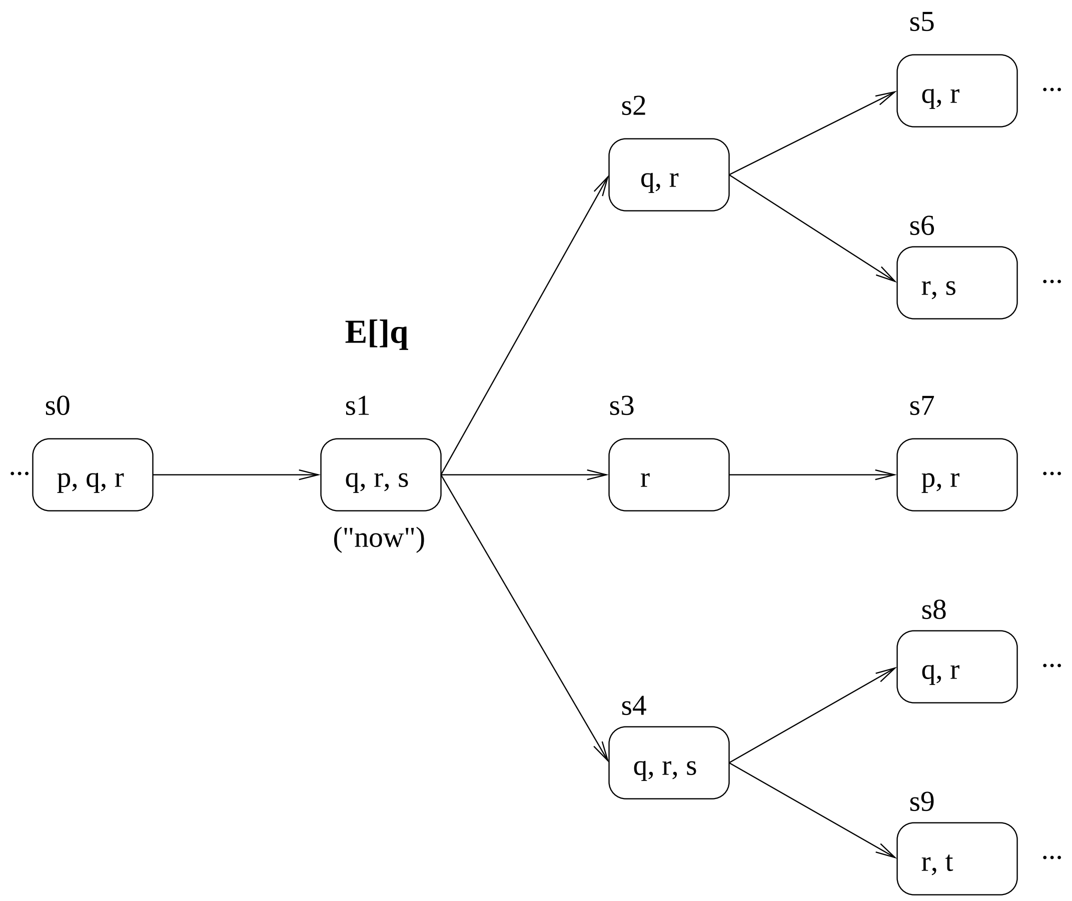
Example 1



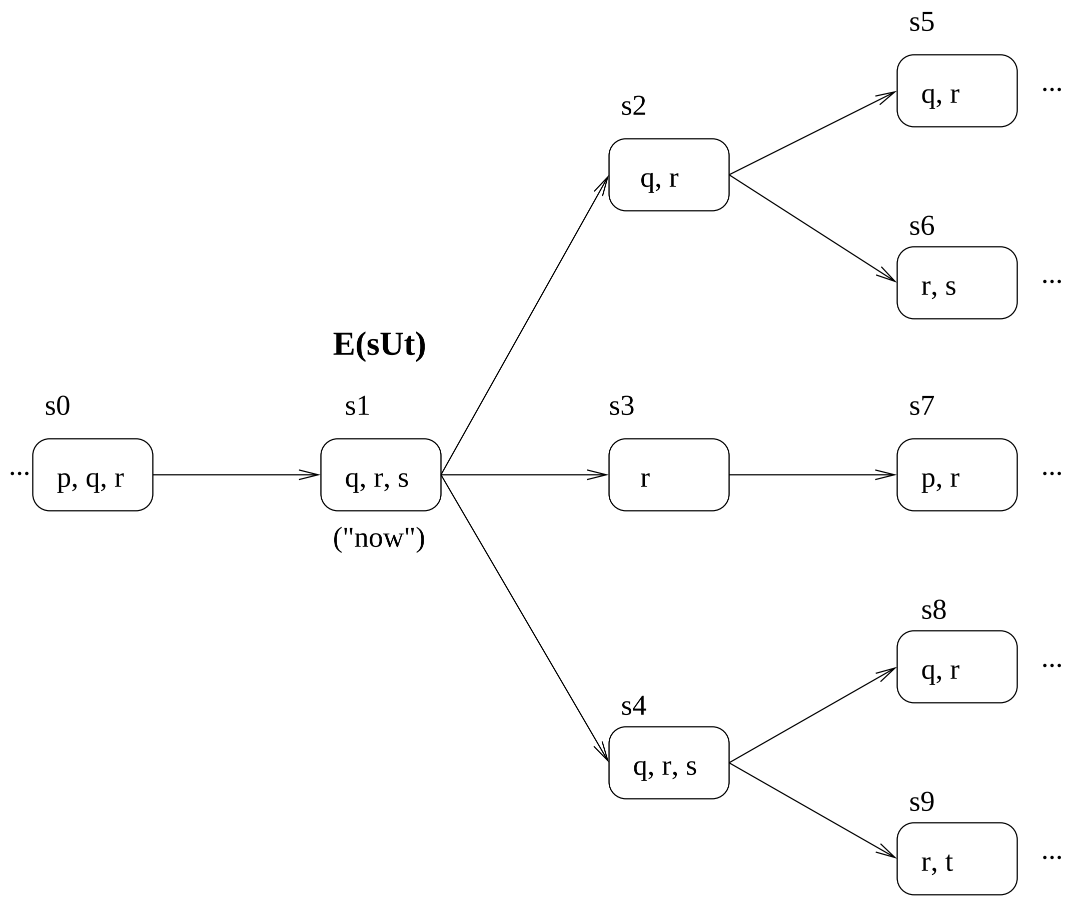
Example 2



Example 3



Example 4



Computational Properties of CTL

- *Satisfiability* problem for CTL:

  Given CTL formula φ is there some model that satisfies φ?

  Time complexity: EXPTIME-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 φ, is φ is true at state s_0 in M?

  Time complexity: $O(|M|.|\varphi|)$.

Alternating-time Temporal Logic

- 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 φ"

  or, equivalently:

  "G have a collective strategy to force φ"

- ATL generalises CTL, since

$\langle\langle\emptyset\rangle\rangle$ is same as A
$\langle\langle\Sigma\rangle\rangle$ 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

$\langle\langle mjw \rangle\rangle \Diamond \mathit{bored\text{-}audience}$

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

$\neg \langle\langle mjw \rangle\rangle \Box \mathit{excited}$

mjw has no strategy for ensuring that the audience is always excited

$\langle\langle gwb, tb \rangle\rangle \Diamond \mathit{peace}$

gwb and tb have a strategy for ensuring that, eventually, there is peace (!)

Social Laws

- Social laws are *coordination mechanisms*.

  A set of rules imposed upon a multiagent system with goal of ensuring that some desirable behaviour will result.

- Work by *prohibiting* the performance of certain actions in certain states.
- Note that this is *not* social sciences *nor* is it law. . . it's computer science!

Offline and Online Design

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*.

Previous Work

Offline design of social laws was first investigated by Moses, Shoham and Tennenholtz

- Moses and Tennenholtz investigate *artificial social systems* and issues that arise in their design
- Shoham and Tennenholtz propose the original framework of social laws and investigate the computational problems associated with them.
- Original framework is extended by Fitoussi and Tennenholtz to incorporate notions of minimal and simple social laws

Useful Social Laws

- A *constraint* is defined to be a pair

  $$\langle E', \alpha \rangle$$

  where

  - $E' \subseteq E$ is a set of environment states; and
  - $\alpha \in Ac$ is an action.

  A *social law* is a set sl of such constraints

- Each agent is associated with set F_i of *focal states*.

  Social law is *useful* if after implementing it, it's possible for every agent to move from any of its focal states to any other focal state.

- Useful social law problem: NP-complete.

Social Laws in Alternating Time

- We have developed a new framework, in which social laws can be expressed in ATL
- ATL provides a natural language for expressing social laws:
  - can capture *liveness* and *safety* properties, as it generalises CTL
  - can capture *powers* that agents can/should have.
- Some problems associated with social laws, such as the effectiveness, feasibility and synthesis problems reduce directly to ATL model checking problems
- Checked with existing model checkers, such as MOCHA

Semantic Structures: AATSs

$$\langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho, \tau, \Phi, \pi \rangle$$

- Q is a finite, non-empty set of *states*;
- $q_0 \in Q$ is the *initial state*;
- $Ag = \{1, \ldots, n\}$ is a set of *agents*;
- Ac_i is a set of *actions*;
- $\rho : Ac_1 \cup \cdots \cup Ac_n \to 2^Q$ is an *action precondition function*;
- $\tau : Q \times (Ac_1 \times \cdots Ac_n) \to Q$ is a *(partial) system transition function*;
- Φ is a set of *atomic propositions*;
- $\pi : Q \to 2^\Phi$ is an interpretation function.

(Some obvious *coherence constraints* are required.)

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.

$\boxed{\text{The Formal Definition}}$

A *social law* is a pair

$$(\varphi, \beta)$$

where:

- φ is an ATL formula called the *objective* of the law
- $\beta : (Ac_1 \cup \cdots \cup Ag_n) \to 2^Q$ is a *behavioural constraint*.

Behavioural Constraints

- Behavioural constraints are required to be “reasonable” (every agent must be able to do *something*).
- Implement β in AATS S = eliminate from S all transitions forbidden by β
- Implementation of β is an *update* on AATSs, resulting in a new AATS
- AATS obtained from S by implementing β denoted by $S \dagger \beta$

Implementing Behavioural Constraints

$$S \dagger \beta = \langle Q, q_0, Ag, Ac_1, \ldots, Ac_n, \rho', \tau', \Phi, \pi \rangle,$$

where:

1. $\forall \alpha \in Ac$,
   $$\rho'(\alpha) = \rho(\alpha) \setminus \beta(\alpha)$$
2. $\forall q \in Q, \forall j \in J_{Ag}$,
   $$\tau'(q, j) = \begin{cases} \tau(q, j) & \text{if } (q, j) \in \text{dom } \tau \text{ and } \forall i \in Ag, q \notin \beta(j_i) \\ \text{undefined} & \text{otherwise} \end{cases}$$
3. All other components of $S \dagger \beta$ are as in S.

Note that $S \dagger \beta$ can be computed in time polynomial in size of S, β.

Update Properties

Consider *universal* and *existential* sublanguages of ATL, denoted $\mathcal{L}^u$ and $\mathcal{L}^e$:

$$\mathcal{L}^u \quad \upsilon ::= p \mid \neg p \mid \upsilon \wedge \upsilon \mid \upsilon \vee \upsilon \mid \langle\langle\rangle\rangle \bigcirc \upsilon \mid \langle\langle\rangle\rangle \Diamond \upsilon \mid \langle\langle\rangle\rangle \Box \upsilon \mid \langle\langle\rangle\rangle \upsilon \mathcal{U} \upsilon$$

$$\mathcal{L}^e \quad \epsilon ::= p \mid \neg p \mid \epsilon \wedge \epsilon \mid \epsilon \vee \epsilon \mid \langle\langle Ag\rangle\rangle \bigcirc \epsilon \mid \langle\langle Ag\rangle\rangle \Diamond \epsilon \mid \langle\langle Ag\rangle\rangle \Box \epsilon \mid \langle\langle Ag\rangle\rangle \epsilon \mathcal{U} \epsilon$$

where $p \in \Phi$.

Properties of Updates

Suppose we have AATS S, a behavioural constraint β, a state q in S, and formulae $\upsilon \in \mathcal{L}^u$, $\epsilon \in \mathcal{L}^e$. Then:

1. *Implementation preserves universal properties*:

   If $S, q \models \upsilon$ then $S \dagger \beta, q \models \upsilon$.

2. *Existential properties of updated systems are there in the original system*:

   If $S \dagger \beta, q \models \epsilon$ then $S, q \models \epsilon$.

Effective Social Laws

- A social law (φ, β) is *effective* in S if

$$S \dagger \beta, q_0 \models \langle\langle\rangle\rangle \Box \varphi$$

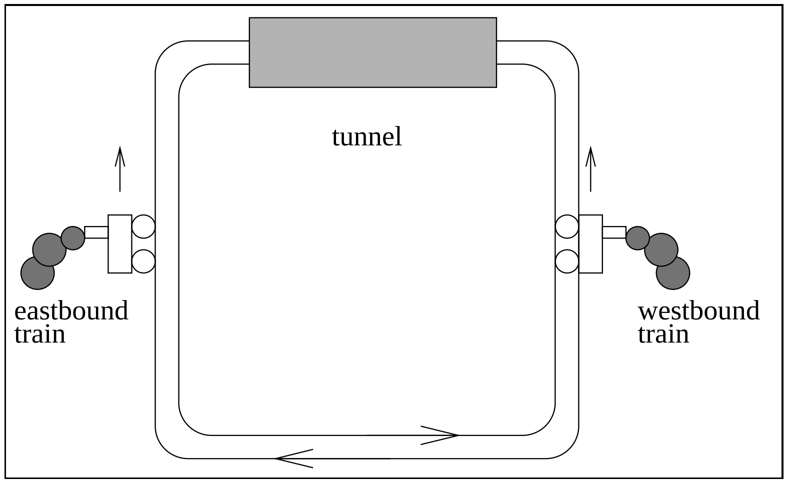
  That is, if after implementing it, the objective is guaranteed to hold.

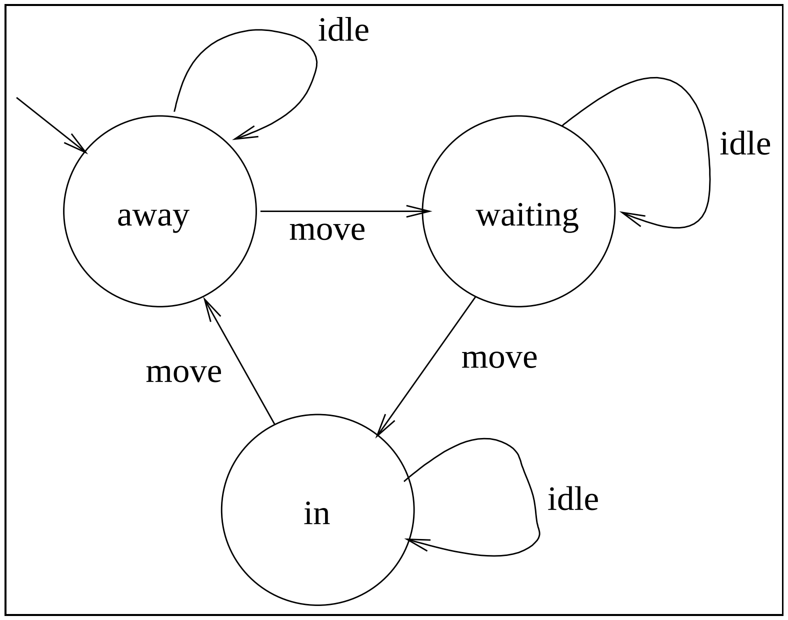
- Thus, the Effectiveness Problem is:

  Given S and (φ, β) over S, determine whether (φ, β) is effective in S.

- Implies effectiveness problem may be solved in time polynomial in the size of S and φ.

An Example System





Example Social Laws

- Obvious requirement: the trains don't crash:

$$O_1 = \neg(in_E \wedge in_W)$$

- Consider the behavioural constraint β_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, β_1) is an effective social law in the trains system.

More Examples

- But consider the social law β_2 that simply *prevents both trains from moving*: (O_1, β_2) is also effective!
- Refine our original objective:

$$O_2 = O_1 \;\wedge \bigwedge_{i \in \{E,W\}} \left(\begin{array}{ll} (away_i \rightarrow \langle\langle i \rangle\rangle \Diamond waiting_i) & \wedge \\ (waiting_i \rightarrow \langle\langle i \rangle\rangle \Diamond (in_i \wedge O_1)) & \wedge \\ (in_i \rightarrow \langle\langle i \rangle\rangle \bigcirc away_i) & \end{array} \right)$$

- β_3 forbids trains from lingering in tunnel, but is otherwise the same as β_1: (O_2, β_3) is effective.

The Feasibility Problem

Given S and a formula φ of ATL representing an objective, does there exist a β such that (φ, β) is an effective social law in S.

- This problem is...

  NP-complete for arbitrary ATL objectives formulae...

  NP-complete for CTL-objectives...

  ...and even NP-complete for $\mathcal{L}^u$ objectives!
- But it is polynomial for propositional logic objectives.

$\boxed{\text{Synthesis}}$

Given S and a formula φ of ATL representing an objective, exhibit a behavioural constraint β such that (φ, β) is an effective social law in S if such a constraint exists, otherwise answer "no".

- In general, requires solving NP-hard optimisation problem. . .

Synthesis for Propositional Objectives

- There is a link between *model checking* and *synthesis*.
- Intuition: An objective is feasible if the agents could cooperate to make it work.

Suppose φ is a propositional logic formula (representing an objective), and S is an AATS. Then:

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

- So, we get synthesis here as a “side effect” of model checking.

This Result Does Not Generalise

- This result does not hold for arbitrary formulae.
- Consider the following objective:

$$p \wedge \langle\langle i \rangle\rangle \Diamond \neg p$$

- On the one hand want to delete all $\neg p$ states (to ensure $\Box p$)
- But on the other hand, we need them to ensure $\Box \langle\langle i \rangle\rangle \Diamond \neg p$.

Synthesis for ATL Objectives

So, for arbitrary ATL objectives, we have:

Suppose $\upsilon \in \mathcal{L}^u$ is a universal ATL formula (representing an objective), and S is an AATS. Then:

$$S \models \langle\langle Ag \rangle\rangle \Box \upsilon \text{ implies } \upsilon \text{ is feasible in } S.$$

Referring to Actions

- Suppose we want to refer to legality of actions *explicitly* in objectives.
- For each action α, let $\ell(\alpha)$ be a proposition meaning "α is legal".

$$\begin{aligned}
O_4 &= \neg(in_E \wedge in_W) \wedge \ell(move_E) \wedge \ell(move_W) \\
O_5 &= \neg(in_E \wedge in_W) \wedge \ell(move_E) \\
O_6 &= \neg(in_E \wedge in_W) \wedge \ell(move_W)
\end{aligned}$$

Implementation using MOCHA

The idea:

- add “controller” agents, which control the $\ell(\alpha)$ variables
- change the pre-condition of each action to include this variable.

$$
\begin{array}{lcl}
\texttt{agent}\ \mathit{name}\ \texttt{reads}\ \mathit{in}\ \texttt{writes}\ \mathit{out} & & \texttt{agent}\ \mathit{name}\ \texttt{reads}\ \mathit{in}\ \texttt{writes}\ \mathit{out} \\
\quad P_1 \mapsto \alpha_1; & & \quad \ell(\alpha_1) \wedge P_1 \mapsto \alpha_1; \\
\quad P_2 \mapsto \alpha_2; & \Rightarrow & \quad \ell(\alpha_2) \wedge P_2 \mapsto \alpha_2; \\
\quad \cdots & & \quad \cdots \\
\quad P_k \mapsto \alpha_k. & & \quad \ell(\alpha_k) \wedge P_k \mapsto \alpha_k.
\end{array}
$$

Feasibility Again

Again, for propositional objectives, feasibility = model checking.

Suppose φ is a propositional logic formula (representing an objective), and S is an AATS. Then

$$S^{\circ} \models \langle\langle controllers \rangle\rangle \Box \left(\varphi \wedge \left[\bigwedge_{i \in Ag} \bigvee_{\alpha \in Ac_i} \ell(\alpha) \right] \right)$$

iff φ is feasible in S.

That Example Again

$$O_4 = \neg(in_E \wedge in_W) \wedge \ell(move_E) \wedge \ell(move_W)$$

…is not feasible, but…

$$O_5 = \neg(in_E \wedge in_W) \wedge \ell(move_E)$$

is feasible, and so is

$$O_6 = \neg(in_E \wedge in_W) \wedge \ell(move_W)$$

Future Work

- Extend the framework to incorporate *knowledge*
  (Feasibility for ATEL is NP-complete.)
- Algorithmic synthesis of social laws using algorithms
- Examples without trains.