

“Prof Kripke, let me introduce Prof Nash”

Logic for Automated Mechanism Design

Michael Wooldridge

(with Ågotnes, Dunne, van der Hoek, Pauly, and Ruan)

Department of Computer Science
University of Liverpool, UK

<http://www.csc.liv.ac.uk/~mjw/>



Mechanisms and Protocols

- Distributed systems research focusses on **protocols**.
- In multi-agent systems, we study **mechanisms**.
mechanism = protocol + self interest
- Mechanisms raise issues beyond deadlock, livelock, etc.
strategic considerations come to the fore.
- Treating mechanisms as if they were simply protocols misses a big part of the story.
example: sniping on eBay
- In MAS, mechanism participants are **software agents**.

Knowledge Representation for Mechanisms

- An agent's **environment** is a **mechanism**, containing **other agents**, which act **strategically** in their **own best interests**.
- This raises classic AI questions:

Knowledge Representation for Mechanisms

- An agent's **environment** is a **mechanism**, containing **other agents**, which act **strategically** in their **own best interests**.
- This raises classic AI questions:
 - how to **represent knowledge about a mechanism**?

Knowledge Representation for Mechanisms

- An agent's **environment** is a **mechanism**, containing **other agents**, which act **strategically** in their **own best interests**.
- This raises classic AI questions:
 - how to **represent knowledge about a mechanism**?
 - how to **apply** this knowledge, to make **good decisions**.

... and that is what this talk is all about.

Overview

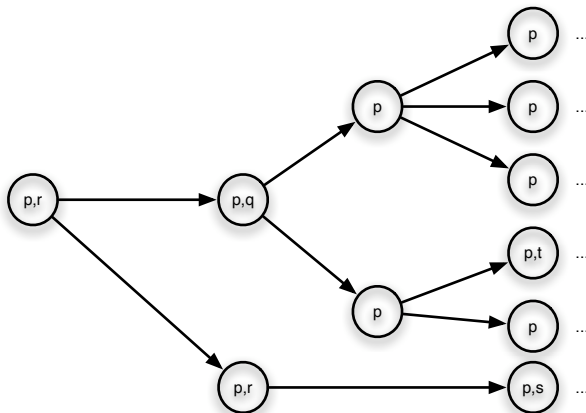
- Introduce ATL – a **cooperation logic**
- Show how ATL can be used for **mechanism specification**
- Discuss **reasoning** with ATL
 - **theorem proving**
 - **model checking**
- Introduce an ATL framework for **social laws**
- Discuss **social laws as mechanism design**
- Future work.

ATL: Alternating-time Temporal Logic

- **Alternating-time Temporal Logic** (“ATL”) introduced in 1997 for reasoning about **game-like distributed systems**.
- Main item of novelty: allows us to talk about **powers** of system components.
- A **branching-time** logic – generalises **Computation Tree Logic** (CTL).

A Branching Time model

(Branching time models = Kripke Structures)



Intuitively, arcs correspond to **possible actions**.

Computation Tree Logic: CTL

A language for talking about branching time structures.
Extends propositional logic with

- **path quantifiers** A, E
- **tense modalities** \bigcirc , \diamond , \square , \mathcal{U}

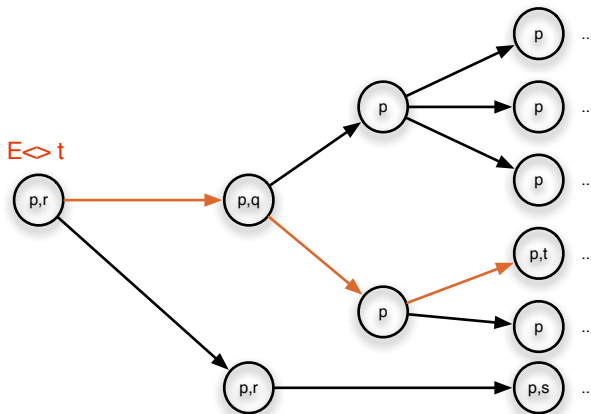
Computation Tree Logic: CTL

A language for talking about branching time structures.
 Extends propositional logic with

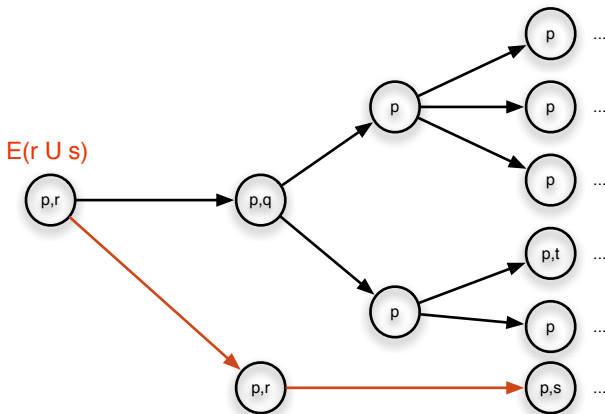
- **path quantifiers** A, E
- **tense modalities** \bigcirc , \diamond , \square , \mathcal{U}

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

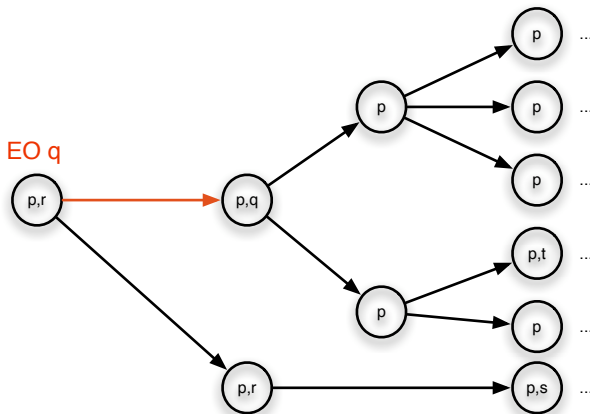
Interpreting CTL Formulae



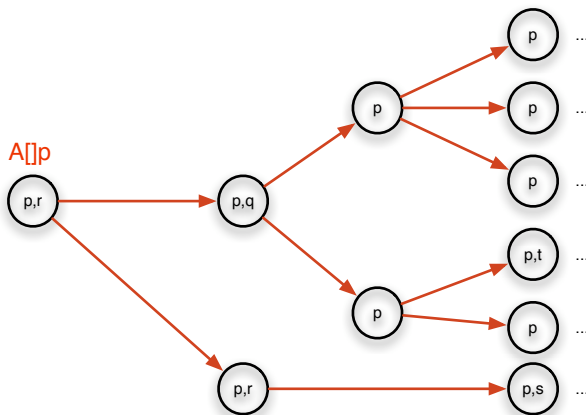
Interpreting CTL Formulae



Interpreting CTL Formulae



Interpreting CTL Formulae



Limitations of CTL

- In CTL we can only say something is **inevitable** or **possible**.
OK for **protocols**
(eg “never reach deadlock”: $A \Box \neg \text{dlock}$)
- Not much use for **mechanisms** – CTL has no notion of
 - **strategic action**;
 - **agency**.
- ATL is intended to overcome these limitations.

ATL

Basic expression in ATL is the **cooperation modality**:

$$\langle\langle C \rangle\rangle\varphi$$

means

“coalition C can cooperate to ensure that φ ”

ATL

Basic expression in ATL is the **cooperation modality**:

$$\langle\langle C \rangle\rangle\varphi$$

means

“coalition C can cooperate to ensure that φ ”

Cooperation modalities are combined with CTL-like **temporal** operators:

- \bigcirc in the next state
- \blacklozenge eventually
- \square always
- \mathcal{U} until

Coalition logic is the “ \bigcirc ”-fragment.

CTL is a Fragment of ATL

Suppose Ag is the set of all agents. Then...

CTL is a Fragment of ATL

Suppose Ag is the set of all agents. Then...

$\langle\langle Ag \rangle\rangle$ is the same as E

$\langle\rangle$ is the same as A

Models for ATL are basically Kripke structures with arcs labelled with **tuples** of actions.

Example ATL Formulae

$$\langle\langle gb, tb \rangle\rangle \diamond peace$$

George Bush and Tony Blair can cooperate to ensure that, eventually, there is peace.

Example ATL Formulae

$$\langle\langle gb, tb \rangle\rangle \diamond peace$$

George Bush and Tony Blair can cooperate to ensure that, eventually, there is peace.

Note:

- does **not** imply that *gb* and *tb* **know what the strategy is**,
- nor that they **choose** this strategy.

Example ATL Formulae

$$\langle\langle gb, tb \rangle\rangle \diamond peace$$

George Bush and Tony Blair can cooperate to ensure that, eventually, there is peace.

Note:

- does **not** imply that *gb* and *tb* **know what the strategy is**,
- nor that they **choose** this strategy.

It just says: **there is in principle some way that they could ensure there is eventually peace.**

Example ATL Formulae

$$\langle\langle i \rangle\rangle \diamond \langle\langle j \rangle\rangle \diamond goal_j$$

i can ensure that eventually *j* has the ability to achieve his goal.

Examples of Mechanisms

- **Social choice mechanisms:**
 - players **collectively choose an outcome** even though they have differing preferences (e.g., political elections)
 - incentivize players to tell the truth about their preferences (impossibility results, Gibbard-Satterthwaite theorem etc)

Examples of Mechanisms

- **Social choice mechanisms:**
 - players **collectively choose an outcome** even though they have differing preferences (e.g., political elections)
 - incentivize players to tell the truth about their preferences (impossibility results, Gibbard-Satterthwaite theorem etc)
- **Auctions:**
 - possible aims: maximise revenue, truth telling
 - “Vickrey’s truth serum” and the VCG mechanism.

A Social Choice Mechanism

Two agents, A and B , must choose between two outcomes, p and q . We want a mechanism that will allow them to choose, which will satisfy the following requirements. First, whatever happens, we definitely want an outcome to result — that is, we want either p or q to be selected. Second, we really do want the agents to be able to collectively choose an outcome. However, we do not want them to be able to bring about both outcomes simultaneously. Similarly, we do not want either agent to dominate: we want them both to have equal power.

Pauly: ATL for Mechanism Specification

Pauly realised you could express these requirements in ATL:

Pauly: ATL for Mechanism Specification

Pauly realised you could express these requirements in ATL:

$$\langle\langle A, B \rangle\rangle \bigcirc p$$

Pauly: ATL for Mechanism Specification

Pauly realised you could express these requirements in ATL:

$$\langle\langle A, B \rangle\rangle \bigcirc p$$

$$\langle\langle A, B \rangle\rangle \bigcirc q$$

Pauly: ATL for Mechanism Specification

Pauly realised you could express these requirements in ATL:

$$\langle\langle A, B \rangle\rangle \bigcirc p$$

$$\langle\langle A, B \rangle\rangle \bigcirc q$$

$$\langle\langle \rangle\rangle \bigcirc (p \vee q)$$

Pauly: ATL for Mechanism Specification

Pauly realised you could express these requirements in ATL:

$$\langle\langle A, B \rangle\rangle \bigcirc p$$

$$\langle\langle A, B \rangle\rangle \bigcirc q$$

$$\langle\langle \rangle\rangle \bigcirc (p \vee q)$$

$$\langle\langle \rangle\rangle \bigcirc \neg(p \wedge q)$$

Pauly: ATL for Mechanism Specification

Pauly realised you could express these requirements in ATL:

$$\langle\langle A, B \rangle\rangle \bigcirc p$$

$$\langle\langle A, B \rangle\rangle \bigcirc q$$

$$\langle\langle \rangle\rangle \bigcirc (p \vee q)$$

$$\langle\langle \rangle\rangle \bigcirc \neg(p \wedge q)$$

$$\neg \langle\langle A \rangle\rangle \bigcirc p$$

Pauly: ATL for Mechanism Specification

Pauly realised you could express these requirements in ATL:

$$\langle\langle A, B \rangle\rangle \bigcirc p$$

$$\langle\langle A, B \rangle\rangle \bigcirc q$$

$$\langle\langle \rangle\rangle \bigcirc (p \vee q)$$

$$\langle\langle \rangle\rangle \bigcirc \neg(p \wedge q)$$

$$\neg \langle\langle A \rangle\rangle \bigcirc p$$

$$\neg \langle\langle B \rangle\rangle \bigcirc p$$

Pauly: ATL for Mechanism Specification

Pauly realised you could express these requirements in ATL:

$$\langle\langle A, B \rangle\rangle \bigcirc p$$

$$\langle\langle A, B \rangle\rangle \bigcirc q$$

$$\langle\langle \rangle\rangle \bigcirc (p \vee q)$$

$$\langle\langle \rangle\rangle \bigcirc \neg(p \wedge q)$$

$$\neg \langle\langle A \rangle\rangle \bigcirc p$$

$$\neg \langle\langle B \rangle\rangle \bigcirc p$$

$$\neg \langle\langle A \rangle\rangle \bigcirc q$$

Pauly: ATL for Mechanism Specification

Pauly realised you could express these requirements in ATL:

$$\langle\langle A, B \rangle\rangle \bigcirc p$$

$$\langle\langle A, B \rangle\rangle \bigcirc q$$

$$\langle\langle \rangle\rangle \bigcirc (p \vee q)$$

$$\langle\langle \rangle\rangle \bigcirc \neg(p \wedge q)$$

$$\neg \langle\langle A \rangle\rangle \bigcirc p$$

$$\neg \langle\langle B \rangle\rangle \bigcirc p$$

$$\neg \langle\langle A \rangle\rangle \bigcirc q$$

$$\neg \langle\langle B \rangle\rangle \bigcirc q$$

Pauly: ATL for Mechanism Specification

Pauly realised you could express these requirements in ATL:

$$\langle\langle A, B \rangle\rangle \bigcirc p$$

$$\langle\langle A, B \rangle\rangle \bigcirc q$$

$$\langle\langle \rangle\rangle \bigcirc (p \vee q)$$

$$\langle\langle \rangle\rangle \bigcirc \neg(p \wedge q)$$

$$\neg \langle\langle A \rangle\rangle \bigcirc p$$

$$\neg \langle\langle B \rangle\rangle \bigcirc p$$

$$\neg \langle\langle A \rangle\rangle \bigcirc q$$

$$\neg \langle\langle B \rangle\rangle \bigcirc q$$

... but you can then **automatically verify** mechanisms.

(Pauly & Wooldridge, **GTD**, 2003)

Case Study 2: Eternal Voting

A political body $\{1, 2, 3, 4\}$ has to decide on passing a new law. There are two versions of the law, and the process begins by a single agent, 2, proposing which of these versions should be adopted. Once 2 has selected a version, the entire body votes on whether to accept the proposal; if there is a majority in favour of acceptance, then the proposed version is accepted; if there is a majority against, then there is deadlock, and the process begins again, with 2 selecting a version of the law to propose; if there is no majority one way or the other, then the vote of the chairman, 1, is decisive, in either accepting the proposed law or returning it to 2.

Properties

- Coalition $\{1, 2\}$ is all-powerful: they can force any outcome (incl deadlock) they want:
- In contrast, $\{3, 4\}$ is weak: they cannot force any outcome at all.

Other Mechanism Properties Specified Using ATL

Capture properties of **qualitative coalitional games** (QCGs).

$SUCC(C)$ means that C are **successful**:

$$SUCC(C) \equiv \langle\langle C \rangle\rangle \bigwedge_{i \in C} goal_i$$

(Wooldridge & Dunne, **AIJ**, 2004)

Other Mechanism Properties Specified Using ATL

Capture **dependencies** between agents.

$VETO(i, j)$ means “ j needs i to achieve his goal”

$$VETO(i, j) \equiv \bigwedge_C \langle\langle C \rangle\rangle \diamond goal_j \Rightarrow \neg \langle\langle C \setminus \{i\} \rangle\rangle \diamond goal_j$$

Other Properties Specified Using ATL

... and so capture **mutual dependencies** between agents.

$$MD(C) \equiv \bigwedge_{i,j \in C} VETO(i,j)$$

(Dunne, Hoek, Wooldridge, **Jnl. App. Non-class. Logic**, 2007)

Reasoning with ATL: Satisfiability

- The **satisfiability problem** for ATL is as follows:
Given ATL formula φ is there some interpretation that makes φ true?

Reasoning with ATL: Satisfiability

- The **satisfiability problem** for ATL is as follows:
Given ATL formula φ is there some interpretation that makes φ true?
- How hard is satisfiability in ATL?
 - For **Coalition Logic**: **PSPACE-complete** (Pauly, 2001).
 - For **CTL**: **EXPTIME-complete** – a lower bound for ATL.

Reasoning with ATL: Satisfiability

Theorem (van Drimmelen, 2003)

For *any fixed set of agents* Ag , satisfiability for ATL formulae over Ag is EXPTIME-complete.

Reasoning with ATL: Satisfiability

Theorem (van Drimmelen, 2003)

For *any fixed set of agents* Ag , satisfiability for ATL formulae over Ag is EXPTIME-complete.

But if the set of agents is not fixed, van Drimmelen's algorithm is 2EXPTIME.

Reasoning with ATL: Satisfiability

Theorem (van Drimmelen, 2003)

For *any fixed set of agents* Ag , satisfiability for ATL formulae over Ag is EXPTIME-complete.

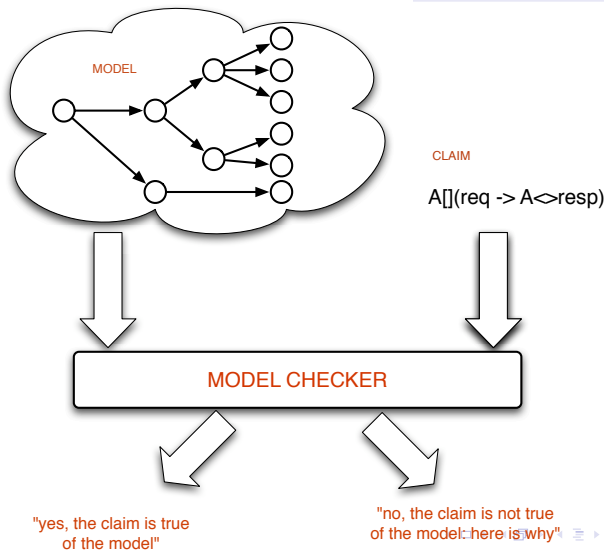
But if the set of agents is not fixed, van Drimmelen's algorithm is 2EXPTIME.

Theorem (Walther, Lutz, Wolter, Wooldridge, 2006)

The satisfiability problem for *arbitrary* ATL formulae is EXPTIME-complete (and hence no harder than CTL).

Reasoning with ATL: Model-checking

Notation: $S \models \varphi$



Reasoning with ATL: Model-checking

- But **how do we represent the model**? This affects the complexity.
- If the model defined as **state transition graph**
... the problem can be solved in time polynomial in the size of the state transition graph.
Hurrah... ?

Reasoning with ATL: Model-checking

- But **how do we represent the model**? This affects the complexity.
- **If the model defined as state transition graph**
... the problem can be solved in time polynomial in the size of the state transition graph.
Hurrah... ?
This is **not** a feasible representation:
State transition graph is **exponential** in the number of system variables.

Reasoning with ATL: Model Checking

- **Practical** model checkers use **high-level** model specification languages.
- **Reactive modules**: a rule-based language for model specification. . .

```
module toggle controls x
  init
  []  $\top \rightsquigarrow x' := \top$ 
  []  $\top \rightsquigarrow x' := \perp$ 
  update
  []  $x \rightsquigarrow x' := \perp$ 
  []  $(\neg x) \rightsquigarrow x' := \top$ 
```

Reasoning with ATL: Model Checking

The complexity of model checking depends on the **language for claims** and the **representation of model**.

Theorem (van der Hoek, Lomuscio, Wooldridge, AAMAS06)

For Reactive Modules models, model checking is exactly as hard as theorem proving in the corresponding language:

<i>ATL</i>	<i>EXPTIME-complete</i>
<i>Coalition Logic</i>	<i>PSPACE-complete</i>
<i>prop logic</i>	<i>co-NP-complete</i>

Another Language for Models

(GDL = The Game Description Language)

- In 2005, AAI introduced **General Game Playing Competition**, intended to test ability of programs to play games in general, rather than just one game.
- **GDL** is a language developed for **defining** the games that participants play.

Another Language for Models

(GDL = The Game Description Language)

- In 2005, AAI introduced **General Game Playing Competition**, intended to test ability of programs to play games in general, rather than just one game.
- **GDL** is a language developed for **defining** the games that participants play.
A rule-based language – stratified logic programs.
- GDL is a **a language for defining mechanisms**.

Example GDL Code

```
(role mw)
(init (clear b))
(init (clear c))
(init (step 1))
(<= (next (on ?x ?y))
  (does mw (stk ?x ?y)))
...
(succ 1 2)
(succ 2 3)
(succ 3 4)
```

```
(<= (legal mw (stk ?x ?y))
  (true (clear ?x))
  (true (table ?x))
  (true (clear ?y))
  (distinct ?x ?y))
...
(<= (goal mw 100)
  (true (on a b))
  (true (on b c)))
...
(<= terminal
  (true (step 4)))
...
```

ATL Model Checking over GDL Models

Theorem (Hoek, Ruan, Wooldridge, LORI, 2007)

Model checking ATL over GDL specifications is as hard as for Reactive Modules:

<i>ATL</i>	<i>EXPTIME-hard</i>
<i>Coalition Logic</i>	<i>PSPACE-hard</i>
<i>prop logic</i>	<i>co-NP-hard</i>

ATL Model Checking over GDL Models

Theorem (Hoek, Ruan, Wooldridge, LORI, 2007)

Model checking ATL over GDL specifications is as hard as for Reactive Modules:

<i>ATL</i>		<i>EXPTIME-hard</i>
<i>Coalition Logic</i>		<i>PSPACE-hard</i>
<i>prop logic</i>		<i>co-NP-hard</i>

Ruan is implementing a model checker for ATL over GDL models.

Main application: verifying game specifications.

Social Laws

(Also known as Normative Systems)

- Mechanism design for **legacy systems**.
- Social laws are **coordination mechanisms** for **pre-existing** systems.

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

- **Prohibit** certain actions in certain states.
- This is **not** social sciences. . .
... **nor** is it law. . .
... it's computer science!

Moses, Shoham, and Tennenholtz

- Offline design of social laws first investigated by Moses, Shoham and Tennenholtz (1991–97)
- Each agent 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.
Note that this is a **power** of the system agents.

Moses, Shoham, and Tennenholtz

- Offline design of social laws first investigated by Moses, Shoham and Tennenholtz (1991–97)
- Each agent 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.
Note that this is a **power** of the system agents.
- Useful social law problem: NP-complete.

Social Laws in Alternating Time

- ATL provides a natural language for expressing social laws, as we can capture:
 - **liveness** and **safety** properties
 - **powers** that agents can/should have

Social Laws Framework

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

- 1 An **objective**, φ
What we **want to achieve** with the social law.
An **ATL formula**.
- 2 A **behavioural constraint**, β
The **mechanism** by which we will achieve it.
For each action, lists states that this action is forbidden in.

Behavioural Constraints

- **Implementing** β in S :
eliminate all transitions forbidden by β
- Behavioural constraints are required to be “reasonable”:
every agent must be able to do **something**.
- Implementation of β in S is denoted

$$S \dagger \beta$$

Effective Social Laws

- A social law (φ, β) is **effective** in S if after implementing it, the objective φ is guaranteed to hold:

$$S \uparrow \beta \models \varphi$$

Effective Social Laws

- A social law (φ, β) is **effective** in S if after implementing it, the objective φ is guaranteed to hold:

$$S \uparrow \beta \models \varphi$$

- The **Effectiveness Problem** is then:
Given S and (φ, β) over S , determine whether (φ, β) is effective in S .

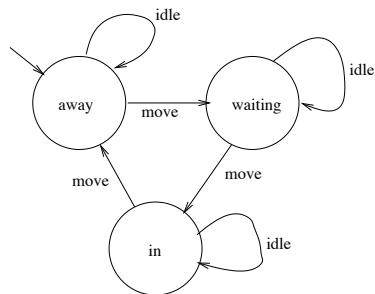
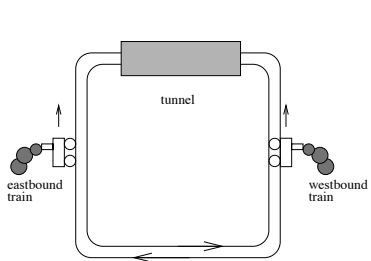
Checking Effectiveness

Theorem (Wooldridge et al, IJCAI 2007)

Complexity of checking effectiveness:

	<i>Explicit representation</i>	<i>Reactive Modules</i>
<i>prop logic</i>	<i>poly time</i>	<i>PSPACE-complete</i>
<i>CTL</i>	<i>poly time</i>	<i>PSPACE-complete</i>
<i>ATL</i>	<i>poly time</i>	<i>PSPACE-hard</i>

An Example System



Example Social Laws

- Obvious requirement: the trains don't crash:

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

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 :

$\beta_2 =$ prevent both trains from ever moving

More Examples

But consider the social law β_2 :

$\beta_2 =$ prevent both trains from ever moving

(O_1, β_2) is also effective!

cf. the **Frame Problem**.

Need to specify properties to be **preserved**.

More Examples

Refine our original objective:

$$O_2 = O_1 \wedge \bigwedge_{i \in \{E, W\}} \left(\begin{array}{l} (away_i \Rightarrow \langle\langle i \rangle\rangle \diamond waiting_i) \quad \wedge \\ (waiting_i \Rightarrow \langle\langle i \rangle\rangle \diamond (in_i \wedge O_1)) \quad \wedge \\ (in_i \Rightarrow \langle\langle i \rangle\rangle \bigcirc away_i) \end{array} \right)$$

More Examples

Refine our original objective:

$$O_2 = O_1 \wedge \bigwedge_{i \in \{E, W\}} \left(\begin{array}{l} (away_i \Rightarrow \langle\langle i \rangle\rangle \diamond waiting_i) \quad \wedge \\ (waiting_i \Rightarrow \langle\langle i \rangle\rangle \diamond (in_i \wedge O_1)) \quad \wedge \\ (in_i \Rightarrow \langle\langle i \rangle\rangle \bigcirc away_i) \end{array} \right)$$

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 ?

The Feasibility Problem

Complexity again depends on the representation of model and the language of the objective.

Theorem (Wooldridge et al, IJCAI 2007)

Complexity of feasibility:

	<i>Explicit state</i>	<i>Reactive Modules</i>
<i>prop logic</i>	<i>poly time</i>	<i>PSPACE-complete</i>
<i>CTL</i>	<i>NP-complete</i>	<i>PSPACE-complete</i>
<i>ATL</i>	<i>NP-complete</i>	<i>EXPTIME-hard</i>

Synthesis and Model Checking

- There is a link between **model checking** and **synthesis**.

Synthesis and Model Checking

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

Synthesis and Model Checking

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

Theorem

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.

Incentive Compatibility

Incentive Compatibility

- The question we got tired of hearing:
How do you deal with non-compliance?

Incentive Compatibility

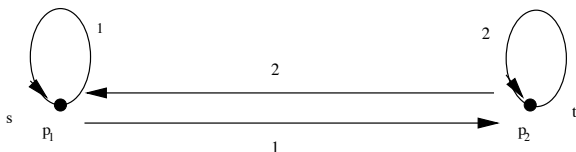
- The question we got tired of hearing:
How do you deal with non-compliance?
- One answer: **incentive compatibility**.
- For this, we need **preferences**:
 - a **prioritised list** of goal formulae.

An 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.



Goals for the Example

- Each agent i wants the resource as **often** and **long** as possible for himself:

$$\gamma_i = (\quad \top, \quad E \diamond p_i, \\ E \square E \diamond p_i, \quad E \diamond E \square p_i, \\ A \square E \diamond p_i, \quad E \diamond A \square p_i \\ A \square A \diamond p_i, \quad A \square (A \diamond p_i \wedge E \square p_i), \\ A \square p_i)$$

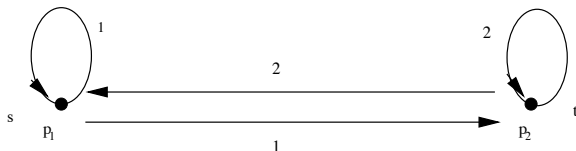
- This gives us **utility**.

Compliance Games

Every agent has to decide whether to comply (C) or deviate (D) from the social law.

This gives us **compliance games**.

The Example Revisited



Consider the social law which prevents agents holding on to the resource.

The payoff matrix for the corresponding compliance game is:

	<i>C</i>	<i>D</i>
<i>C</i>	(2, 2)	(0, 3)
<i>D</i>	(3, 0)	(0, 0)

Pareto Efficiency

- A benign leader asks: **is it possible to make some agents better off without making anybody else worse off?**
- A system is **Pareto efficient** if no such **Pareto improvements** are possible.

Theorem

Checking Pareto efficiency is co-NP-complete, even for one-agent systems.

Nash Implementations

- A social law is a **Nash implementation** if **everyone complying** is a Nash equilibrium the corresponding **compliance game**.

Nash Implementations

- A social law is a **Nash implementation** if **everyone complying** is a Nash equilibrium the corresponding **compliance game**.
- The **Nash implementation** problem:
Given: System S , goal lists $\gamma_1, \dots, \gamma_n$, and objective φ ?
Question: Does there exist a Nash implementation social law which is effective for φ ?

Theorem

NI is NP-complete, even for two-agent systems.

Conclusions

- Cooperation logics are a powerful and fascinating tool for specification and analysis of mechanisms.
- We have explored many issues surrounding their use: reasoning, social laws,
- Our work that I didn't mention: knowledge, obligations, cooperative games, explicit strategies, . . .
- Our medium term goal (2 years): a practical (model checking) tool for programming and analysing mechanisms.

Thanks to...

Thomas Ågotnes

Paul E. Dunne

Michael Fisher

Wiebe van der Hoek

Wojtek Jamroga

Sarit Kraus

Carsten Lutz

Sieuwert van Otterloo

Marc Pauly

Ji Ruan

Dirk Walther

Rafael Bordini

Shaheen Fatima

Jelle Gerbrandy

Jomi Hubner

Nick Jennings

Alessio Lomuscio

Peter McBurney

Simon Parsons

Mark Roberts

Luigi Sauro

Frank Wolter

Find Out More

D. Walther, C. Lutz, F. Wolter, and M. Wooldridge. [ATL Satisfiability is Indeed EXPTIME-complete](#). *Journal of Logic & Computation*, 16:765-787, 2006.

W. van der Hoek, M. Roberts, M. Wooldridge [Social Laws in Alternating Time](#), *Synthese*, 156(1):1–19, May 2007.

M. Wooldridge, P. E. Dunne. [On the Computational Complexity of Coalitional Resource Games](#), In *Artificial Intelligence*, July 2006

M. Wooldridge and W. van der Hoek. [On Obligations and Normative Ability](#). In *Journal of Applied Logic*, Vol 3, 2005

W. van der Hoek and M. Wooldridge. [On the Logic of Cooperation and Propositional Control](#). In *Artificial Intelligence*, May 2005.

M. Wooldridge, P. E. Dunne. [On the Computational Complexity of Qualitative Coalitional Games](#), In *Artificial Intelligence*, Sept 2004

M. Pauly and M. Wooldridge. [Logic for Mechanism Design — A Manifesto](#). In *Proceedings GTDT 2003*, Melbourne, Aus., 2003.

... and papers at AAMAS02-07, IJCAI07.