# Games on Concurrent Processes: Epistemic Strategies

Sophia Knight
McGill University

# Outline

- Introduction: problems with traditional process algebra.
- Games and strategies
- Epistemic restrictions strategies
- Conclusions

# Goals

# Goals

- Use process algebra to model interacting agents in security protocols.

# Goals

- Use process algebra to model interacting agents in security protocols.
- Control what the agents know by restricting their allowed strategies.

# Goals

- Use process algebra to model interacting agents in security protocols.
- Control what the agents know by restricting their allowed strategies.
- Understand and control information flow.

# Problems with schedulers

# Problems with schedulers

- Nondeterminism is resolved by a scheduler.

# Problems with schedulers

- Nondeterminism is resolved by a scheduler.
- The scheduler is assumed to be omniscient.

# Problems with schedulers

- Nondeterminism is resolved by a scheduler.
- The scheduler is assumed to be omniscient.
- It is hard to **require** it to respect independence constraints without controlling it somehow.

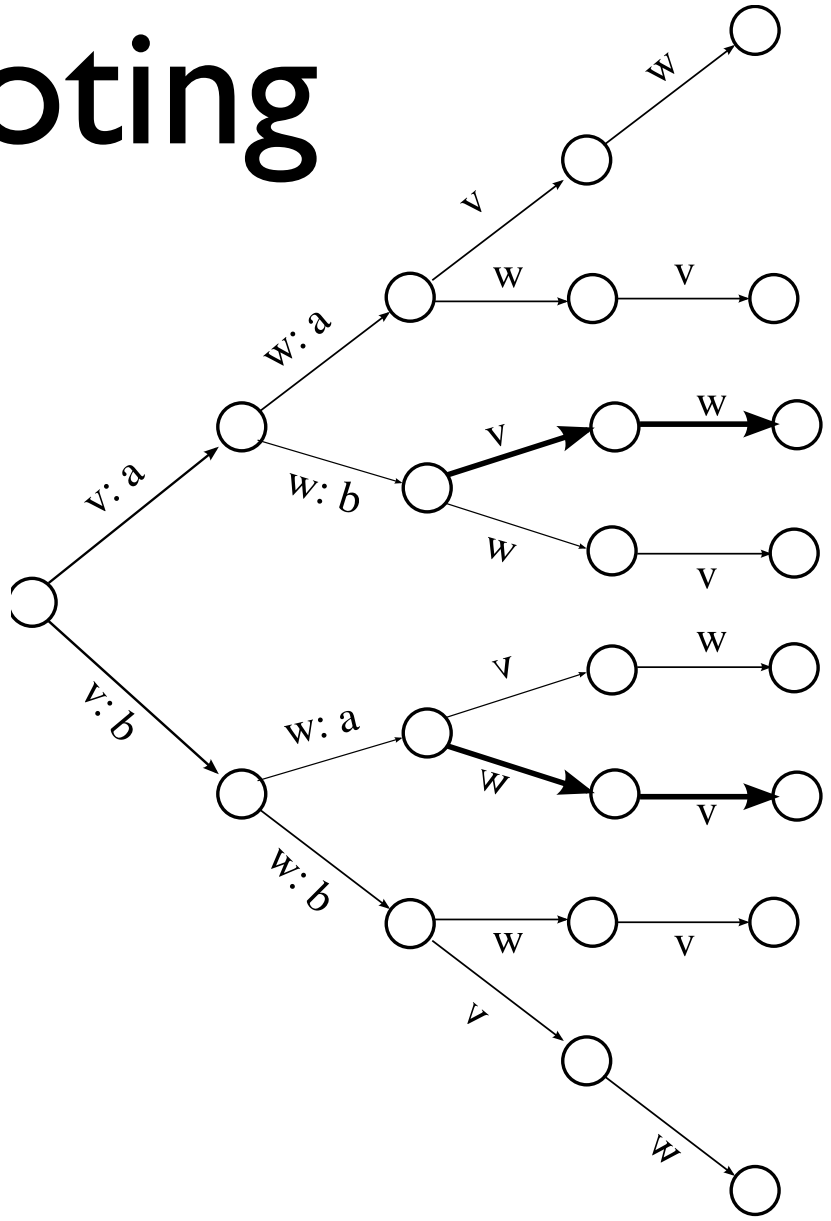
- A perverse scheduler can leak information to the outside world.

- A perverse scheduler can leak information to the outside world.
- Safety properties are usually required to hold with **universal quantification** over the possible schedulers,

- A perverse scheduler can leak information to the outside world.
- Safety properties are usually required to hold with **universal quantification** over the possible schedulers,
- so it is impossible to prove any reasonable security properties in this setting.

# Example: voting

- Two candidates: a,b
- Two electors: v,w
- Must output who voted
- But not for whom they voted
- Thick arrows show scheduler that violates anonymity.



# Processes with labels

| | |
|---|---|
| $a, b$ | actions |
| $\bar{a}, \bar{b}$ | co-actions |
| $\tau$ | silent action |
| $\alpha, \beta$ | generic actions, co-actions, or silent action |

$$P, Q ::= 0 \mid l : \alpha.P \mid P|Q \mid P + Q \mid (\nu a)P \mid l : \{P\}$$

# Operational Semantics

ACT $\dfrac{}{l{:}\alpha.P \xrightarrow{\alpha} P}$

RES $\dfrac{P \xrightarrow{\alpha} P' \qquad \alpha \neq a, \bar{a}}{(\nu a)P \xrightarrow{\alpha} (\nu a)P'}$

SUM1 $\dfrac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'}$

SUM2 $\dfrac{Q \xrightarrow{\alpha} Q'}{P+Q \xrightarrow{\alpha} Q'}$

PAR1 $\dfrac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}$

PAR2 $\dfrac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}$

COM $\dfrac{P \xrightarrow{a} P' \qquad Q \xrightarrow{\bar{a}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}$

SWITCH $\dfrac{P \xrightarrow{\tau} P'}{l{:}\{P\} \xrightarrow{\tau} P'}$

# The SWITCH rule

# The SWITCH rule

$$\text{SWITCH}\ \frac{P \xrightarrow{\tau} P'}{l{:}\{P\} \xrightarrow{\tau} P'}$$

# The SWITCH rule

$$\text{SWITCH}\ \frac{P \xrightarrow{\tau} P'}{l{:}\{P\} \xrightarrow{\tau} P'}$$

Represents the choices made independently of other choices in the process

# The SWITCH rule

$$\text{SWITCH}\ \frac{P \xrightarrow{\tau} P'}{l{:}\{P\} \xrightarrow{\tau} P'}$$

Represents the choices made independently of other choices in the process

Required to do a silent action because otherwise the outcome of the protected choice would be visible to the scheduler.

# Games

# Games

- A game is defined for each specific process: the process is the game board.

# Games

- A game is defined for each specific process: the process is the game board.
- Two-player games

# Games

- A game is defined for each specific process: the process is the game board.
- Two-player games
  - two players are sufficient to model interaction

# Games

- A game is defined for each specific process: the process is the game board.
- Two-player games
  - two players are sufficient to model interaction
  - the players are called X and Y

# Games

- A game is defined for each specific process: the process is the game board.
- Two-player games
  - two players are sufficient to model interaction
  - the players are called X and Y
- Players are independent and act according to their strategies.

# Games

- A game is defined for each specific process: the process is the game board.
- Two-player games
  - two players are sufficient to model interaction
  - the players are called X and Y
- Players are independent and act according to their strategies.
- Players interact to determine how process will execute.

# Valid Positions

# Valid Positions

- Players' moves are labels in the process.

# Valid Positions

- Players' moves are labels in the process.
- A string of allowable moves is called a valid position.

# Valid Positions

- Players' moves are labels in the process.
- A string of allowable moves is called a valid position.
- A valid position is like a trace, but with labels instead of actions.

# Valid Positions: Example

# Valid Positions: Example

$$P = l_1 : \{k_1 : \tau.l_3 : a + k_2 : \tau.l_3 : b\} + l_2 : c$$

# Valid Positions: Example

$$P = l_1 : \{k_1 : \tau.l_3 : a + k_2 : \tau.l_3 : b\} + l_2 : c$$

Valid positions:
$\{\varepsilon,\ l_1,\ l_2,\ l_1.k_1,\ l_1.k_2,\ l_1.k_1.l_3,\ l_1.k_2.l_3\}$

# Valid Positions: Example

$$P = l_1 : \{k_1 : \tau.l_3 : a + k_2 : \tau.l_3 : b\} + l_2 : c$$

Valid positions:
$\{\varepsilon,\ l_1,\ l_2,\ l_1.k_1,\ l_1.k_2,\ l_1.k_1.l_3,\ l_1.k_2.l_3\}$

We require the positions be *deterministically labelled*,
so that each valid position determines a unique execution.
There are never two identical labels available at the same step.

# Valid Positions: Example

$$P = l_1 : \{k_1 : \tau.l_3 : a + k_2 : \tau.l_3 : b\} + l_2 : c$$

Valid positions:
$\{\varepsilon,\ l_1,\ l_2,\ l_1.k_1,\ l_1.k_2,\ l_1.k_1.l_3,\ l_1.k_2.l_3\}$

We require the positions be *deterministically labelled*,
so that each valid position determines a unique execution.
There are never two identical labels available at the same step.

The set of valid positions is prefix closed.

# Strategies

# Strategies

**Definition**: In the game for $P$, a *strategy for player* $Z$ is a set $S$ of valid positions such that $\varepsilon \in S$ and if $s.m \in S$, then $m$ is a $Z$ move and every prefix of $s$ ending with a $Z$ move is in $S$.

# Strategies

**Definition**: In the game for $P$, a *strategy for player* $Z$ is a set $S$ of valid positions such that $\varepsilon \in S$ and if $s.m \in S$, then $m$ is a $Z$ move and every prefix of $s$ ending with a $Z$ move is in $S$.

A strategy tells the player what move to make in a possible partial execution of the process.

# An example strategy

A strategy for player $X$ for process $P$ is: $\{\varepsilon,\ l_1,\ l_1.k_1.l_3\}$.

# An example strategy

$$P = l_1 : \{k_1 : \tau.l_3 : a + k_2 : \tau.l_3 : b\} + l_2 : c$$

A strategy for player $X$ for process $P$ is: $\{\varepsilon,\ l_1,\ l_1.k_1.l_3\}$.

# Restrictions 1

**Definition** A strategy $S$ is *deterministic* if for all sequences $s$, $s.m_1 \in S$ and $s.m_2 \in S$ implies $m_1 = m_2$.

# Complete strategies

- We want some way of ensuring that the strategy tells the player what to do in every possible situation.
- This is formalized by the definition of complete strategy.
- The details are in the paper in the proceedings.

# Executions

# Executions

- A pair of complete, deterministic strategies

# Executions

- A pair of complete, deterministic strategies
- one for each player

# Executions

- A pair of complete, deterministic strategies
- one for each player
- defines an execution of the process.

# Epistemic restrictions

# Epistemic restrictions

- We define two equivalences on valid positions, one for each player.

# Epistemic restrictions

- We define two equivalences on valid positions, one for each player.
- These equivalences capture what players "know" in the usual (Kripke) way.

# Epistemic restrictions

- We define two equivalences on valid positions, one for each player.
- These equivalences capture what players "know" in the usual (Kripke) way.
- If $s_1$ and $s_2$ are equivalent for Z then $s_1.m$ is in Z's strategy if and only if $s_2.m$ is in the strategy.

# Epistemic restrictions

- We define two equivalences on valid positions, one for each player.
- These equivalences capture what players "know" in the usual (Kripke) way.
- If $s_1$ and $s_2$ are equivalent for Z then $s_1.m$ is in Z's strategy if and only if $s_2.m$ is in the strategy.
- We are saying that strategies can only be based on what players know.

# Epistemic restrictions

- We define two equivalences on valid positions, one for each player.
- These equivalences capture what players "know" in the usual (Kripke) way.
- If $s_1$ and $s_2$ are equivalent for Z then $s_1.m$ is in Z's strategy if and only if $s_2.m$ is in the strategy.
- We are saying that strategies can only be based on what players know.
- One can design different equivalences to "engineer" the appropriate epistemic concept.

# Introspection

# Introspection

- An example epistemic restriction: introspection.

# Introspection

- An example epistemic restriction: introspection.
- The player knows his own history and what moves were available to him at every point in the past.

# The main technical result

# The main technical result

- The introspective restriction exactly captures the independence requirement that one expects

# The main technical result

- The introspective restriction exactly captures the independence requirement that one expects
- In particular,

# The main technical result

- The introspective restriction exactly captures the independence requirement that one expects
- In particular,
- they are equivalent to the syntactic schedulers of Chatzikokolakis and Palamidessi.

# Conclusions

# Conclusions

- A semantic description of limitations of the power of agents.

# Conclusions

- A semantic description of limitations of the power of agents.
- A framework that can be used for other such limitations.

# Conclusions

- A semantic description of limitations of the power of agents.
- A framework that can be used for other such limitations.
- A better conceptual understanding of knowledge and interaction.