Lecture 9: Part II: Strategy Logics

Eric Pacuit

ILLC, University of Amsterdam

staff.science.uva.nl/∼epacuit

epacuit@science.uva.nl

Lecture Date: April 13, 2006

Caput Logic, Language and Information: Social Software

staff.science.uva.nl/∼epacuit/caputLLI.html

Introduction

Imagine a set of agents (players, system components) taking actions, simultaneously or in turns, on a common set of states — and thus effecting transitions between these states.

Assume the agents pursue certain goals and in that pursuit they can form **coalitions**

The objective is to develop formal tools for reasoning about coalitions of agents and their ability to achieve specified outcomes in such situations.

(Temporal) Logics for Reasoning about Actions

- *Linear Time Temporal Logic* (**LTL**) [Pnuelli, 1977]: Reasoning about computations:

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

- *Branching-time Temporal Logic* (**CTL**, **CTL***) [Clarke and Emerson, 1981, Emerson and Halpern, 1986]: Allows quantification over paths:

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

- *Alternating-time Temporal Logic* (**ATL**, **ATL***) [Alur, Henzinger, Kupferman, 1997]: Selective quantification over paths:

  $\langle\langle A\rangle\rangle \bigcirc \phi$: The coalition A has a joint strategy to ensure that ϕ is true at the next moment.

Logics for Reasoning about Actions

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

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

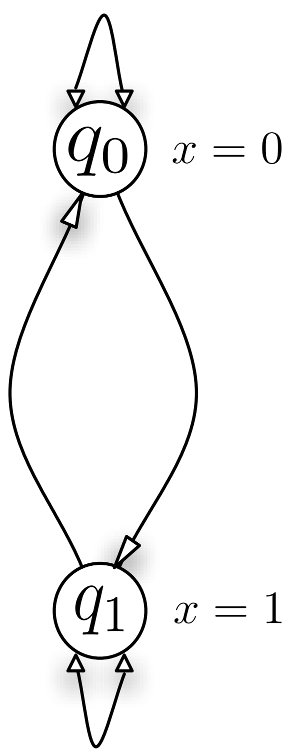
- *Game Logic* (**GL**) [Parikh, 1985]: Reasoning about games:

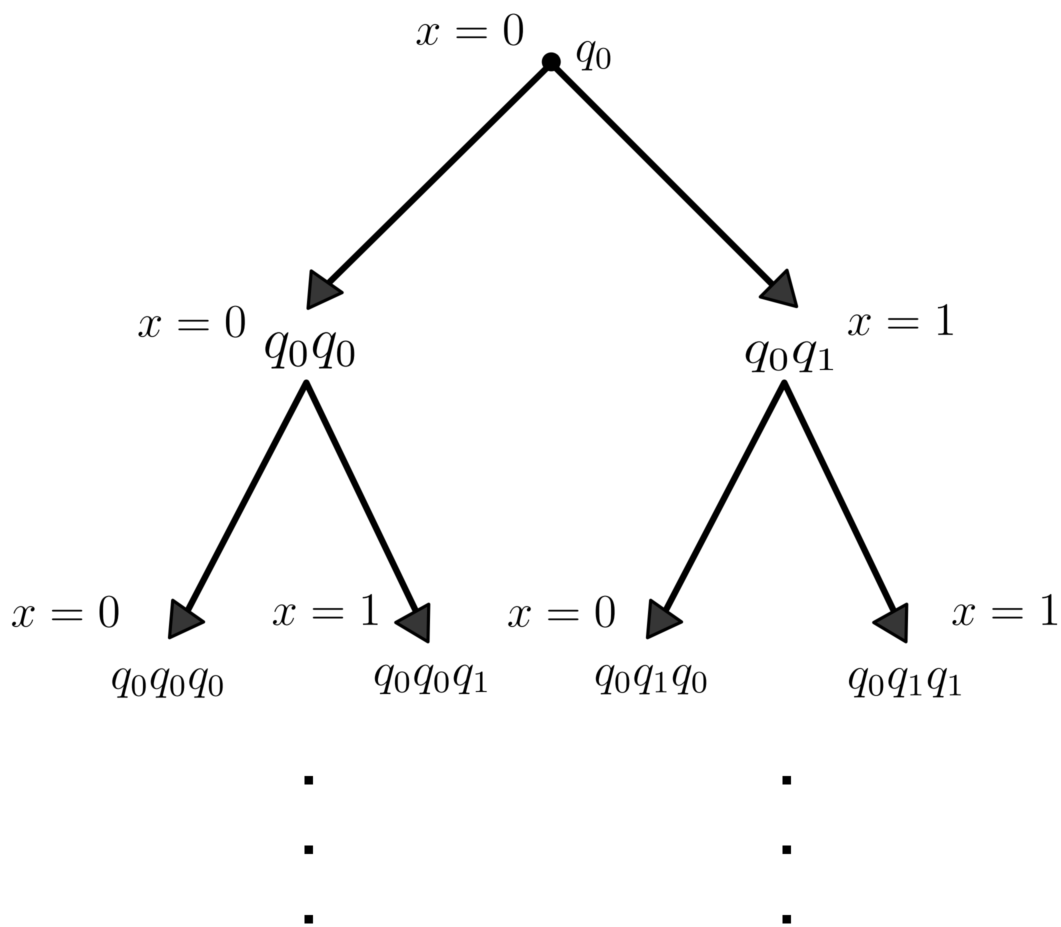
  $(\gamma)\phi$: Agent I has a strategy to bring about ϕ in game γ.

- *Coalitioanal Logic* (**CL**, **ECL**) [Pauly, 2000]: Reasoning about group power:

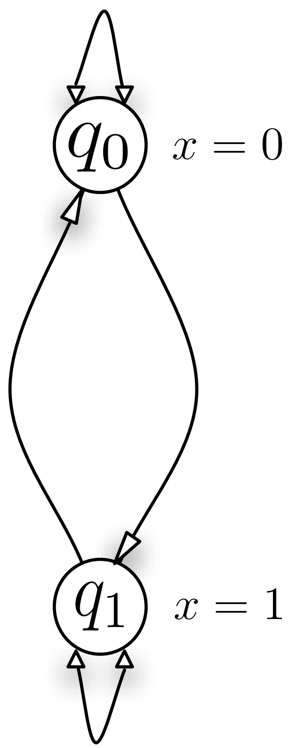
  $[C]\phi$: coalition C has a joint strategy to bring about ϕ.

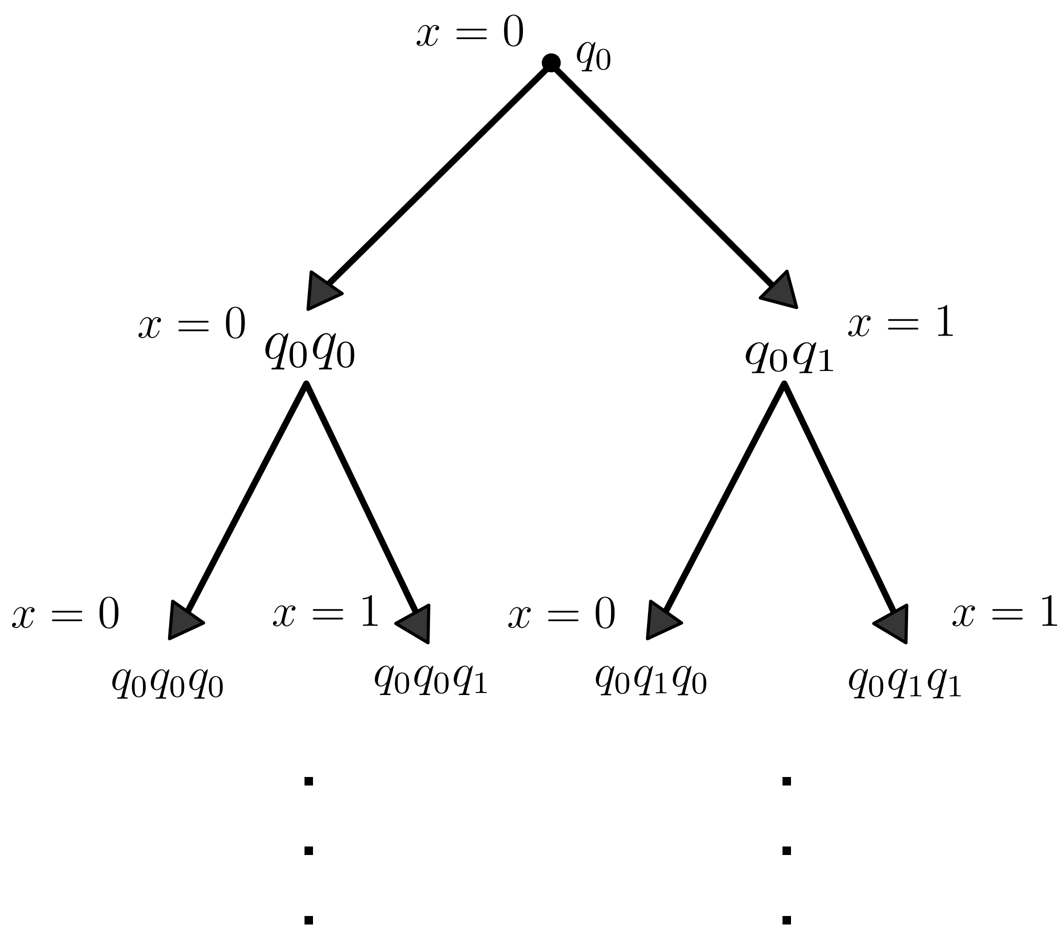
Computational vs. Behavioral Structures





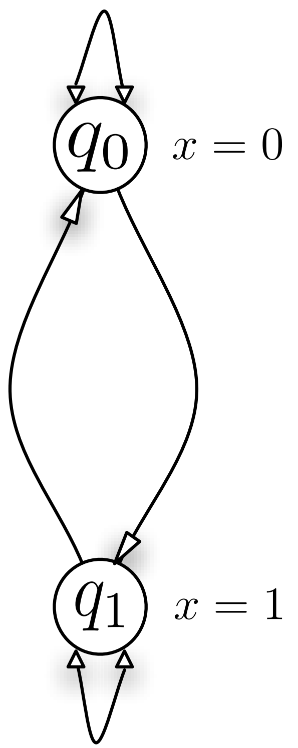
Computational vs. Behavioral Structures

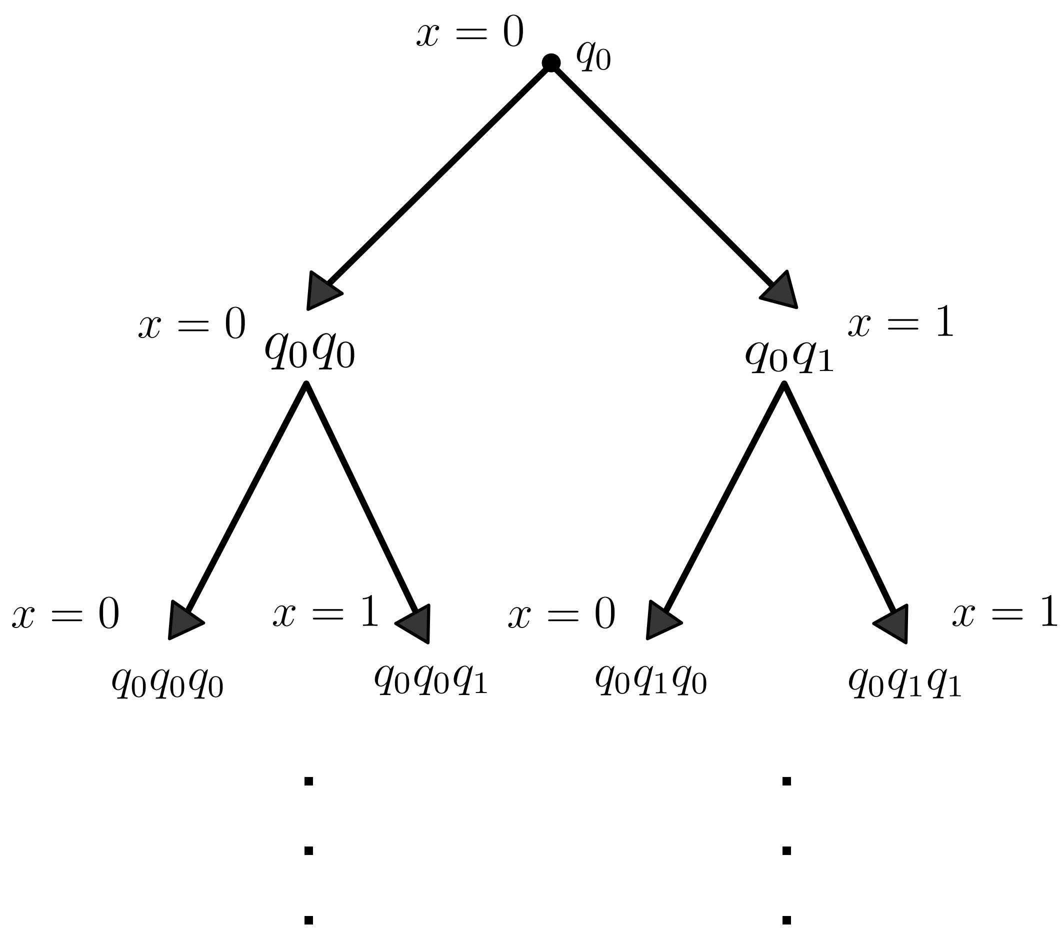




$\exists\Diamond P_{x=1}$

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.

	$deny$	$grant$
$set0$		
$set1$		

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.

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

Alternating Transition Systems

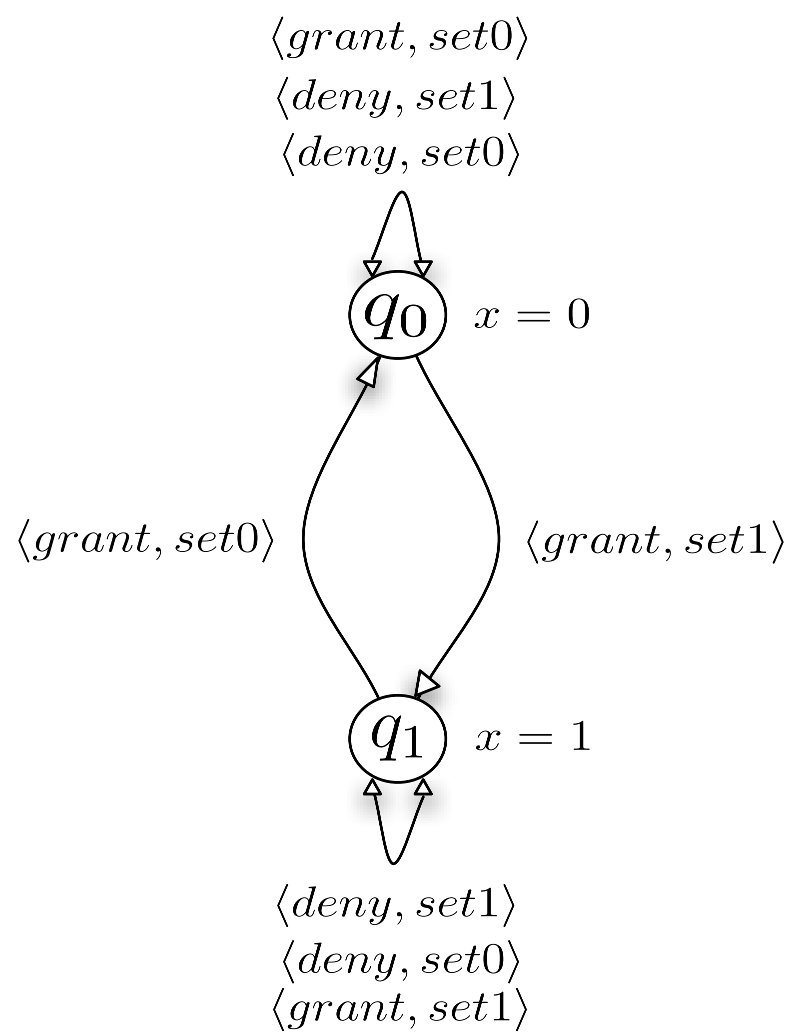
The previous model assumes there is *one* agent that "controls" the transition system.

What if there is more than one agent?

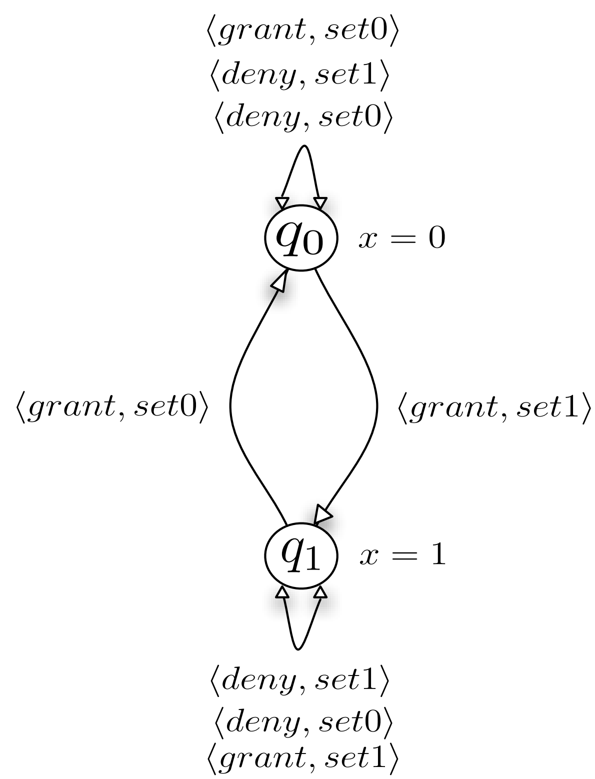
Example: Suppose that there are two agents: a server (s) and a client (c). The client asks to set the value of x and the server can either grant or deny the request. Assume the agents make simultaneous moves.

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

Multi-agent Transition Systems

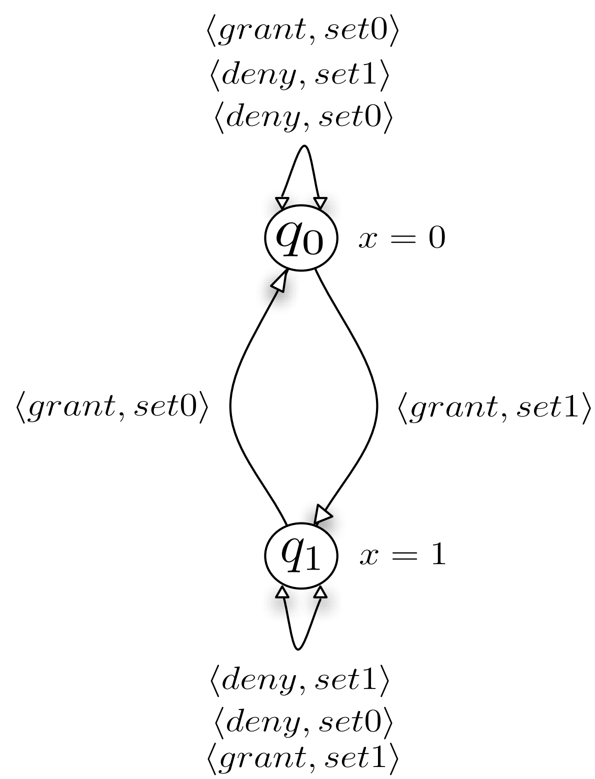


Multi-agent Transition Systems



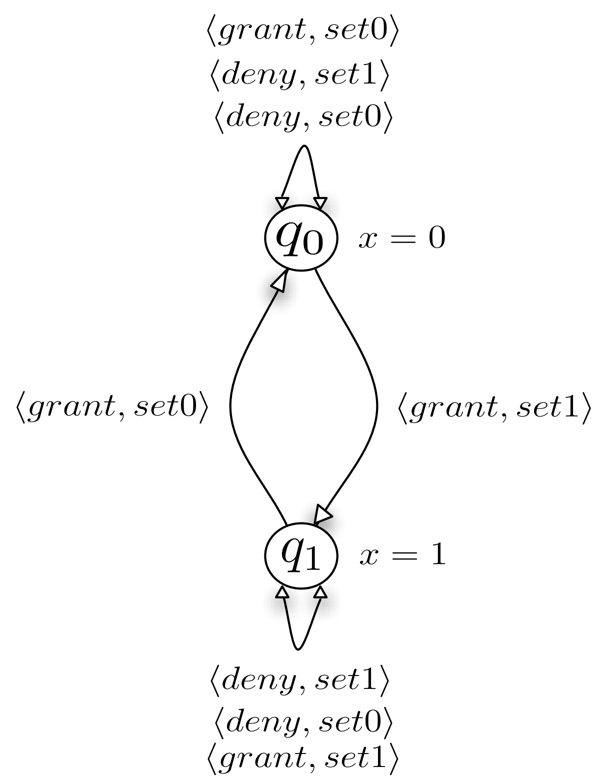
$$(P_{x=0} \rightarrow [s]P_{x=0}) \wedge (P_{x=1} \rightarrow [s]P_{x=1})$$

Multi-agent Transition Systems



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

Multi-agent Transition Systems



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

ATL:Syntax

Let Σ be a set of agents, Π a set of propositional variables and $A \subseteq \Sigma$.

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

A Logic for Mechanism Design

Recently, Pauly and Wooldridge in *Logic for Mechanism Design — A Manifesto*, argue that **ATL** can be used to reason about properties of game theoretic mechanisms.

- Logics have an important role to play in the specification, implementation and verification of mechanism

- Techniques developed for the automated verification of computer systems can be usefully applied to mechanism design problems in the game-theoretic sense. In particular, model checking.

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: $\langle\langle\rangle\rangle \bigcirc (p \wedge q)$
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: $\langle\langle\rangle\rangle \bigcirc (p \wedge q)$
2. We want the agents to be able to collectively choose and outcome: $\langle\langle A, B\rangle\rangle \bigcirc p \wedge \langle\langle A, B\rangle\rangle \bigcirc q$
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: $\langle\langle\rangle\rangle \bigcirc (p \wedge q)$
2. We want the agents to be able to collectively choose and outcome: $\langle\langle A, B\rangle\rangle \bigcirc p \wedge \langle\langle A, B\rangle\rangle \bigcirc q$
3. We do not want them to be able to bring about both outcomes simultaneously: $\neg\langle\langle A, B\rangle\rangle \bigcirc (p \wedge q)$
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: $\langle\langle\rangle\rangle \bigcirc (p \wedge q)$
2. We want the agents to be able to collectively choose and outcome: $\langle\langle A, B\rangle\rangle \bigcirc p \wedge \langle\langle A, B\rangle\rangle \bigcirc q$
3. We do not want them to be able to bring about both outcomes simultaneously: $\neg\langle\langle A, B\rangle\rangle \bigcirc (p \wedge q)$
4. We want them both to have equal power: $\neg\langle\langle x\rangle\rangle \bigcirc p \wedge \neg\langle\langle x\rangle\rangle \bigcirc q$ 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.

Coalitional Logic: 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.

$$X \in E_G^{\alpha}(C) \text{iff } \exists \sigma_C \forall \sigma_{\overline{C}} \quad o(\sigma_C, \sigma_{\overline{C}}) \in X$$

$$X \in E_G^{\beta}(C) \text{iff } \forall \sigma_{\overline{C}} \exists \sigma_C \quad o(\sigma_C, \sigma_{\overline{C}}) \in X$$

$E_G^{\alpha} \subseteq E_G^{\beta}$

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

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

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

Coalitional Logic: 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 $Q - X \in \mathsf{E}(\varnothing)$
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. *Logics for Social Software.* Ph.D. Thesis, ILLC.

Coalitional Logic: Coalition Effectivity Models

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

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

Coalitional Logic: Axiomatization

$(\bot)$ $\neg[C]\bot$

$(\top)$ $[C]\top$

(N) $(\neg[\varnothing]\neg\phi \to [N]\phi)$

(M) $[C](\phi \wedge \psi) \to [C]\psi$

(S) $([C_1]\phi_1 \wedge [C_2]\phi_2) \to [C_1 \cup C_2](\phi_1 \wedge \phi_2)$

provided $C_1 \cap C_2 = \varnothing$

$$\frac{\phi \qquad \phi \to \psi}{\psi}$$

$$\frac{\phi \leftrightarrow \psi}{[C]\phi \leftrightarrow [C]\psi}$$

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).

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.

ATL, Semantics I (Concurrent Game Structure)

A **Concurrent Game Structure** is a tuple $\mathcal{S} = \langle n, Q, \Pi, \pi, d, \delta \rangle$ with

- A natural number $n \geq 1$ of players: $\Sigma = \{1, \ldots, n\}$
- A set Q of states
- A finite set Π of atomic propositions
- For each $q \in Q$, a set $\pi(q) \subseteq \Pi$ of atomic propositions true at q.

ATL, Semantics I (Concurrent Game Structure)

A **Concurrent Game Structure** is a tuple $\mathcal{S} = \langle n, Q, \Pi, \pi, d, \delta \rangle$ with

- For each $a \in \Sigma$ and each $q \in Q$, a natural number $d_a(q) \geq 1$ of moves available at state q to player a. Moves of player a at q are identified with numbers $1, \ldots, d_a(q)$.

  A **move vector** at q is a tuple $\langle j_1, \ldots, j_k \rangle$ such that $1 \leq j_a \leq d_a(q)$ for each player a.

  Write $D(q) = \{1, \ldots, d_1(q)\} \times \cdots \times \{1, \ldots, d_n(q)\} \subseteq \mathbb{N}^n$.

- For each $q \in Q$ and each move vector $\langle j_1, \ldots, j_n \rangle \in D(q)$, a state $\delta(q, j_1, \ldots, j_n) \in Q$ that results from the state q if each player $a \in \{1, \ldots, n\}$ chooses move j_a. δ is called the **transition function**.

Notation

- q' is a **successor** of q iff $\exists \langle j_1, \ldots, j_n \rangle \in D(q)$ such that $q' = \delta(q, j_1, \ldots, j_n)$

- A **computation** is an infinite sequence $\lambda = q_0 q_1 \ldots$ of states such that for all $i \geq 0$, q_{i+1} is a successor of q_i

- If a computation $\lambda = q_0 q_1 q_2 \cdots$ starts at q (i.e., $q_0 = q$), λ is called a q-computation

- Given a computation λ, $\lambda[i]$, $\lambda[0, i]$, and $\lambda[i, \infty]$ denote respectively the ith state of λ, the finite prefix $q_0 q_1 \cdots q_i$ and the infinite suffix $q_i q_{i+1} \cdots$

Strategies I

A **strategy** for player $a \in \Sigma$ is a function $f_a : Q^+ \to \mathbb{N}$ that maps every nonempty finite state sequence λ to a natural number such that if the last state is q, $1 \leq f_a(\lambda) \leq d_a(q)$.

For a state $q \in Q$ and $A \subseteq \Sigma$ an A-move is a tuple $(\sigma_a)_{a \in A}$ such that $1 \leq \sigma_a \leq d_a(q)$. Let $D_A(q)$ denote the set of A-moves.

A state q' is **consistent** with an A-move $\sigma \in D_A(q)$ when there is a move vector $\langle j_1, \ldots, j_n \rangle \in D(q)$ such that

1. $j_a = \sigma_a$ for all $a \in A$
2. $q' = \delta(q, j_1, \ldots, j_n)$

Strategies II

Let out(σ) denote the set of state consistent with σ

For $A \subseteq \Sigma$ an A-strategy is a mapping $F_A : Q^+ \rightarrow \cup\{D_A(q) \mid q \in Q\}$ such that for all $\lambda \in Q^*$, and for all $q \in Q$, $F_A(\lambda \cdot q) \in D_A(q)$.

A q-computation $\lambda = q_0 q_1 \ldots$ with $q_0 = q$ is **consistent** with F_A written $\lambda \in out(q, F_A)$ if for all $i \geq 0$, q_{i+1} is a successor of q_i and $q_{i+1} \in out(F_A(\lambda[0, i]))$.

Definition of Truth

- For $p \in \Pi$, $q \models p$ iff $p \in \pi(q)$
- $q \models \neg\phi$ iff $q \not\models \phi$
- $q \models \phi \vee \psi$ iff $q \models \phi$ of $q \models \psi$
- $q \models \langle\langle A \rangle\rangle \bigcirc \phi$ iff there exists an A-strategy F_A such that for each computation $\lambda \in out(q, F_A)$ we have $\lambda[1] \models \phi$; equivalently, iff there exists an A-move $\sigma \in D_A(q)$ such that for all $q' \in out(\sigma)$, $q' \models \phi$.
- $q \models \langle\langle A \rangle\rangle \Box \phi$ iff there exists an A-strategy F_A such that for each computation $\lambda \in out(q, F_A)$ and all $i \geq 0$, we have $\lambda[i] \models \phi$
- $q \models \langle\langle A \rangle\rangle \phi U \psi$ iff there exists an A-strategy F_A such that for each computation $\lambda \in out(q, F_A)$ there exists a position $i \geq 0$ such that $\lambda[i] \models \psi$ and for all $0 \leq j < i$ we have $\lambda[j] \models \phi$

Definition of Truth

- For $p \in \Pi$, $q \models p$ iff $p \in \pi(q)$
- $q \models \neg\phi$ iff $q \not\models \phi$
- $q \models \phi \vee \psi$ iff $q \models \phi$ of $q \models \psi$
- $q \models \langle\langle A \rangle\rangle \bigcirc \phi$ iff there exists an A-strategy F_A such that for each computation $\lambda \in out(q, F_A)$ we have $\lambda[1] \models \phi$; equivalently, iff there exists an A-move $\sigma \in D_A(q)$ such that for all $q' \in out(\sigma)$, $q' \models \phi$.
- $q \models \langle\langle A \rangle\rangle \Box \phi$ iff there exists an A-strategy F_A such that for each computation $\lambda \in out(q, F_A)$ and all $i \geq 0$, we have $\lambda[i] \models \phi$
- $q \models \langle\langle A \rangle\rangle \phi U \psi$ iff there exists an A-strategy F_A such that for each computation $\lambda \in out(q, F_A)$ there exists a position $i \geq 0$ such that $\lambda[i] \models \psi$ and for all $0 \leq j < i$ we have $\lambda[j] \models \phi$

Definition of Truth

- For $p \in \Pi$, $q \models p$ iff $p \in \pi(q)$
- $q \models \neg\phi$ iff $q \not\models \phi$
- $q \models \phi \vee \psi$ iff $q \models \phi$ of $q \models \psi$
- $q \models \langle\langle A \rangle\rangle \bigcirc \phi$ iff there exists an A-strategy F_A such that for each computation $\lambda \in out(q, F_A)$ we have $\lambda[1] \models \phi$; equivalently, iff there exists an A-move $\sigma \in D_A(q)$ such that for all $q' \in out(\sigma)$, $q' \models \phi$.
- $q \models \langle\langle A \rangle\rangle \Box \phi$ iff there exists an A-strategy F_A such that for each computation $\lambda \in out(q, F_A)$ and all $i \geq 0$, we have $\lambda[i] \models \phi$
- $q \models \langle\langle A \rangle\rangle \phi U \psi$ iff there exists an A-strategy F_A such that for each computation $\lambda \in out(q, F_A)$ there exists a position $i \geq 0$ such that $\lambda[i] \models \psi$ and for all $0 \leq j < i$ we have $\lambda[j] \models \phi$

Definition of Truth

- For $p \in \Pi$, $q \models p$ iff $p \in \pi(q)$
- $q \models \neg\phi$ iff $q \not\models \phi$
- $q \models \phi \vee \psi$ iff $q \models \phi$ of $q \models \psi$
- $q \models \langle\langle A \rangle\rangle \bigcirc \phi$ iff there exists an A-strategy F_A such that for each computation $\lambda \in out(q, F_A)$ we have $\lambda[1] \models \phi$; equivalently, iff there exists an A-move $\sigma \in D_A(q)$ such that for all $q' \in out(\sigma)$, $q' \models \phi$.
- $q \models \langle\langle A \rangle\rangle \Box \phi$ iff there exists an A-strategy F_A such that for each computation $\lambda \in out(q, F_A)$ and all $i \geq 0$, we have $\lambda[i] \models \phi$
- $q \models \langle\langle A \rangle\rangle \phi U \psi$ iff there exists an A-strategy F_A such that for each computation $\lambda \in out(q, F_A)$ there exists a position $i \geq 0$ such that $\lambda[i] \models \psi$ and for all $0 \leq j < i$ we have $\lambda[j] \models \phi$

Alternative Semantics for ATL: Alternating Transition Systems

An **alternating transition system** is a tuple $\langle \Pi, N, Q, \pi, \delta \rangle$ where

- Π is a set of atomic propositions
- N is a set of agents (nonempty and finite)
- Q is a set of states (nonempty)
- $\pi : Q \to 2^{\Pi}$ is a valuation funciton
- $\delta : Q \times N \to 2^{2^Q}$ is a **transition funciton**. Foreach $i \in N$, $Q_i \in \delta(q, i)$ means i can force the state to be in Q_i. We assume that for each that the system is deterministic: given a state q and the agents choices in q, $Q_1, \ldots, Q_n$, $Q_1 \cap \cdots Q_n$ is a singleton.

Axiomatization

Coalitional Logic Axioms:

(TAUT) Enough propositional tautologies

($\bot$) $\neg\langle\langle A\rangle\rangle \bigcirc \bot$

($\top$) $\langle\langle A\rangle\rangle \bigcirc \top$

(Σ) $\neg\langle\langle \emptyset\rangle\rangle \bigcirc \neg\phi \rightarrow \langle\langle \Sigma\rangle\rangle \bigcirc \phi$

(S) $\langle\langle A_1\rangle\rangle \bigcirc \phi_1 \wedge \langle\langle A_2\rangle\rangle \bigcirc \phi_2 \rightarrow \langle\langle A_1 \cup A_2\rangle\rangle \bigcirc (\phi_1 \wedge \phi_2)$

Axiomatization

Fixed Point Axioms:

(FP$_\Box$) $\langle\!\langle A\rangle\!\rangle\Box\phi \leftrightarrow \phi \wedge \langle\!\langle A\rangle\!\rangle \bigcirc \langle\!\langle A\rangle\!\rangle\Box\phi$

(GFP$_\Box$) $\langle\!\langle \varnothing\rangle\!\rangle\Box(\theta \rightarrow (\phi \wedge \langle\!\langle A\rangle\!\rangle \bigcirc \theta)) \rightarrow \langle\!\langle \varnothing\rangle\!\rangle\Box(\theta \rightarrow \langle\!\langle A\rangle\!\rangle\Box\phi)$

(FP$_U$) $\langle\!\langle A\rangle\!\rangle\phi U\psi \leftrightarrow \psi \vee (\phi \wedge \langle\!\langle A\rangle\!\rangle \bigcirc \langle\!\langle A\rangle\!\rangle\phi U\psi)$

(LFP$_U$) $\langle\!\langle \varnothing\rangle\!\rangle\Box((\psi \vee (\phi \wedge \langle\!\langle A\rangle\!\rangle \bigcirc \theta)) \rightarrow \theta) \rightarrow \langle\!\langle \varnothing\rangle\!\rangle\Box(\langle\!\langle A\rangle\!\rangle\phi U\psi \rightarrow \theta)$

Axiomatization

Rules:

(Modus Ponens) $$\frac{\phi \qquad \phi \rightarrow \psi}{\psi}$$

($\langle\langle A \rangle\rangle \bigcirc$-Monotonicity) $$\frac{\phi \rightarrow \psi}{\langle\langle A \rangle\rangle \bigcirc \phi \rightarrow \langle\langle A \rangle\rangle \bigcirc \psi}$$

($\langle\langle A \rangle\rangle \Box$-Necessitation) $$\frac{\phi}{\langle\langle A \rangle\rangle \Box \phi}$$

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).

Next Week: ???