Towards a Theory of Correctness of Social Procedures

Eric Pacuit

April 20, 2006

ILLC, University of Amsterdam
staff.science.uva.nl/∼epacuit
epacuit@science.uva.nl

Introduction: Social Software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Introduction: Social Software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Research in Social Software can be divided into three different but related categories:

Introduction: Social Software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Research in Social Software can be divided into three different but related categories:

- Mathematical Models of Social Situations

Introduction: Social Software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Research in Social Software can be divided into three different but related categories:

- Mathematical Models of Social Situations
- A Theory of Correctness of Social Procedures

Introduction: Social Software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Research in Social Software can be divided into three different but related categories:

- Mathematical Models of Social Situations
- A Theory of Correctness of Social Procedures
- Designing Social Procedures

Introduction: Social Software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Research in Social Software can be divided into three different but related categories:

- Mathematical Models of Social Situations
- A Theory of Correctness of Social Procedures
- Designing Social Procedures

Introduction: Social Software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

For more information see

R. Parikh. *Social Software*. *Synthese* **132** (2002).

R. Parikh. *Language as Social Software*. in *Future Pasts: The Analytic Tradition in the Twentieth Century* (2001).

EP and R. Parikh. *Social Interaction, Knowledge, and Social Software*. in *Interactive Computation: The New Paradigm* (forthcoming).

Logic for Mechanism Design

Computational aspects of computer science vs. using ideas from computer science (eg. program verification) in game theory.

Logic for Mechanism Design

Computational aspects of computer science vs. using ideas from computer science (eg. program verification) in game theory.

Formally verifying mechanisms:

J. Halpern. *A Computer Scientist Looks at Game Theory. Games and Economic Behavior* **45** (2003).

J. van Benthem. *Extensive Games as Process Models. JOLLI* **11** (2002).

M. Pauly and M. Wooldridge. *Logics for Mechanism Design — A Manifesto.* available at the author's websites.

S. van Otterloo. *Strategic Analysis of Multi-agent Protocols.* Ph.D. Thesis, University of Liverpool (2005).

Outline of the Talk

- Strategy Logics
  - Coalitional Logic
    - A Simple Example
  - Alternating Time Temporal Logic
- From Hoare Logic to **PDL**
- Game Logic
  - Banach-Knaster Cake Cutting Algorithm
- Pauly's Mechanism Programming Language
  - Example
- Case Study: Adjusted Winner

From Temporal Logic to Strategy Logic

From Temporal Logic to Strategy Logic

- *Linear Time Temporal Logic*: Reasoning about computation paths:

$$\Diamond\phi\text{: } \phi \text{ is true some time in } \textit{the} \text{ future.}$$

A. Pnuelli. *A Temporal Logic of Programs.* in *Proc. 18th IEEE Symposium on Foundations of Computer Science* (1977).

From Temporal Logic to Strategy Logic

- *Linear Time Temporal Logic*: Reasoning about computation paths:

  $\Diamond\phi$: ϕ is true some time in *the* future.

A. Pnuelli. *A Temporal Logic of Programs.* in *Proc. 18th IEEE Symposium on Foundations of Computer Science* (1977).

- *Branching Time Temporal Logic*: Allows quantification over paths:

  $\exists\Diamond\phi$: there is a path in which ϕ is eventually true.

E. M. Clarke and E. A. Emerson. *Design and Synthesis of Synchronization Skeletons using Branching-time Temproal-logic Specifications.* In *Proceedings Workshop on Logic of Programs*, LNCS (1981).

From Temporal Logic to Strategy Logic

- *Alternating-time Temporal Logic*: Reasoning about (local and global) group power:

  $\langle\langle A \rangle\rangle \Box \phi$: The coalition A has a joint strategy to ensure that ϕ will remain true.

R. Alur, T. Henzinger and O. Kupferman. *Alternating-time Temproal Logic. Jouranl of the ACM* (2002).

From Temporal Logic to Strategy Logic

- *Alternating-time Temporal Logic*: Reasoning about (local and global) group power:

  $\langle\langle A \rangle\rangle \Box \phi$: The coalition A has a joint strategy to ensure that ϕ will remain true.

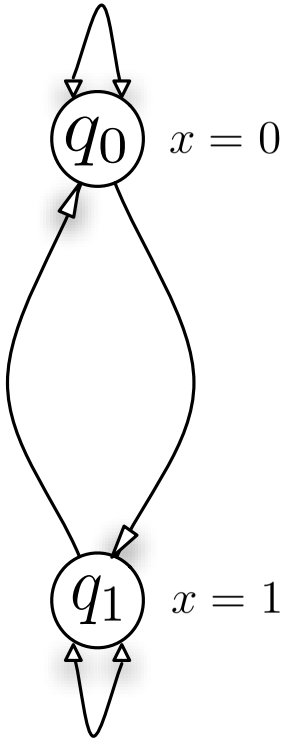
R. Alur, T. Henzinger and O. Kupferman. *Alternating-time Temproal Logic. Jouranl of the ACM* (2002).

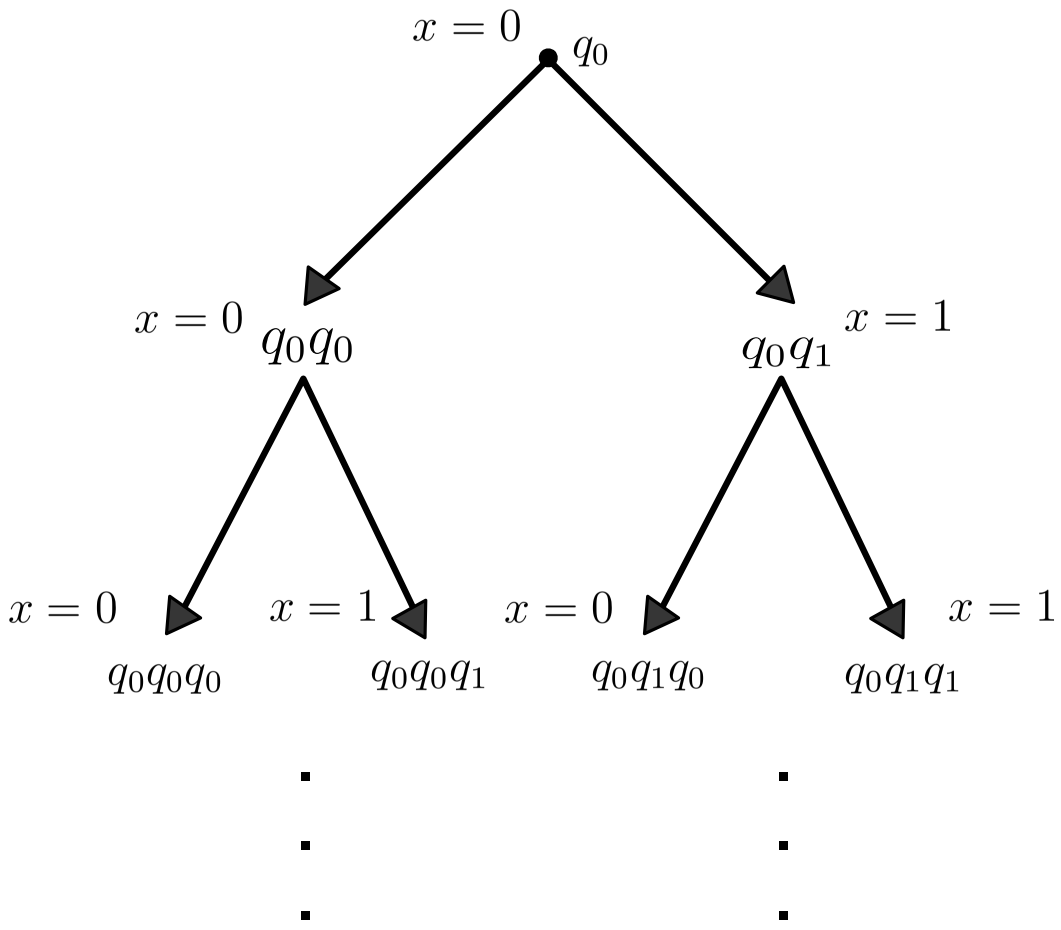
- *Coalitional Logic*: Reasoning about (local) group power (fragment of **ATL**).

  $[C]\phi$ (equivalently $\langle\langle C \rangle\rangle \bigcirc \phi$): coalition C has a joint strategy to bring about ϕ.

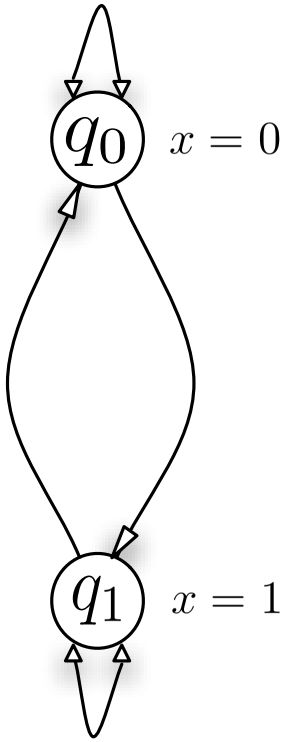
M. Pauly. *A Modal Logic for Coalition Powers in Games. Journal of Logic and Computation* **12** (2002).

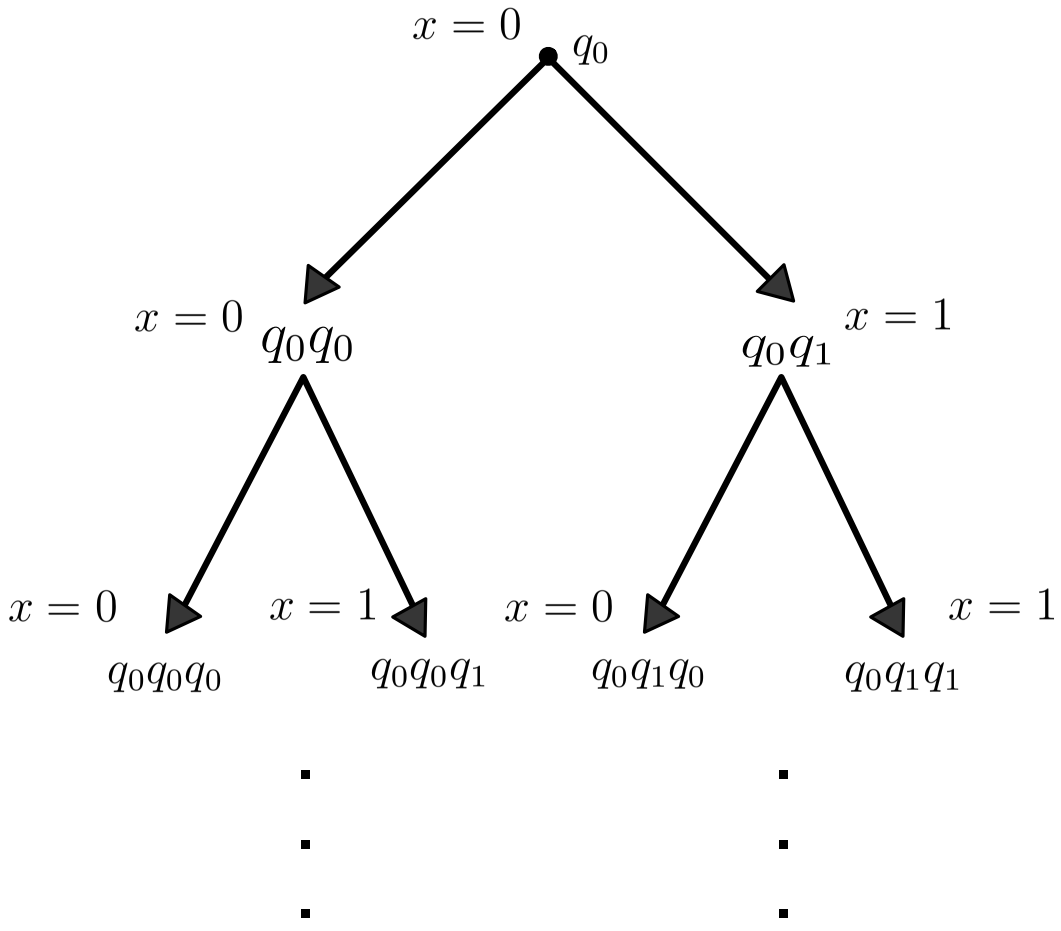
Computational vs. Behavioral Structures





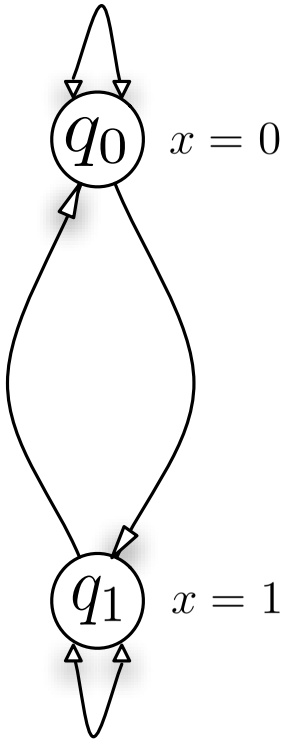
Computational vs. Behavioral Structures

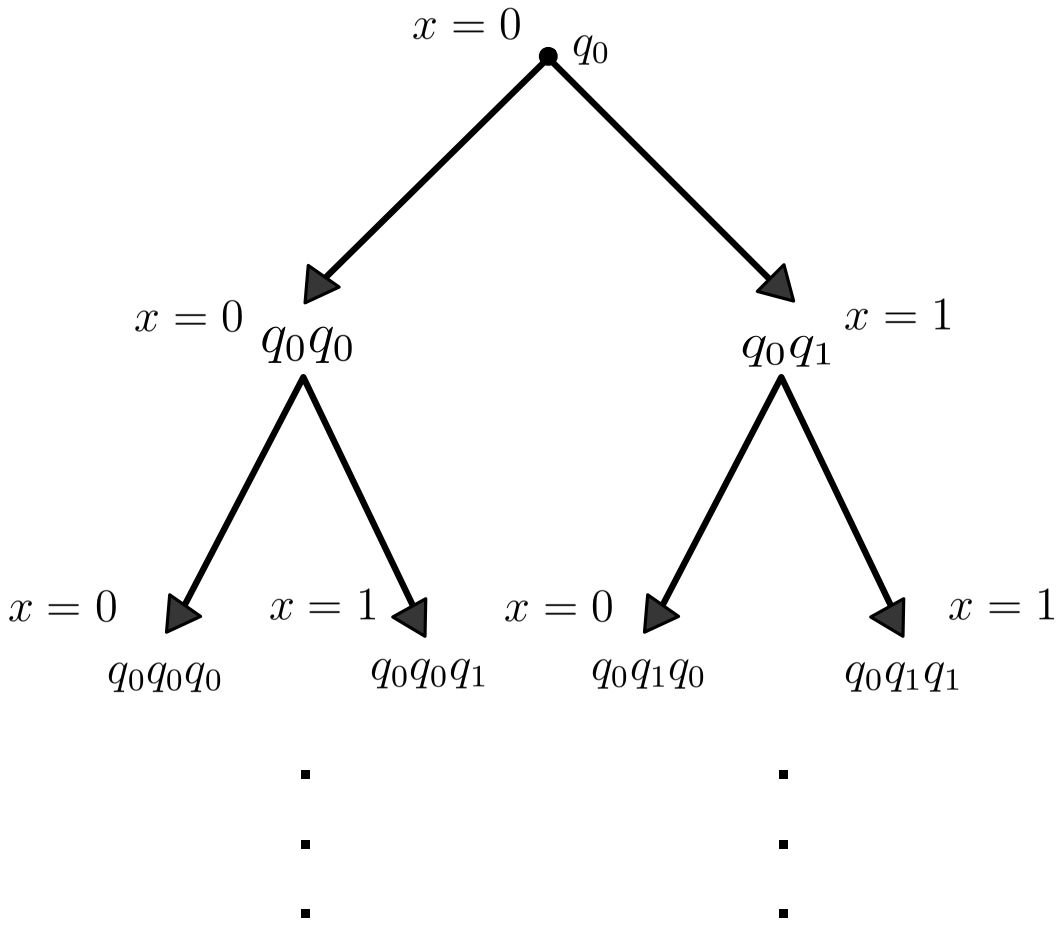




$$\exists \Diamond P_{x=1}$$

Computational vs. Behavioral Structures





$$\neg\forall\Diamond P_{x=1}$$

Alternating Transition Systems

The previous model assumes there is *one* agent that "controls" the transition system.

Alternating Transition Systems

The previous model assumes there is *one* agent that "controls" the transition system.

What if there is more than one agent?

Alternating Transition Systems

The previous model assumes there is *one* agent that "controls" the transition system.

What if there is more than one agent?

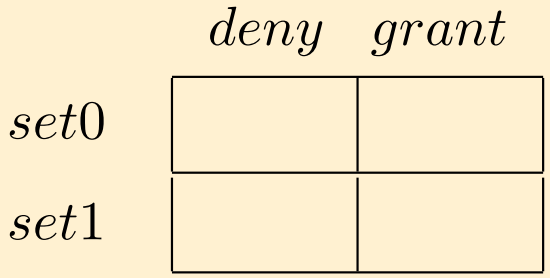
Example: Suppose that there are two agents: a server (s) and a client (c). The client asks to set the value of x and the server can either grant or deny the request. Assume the agents make simultaneous moves.

Alternating Transition Systems

The previous model assumes there is *one* agent that "controls" the transition system.

What if there is more than one agent?

Example: Suppose that there are two agents: a server (s) and a client (c). The client asks to set the value of x and the server can either grant or deny the request. Assume the agents make simultaneous moves.



Alternating Transition Systems

The previous model assumes there is *one* agent that "controls" the transition system.

What if there is more than one agent?

Example: Suppose that there are two agents: a server (s) and a client (c). The client asks to set the value of x and the server can either grant or deny the request. Assume the agents make simultaneous moves.

	deny	*grant*
*set*0		$q_0 \Rightarrow q_0,\ q_1 \Rightarrow q_0$
*set*1		$q_0 \Rightarrow q_1,\ q_1 \Rightarrow q_1$

Alternating Transition Systems

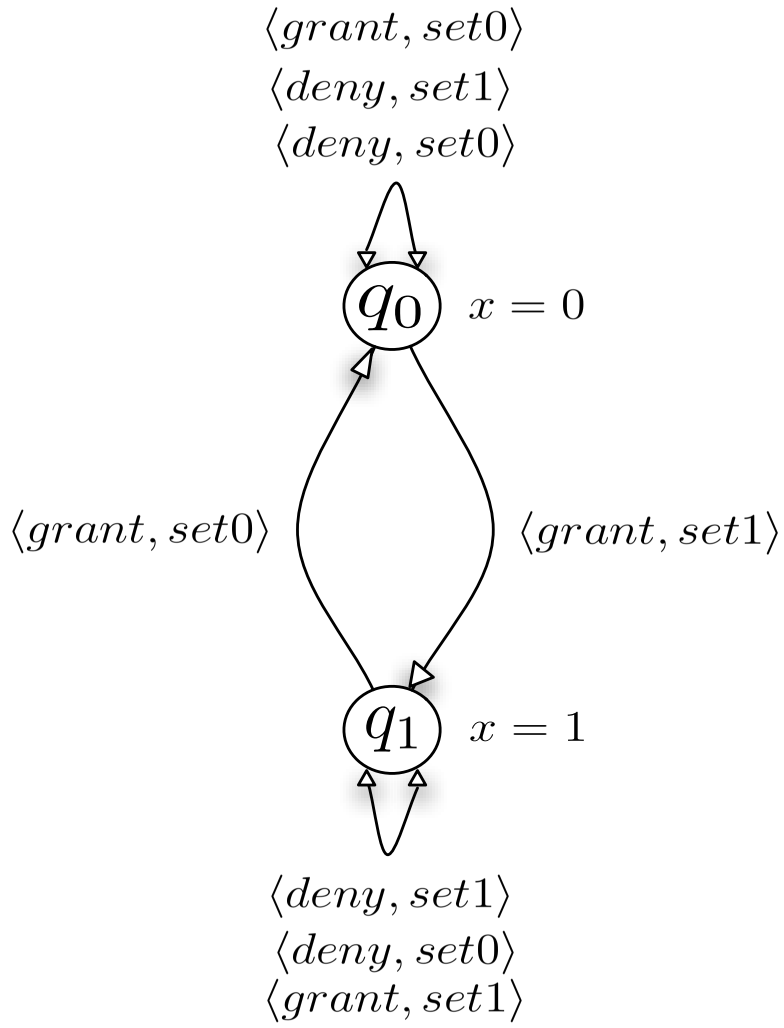
The previous model assumes there is *one* agent that "controls" the transition system.

What if there is more than one agent?

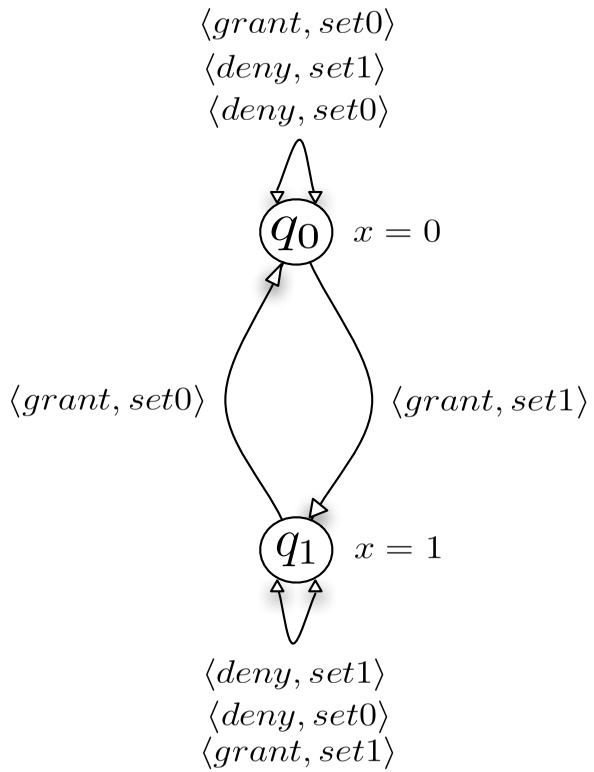
Example: Suppose that there are two agents: a server (s) and a client (c). The client asks to set the value of x and the server can either grant or deny the request. Assume the agents make simultaneous moves.

	deny	*grant*
$set0$	$q \Rightarrow q$	$q_0 \Rightarrow q_0$, $q_1 \Rightarrow q_0$
$set1$	$q \Rightarrow q$	$q_0 \Rightarrow q_1$, $q_1 \Rightarrow q_1$

Multi-agent Transition Systems

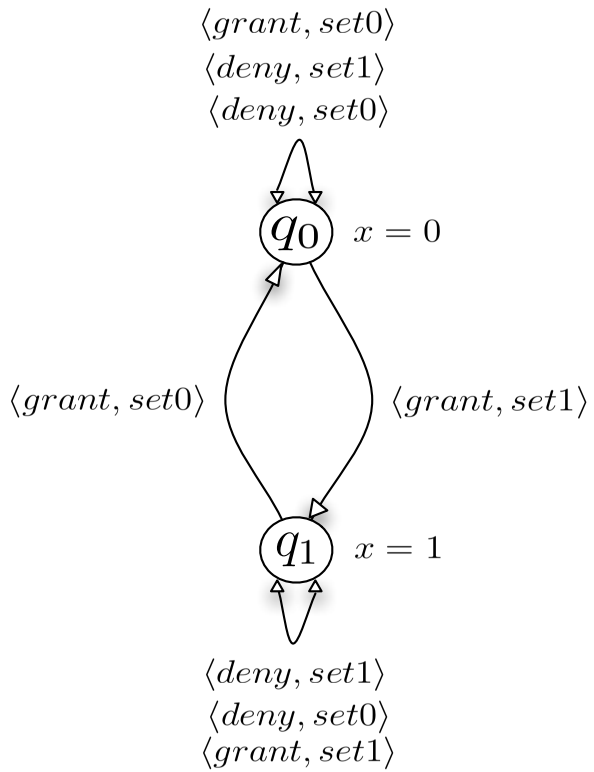


Multi-agent Transition Systems



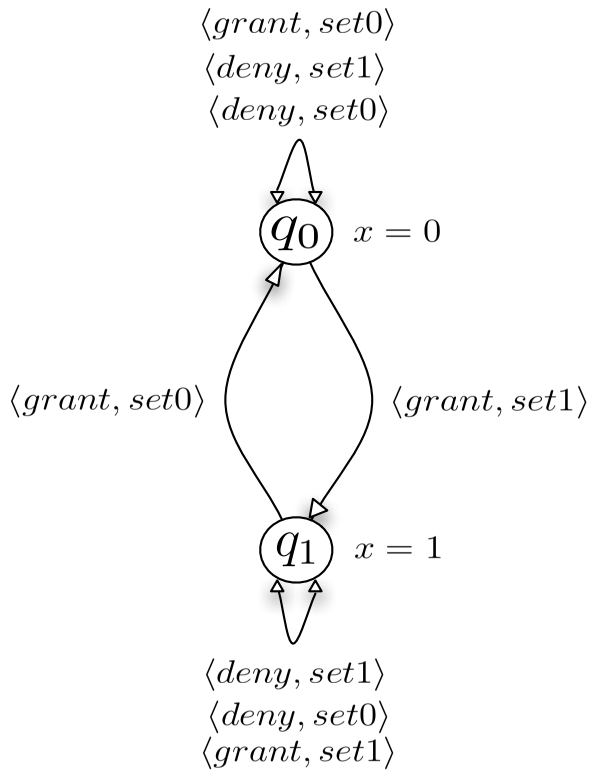
$$(P_{x=0} \rightarrow [s]P_{x=0}) \wedge (P_{x=1} \rightarrow [s]P_{x=1})$$

Multi-agent Transition Systems



$$P_{x=0} \rightarrow \neg[s]P_{x=1}$$

Multi-agent Transition Systems



$$P_{x=0} \rightarrow [s, c]P_{x=1}$$

An Example

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:

1. We definitely want an outcome to result, i.e., either p or q must be selected
2. We want the agents to be able to collectively choose and outcome
3. We do not want them to be able to bring about both outcomes simultaneously
4. We want them both to have equal power

An Example

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:

1. We definitely want an outcome to result, i.e., either p or q must be selected: $[\emptyset](p \vee q)$
2. We want the agents to be able to collectively choose and outcome
3. We do not want them to be able to bring about both outcomes simultaneously
4. We want them both to have equal power

An Example

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:

1. We definitely want an outcome to result, i.e., either p or q must be selected: $[\emptyset](p \vee q)$
2. We want the agents to be able to collectively choose and outcome: $[A, B]p \wedge [A, B]q$
3. We do not want them to be able to bring about both outcomes simultaneously
4. We want them both to have equal power

An Example

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:

1. We definitely want an outcome to result, i.e., either p or q must be selected: $[\emptyset](p \vee q)$
2. We want the agents to be able to collectively choose and outcome: $[A, B]p \wedge [A, B]q$
3. We do not want them to be able to bring about both outcomes simultaneously: $\neg[A, B](p \wedge q)$
4. We want them both to have equal power

An Example

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:

1. We definitely want an outcome to result, i.e., either p or q must be selected: $[\emptyset](p \vee q)$
2. We want the agents to be able to collectively choose and outcome: $[A, B]p \wedge [A, B]q$
3. We do not want them to be able to bring about both outcomes simultaneously: $\neg[A, B](p \wedge q)$
4. We want them both to have equal power: $\neg[x]p \wedge \neg[x]q$ where $x \in \{A, B\}$

An Example

Consider the following mechanism:

The two agents vote on the outcomes, i.e., they choose either p or q. If there is a consensus, then the consensus is selected; if there is no consensus, then an outcome p or q is selected non-deterministically.

Pauly and Wooldridge use the MOCHA model checking system to verify that the above procedure satisfies the previous specifications.

Pauly's Coalitional Logic: Syntax

Given a finite non-empty set of agents N and a set of atomic propositions Φ_0, a formula ϕ can have the following syntactic form

$$\phi \;:=\; \bot \mid p \mid \neg\phi \mid \phi \vee \phi \mid [C]\phi$$

where $p \in \Phi_0$ and $C \subseteq N$.

$[C]\phi$ is intended to mean "coalition C can (locally) force ϕ to be true"

M. Pauly. *Logics for Social Software*. Ph.D. Thesis, ILLC (2001).

Multi-player Game Models

A **Strategic Game Form** is a tuple $\langle N, \{\Sigma_i \mid i \in N\}, Q, o\langle$ where

- N is a set of agents
- Σ_i is a set of actions
- Q is a set of states
- $o : \Pi_{i \in N} \Sigma_i \rightarrow Q$ assigns an outcome to each choice of aciton.

Let Γ_Q^N be the set of all such strategic game forms.

A **Multi-Player Game Model** is a tuple $\langle Q, \gamma, \pi \rangle$ where Q is a set of states and $\gamma : Q \rightarrow \Gamma_Q^N$ associates strategic games form to each state

$$q \models [C]\phi \text{iff } \exists\ \sigma_C\ \forall\ \sigma_{N-C},\ o(\sigma_C, \sigma_{N-C}) \models \phi$$

Effectivity Functions

Let G be a strategic game.

$$X \in E^{\alpha}_{G}(C) \text{iff } \exists \sigma_C \forall \sigma_{\overline{C}} \;\; o(\sigma_C, \sigma_{\overline{C}}) \in X$$

$$X \in E^{\beta}_{G}(C) \text{iff } \forall \sigma_{\overline{C}} \exists \sigma_C \;\; o(\sigma_C, \sigma_{\overline{C}}) \in X$$

$E^{\alpha}_{G} \subseteq E^{\beta}_{G}$

$$E_G^\beta \not\subseteq E_G^\alpha$$

Player 1 chooses the row, Player 2 chooses the column, Player 3 chooses the table

	l	m	r
l	s_1	s_2	s_1
r	s_2	s_1	s_3

	l	m	r
l	s_3	s_1	s_2
r	s_2	s_3	s_3

$$\{s_2\} \in E_G^\beta(\{2\}) \text{ but } \{s_2\} \notin E_G^\alpha(\{2\})$$

Coalition Effectivity Models

An effectivity function is **playable** iff

1. For each $C \subseteq N$, $\varnothing \notin E(C)$
2. For each $C \subseteq N$, $Q \in E(C)$
3. If $X \notin E(N)$, then $Q - X \in E(\varnothing)$
4. If $X \subseteq Y$ and $X \in E(C)$ then $Y \in E(C)$
5. for all $C_1, C_2 \subseteq N$ and $X_1, X_2 \subseteq Q$, if $C_1 \cap C_2 = \varnothing$, $X_1 \in E(C_1)$ and $X_2 \in E(C_2)$ then $X_1 \cap X_2 \in E(C_1 \cup C_2)$

Characterization Theorem: An α-effectivity function E is playable iff it is the effectivity function of some strategic game.

M. Pauly. *A Modal Logic for Coalition Powers in Games. Journal of Logic and Computation* **12** (2002).

Coalitional Logic: Coalition Effectivity Models

A **coalitional effectivity model** is a tuple $\langle Q, E, V \rangle$ where $E : Q \to (2^N \to 2^{2^Q})$ assigns a playable effectivity function to each state and V is a valuation function.

$$q \models [C]\phi \text{ iff } (\phi)^{\mathcal{M}} \in E_q(C)$$

Main Results

Theorem Coalitional Logic is sound and strongly complete with respect to the class of effectivity models.

Theorem The complexity of the satisfiability problem of coalitional logic is PSPACE-complete.

M. Pauly. *A Modal Logic for Coalitional Powers in Games. Journal of Logic and Computation* (2002).

M. Pauly. *On the Complexity of Coalitional Reasoning . International Game Theory Review* (2002).

ATL: Syntax

Let $\mathcal{A}$ be a set of agents, Π a set of propositional variables and $A \subseteq \mathcal{A}$.

1. p where $p \in \Pi$
2. $\neg\phi$
3. $\phi \vee \psi$
4. $\langle\langle A \rangle\rangle \bigcirc \phi$ meaning 'The coalition A can force in the next move an outcome satisfying ϕ'
5. $\langle\langle A \rangle\rangle \Box \phi$ meaning 'The coalition A can maintain forever outcomes satisfying ϕ'
6. $\langle\langle A \rangle\rangle \phi U \psi$ meaning 'The coalition A can eventually force an outcome satisfying ψ while meanwhile maintaining the truth of ϕ

Coalition Logic is a Fragment of ATL

Define $[A]\phi$ to be $\langle\langle A\rangle\rangle \bigcirc \phi$

Multi-player Game Models and Concurrent-game Models only differ in notation

Coalitional Effectivity Models can be used as a semantics for **ATL**

Goranko and Jamroga. *Comparing Semantics of Logics fro Multi-Agent Systems*. See the website.

Results

Theorem All of the semantics (concurrent game structures, alternating transitions systems and coalitional effectivity models) are equivalent.

Goranko and Jamroga. *Comparing Semantics of Logics for Multi-Agent Systems.* See the website.

Theorem **ATL** is sound and (weakly) complete.

Theorem Given a finite set of players, the satisfiability problem for **ATL**-formulas over N with respect to concurrent game structures over N is EXPTIME-complete.

Goranko and van Drimmelen. *Complete Axiomatization and Decidability of the Alternating-Time Temporal Logic.* Theoretical Computer Science (2005).

From Hoare Logic to Game Logic

- *Hoare Logic* Partial Correctness of Procedures

  $\{\phi\}\alpha\{\psi\}$: If the program α begins in a state in which ϕ is true, then after α terminates (!), ψ will be true.

C. A. R. Hoare. *An Axiomatic Basis for Computer Programming.*. Comm. Assoc. Comput. Mach. 1969.

- *Propositional Dynamic Logic* (**PDL**) [Pratt, 1976]: Reason about programs explicitly:

  $[\alpha]\phi$: after executing α, ϕ is true.

C. A. R. Hoare. *An Axiomatic Basis for Computer Programming.*. Comm. Assoc. Comput. Mach. 1969.

From Hoare Logic to Game Logic

- *Game Logic* (**GL**) [Parikh, 1985]: Reasoning about games:

  $(\gamma)\phi$: Agent I has a strategy to bring about ϕ in game γ.

R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

- More information:

K.R. Apt and E.R. Olderog. Verification of Sequential and Concurrent Programs, *Second Edition* . Springer-Verlag (1997).

D. Harel, D. Kozen and J. Tiuryn. Dynamic Logic. MIT Press (2000).

Background: Hoare Logic

Background: Hoare Logic

Motivation: Formally verify the "correctness" of a program via *partial correctness assertions*:

$$\{\phi\}\alpha\{\psi\}$$

Background: Hoare Logic

Motivation: Formally verify the "correctness" of a program via *partial correctness assertions*:

$$\{\phi\}\alpha\{\psi\}$$

Intended Interpretation: If the program α begins in a state in which ϕ is true, then after α terminates (!), ψ will be true.

Background: Hoare Logic

Motivation: Formally verify the "correctness" of a program via *partial correctness assertions*:

$$\{\phi\}\alpha\{\psi\}$$

Intended Interpretation: If the program α begins in a state in which ϕ is true, then after α terminates (!), ψ will be true.

C. A. R. Hoare. *An Axiomatic Basis for Computer Programming.*. Comm. Assoc. Comput. Mach. 1969.

Background: Hoare Logic

Main Rules:

Background: Hoare Logic

Main Rules:

Assignment Rule: $\{\phi[x/e]\}\ x := e\ \{\phi\}$

Background: Hoare Logic

Main Rules:

Assignment Rule: $\{\phi[x/e]\}\ x := e\ \{\phi\}$

Composition Rule: $$\frac{\{\phi\}\ \alpha\ \{\sigma\} \qquad \{\sigma\}\ \beta\ \{\psi\}}{\{\phi\}\ \alpha;\beta\ \{\psi\}}$$

Background: Hoare Logic

Main Rules:

Assignment Rule: $\{\phi[x/e]\}\ x := e\ \{\phi\}$

Composition Rule: $$\frac{\{\phi\}\ \alpha\ \{\sigma\} \qquad \{\sigma\}\ \beta\ \{\psi\}}{\{\phi\}\ \alpha;\beta\ \{\psi\}}$$

Conditional Rule: $$\frac{\{\phi \wedge \sigma\}\ \alpha\ \{\psi\} \qquad \{\phi \wedge \neg\sigma\}\ \beta\ \{\psi\}}{\{\phi\}\ \textbf{if}\ \sigma\ \textbf{then}\ \alpha\ \textbf{else}\ \beta\ \{\psi\}}$$

Background: Hoare Logic

Main Rules:

Assignment Rule: $\{\phi[x/e]\}\ x := e\ \{\phi\}$

Composition Rule: $$\frac{\{\phi\}\ \alpha\ \{\sigma\} \qquad \{\sigma\}\ \beta\ \{\psi\}}{\{\phi\}\ \alpha;\beta\ \{\psi\}}$$

Conditional Rule: $$\frac{\{\phi \wedge \sigma\}\ \alpha\ \{\psi\} \qquad \{\phi \wedge \neg\sigma\}\ \beta\ \{\psi\}}{\{\phi\}\ \textbf{if}\ \sigma\ \textbf{then}\ \alpha\ \textbf{else}\ \beta\ \{\psi\}}$$

While Rule: $$\frac{\{\phi \wedge \sigma\}\ \alpha\ \{\phi\}}{\{\phi\}\ \textbf{while}\ \sigma\ \textbf{do}\ \alpha\ \{\phi \wedge \neg\sigma\}}$$

Example: Euclid's Algorithm

$x := u;$

$y := v;$

`while` $x \neq y$ `do`

    `if` $x < y$ `then`

        $y := y - x;$

    `else`

        $x := x - y;$

Let $\phi := \gcd(x, y) = \gcd(u, v)$

Example: Euclid's Algorithm

$x := u;$

$y := v;$

`while` $x \neq y$ `do`

    `if` $x < y$ `then`

        $y := y - x;$

    `else`

        $x := x - y;$

Let α be the inner `if` statement.

Example: Euclid's Algorithm

$x := u;$

$y := v;$

`while` $x \neq y$ `do`

    `if` $x < y$ `then`

        $y := y - x;$

    `else`

        $x := x - y;$

Let α be the inner `if` statement.

Then $\{\gcd(x, y) = \gcd(u, v)\}\ \alpha\ \{\gcd(x, y) = \gcd(u, v)\}$

Example: Euclid's Algorithm

$x := u;$

$y := v;$

`while` $x \neq y$ `do`

    `if` $x < y$ `then`

        $y := y - x;$

    `else`

        $x := x - y;$

Hence by the `while`-rule (using a "weakening rule")

$$\frac{\{(\gcd(x,y) = \gcd(u,v)) \wedge (x \neq y)\}\ \alpha\ \{\gcd(x,y) = \gcd(u,v))\}}{\{\gcd(x,y) = \gcd(u,v)\}\ \textbf{while}\ \sigma\ \textbf{do}\ \alpha\ \{(\gcd(x,y) = \gcd(u,v)) \wedge \neg(x \neq y)\}}$$

Background: Propositional Dynamic Logic

Let P be a set of atomic programs and At a set of atomic propositions.

Formulas of **PDL** have the following syntactic form:

$$\phi := p \mid \bot \mid \neg\phi \mid \phi \vee \psi \mid [\alpha]\phi$$

$$\alpha := a \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^* \mid \phi?$$

where $p \in \mathsf{At}$ and $a \in \mathsf{P}$.

Background: Propositional Dynamic Logic

Let P be a set of atomic programs and At a set of atomic propositions.

Formulas of **PDL** have the following syntactic form:

$$\phi := p \mid \bot \mid \neg\phi \mid \phi \vee \psi \mid [\alpha]\phi$$

$$\alpha := a \mid \alpha \cup \beta \mid \alpha; \beta \mid \alpha^* \mid \phi?$$

where $p \in \mathsf{At}$ and $a \in \mathsf{P}$.

$\{\phi\}\ \alpha\ \{\psi\}$ **is replaced with** $\phi \rightarrow [\alpha]\psi$

From PDL to Game Logic

Game Logic (**GL**) was introduced by Rohit Parikh in

R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

From PDL to Game Logic

Game Logic (**GL**) was introduced by Rohit Parikh in

R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

Main Idea:

In **PDL**: $w \models \langle \pi \rangle \phi$: there is a run of the program π starting in state w that ends in a state where ϕ is true.

The programs in **PDL** can be thought of as *single player games*.

From PDL to Game Logic

Game Logic (**GL**) was introduced by Rohit Parikh in

R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

Main Idea:

In **PDL**: $w \models \langle \pi \rangle \phi$: there is a run of the program π starting in state w that ends in a state where ϕ is true.

The programs in **PDL** can be thought of as *single player games*.

Game Logic generalized **PDL** by considering two players:

In **GL**: $w \models \langle \gamma \rangle \phi$: Angel has a **strategy** in the game γ to ensure that the game ends in a state where ϕ is true.

From PDL to Game Logic

Consequences of two players:

From PDL to Game Logic

Consequences of two players:

$\langle\gamma\rangle\phi$: Angel has a strategy in γ to ensure ϕ is true

$[\gamma]\phi$: Demon has a strategy in γ to ensure ϕ is true

From PDL to Game Logic

Consequences of two players:

$\langle\gamma\rangle\phi$: Angel has a strategy in γ to ensure ϕ is true

$[\gamma]\phi$: Demon has a strategy in γ to ensure ϕ is true

Either Angel or Demon can win: $\langle\gamma\rangle\phi \vee [\gamma]\neg\phi$

From PDL to Game Logic

Consequences of two players:

$\langle\gamma\rangle\phi$: Angel has a strategy in γ to ensure ϕ is true

$[\gamma]\phi$: Demon has a strategy in γ to ensure ϕ is true

Either Angel or Demon can win: $\langle\gamma\rangle\phi \vee [\gamma]\neg\phi$

But not both: $\neg(\langle\gamma\rangle\phi \wedge [\gamma]\neg\phi)$

From PDL to Game Logic

Consequences of two players:

$\langle\gamma\rangle\phi$: Angel has a strategy in γ to ensure ϕ is true

$[\gamma]\phi$: Demon has a strategy in γ to ensure ϕ is true

Either Angel or Demon can win: $\langle\gamma\rangle\phi \vee [\gamma]\neg\phi$

But not both: $\neg(\langle\gamma\rangle\phi \wedge [\gamma]\neg\phi)$

Thus, $[\gamma]\phi \leftrightarrow \neg\langle\gamma\rangle\neg\phi$ is a valid principle

From PDL to Game Logic

Consequences of two players:

$\langle\gamma\rangle\phi$: Angel has a strategy in γ to ensure ϕ is true

$[\gamma]\phi$: Demon has a strategy in γ to ensure ϕ is true

Either Angel or Demon can win: $\langle\gamma\rangle\phi \vee [\gamma]\neg\phi$

But not both: $\neg(\langle\gamma\rangle\phi \wedge [\gamma]\neg\phi)$

Thus, $[\gamma]\phi \leftrightarrow \neg\langle\gamma\rangle\neg\phi$ is a valid principle

However, $[\gamma]\phi \wedge [\gamma]\psi \rightarrow [\gamma](\phi \wedge \psi)$ is **not** a valid principle

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds
- $\gamma_1;\gamma_2$: First play γ_1 then γ_2

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds
- $\gamma_1;\gamma_2$: First play γ_1 then γ_2
- $\gamma_1 \cup \gamma_2$: Angel choose between γ_1 and γ_2

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds
- $\gamma_1;\gamma_2$: First play γ_1 then γ_2
- $\gamma_1 \cup \gamma_2$: Angel choose between γ_1 and γ_2
- γ^*: Angel can choose how often to play γ (possibly not at all); each time she has played γ, she can decide whether to play it again or not.

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds
- $\gamma_1;\gamma_2$: First play γ_1 then γ_2
- $\gamma_1 \cup \gamma_2$: Angel choose between γ_1 and γ_2
- γ^*: Angel can choose how often to play γ (possibly not at all); each time she has played γ, she can decide whether to play it again or not.
- γ^d: Switch roles, then play γ

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds
- $\gamma_1;\gamma_2$: First play γ_1 then γ_2
- $\gamma_1 \cup \gamma_2$: Angel choose between γ_1 and γ_2
- γ^*: Angel can choose how often to play γ (possibly not at all); each time she has played γ, she can decide whether to play it again or not.
- γ^d: Switch roles, then play γ
- $\gamma_1 \cap \gamma_2 := (\gamma_1^d \cup \gamma_2^d)^d$: Demon chooses between γ_1 and γ_2

From PDL to Game Logic

Reinterpret operations and invent new ones:

- $?\phi$: Check whether ϕ currently holds
- $\gamma_1;\gamma_2$: First play γ_1 then γ_2
- $\gamma_1 \cup \gamma_2$: Angel choose between γ_1 and γ_2
- γ^*: Angel can choose how often to play γ (possibly not at all); each time she has played γ, she can decide whether to play it again or not.
- γ^d: Switch roles, then play γ
- $\gamma_1 \cap \gamma_2 := (\gamma_1^d \cup \gamma_2^d)^d$: Demon chooses between γ_1 and γ_2
- $\gamma^x := ((\gamma^d)^*)^d$: Demon can choose how often to play γ (possibly not at all); each time he has played γ, he can decide whether to play it again or not.

Syntax

Let Γ_0 be a set of atomic games and At a set of atomic propositions. Then formulas of Game Logic are defined inductively as follows:

$$\gamma \;:=\; g \mid \phi? \mid \gamma;\gamma \mid \gamma \cup \gamma \mid \gamma^* \mid \gamma^d$$

$$\phi \;:=\; \bot \mid p \mid \neg\phi \mid \phi \vee \phi \mid \langle\gamma\rangle\phi \mid [\gamma]\phi$$

where $p \in \mathsf{At}, g \in \Gamma_0$.

Game Logic: Semantics I

A **neighborhood game model** is a tuple
$\mathcal{M} = \langle W, \{E_g \mid g \in \Gamma_0\}, V\rangle$ where

W is a nonempty set of states

For each $g \in \Gamma_0$, $E_g : W \to 2^{2^W}$ is an **effectivity function** such that if $X \subseteq X'$ and $X \in E_g(w)$ then $X' \in E_g(w)$.

$X \in E_g(w)$ means in state s, Angel has a strategy to force the game to end in *some* state in X (we may write wE_gX)

$V : \mathsf{At} \to 2^W$ is a valuation function.

Game Logic: Semantics

A **neighborhood game model** is a tuple
$\mathcal{M} = \langle W, \{E_g \mid g \in \Gamma_0\}, V \rangle$ where

Propositional letters and boolean connectives are as usual.

$\mathcal{M}, w \models \langle \gamma \rangle \phi$ iff $(\phi)^{\mathcal{M}} \in E_\gamma(w)$

Game Logic: Semantics

A **neighborhood game model** is a tuple
$\mathcal{M} = \langle W, \{E_g \mid g \in \Gamma_0\}, V \rangle$ where

Propositional letters and boolean connectives are as usual.

$\mathcal{M}, w \models \langle\gamma\rangle\phi$ iff $(\phi)^{\mathcal{M}} \in E_\gamma(w)$

Suppose $E_\gamma(Y) = \{s \mid Y \in E_g(s)\}$

- $E_{\gamma_1;\gamma_2}(Y) := E_{\gamma_1}(E_{\gamma_2}(Y))$
- $E_{\gamma_1 \cup \gamma_2}(Y) := E_{\gamma_1}(Y) \cup E_{\gamma_2}(Y)$
- $E_{\phi?}(Y) := (\phi)^{\mathcal{M}} \cap Y$
- $E_{\gamma^d}(Y) := \overline{E_\gamma(\overline{Y})}$
- $E_{\gamma^*}(Y) := \mu X.Y \cup E_\gamma(X)$

Some Results

Fact Game Logic is more expressive than **PDL**

Some Results

Fact Game Logic is more expressive than **PDL**

$\langle (g^d)^* \rangle \bot$

Some Results

Fact Game Logic is more expressive than **PDL**

$\langle (g^d)^* \rangle \bot$

Theorem Game Logic^{-x}, where $x \in \{*, d\}$ is sound and complete with respect to the class of all game models.

Open Question Is (full) game logic complete with respect to the class of all game models?

R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

M. Pauly. *Logic for Social Software*. Ph.D. Thesis, University of Amsterdam (2001)..

Some Results

Theorem [2] Given a game logic formula ϕ and a finite game model $\mathcal{M}$, model checking can be done in time $O(|\mathcal{M}|^{ad(\phi)+1} \times |\phi|)$

Theorem [1,2] The satisfiability problem for game logic is in EXPTIME.

Theorem [1] Game logic can be translated into the modal μ-calculus

[1] R. Parikh. *The Logic of Games and its Applications.*. Annals of Discrete Mathematics. (1985) .

[2] M. Pauly. *Logic for Social Software.* Ph.D. Thesis, University of Amsterdam (2001)..

More Information

Editors: M. Pauly and R. Parikh. *Special Issue on Game Logic*. Studia Logica **75**, 2003.

M. Pauly and R. Parikh. *Game Logic — An Overview*. Studia Logica **75**, 2003.

R. Parikh. *The Logic of Games and its Applications*.. Annals of Discrete Mathematics. (1985) .

M. Pauly. *Game Logic for Game Theorists*. Available at `http://www.stanford.edu/ pianoman/`.

Example: Banach-Knaster Cake Cutting Algorithm

The Algorithm:

Example: **Banach-Knaster Cake Cutting Algorithm**

The Algorithm:

- The first person cuts out a piece which he claims is his fair share.

Example: Banach-Knaster Cake Cutting Algorithm

The Algorithm:

- The first person cuts out a piece which he claims is his fair share.

- The piece goes around being inspected by each agent.

Example: Banach-Knaster Cake Cutting Algorithm

The Algorithm:

- The first person cuts out a piece which he claims is his fair share.
- The piece goes around being inspected by each agent.
- Each agent, in turn, can either reduce the piece, putting some back to the main part, or just pass it.

Example: Banach-Knaster Cake Cutting Algorithm

The Algorithm:

- The first person cuts out a piece which he claims is his fair share.
- The piece goes around being inspected by each agent.
- Each agent, in turn, can either reduce the piece, putting some back to the main part, or just pass it.
- After the piece has been inspected by p_n, the last person who reduced the piece, takes it. If there is no such person, then the piece is taken by p_1.

Example: Banach-Knaster Cake Cutting Algorithm

The Algorithm:

- The first person cuts out a piece which he claims is his fair share.
- The piece goes around being inspected by each agent.
- Each agent, in turn, can either reduce the piece, putting some back to the main part, or just pass it.
- After the piece has been inspected by p_n, the last person who reduced the piece, takes it. If there is no such person, then the piece is taken by p_1.
- The algorithm continues with $n-1$ participants.

Example: Banach-Knaster Cake Cutting Algorithm

Correctness: The algorithm is "correct" iff each player has a winning strategy for achieving a fair outcome ($1/n$ of the pie according to p_i's own valuation).

Example: Banach-Knaster Cake Cutting Algorithm

Correctness: The algorithm is "correct" iff each player has a winning strategy for achieving a fair outcome ($1/n$ of the pie according to p_i's own valuation).

Towards a Formal Proof:

- $F(m, k)$: the piece m is big enough for k people.
- $F(m, k) \rightarrow (c, i)(F(m, k-1) \wedge H(x))$

Example: Banach-Knaster Cake Cutting Algorithm

Correctness: The algorithm is "correct" iff each player has a winning strategy for achieving a fair outcome ($1/n$ of the pie according to p_i's own valuation).

Towards a Formal Proof:

- $F(m, k)$: the piece m is big enough for k people.
- $F(m, k) \rightarrow (c, i)(F(m, k-1) \wedge H(x))$

Goal: *Derive* a formula expressing that every individual has a strategy that guarantees her fair share.

M. Pauly and R. Parikh. *Game Logic — An Overview.* Studia Logica **75**, 2003.

R. Parikh. *The Logic of Games and its Applications..* Annals of Discrete Mathematics. (1985) .

A Hoare-style Logic for Reasoning about Mechanisms

A Hoare-style Logic for Reasoning about Mechanisms

Add a (simultaneous) choice construct to the WHILE-language:
$\mathsf{ch}_A(\{x_a \mid a \in A\})$

A Hoare-style Logic for Reasoning about Mechanisms

Add a (simultaneous) choice construct to the WHILE-language: $\mathsf{ch}_A(\{x_a \mid a \in A\})$

A state is a function s that assigns element of some domain $\mathcal{D}$ to variables

A Hoare-style Logic for Reasoning about Mechanisms

Add a (simultaneous) choice construct to the WHILE-language: $\mathsf{ch}_A(\{x_a \mid a \in A\})$

A state is a function s that assigns element of some domain $\mathcal{D}$ to variables

An **interpretation** $\mathcal{I}$ is a first order structure (a domain $\mathcal{D}_\mathcal{I}$ and an interpretation of function and relation symbols) and preference relations $\succeq_a^\mathcal{I}$ on $\mathcal{D}_\mathcal{I}$ for each agent a.

A Hoare-style Logic for Reasoning about Mechanisms

Add a (simultaneous) choice construct to the WHILE-language: $\mathsf{ch}_A(\{x_a \mid a \in A\})$

A state is a function s that assigns element of some domain $\mathcal{D}$ to variables

An **interpretation** $\mathcal{I}$ is a first order structure (a domain $\mathcal{D}_\mathcal{I}$ and an interpretation of function and relation symbols) and preference relations $\succeq_a^\mathcal{I}$ on $\mathcal{D}_\mathcal{I}$ for each agent a.

Associate with each expression γ and state s a *semi-game* $G(\gamma, s, \mathcal{I})$

A Hoare-style Logic for Reasoning about Mechanisms

A semi-game $G(\gamma, s, \mathcal{I})$ can be turned into a **game** by adding an outcome function $\hat{o}$ that assigns an element of $\mathcal{D}_{\mathcal{I}}$ to terminal histories.

A Hoare-style Logic for Reasoning about Mechanisms

A semi-game $G(\gamma, s, \mathcal{I})$ can be turned into a **game** by adding an outcome function $\hat{o}$ that assigns an element of $\mathcal{D}_\mathcal{I}$ to terminal histories.

- A **predicate** is *any* set of states $P \subseteq \mathcal{S}_\mathcal{I}$
- An **e-predicate** is *any* subset $P \subseteq \mathcal{S}_\mathcal{I} \times \mathcal{D}_\mathcal{I}$

A Hoare-style Logic for Reasoning about Mechanisms

A semi-game $G(\gamma, s, \mathcal{I})$ can be turned into a **game** by adding an outcome function $\hat{o}$ that assigns an element of $\mathcal{D}_{\mathcal{I}}$ to terminal histories.

- A **predicate** is *any* set of states $P \subseteq \mathcal{S}_{\mathcal{I}}$
- An **e-predicate** is *any* subset $P \subseteq \mathcal{S}_{\mathcal{I}} \times \mathcal{D}_{\mathcal{I}}$

Define strategies and strategy profiles (σ) as usual. Each strategy profile corresponds to a run $run(\sigma)$. Let s_σ denote the last state of (a finite) $run(\sigma)$.

Given an e-predicate Q, let

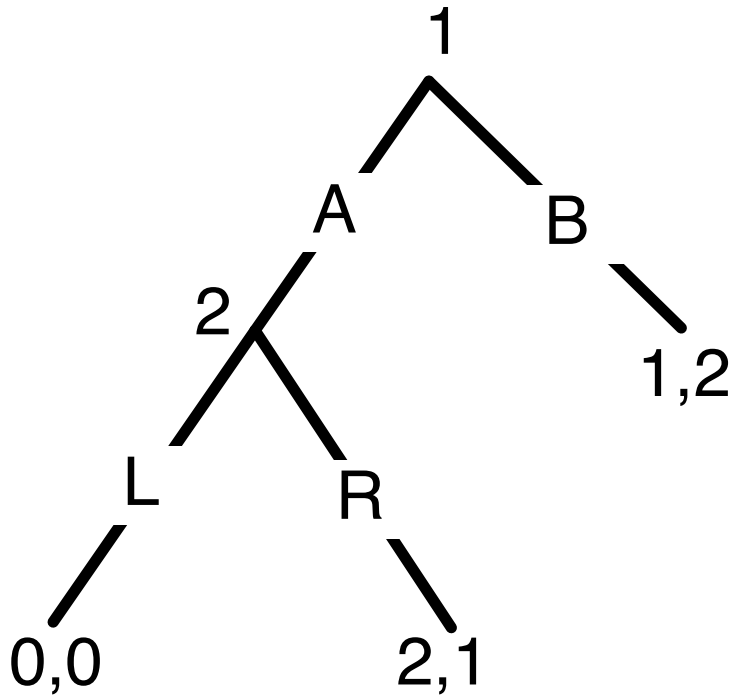
$\hat{O}_Q = \{\hat{o} \in \hat{O} \mid \text{for each terminal run } \sigma, \text{ if } \sigma \text{ is finite, then } (\mathsf{last}(\sigma), \hat{o}(\sigma)) \in Q\}$

Digression: Subgame Perfect Equilibrium

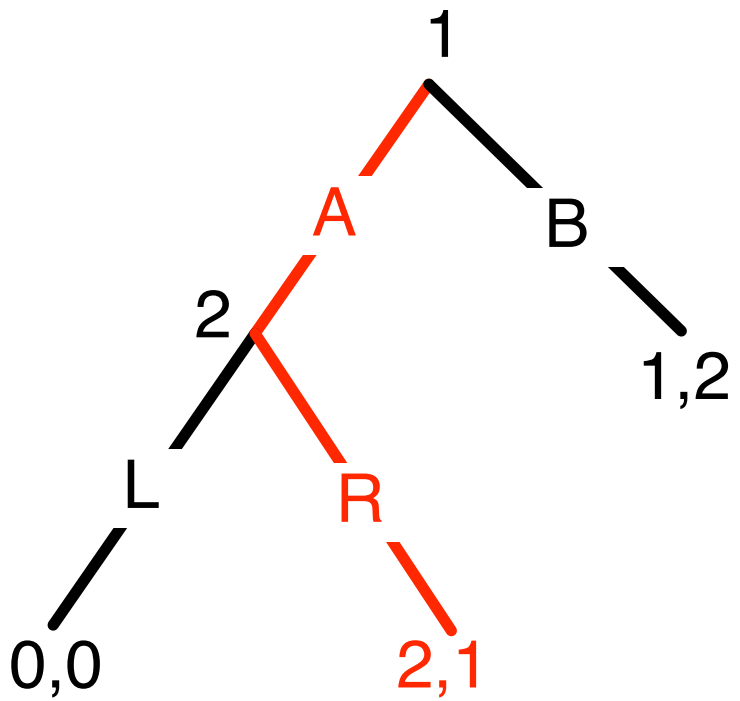
A **Nash Equilibrium** is a strategy profile in which no agent has an incentive to (unilaterally) deviate from their chosen strategy.

A **Subgame Perfect Equilibrium** is a strategy profile that is a Nash equilibrium in *every subgame*.

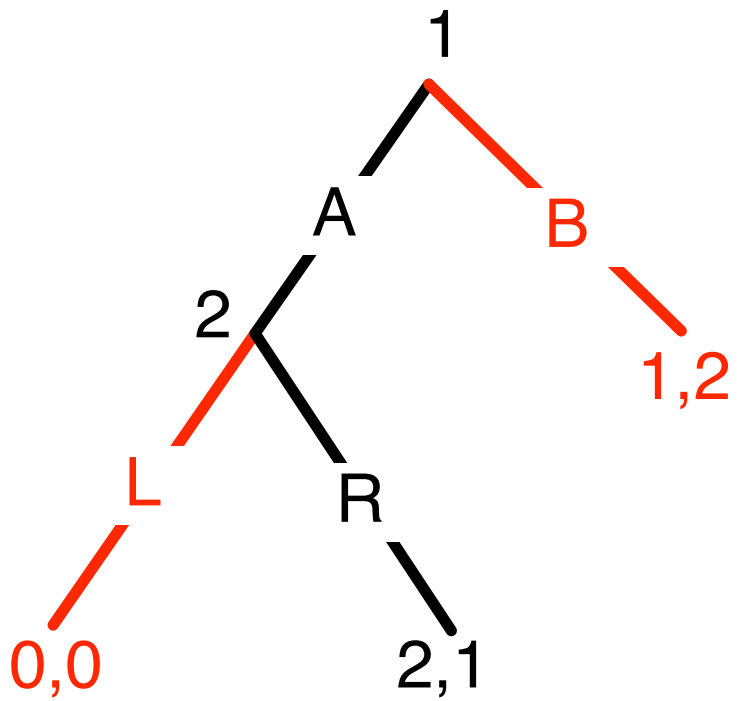
Digression: Subgame Perfect Equilibrium



Digression: Subgame Perfect Equilibrium



Digression: Subgame Perfect Equilibrium



A Hoare-style Logic for Reasoning about Mechanisms

A **correctness assertion** is an expression of the form $\{P\}\gamma\{Q\}$ where P, Q are **e-predicate**

A Hoare-style Logic for Reasoning about Mechanisms

A **correctness assertion** is an expression of the form $\{P\}\gamma\{Q\}$ where P, Q are **e-predicate**

We say $\mathcal{I} \models \{P\}\gamma\{Q\}$ provided

> For each $(s, o) \in P$ there is a outcome function $\hat{f} \in \hat{O}_Q$ and a strategy profile σ such that σ is a **subgame perfect equilibrium** in $G(\gamma, s, \mathcal{I}, \hat{f})$ and $\hat{(}f)(s_\sigma) = o$.

Mechanism Design Problem

A **social choice correspondence** f maps a preference profile $(\succeq_i)_{i \in \mathcal{A}}$ to a set of outcomes $X \subseteq \mathcal{D}_{\mathcal{I}}$.

Mechanism Design Problem

A **social choice correspondence** f maps a preference profile $(\succeq_i)_{i \in \mathcal{A}}$ to a set of outcomes $X \subseteq \mathcal{D}_{\mathcal{I}}$.

Mechanism Design Problem: find a mechanism which implements the social choice correspondence such that no matter what the preferences of the agents are, self-interested agents will have an incentive to play so that the outcome intended by the designer will obtain.

M. Osborne and A. Rubinstein. A Course in Game Theory. Chapter 10.

Mechanism Design Problem

Give a social choice correspondence f, let
$f^*(x) = \{(s, o) \in S_{\mathcal{I}} \times \mathcal{D}_{\mathcal{I}} \mid o \in f(x)\}$ and let Q be any functional e-predicate.

Mechanism Design Problem

Give a social choice correspondence f, let
$f^*(x) = \{(s, o) \in S_{\mathcal{I}} \times \mathcal{D}_{\mathcal{I}} \mid o \in f(x)\}$ and let Q be any functional e-predicate.

Then we say that (γ, Q) SPE-implements a social choice correspondence f iff for all preference profiles $(\geq_i)_{i \in \mathcal{A}}$ we have

$$\mathcal{I}[(\geq_i)_{i \in \mathcal{A}}] \models \{f^*((\geq_i)_{i \in \mathcal{A}})\}\ \gamma\ \{Q\}$$

Solomon's Dilema

Two women have come before him with a small child, both claiming to be the mother of the child.

Solomon's Dilema

Two women have come before him with a small child, both claiming to be the mother of the child.

Suppose that a is 'give the baby to A', b is 'give the baby to B' and c is 'cut the baby in half'.

Suppose

- $\Theta_1 : a >_1 b >_1 c$ and $b >_2 c >_2 a$
- $\Theta_2 : a >_1 c >_1 b$ and $b >_2 a >_2 c$

Solomon's Dilema

Two women have come before him with a small child, both claiming to be the mother of the child.

Suppose that a is 'give the baby to A', b is 'give the baby to B' and c is 'cut the baby in half'.

Suppose

- $\Theta_1 : a >_1 b >_1 c$ and $b >_2 c >_2 a$
- $\Theta_2 : a >_1 c >_1 b$ and $b >_2 a >_2 c$

Solomon must find a mechanism which implements the social choice rule $f(\Theta_1) = \{a\}$ and $f(\Theta_2) = \{b\}$.

It is well-known that f is not Nash-implementable

M. Osborne and A. Rubinstein. *A Course on Game Theory*. .

Solomon's Dilema

However, there is a solution involving money.

Allow Solomon to impose fines on the women, so outcomes are of the form:

$$(x, m_1, m_2)$$

where $x \in \{0, 1, 2\}$

Suppose the legitimate owner has valuation v_H and the other women has valuation v_L where

$$v_H > v_L > 0$$

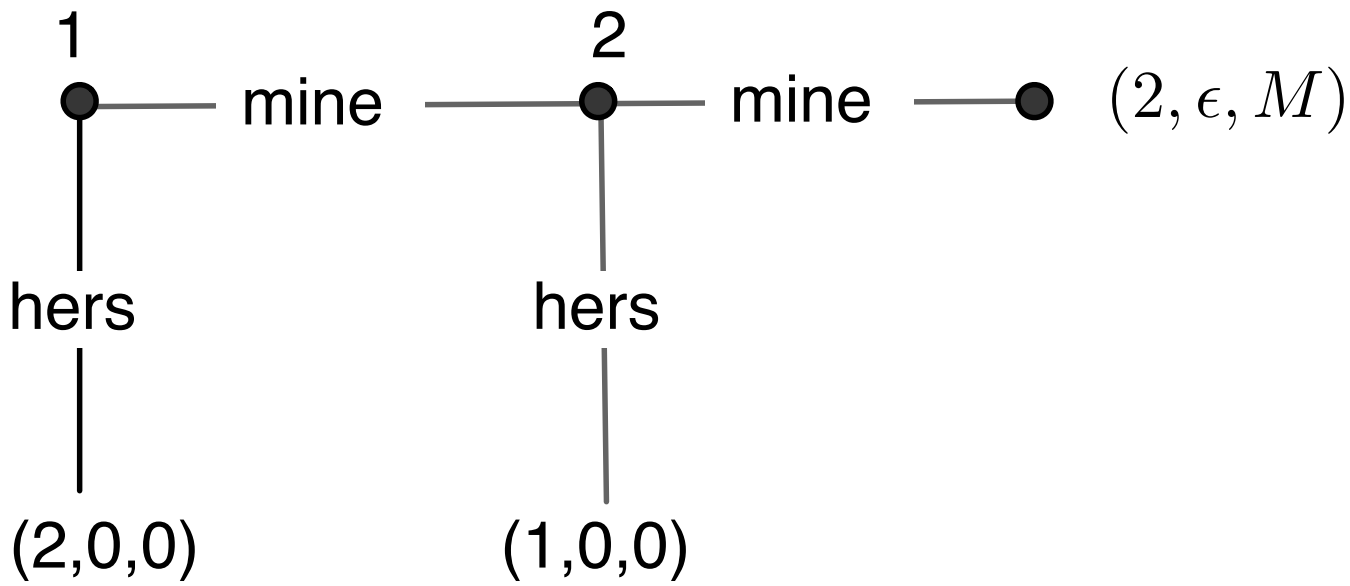
Solomon's Dilema

- If i does not get the painting then i's payoff is $-m_i$
- If i gets the painting and i is the legitimate owner then i's payoff is $v_H - m_i$
- If i gets the painting and i is not the legitimate owner then i's payoff is $v_L - m_i$

Solomon wishes to find a γ and Q such that $f(\Theta_i) = (i, 0, 0)$.

Let $\epsilon > 0$ and M be such that $v_L < M < v_H$.

Solomon's Dilema



Solomon's Dilema: A Formal Approach

$\mathsf{ch}_{\{1\}}(\{x_1\});$
$\texttt{if } x_1 > 0 \texttt{ then } owner := 2$
$\qquad \texttt{else } \mathsf{ch}_{\{2\}}(\{x_2\});$
$\qquad\qquad \texttt{if } x_2 > 0 \texttt{ then } owner := 1 \texttt{ else } owner := 0;$

Solomon's Dilema: A Formal Approach

$\mathsf{ch}_{\{1\}}(\{x_1\});$
$\texttt{if } x_1 > 0 \texttt{ then } owner := 2$
$\qquad \texttt{else } \mathsf{ch}_{\{2\}}(\{x_2\});$
$\qquad\qquad \texttt{if } x_2 > 0 \texttt{ then } owner := 1 \texttt{ else } owner := 0;$

Q is the conjunction of

- $owner = 1 \rightarrow o = (1, 0, 0)$
- $owner = 2 \rightarrow o = (1, 0, 0)$
- $owner = 0 \rightarrow o = (2, \epsilon, M)$

Solomon's Dilema: A Formal Approach

$\mathsf{ch}_{\{1\}}(\{x_1\})$;
if $x_1 > 0$ **then** $owner := 2$
else $\mathsf{ch}_{\{2\}}(\{x_2\})$;
if $x_2 > 0$ **then** $owner := 1$ **else** $owner := 0$;

Q is the conjunction of

- $owner = 1 \rightarrow o = (1, 0, 0)$
- $owner = 2 \rightarrow o = (1, 0, 0)$
- $owner = 0 \rightarrow o = (2, \epsilon, M)$

$\mathcal{I}[\Theta_1] \models \{o = (1, 0, 0)\}\gamma\{Q\}$ and $\mathcal{I}[\Theta_2] \models \{o = (2, 0, 0)\}\gamma\{Q\}$

Adjusted Winner

Adjusted winner (AW) is an algorithm for dividing n divisible goods among two people (invented by Steven Brams and Alan Taylor).

For more information see

- *Fair Division: From cake-cutting to dispute resolution* by Brams and Taylor, 1998
- *The Win-Win Solution* by Brams and Taylor, 2000
- www.nyu.edu/projects/adjustedwinner

Adjusted Winner: Example

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 1. Both Ann and Bob divide 100 points among the three goods.

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 1. Both Ann and Bob divide 100 points among the three goods.

Item	Ann	Bob
A	5	4
B	65	46
C	30	50
Total	100	100

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 2. The agent who assigns the most points receives the item.

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 2. The agent who assigns the most points receives the item.

Item	Ann	Bob
A	5	4
B	65	46
C	30	50
Total	100	100

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 2. The agent who assigns the most points receives the item.

Item	Ann	Bob
A	5	0
B	65	0
C	0	50
Total	70	50

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 3. Equitability adjustment:

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 3. Equitability adjustment:

Notice that $65/46 \geq 5/4 \geq 1 \geq 30/50$

Item	Ann	Bob
A	5	4
B	65	46
C	30	50
Total	100	100

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 3. Equitability adjustment:

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 3. Equitability adjustment:

Give A to Bob (the item whose ratio is closest to 1)

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 3. Equitability adjustment:

Give A to Bob (the item whose ratio is closest to 1)

Item	Ann	Bob
A	5	0
B	65	0
C	0	50
Total	70	50

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 3. Equitability adjustment:

Give A to Bob (the item whose ratio is closest to 1)

Item	Ann	Bob
A	0	4
B	65	0
C	0	50
Total	65	54

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 3. Equitability adjustment:

Still not equal, so give (some of) B to Bob: $65p = 100 - 46p$.

Item	Ann	Bob
A	0	4
B	65	0
C	0	50
Total	65	54

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 3. Equitability adjustment:

yielding $p = 100/111 = 0.9009$

Item	Ann	Bob
A	0	4
B	65	0
C	0	50
Total	65	54

Adjusted Winner: Example

Suppose Ann and Bob are dividing three goods: A, B, and C.

Step 3. Equitability adjustment:

yielding $p = 100/111 = 0.9009$

Item	Ann	Bob
A	0	4
B	58.559	4.559
C	0	50
Total	58.559	58.559

Adjusted Winner: Formal Definition

Suppose that $G_1, \ldots, G_n$ is a fixed set of goods.

Adjusted Winner: Formal Definition

Suppose that $G_1, \ldots, G_n$ is a fixed set of goods.

A **valuation** of these goods is a vector of natural numbers $\langle a_1, \ldots, a_n \rangle$ whose sum is 100.

Let $\alpha, \alpha', \alpha'', \ldots$ denote possible valuations for Ann and $\beta, \beta', \beta'', \ldots$ denote possible valuations for Bob.

Adjusted Winner: Formal Definition

Suppose that $G_1, \ldots, G_n$ is a fixed set of goods.

Adjusted Winner: Formal Definition

Suppose that $G_1, \ldots, G_n$ is a fixed set of goods.

An **allocation** is a vector of n real numbers where each component is between 0 and 1 (inclusive). An allocation $\sigma = \langle s_1, \ldots, s_n \rangle$ is interpreted as follows.

For each $i = 1, \ldots, n$, s_i is the proportion of G_i given to Ann.

Thus if there are three goods, then $\langle 1, 0.5, 0 \rangle$ means, "Give all of item 1 and half of item 2 to Ann and all of item 3 and half of item 2 to Bob."

Adjusted Winner: Formal Definition

Suppose that $G_1, \ldots, G_n$ is a fixed set of goods.

Adjusted Winner: Formal Definition

Suppose that $G_1, \ldots, G_n$ is a fixed set of goods.

$V_A(\alpha, \sigma) = \sum_{i=1}^{n} a_i s_i$ is the total number of points that Ann receives.

$V_B(\beta, \sigma) = \sum_{i=1}^{n} b_i(1 - s_i)$ is the total number of points that Bob receives.

Thus AW can be viewed as a function from pairs of valuations to allocations: $\mathsf{AW}(\alpha, \beta) = \sigma$ if σ is the allocation produced by the AW algorithm.

Adjusted Winner is Fair

Theorem *AW produces allocations that are efficient, equitable and envy-free (with respect to the announced valuations)*

S. Brams and A. Taylor. Fair Division. Cambridge University Press.

Adjusted Winner is Fair

Theorem *AW produces allocations that are efficient, equitable and envy-free (with respect to the announced valuations)*

S. Brams and A. Taylor. Fair Division. Cambridge University Press.

$\mathsf{ch}_{\mathcal{A}}(\{x_1, x_2\})$;
$s := \mathrm{wta}(x_1, x_2)$;
`while` $\neg Eq(s, x_1, x_2)$ `do`
    $s := \mathrm{t}(s, x_1, x_2)$;

Adjusted Winner: Strategizing

Item	Ann	Bob
Matisse	75	25
Picasso	25	75

Ann will get the Matisse and Bob will get the Picasso and each gets 75 of his or her points.

Adjusted Winner: Strategizing

Suppose Ann knows Bob's preferences, but Bob does not know Ann's.

Item	Ann	Bob
M	75	25
P	25	75

Item	Ann	Bob
M	26	25
P	74	75

So Ann will get M plus a portion of P.

According to Ann's announced allocation, she receives 50 points

According to Ann's actual allocation, she receives
$75 + 0.33 * 25 = 83.33$ points.

Conclusion

- How should we deal with strategizing?

  EP. *Towards a Logical Analysis of Adjusted Winner.* working paper.

- Expressivity issues.

- Other equilibrium notions.

- Apply these ideas to more sophisticated mechanisms

Thank you.