Logics for Social Software

Eric Pacuit

January 4, 2007

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.

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).

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).

Towards a Theory of Correctness of Social Procedures

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

Towards a Theory of Correctness of Social Procedures

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

Formally verifying game-theoretic mechanisms:

Towards a Theory of Correctness of Social Procedures

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

Formally verifying game-theoretic mechanisms:

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

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.

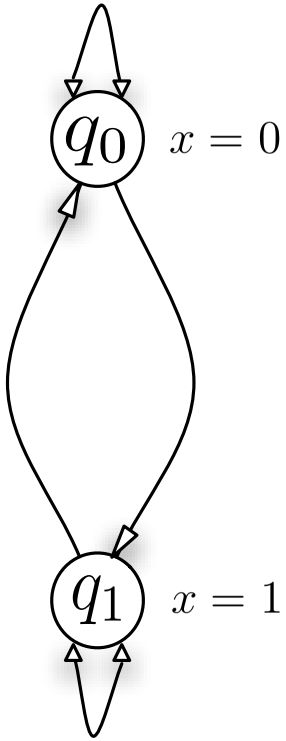
S. van Otterloo. *Strategic Analysis of Multi-agent Protocols.* Ph.D. Thesis, University of Liverpool (2005).

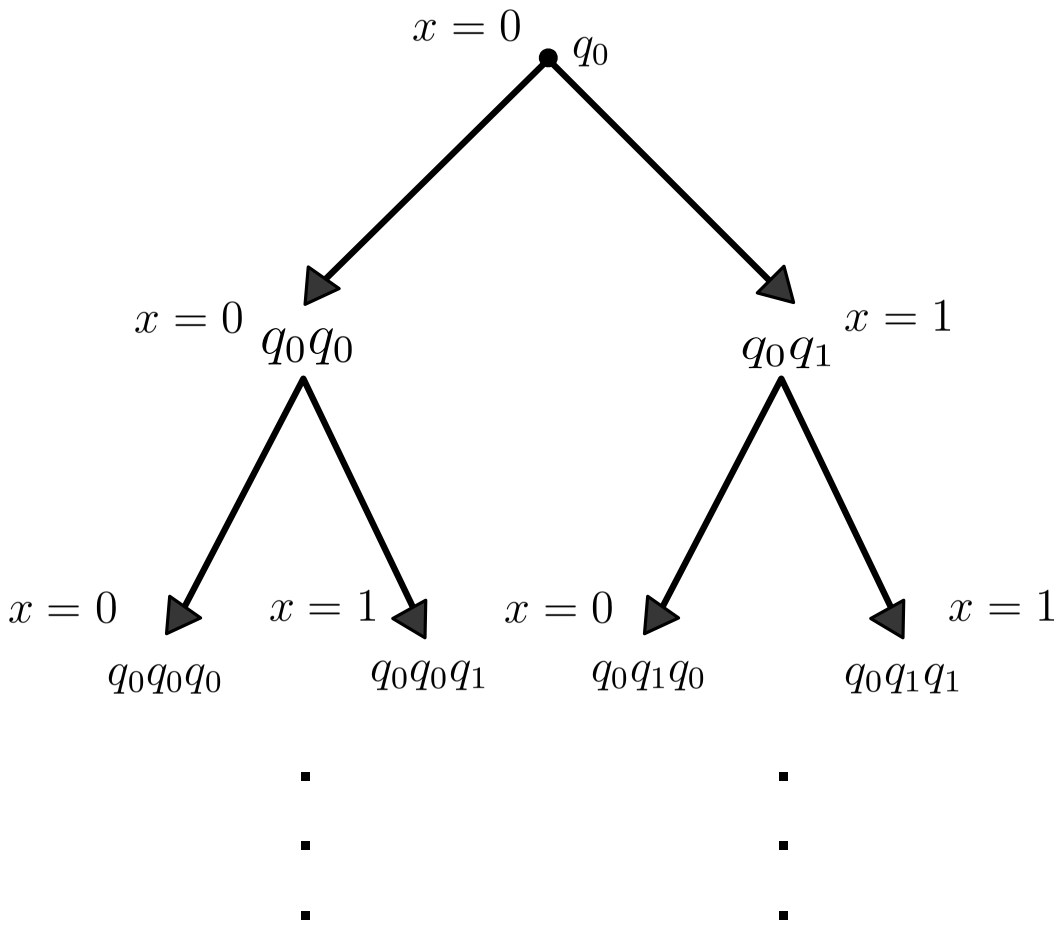
Outline of the Talk

- Strategy Logics
- Hoare Logic
- Pauly's Mechanism Programming Language
  - Example
- Case Study: Adjusted Winner

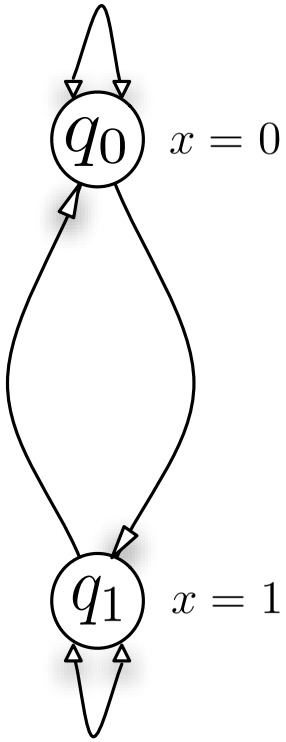
Model Checking vs. Deriving Correctness Conditions

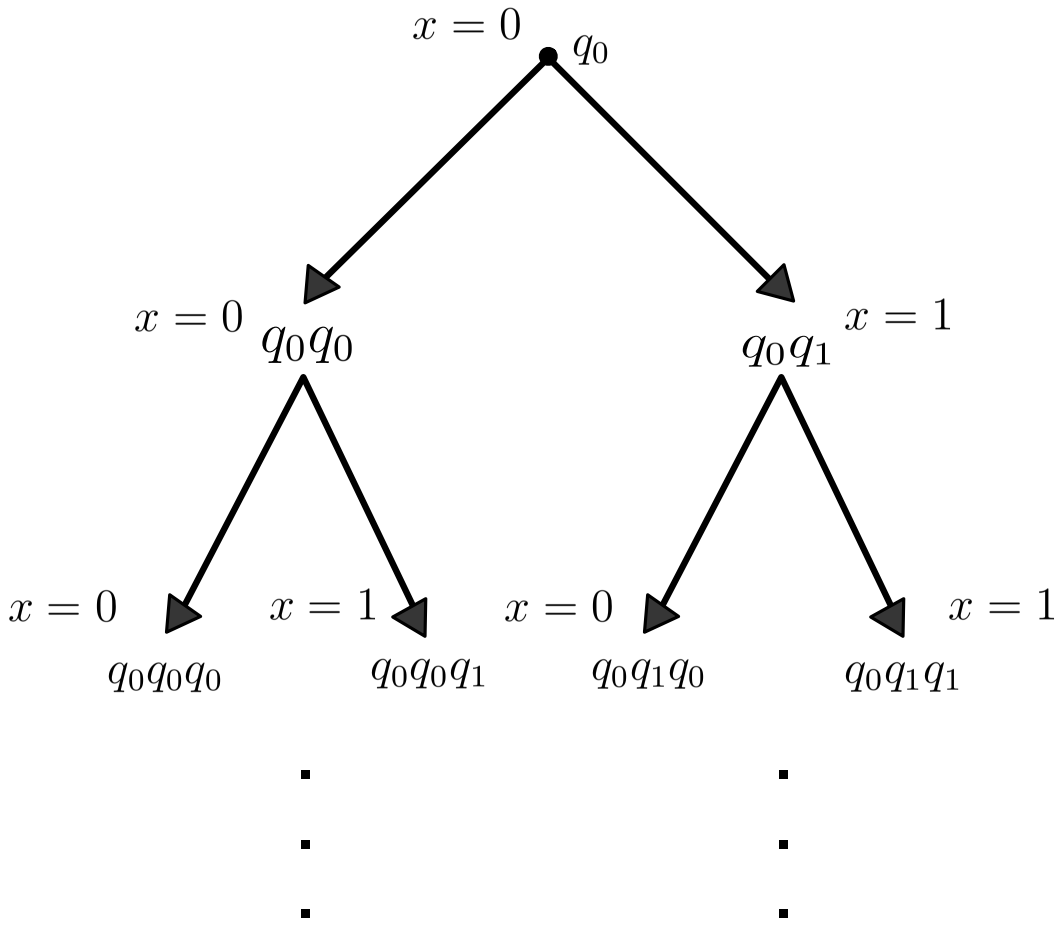
Computational vs. Behavioral Structures





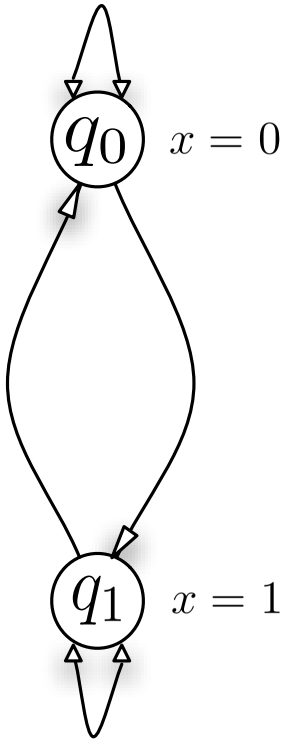
Computational vs. Behavioral Structures

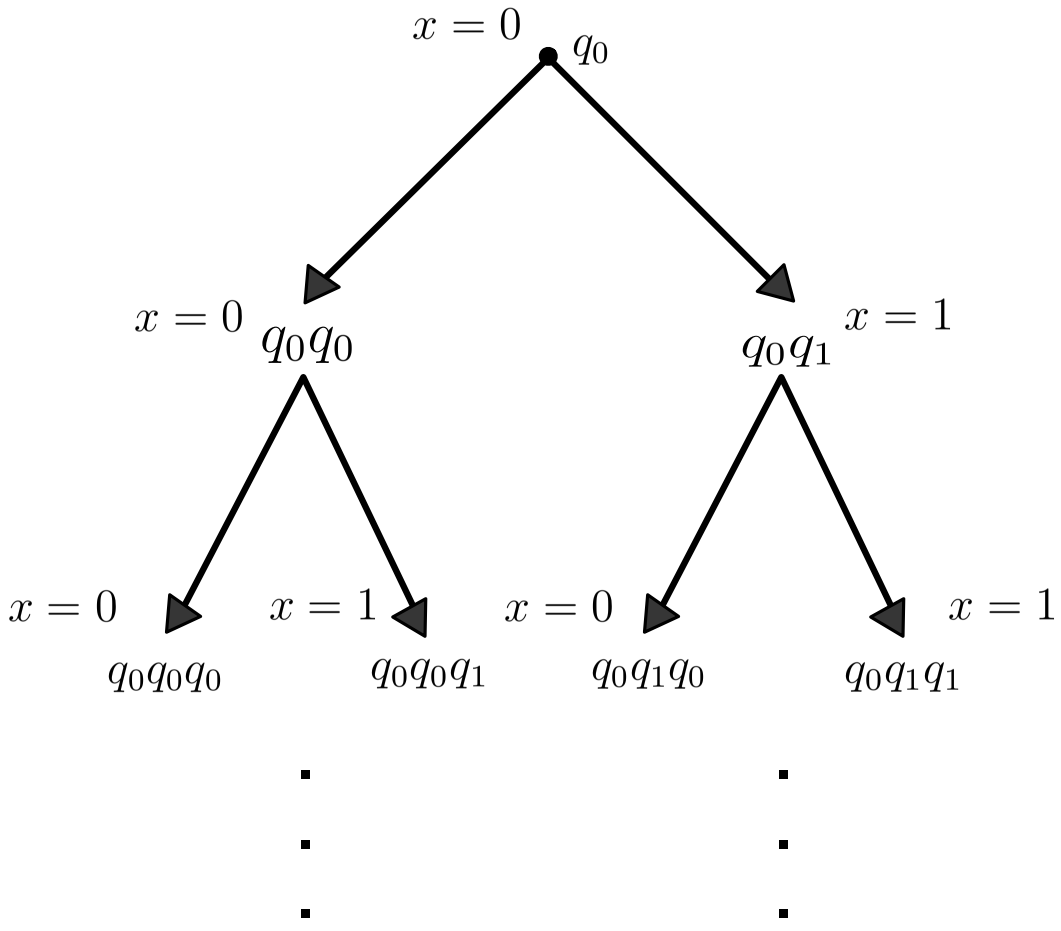




$$\exists \Diamond P_{x=1}$$

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?

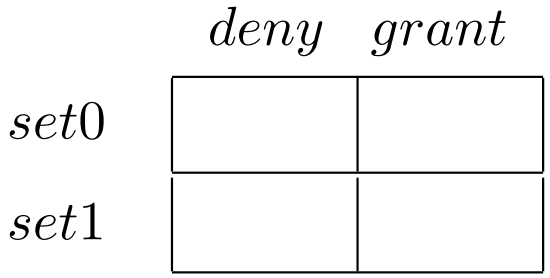
Example: Suppose that there are two agents: a server (s) and a client (c). The client asks to set the value of x and the server can either grant or deny the request. Assume the agents make simultaneous moves.

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

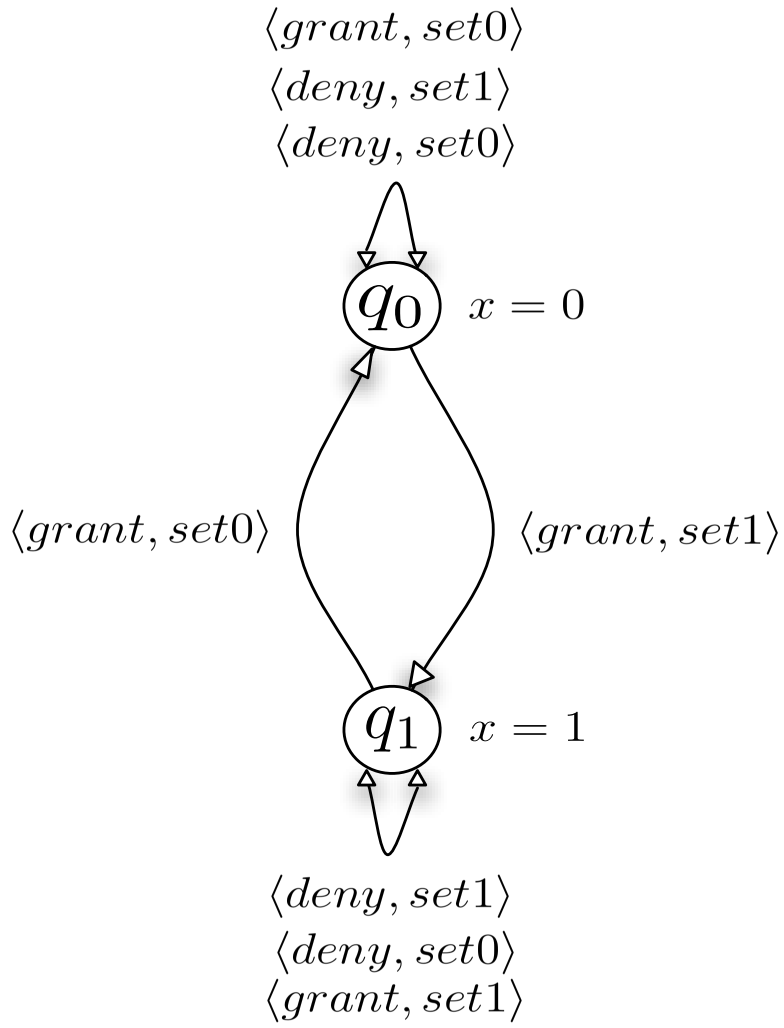
The previous model assumes there is *one* agent that "controls" the transition system.

What if there is more than one agent?

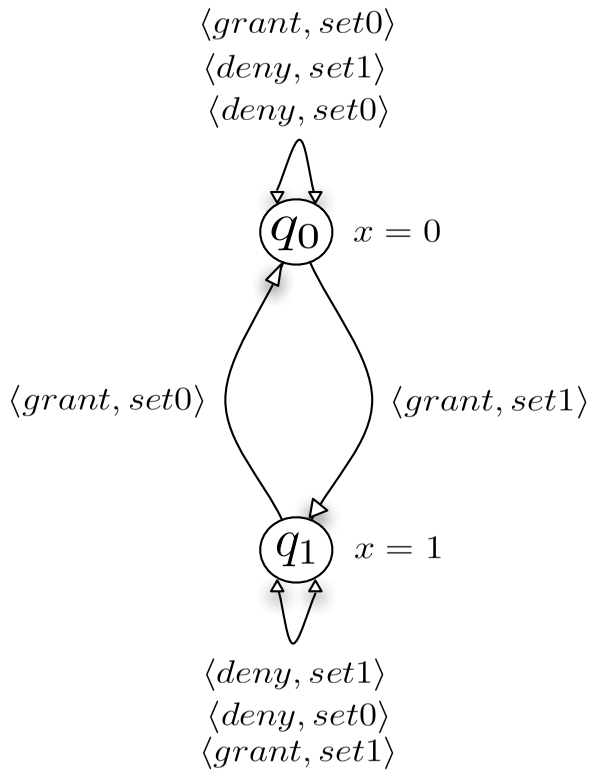
Example: Suppose that there are two agents: a server (s) and a client (c). The client asks to set the value of x and the server can either grant or deny the request. Assume the agents make simultaneous moves.

	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

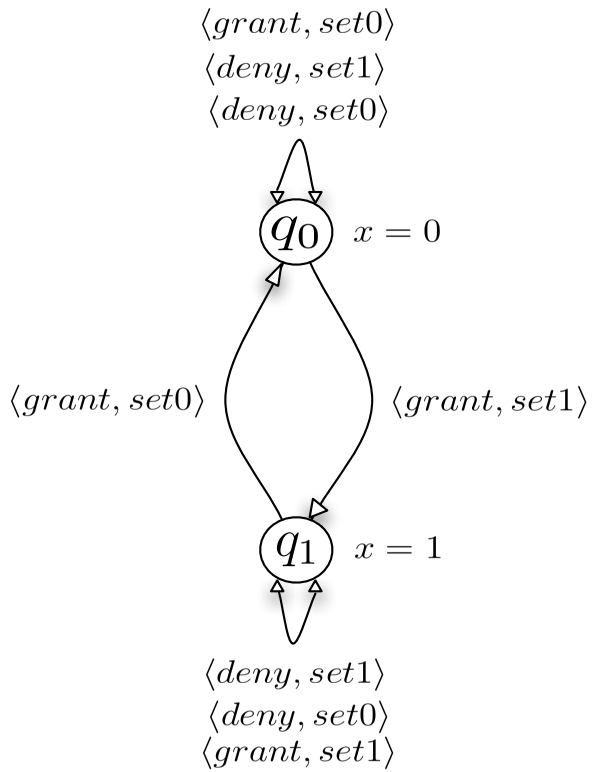


Multi-agent Transition Systems



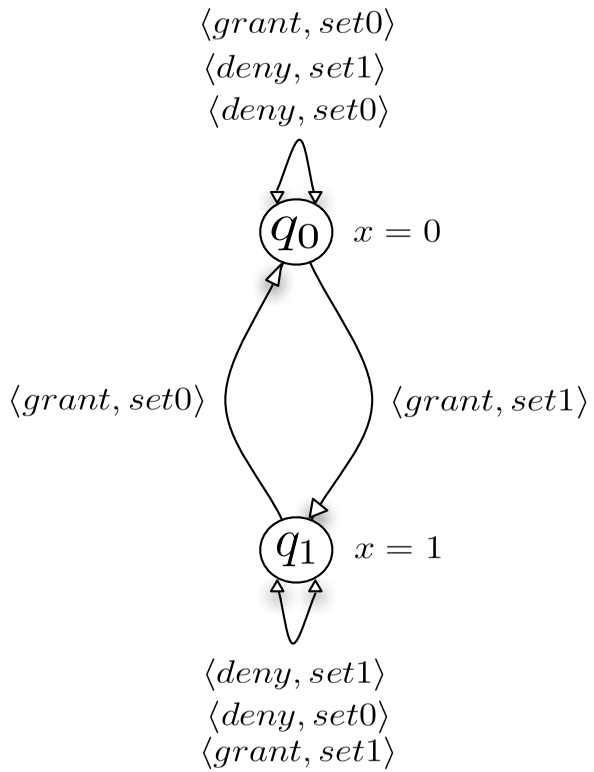
$$(P_{x=0} \rightarrow [s]P_{x=0}) \wedge (P_{x=1} \rightarrow [s]P_{x=1})$$

Multi-agent Transition Systems



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

Multi-agent Transition Systems



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

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).

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.

See, for example,

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

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

Game Logic

$(\gamma, i)\phi$: "Agent i has a strategy that forces ϕ to be true"

where γ is a formal description of a game (eg. $(g^d \cup h)^*$)

Game Logic

$(\gamma, i)\phi$: "Agent i has a strategy that forces ϕ to be true"

where γ is a formal description of a game (eg. $(g^d \cup h)^*$)

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) .

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).

Towards a Formal Proof: A state will consist of the values of $n + 2$ variables.

- The variable m has as its value the main part of the cake.
- The variable x is the piece under consideration.
- For $i = 1, \ldots, n$, the variable x_i has as its value the piece, if any, assigned to the person p_i.

Variables $m, x, x_1, \ldots, x_n$ range over subsets of the cake.

Example: Banach-Knaster Cake Cutting Algorithm

The algorithm uses three basic actions.

- c cuts a piece from m and assigns it to x. c works only if x is 0.
- r (reduce) transfers some (non-zero) portion from x back to m.
- a_i (assign) assigns the piece x to person p_i. Thus a_i is simply, $(x_i, x) := (x, 0)$.

Example: Banach-Knaster Cake Cutting Algorithm

The algorithm uses three basic actions.

- c cuts a piece from m and assigns it to x. c works only if x is 0.
- r (reduce) transfers some (non-zero) portion from x back to m.
- a_i (assign) assigns the piece x to person p_i. Thus a_i is simply, $(x_i, x) := (x, 0)$.

And predicates:

- $F(u, k)$: the piece u is big enough for k people.
- $F(u)$ abbreviates $F(u, 1)$ and F_i abbreviates $F(x_i)$.

Example: Banach-Knaster Cake Cutting Algorithm

Assume the following propositions

Example: Banach-Knaster Cake Cutting Algorithm

Assume the following propositions

1. $F(m,k) \rightarrow (c,i)(F(m,k-1) \wedge F(x))$

Example: Banach-Knaster Cake Cutting Algorithm

Assume the following propositions

1. $F(m,k) \rightarrow (c,i)(F(m,k-1) \wedge F(x))$

2. $F(m,k) \rightarrow [r*]F(m,k)$

Example: Banach-Knaster Cake Cutting Algorithm

Assume the following propositions

1. $F(m,k) \rightarrow (c,i)(F(m,k-1) \wedge F(x))$
2. $F(m,k) \rightarrow [r*]F(m,k)$
3. $F(m,k) \rightarrow [c][r*](F(m,k-1) \vee \langle r \rangle (F(m,k-1) \wedge F(x)))$

Example: Banach-Knaster Cake Cutting Algorithm

Assume the following propositions

1. $F(m,k) \rightarrow (c,i)(F(m,k-1) \wedge F(x))$
2. $F(m,k) \rightarrow [r*]F(m,k)$
3. $F(m,k) \rightarrow [c][r*](F(m,k-1) \vee \langle r \rangle (F(m,k-1) \wedge F(x)))$
4. $F(x) \rightarrow [a_i]F_i$

Example: Banach-Knaster Cake Cutting Algorithm

Assume the following propositions

1. $F(m,k) \rightarrow (c,i)(F(m,k-1) \wedge F(x))$
2. $F(m,k) \rightarrow [r*]F(m,k)$
3. $F(m,k) \rightarrow [c][r*](F(m,k-1) \vee \langle r \rangle (F(m,k-1) \wedge F(x)))$
4. $F(x) \rightarrow [a_i]F_i$

There are tacit assumptions of relevance, e.g. that r and c can only affect statements in which m or x occurs.

We assume moreover that $F(m,n)$ is true at the beginning.

Hoare Logic

Hoare Logic

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

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

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.

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")

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

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

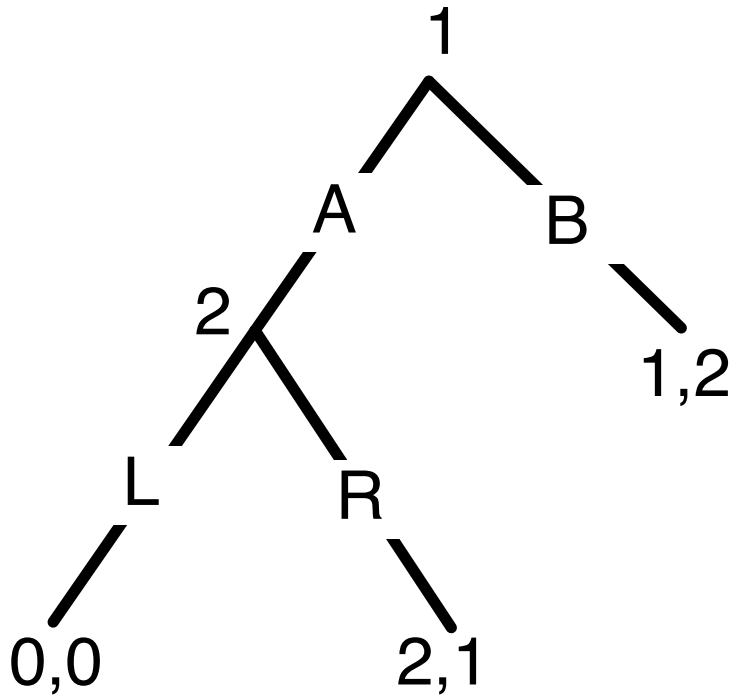
$\hat{O}_Q = \{\hat{o} \in \hat{O} \mid \text{for each terminal run } \sigma, \text{ if } \sigma \text{ is finite, then } (\mathsf{last}(\sigma), \hat{o}(\sigma)) \in Q\}$

Digression: Subgame Perfect Equilibrium

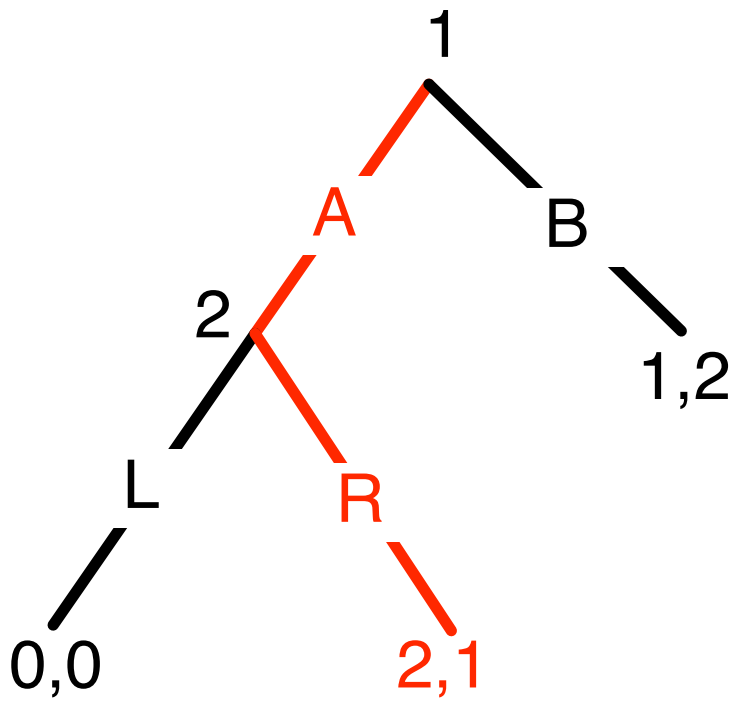
A **Nash Equilibrium** is a strategy profile in which no agent has an incentive to (unilaterally) deviate from their chosen strategy.

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

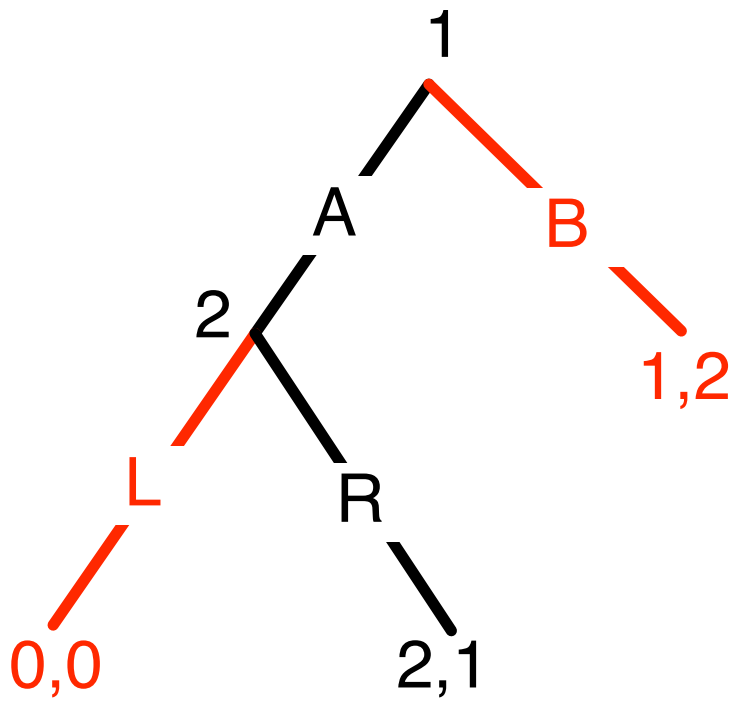
Digression: Subgame Perfect Equilibrium



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 $\hat{(}f)(s_\sigma) = o$.

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.

See also

EP, R. Parikh and Samer Salame. *Some Results on Adjusted Winner*. Available on my website..

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.

See also

EP, R. Parikh and Samer Salame. *Some Results on Adjusted Winner*. Available on my website..

$\mathsf{ch}_{\mathcal{A}}(\{x_1, x_2\})$;
$s := \mathrm{wta}(x_1, x_2)$;
`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.

Strategizing: A Theorem

Theorem (Brams and Taylor) *Assume there are two goods,* G_1 *and* G_2*, all true and announced values are restricted to integers, and suppose Bob's announced valuation of* G_1 *is* x*, where* $x \geq 50$*. Assume Ann's true valuation of* G_1 *is* b*. Then her optimal announced valuation of* G_1 *is:*

$$\begin{cases} x+1 & \textit{if } b > x \\ x & \textit{if } b = x \\ x-1 & \textit{if } b < x \end{cases}$$

Strategizing: Example

Suppose *both* players know each other's preferences but neither knows that the other knows their own preference.

Item	Ann	Bob
M	75	25
P	25	75

Item	Ann	Bob
M	26	74
P	74	26

Each will get 74 of his or her announced points, but each one is really getting only 25 of his or her *true* points.

Strategizing: Example

Suppose *both* players know each other's preferences. Moreover, Ann knows that Bob knows her preference and Bob doesn't know that Ann knows.

Item	Ann	Bob
M	26	74
P	74	26

Item	Ann	Bob
M	73	74
P	27	26

What happens as the level of knowledge increases?

Conclusion

- How should we deal with strategizing? And information?

  EP and R. Parikh. *Some Comments on Strategizing.* Available on my website.

  J. van Benthem and EP. *The Tree of Knowledge in Action: Towards a Common Perspective.* AiML 2006.

- Expressivity issues.

- Other equilibrium notions.

- What about designing social procedures?

Thank you.