Game Theory with Costly Computation

Rafael Pass Joe Halpern

Cornell

A Computational Game

You are given random odd n-bit number (n big)

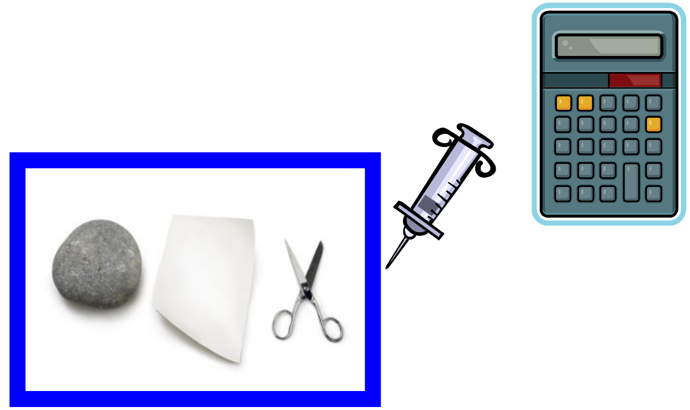
You either GIVE UP, or guess PRIME/NOT PRIME

Payoff:		
	GIVE UP	$10
	Correct answer	$1000
	Wrong answer	-$10000n

GT predicts

What do you do? Depends on cost of comp!

Add Computation into Game Theory



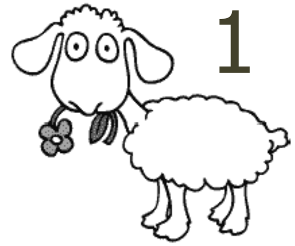
Idea goes back to **Herbert Simon** '55 "bounded rationality"

Two lines of study:

- restricted strategies of player [**Neyman'85**, MW'86, ..,PY'94, UV'99, DHR'00]
- charging for size of strategy [**Rubinstein'87**, …,BKK'06]

Our goal: provide a general model, investigate its properties

Games



1 …  m

Players 1,…,m

Available Actions — Rock | Paper | Scissors

Utility — $u_i(a_1,\ldots,a_m)$

Strategy σ — distribution over actions

Expected Utility — $U_i(\sigma_i,\sigma_{-i}) = \mathrm{Exp}[u_i(a)]$

Nash Eq $(\sigma_1, \ldots, \sigma_m)$ s.t $\forall i\ \forall \sigma'_i\ U_i(\sigma_i,\sigma_{-i}) \geq U_i(\sigma'_i,\sigma_{-i})$

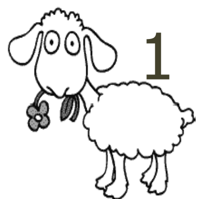
"If others are playing their strategies, I better stick to mine!"

Always exists! [N51]

Bayesian Games

Δ

Types (private information): t_1 … t_m



…

Strategy: $\sigma_1(t_1)$ … $\sigma_m(t_m)$

Actions: a_1 … a_m

Utility: $u_1(\vec{t},\vec{a})$ … $u_m(\vec{t},\vec{a})$

PRIME/NOT PRIME Game

t = odd n-bit number (n big)

Actions: GIVE UP, PRIME, NOT PRIME

Payoff:	GIVE UP	$10
	Correct answer	$1000
	Wrong answer	-$10000n

Only NE

Bayesian Machine Games

Strategy = randomized TM

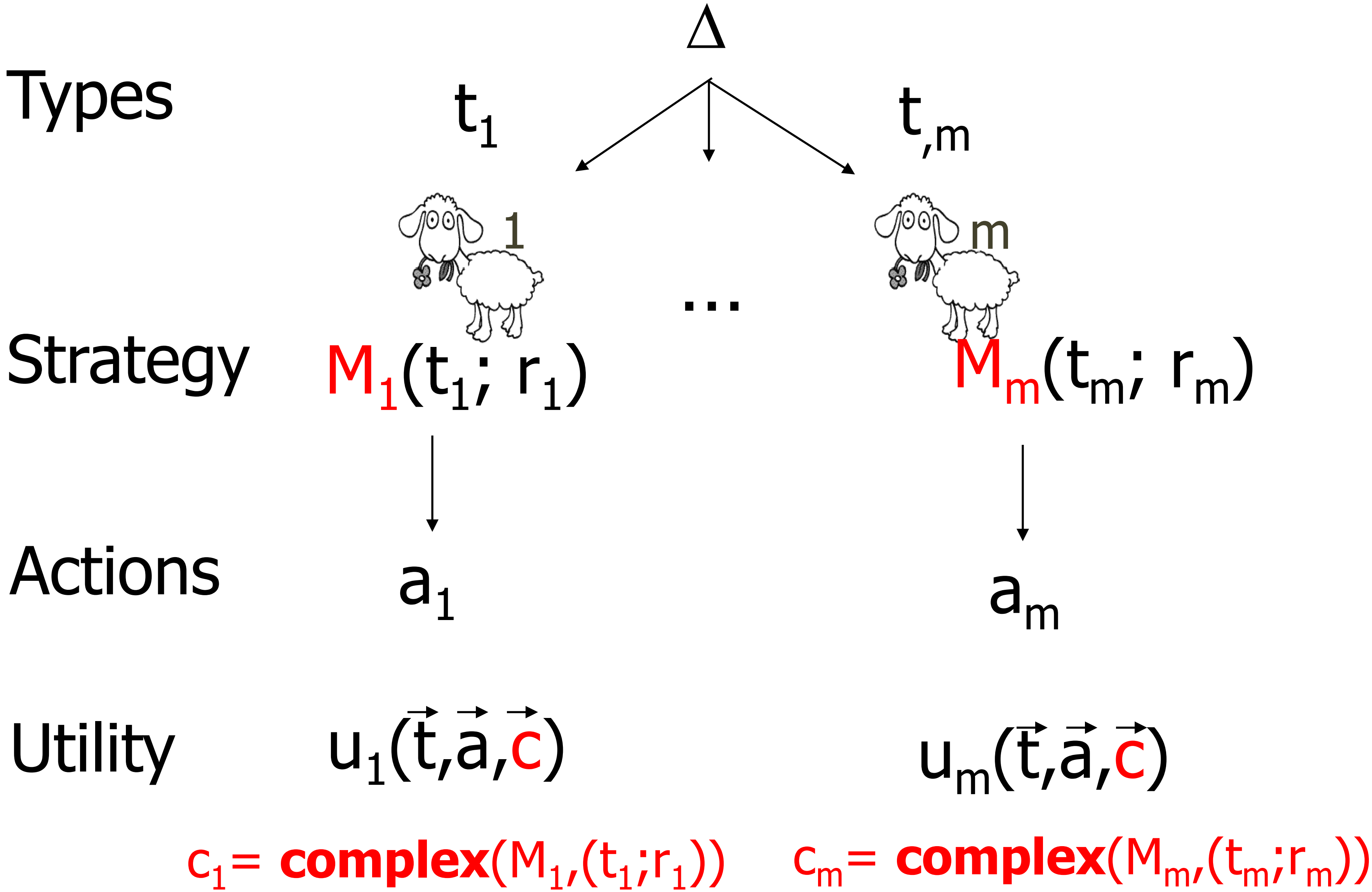
Complexity function complex: M x {0,1}* → N

- **complex**(M,v) = complexity of M on view v = t ; r
- (e.g., time, space, size, communication…)

Utility depends on:

- types
- actions
- complexities of machines

Bayesian Machine Game ([m],Δ,C,u):



Nash Eq in Machine Games

As before:

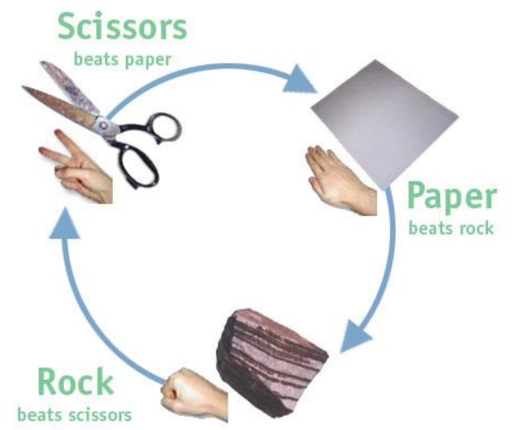
$(M_1, \ldots, M_m)$ s.t $\forall i \; \forall M'_i \; U_i(M_i, M_{-i}) \geq U_i(M'_i, M_{-i})$

But do they ALWAYS exist?

NO!

Roshambo with Costly Comp

Utility as usual but **subtract** # of coin tosses



	Rock	Paper	Scissors
Rock	(0,0)	(1,2)	(2,1)
Paper	(2,1)	(0,0)	(1,2)
Scissors	(1,2)	(2,1)	(0,0)

Assume exist NE such that is randomizing

$\Rightarrow$ exist det. strategy that does better

same original payoff, better complexity

Only possible NE is det.

But there are no deterministic NE! □

Strange? Natural?

The World Champion

Why? Coin tossing is hard?

Nash Eq in Machine Games

Thm: Every finite computable machine game (i.e., utilities and probabilities are computable) where "randomization is free" has a NE.

Main Lemma [Computational Analog of Nash Thm]:
Bayesian games where u_i and Δ are computable, have a NE that is implementable by randomized Turing Machines terminating with probability 1.

Explaining Observed Behavior

There exist many "paradoxical" games where traditional GT solutions concepts provide the "wrong" answer:

- repeated prisoner's dilemma [LR'51]
- first-impression's matter bias [R'99]
- belief polarization [LRL'79]
- use of "bad" randomness in sports competition [WW'01]
- …

One remedy: behavioral economics [KT'81]

Use psychology or models of brain to explain "irrational" behavior

SIMPLE assumptions about cost of computation suffice! (at a qualitative level)

Secure Computation [Y,GMW]

m parties, each with private input x_i

Goal: secure compute function F

Election:
$F(x_1, ..., x_m)$=tally

(M) securely computes F if it provides the same privacy and correctness guarantees as if a trusted party had computed F

- Formalized using **ZK simulation** [GMR]

Does this mean players want to run (M)?

Our Goal: capture intuition that

(M) securely implements F if

WHENEVER:

players **WANT** to compute F using their inputs

THEN:

players **WANT** to run (M) on their inputs

Our Goal: capture intuition that

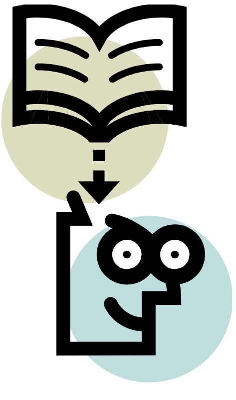
(M) securely implements F if

In every situation where:

players **WANT** to compute F using their inputs

It holds that:

players **WANT** to run (M) on their inputs



situation	=	game
WANT	=	"is a NE"

(M) **universally implements** F if:

In every machine game G where:

providing true input to a trusted party computing F and outputting what the trusted party replies, **is a NE in G.**

It holds that:

running (M) on true inputs **is a NE in G.**

Notes:

- similar to [Forges'87, ILM'06] but with **costly comp**
- guarantees that (M) does not "leak more" information than F

The Theorem

Tight connection between cryptographic notion of secure computation and universal implementation.

The notions are "essentially" equivalent.

Nash equilibrium and "**ZK simulation**"

are intimately connected

Framework for GT with Costly Computation

- Give simple computational explanations observed behavior in "paradoxical" games.
  - Can we use behavioral experiments to determine "cost of computation"?
- **Nash Equilibrium** v.s. **Sequential Equilibrium**
- We have assumed that players "**understand the game**" (i.e., they know how well a machine does, and what its complexity is).
  - Can also model players that have "beliefs" about how well a machine does.
  - But computing these beliefs might itself be costly!

GT definition of security

- "Equivalent" to secure computation in the most general setting,
- But helpful in circumventing lower-bound by considering restricted classes of games (e.g., players strictly prefer to computer less, or don't want to be caught cheating)

LOVE

- Alice and Bob want to find out it they love each other: compute AND of their inputs.
- Desiderata:
  - Want to know the output.
  - Don't want reveal input.
  - If I get the output, (slightly) prefer to trick you.
- [Shoham-Tennenholtz'03] **Impossible** even with a trusted party computing AND!
  - If I have input 0, always better to say 1 to trusted party (since I already know the output).

LOVE

- Use a crypto protocol, where players need to solve a "computational riddle" [DN] if they use input 1.
  - But still "indistinguishable" if you use input 0 or 1.
- Rational to provide true input if:
  - Cost of solving riddle > gain to trying to trick other player.
  - Value of Privacy > cost of solving riddle