Distributed games with a central decision maker

Nehul Jain

IIT Bombay

FSTTCS 2025, 18th December 2025

Advisor: Bharat Adsul

Motivation: Distributed Synthesis

- Reactive synthesis: given a specification, synthesize a system that meets the spec. Active research area.
- Distributed synthesis: given a distributed spec, synthesize a distributed system that meets the spec.
  1. Two actively studied models
     - Petri games by Finkbeiner et al and others [FO17]
     - Asynchronous control games by Muscholl, Gimbert and others. [GGMW13]
  2. Several interesting (acyclic architectures) decidable cases. But undecidable in general!

Games on asynchronous transition systems with a Central Decision Maker

CDM systems

Distributed settings with a designated process, **c**entral **d**ecision **m**aker, which participates in all key decisions

- Server-client architectures – server maintains overall integrity
- Distributed version control systems which maintain a single master copy; Users make concurrent changes to local copies; Changes to the master copy are made by acquiring an exclusive access.
- Working of an organization with multiple agents but a designated head; Several committees are formed and work concurrently; Head is a member of all the decision-making committees.

The cdm participates in all decision making activities.

Distributed safety games with a CDM

- Distributed system is a **team** of processes vs Environment

Does there exist a “distributed co-operative strategy” for the NFA components to win?

Distributed safety games with a CDM

- Distributed system is a **team** of processes vs Environment
- Played on NFAs owned by the each processes which **communicate** via shared/joint actions

Does there exist a “distributed co-operative strategy” for the NFA components to win?

Distributed safety games with a CDM

- Distributed system is a **team** of processes vs Environment
- Played on NFAs owned by the each processes which **communicate** via shared/joint actions
- Actions

Does there exist a “distributed co-operative strategy” for the NFA components to win?

Distributed safety games with a CDM

- Distributed system is a **team** of processes vs Environment
- Played on NFAs owned by the each processes which **communicate** via shared/joint actions
- Actions
  - joint - a fixed set of processes/NFAs participate.

Does there exist a “distributed co-operative strategy” for the NFA components to win?

Distributed safety games with a CDM

- Distributed system is a **team** of processes vs Environment
- Played on NFAs owned by the each processes which **communicate** via shared/joint actions
- Actions
  - joint - a fixed set of processes/NFAs participate.
  - local - only one process participates.

Does there exist a "distributed co-operative strategy" for the NFA components to win?

Distributed safety games with a CDM

- Distributed system is a **team** of processes vs Environment
- Played on NFAs owned by the each processes which **communicate** via shared/joint actions
- Actions
  - joint - a fixed set of processes/NFAs participate.
  - local - only one process participates.
- Initial (global) state

Does there exist a "distributed co-operative strategy" for the NFA components to win?

Distributed safety games with a CDM

- Distributed system is a **team** of processes vs Environment
- Played on NFAs owned by the each processes which **communicate** via shared/joint actions
- Actions
  - joint - a fixed set of processes/NFAs participate.
  - local - only one process participates.
- Initial (global) state
- A winning condition

Does there exist a “distributed co-operative strategy” for the NFA components to win?

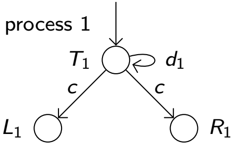
CDM Game: Example

CDM games

A CDM game is a distributed game with a designated cdm process ℓ such that if an action a is not deterministic, then ℓ participates in a.

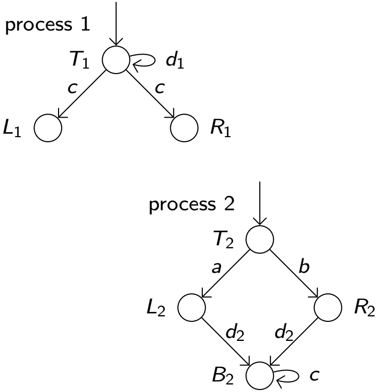
CDM games

A CDM game is a distributed game with a designated cdm process ℓ such that if an action a is not deterministic, then ℓ participates in a.



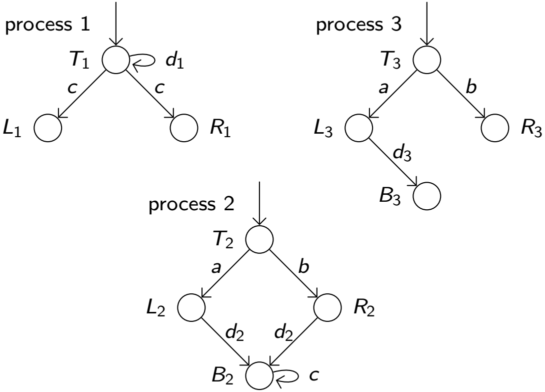
CDM games

A CDM game is a distributed game with a designated cdm process ℓ such that if an action a is not deterministic, then ℓ participates in a.



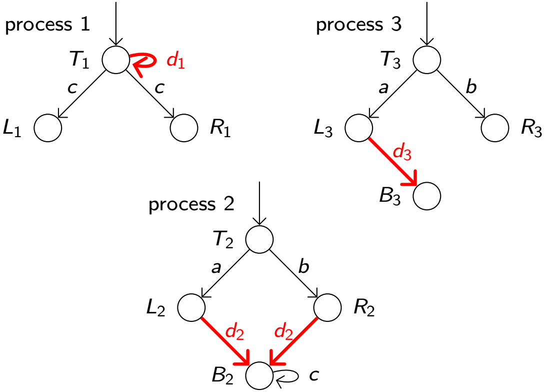
CDM games

A CDM game is a distributed game with a designated cdm process ℓ such that if an action a is not deterministic, then ℓ participates in a.



CDM games

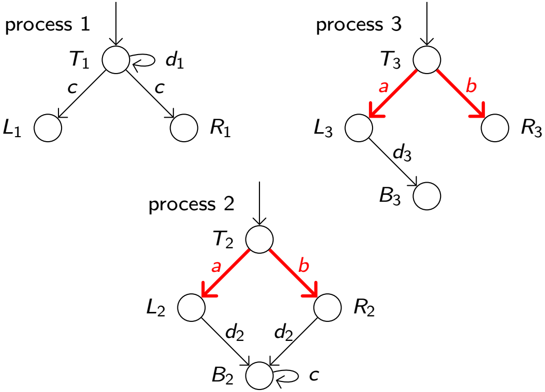
A CDM game is a distributed game with a designated cdm process ℓ such that if an action a is not deterministic, then ℓ participates in a.



- d_i is a local action of process i

CDM games

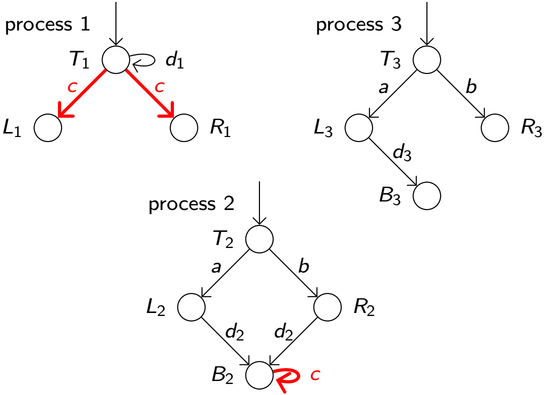
A CDM game is a distributed game with a designated cdm process ℓ such that if an action a is not deterministic, then ℓ participates in a.



- d_i is a local action of process i
- processes 2 and 3 share deterministic actions a and b

CDM games

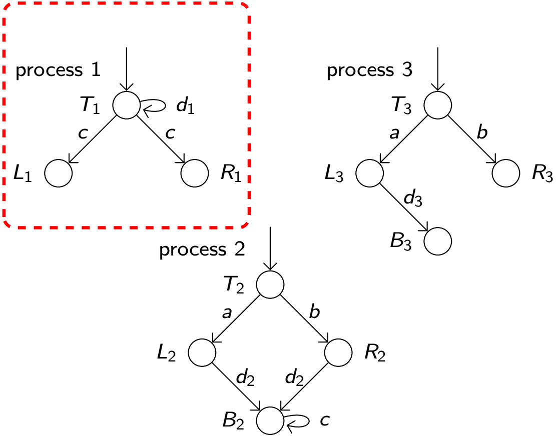
A CDM game is a distributed game with a designated cdm process ℓ such that if an action a is not deterministic, then ℓ participates in a.



- d_i is a local action of process i
- processes 2 and 3 share deterministic actions a and b
- processes 1 and 2 share non deterministic action c $(T_1, B_2) \xrightarrow{c} (L_1, B_2)$, $(T_1, B_2) \xrightarrow{c} (R_1, B_2)$

CDM games

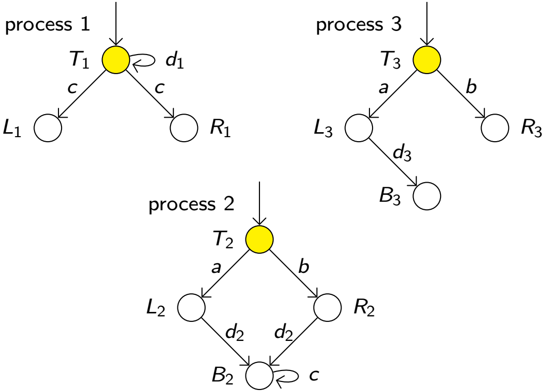
A CDM game is a distributed game with a designated cdm process ℓ such that if an action a is not deterministic, then ℓ participates in a.



- d_i is a local action of process i
- processes 2 and 3 share deterministic actions a and b
- processes 1 and 2 share non deterministic action c $(T_1, B_2) \xrightarrow{c} (L_1, B_2)$, $(T_1, B_2) \xrightarrow{c} (R_1, B_2)$
- 1 is the designated cdm

CDM games

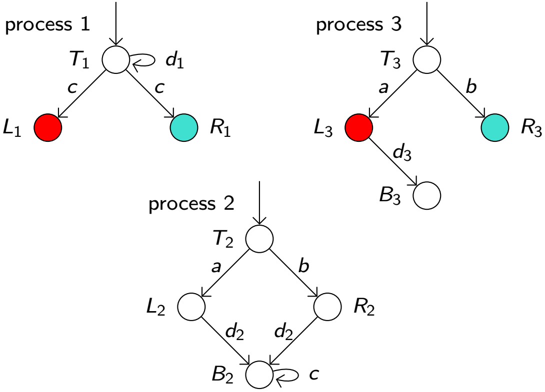
A CDM game is a distributed game with a designated cdm process ℓ such that if an action a is not deterministic, then ℓ participates in a.



- d_i is a local action of process i
- processes 2 and 3 share
  deterministic actions a and b
- processes 1 and 2 share
  non deterministic action c
  $(T_1, B_2) \xrightarrow{c} (L_1, B_2)$,
  $(T_1, B_2) \xrightarrow{c} (R_1, B_2)$
- 1 is the designated cdm
- Initial state: (T_1, T_2, T_3)

CDM games

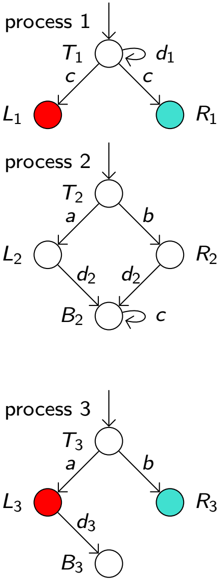
A CDM game is a distributed game with a designated cdm process ℓ such that if an action a is not deterministic, then ℓ participates in a.



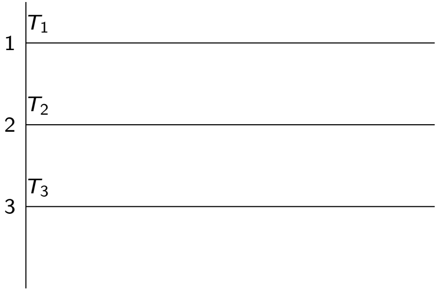
- d_i is a local action of process i
- processes 2 and 3 share deterministic actions a and b
- processes 1 and 2 share non deterministic action c $(T_1, B_2) \xrightarrow{c} (L_1, B_2)$, $(T_1, B_2) \xrightarrow{c} (R_1, B_2)$
- 1 is the designated cdm
- Initial state: (T_1, T_2, T_3)
- Unsafe set $\{(L_1, B_2, R_3), (R_1, B_2, L_3)\}$

Play: Example

Example: Play dynamics

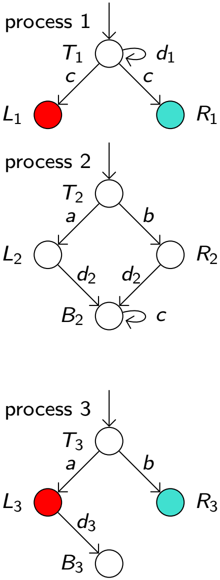


Example of a play from initial state (T_1, T_2, T_3)

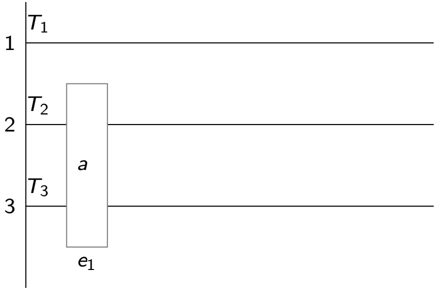


We illustrate how a play proceeds and how a safety winning condition is interpreted, using an example play

Example: Play dynamics

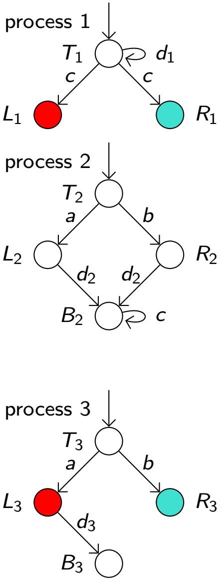


Example of a play from initial state (T_1, T_2, T_3)

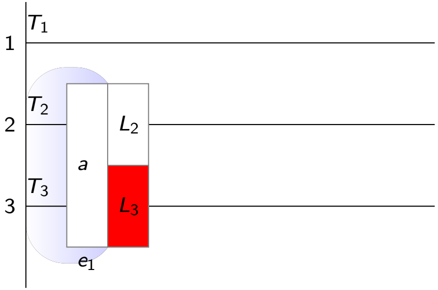


A play starts in the initial state. The environment can play an a or b. Say it plays action a

Example: Play dynamics

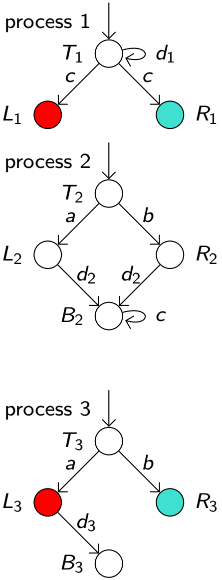


Example of a play from initial state (T_1, T_2, T_3)

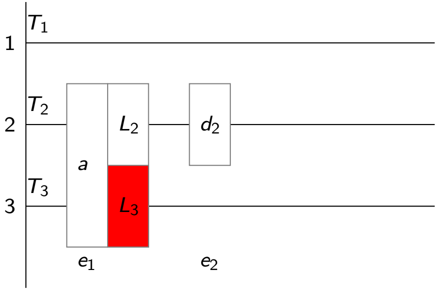


The participating processes respond by moving to the next state

Example: Play dynamics

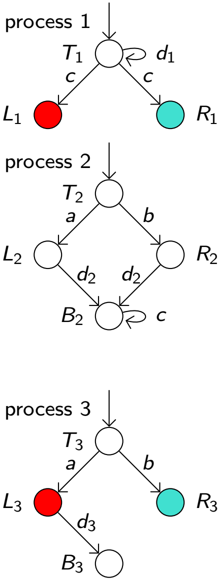


Example of a play from initial state (T_1, T_2, T_3)

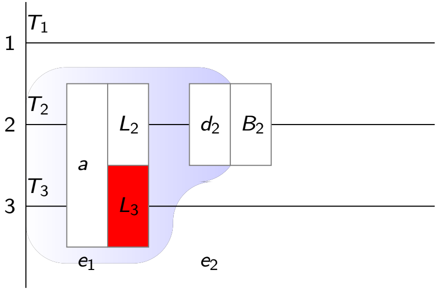


Next environment plays action d_2.

Example: Play dynamics

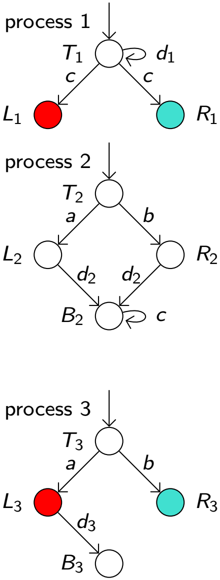


Example of a play from initial state (T_1, T_2, T_3)

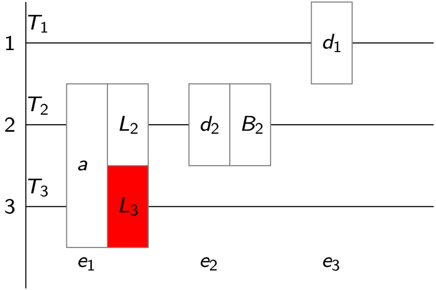


Process 2 participates in both e_1 and e_2; hence e_1 causally precedes e_2.

Example: Play dynamics

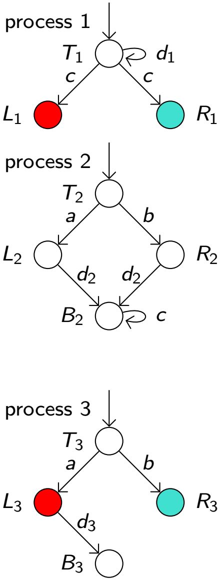


Example of a play from initial state (T_1, T_2, T_3)

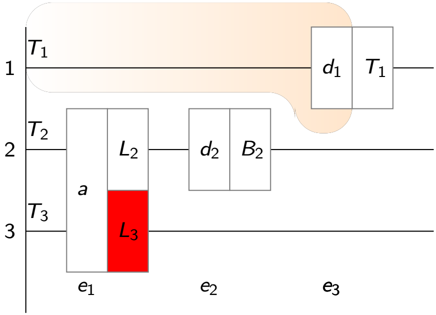


Next on d_1 at event e_3 only process 1 participates.

Example: Play dynamics

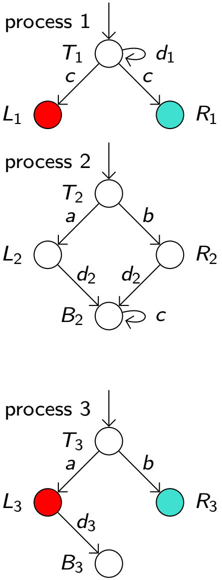


Example of a play from initial state (T_1, T_2, T_3)

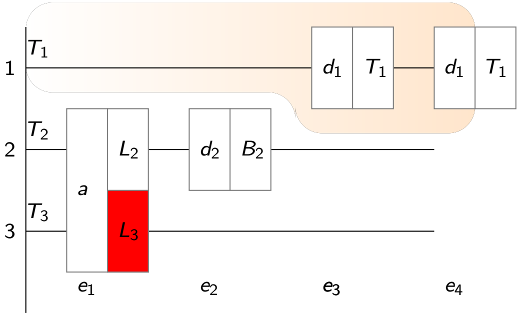


As Process 1 is unaware of e_1 and e_2, event e_3 is concurrent with them, yielding a partial order of events.

Example: Play dynamics

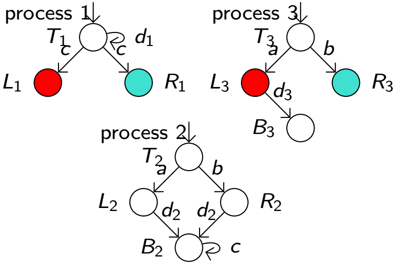


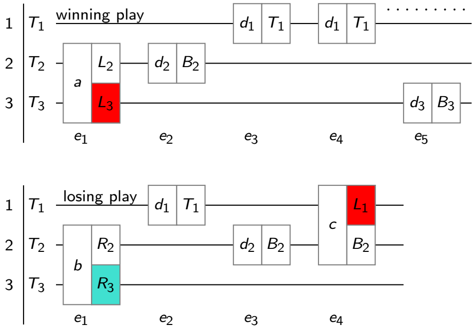
Example of a play from initial state (T_1, T_2, T_3)



And the play goes on

CDM game plays: Winning condition





A winning distributed strategy

A play can be modeled as a labeled partial order (Mazurkiewicz trace)
A distributed strategy is an advice function for the processes to decide matching transitions.

- On local actions, the local process can use its entire **causal past**
- On joint actions, participating processes exchange causal pasts and use this *entire* shared information.

A (finite-state) memory-based distributed strategy

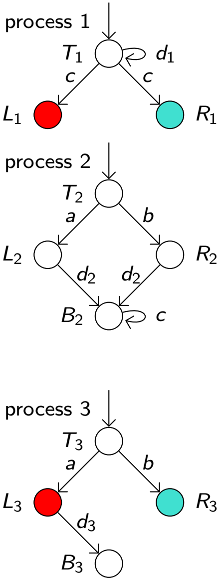
- maintains and updates the relevant key past *distributed* information
- the decisions are completely based on this distributed memory.

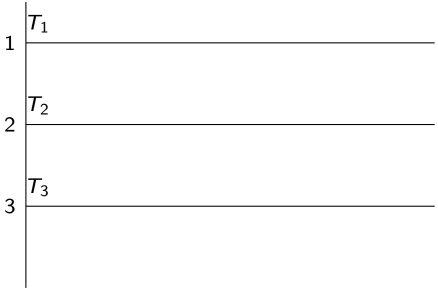
Problem Statement

Given a CDM safety game, decide if there exists a winning distributed strategy. Further, construct a memory-based winning strategy if one exists.

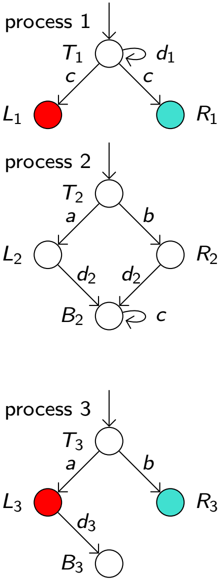
Strategy: Example

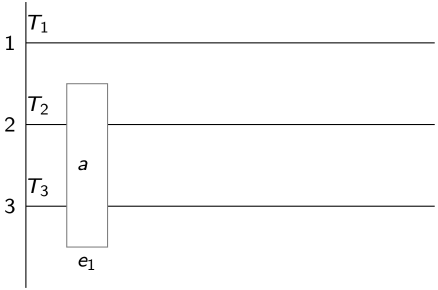
Example: A winning memory-based strategy



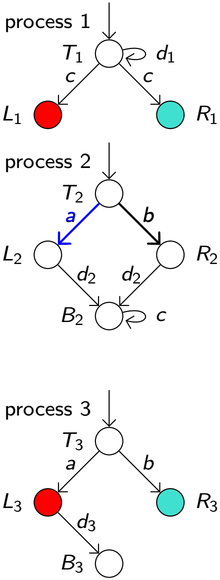


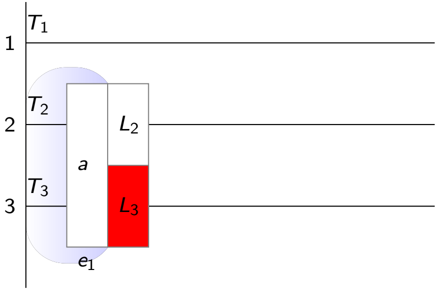
Example: A winning memory-based strategy





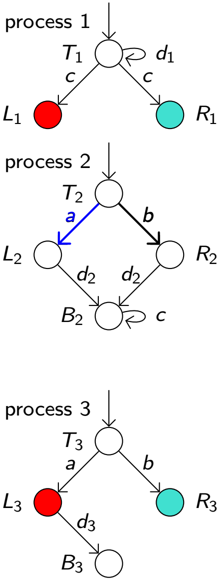
Example: A winning memory-based strategy

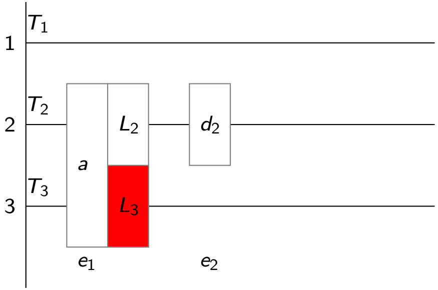




Process 2 remembers if the environment chose Left: a or Right: b

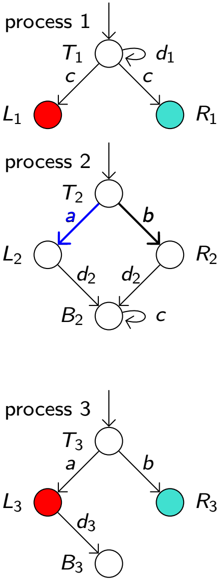
Example: A winning memory-based strategy

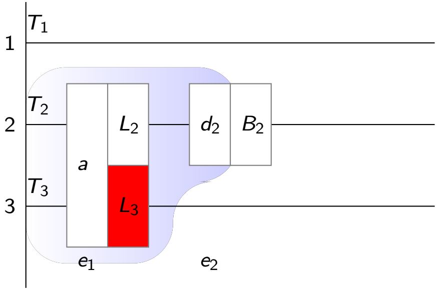




Process 2 remembers if the environment chose Left: a or Right: b

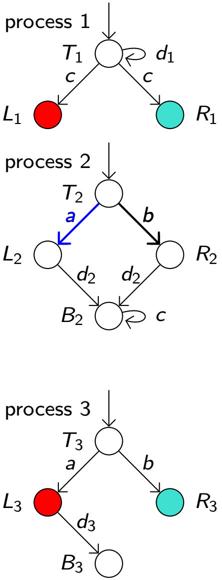
Example: A winning memory-based strategy

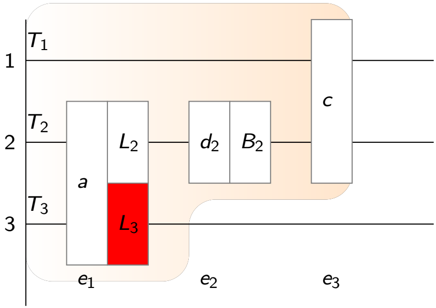




Process 2 remembers if the environment chose Left: a or Right: b

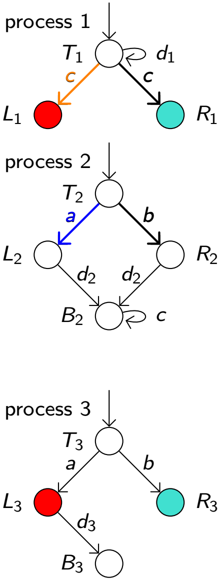
Example: A winning memory-based strategy

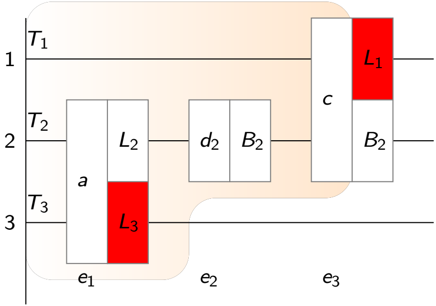




Process 2 remembers if the environment chose Left: a or Right: b

Example: A winning memory-based strategy

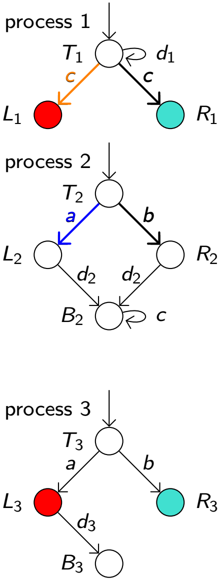


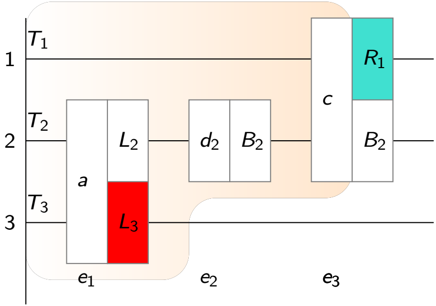


Process 2 remembers if the environment chose Left: a or Right: b

Process 1 goes left

Example: A winning memory-based strategy





Process 2 remembers if the environment chose Left: a or Right: b

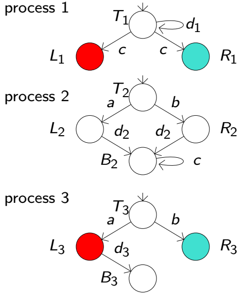
Process 1 goes left

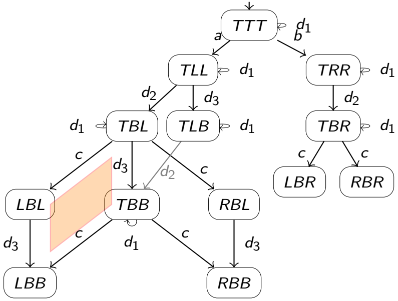
Solution approach

Solution approach - sequential game

Given a safety CDM game G, construct a sequential safety game G_{seq} on the global-states of G

Derived sequential game G_{seq}



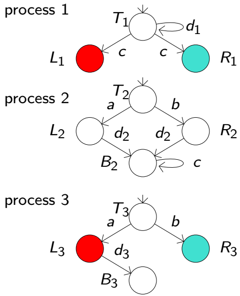


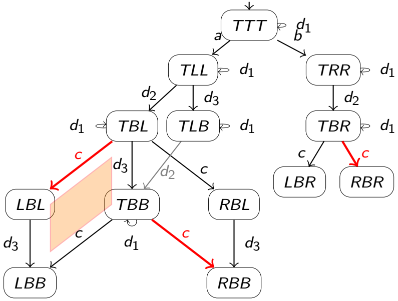
easy: Distributed strategy ——> Sequential strategy

Solution approach - sequential game

Given a safety CDM game G, construct a sequential safety game G_{seq} on the global-states of G

Derived sequential game G_{seq}





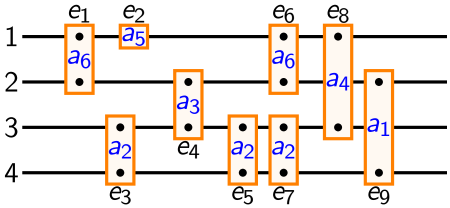
????: Distributed strategy <—— Sequential strategy

Solution approach - Linearization

Given a safety CDM game G, construct a sequential safety game G_{seq} on the global-states of G

There is a distributed winning strategy in G iff there is a sequential winning strategy in G_{seq}.

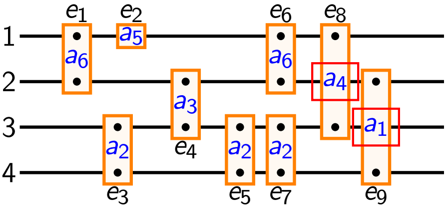
Key idea: given a seq winning strategy in G_{seq}, extract a distributed winning strategy in G. To achieve this, we develop special linearizations of plays/process-diagrams/traces in which cdm events appear the earliest and on any trace we copy the system move on this linearization.



$$e_1 e_2 e_3 e_4 e_6 e_5 e_7 e_8 e_9 = a_6 a_5 a_2 a_3 a_6 a_2 a_2 a_4 a_1$$

Special Linearization - Definition

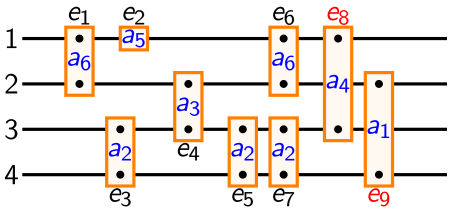
- Fix a total order $\lhd_\Sigma$ on Σ such that $\Sigma \setminus \Sigma_1 \lhd_\Sigma \Sigma_1$.



Process 1 participates in a_4 but not in a_1: $a_1 \lhd_\Sigma a_4$

Special Linearization - Definition

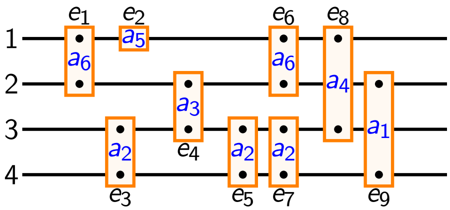
- Fix a total order $\lhd_\Sigma$ on Σ such that $\Sigma \setminus \Sigma_1 \lhd_\Sigma \Sigma_1$.
- To compute linearization $\mathrm{Lin}(t)$ of a play t, start from the **maximal** (last) actions:



maximal events: e_8, e_9

Special Linearization - Definition

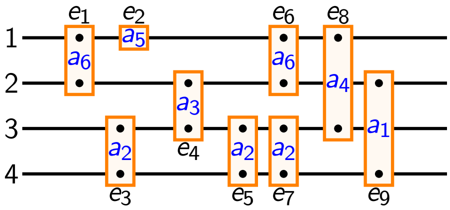
- Fix a total order $\lhd_\Sigma$ on Σ such that $\Sigma \setminus \Sigma_1 \lhd_\Sigma \Sigma_1$.
- To compute linearization $\mathrm{Lin}(t)$ of a play t, start from the **maximal** (last) actions:
- Peel off the $\lhd_\Sigma$-**least** event e_9 among the maximal events.



$\lhd_\Sigma$-least among the maximal events e_9

Special Linearization - Definition

- Fix a total order $\triangleleft_\Sigma$ on Σ such that $\Sigma \setminus \Sigma_1 \triangleleft_\Sigma \Sigma_1$.
- To compute linearization $\mathrm{Lin}(t)$ of a play t, start from the **maximal** (last) actions:
- Peel off the $\triangleleft_\Sigma$-**least** event e_9 among the maximal events.
- Define the linearization recursively: $\mathrm{Lin}(t) = \mathrm{Lin}(t \setminus e_9)\,\lambda(e_9)$



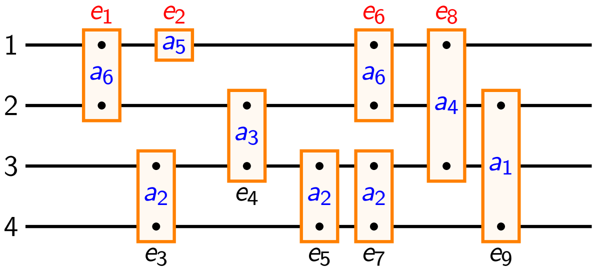
Special Linearization: Property

Resulting property

- Let t be a play and e its event where process 1 participates. Then

$$\mathrm{Lin}(\textbf{causal past of } e) \sqsubseteq \mathrm{Lin}(t)$$

process 1 is the cdm



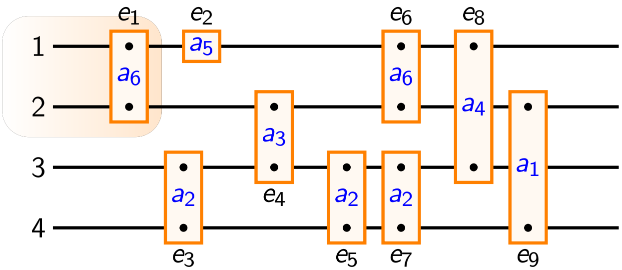
Special Linearization: Property

Resulting property

- Let t be a play and e its event where process 1 participates. Then

$$\mathrm{Lin}(\textbf{causal past of } e) \sqsubseteq \mathrm{Lin}(t)$$

process 1 is the cdm



e_1

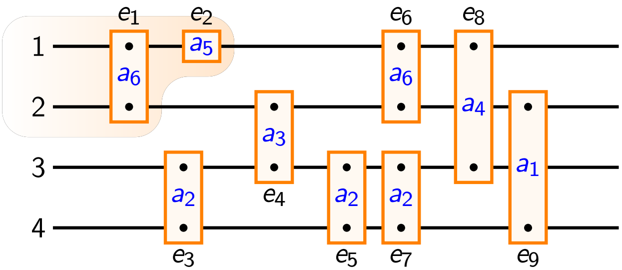
Special Linearization: Property

Resulting property

- Let t be a play and e its event where process 1 participates. Then

$$\mathrm{Lin}(\textbf{causal past of } e) \sqsubseteq \mathrm{Lin}(t)$$

process 1 is the cdm



e_1 e_2

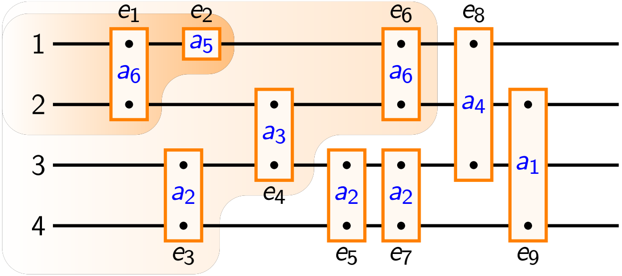
Special Linearization: Property

Resulting property

- Let t be a play and e its event where process 1 participates. Then

$$\mathrm{Lin}(\textbf{causal past of } e) \sqsubseteq \mathrm{Lin}(t)$$

process 1 is the cdm



e_1 e_2 e_3 e_4 e_6

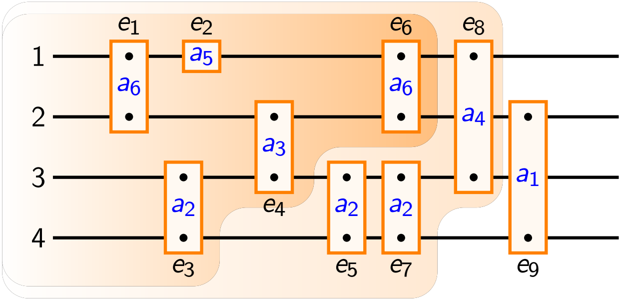
Special Linearization: Property

Resulting property

- Let t be a play and e its event where process 1 participates. Then

$$\mathrm{Lin}(\textbf{causal past of } e) \sqsubseteq \mathrm{Lin}(t)$$

process 1 is the cdm



e_1 e_2 $e_3 e_4 e_6$ $e_5 e_7 e_8$

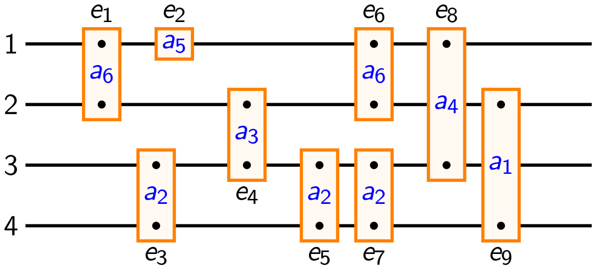
Special Linearization: Property

Resulting property

- Let t be a play and e its event where process 1 participates. Then

$$\mathrm{Lin}(\textbf{causal past of } e) \sqsubseteq \mathrm{Lin}(t)$$

process 1 is the cdm



e_1 e_2 $e_3 e_4 e_6$ $e_5 e_7 e_8$ e_9

Distributed strategy and Why this works

Given τ in G_{seq} we define $\hat{\tau}$ in G at trace t as:

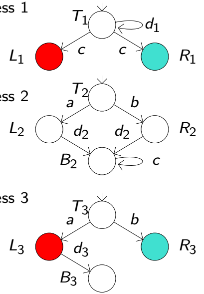
$$\hat{\tau}(t) = \tau(\mathrm{Lin}(t))$$

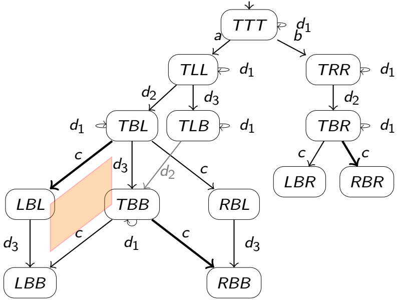
We prove that the system choices still respect the transition system of the distributed game.

- The cdm events appear the earliest, so on cdm events distributed strategy uses linearization of the causal past that is actually available.
- The CDM choices on each CDM event get extended as the plays continues
- When a trace gets extended by a
  - deterministic action: no matter what information the sequential game and distributed game offer the same unique next move is available
  - non deterministic action: the choice made by the distributed strategy copied from sequential strategy remains valid even after some concurrent actions have been played.

An illustration: When a trace gets extended by a non deterministic action

Derived sequential game G_{seq}

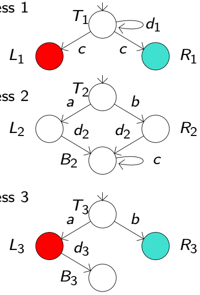


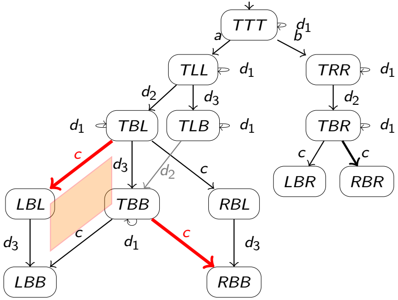


FSTTCS 2025, 18th December 2025

An illustration: When a trace gets extended by a non deterministic action

Derived sequential game G_{seq}

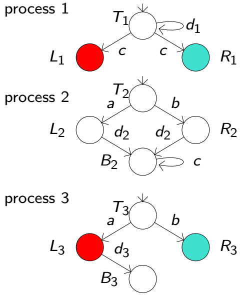


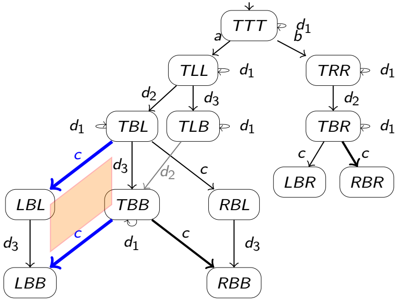


Sequential strategy

An illustration: When a trace gets extended by a non deterministic action

Derived sequential game G_{seq}





Derived distributed strategy: On non deterministic action c the choice made at state TBL remains valid even after concurrent action d_3 have been played.

Finite-state distributed strategy

Finite-state distributed strategies

Let us fix a sequential zero-memory strategy τ in G_{seq}. What is the distributed memory required by the extracted distributed strategy $\hat{\tau}$ in G?

When cdm needs to respond, it computes the effect of τ on the special linearization of its strict causal past. As τ is zero-memory, this amounts to the computation of the best global-state that the cdm is aware of.

So, every process keeps track of the best global-state that they are aware of. In order to update this information on a synchronization, they need to decide, for every other process k, which of the synchronizing process has the latest information about process k.

Finite-state distributed strategies

Let us fix a sequential zero-memory strategy τ in G_{seq}. What is the distributed memory required by the extracted distributed strategy $\hat{\tau}$ in G?

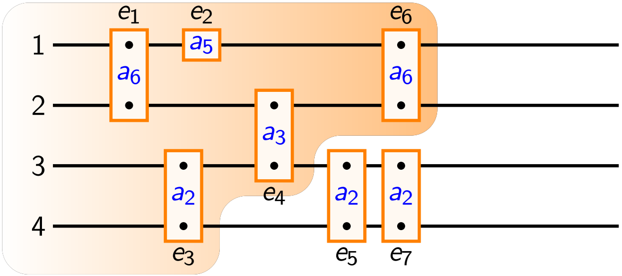
When cdm needs to respond, it computes the effect of τ on the special linearization of its strict causal past. As τ is zero-memory, this amounts to the computation of the best global-state that the cdm is aware of.

So, every process keeps track of the best global-state that they are aware of. In order to update this information on a synchronization, they need to decide, for every other process k, which of the synchronizing process has the latest information about process k.

The gossip asynchronous automaton (Mukund-Sohoni)[MS97] essentially solves the same problem and can be used to update the latest global-state

Computing the latest global state

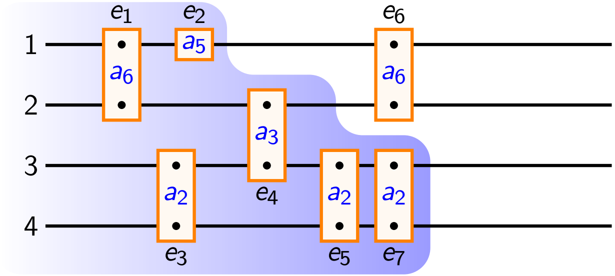
process 1 is the cdm



In an ongoing play
Causal past of process 1
Process 1 tracks global state here

Computing the latest global state

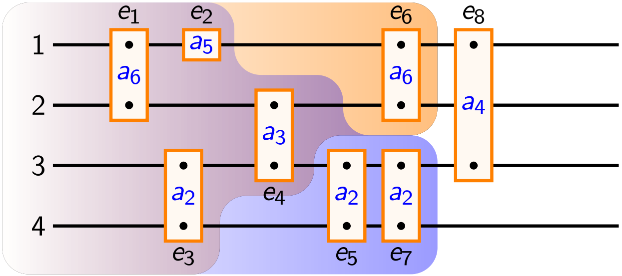
process 1 is the cdm



Causal past of process 3
Process 3 tracks global state here

Computing the latest global state

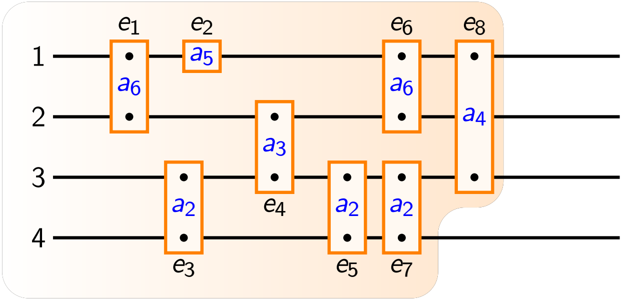
process 1 is the cdm



Who knows better about process 2: process 1(e_6)
Who knows better about process 4: process 3(e_7)

Computing the latest global state

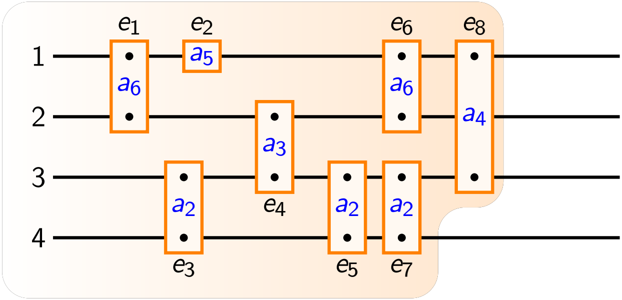
process 1 is the cdm



The gossip asynchronous automaton (Mukund-Sohoni)[MS97] essentially solves this problem and can be used to update the latest global-state

Computing the latest global state

process 1 is the cdm



Thus, Process 1 and 3 update to the next global state they keep track of

Safety CDM games results

Decision complexity

Safety CDM games are EXPTIME-complete

Memory complexity

For a YES-instance, there exists a finite-state distributed winning strategy wherein each process essentially keeps track of the best global-state that it is aware of.

Suppose each process has atmost m local states and there are n processes. So, overall there are m^n global-states and each process needs exponential (in n) many local memory-states.

We show that this exponential dependence on the number of processes is necessary.

More decision makers?

Safety games with two decision makers are undecidable.

We heavily reuse ideas from the recent work (Hugo Gimbert)[Gim22] which showed that asynchronous control games are undecidable.

Asynchronous control games and our games are equivalent.

Summary

- Our model: games on asynchronous transition systems
- Special cases: decision complexity and memory complexity
  1. Safety central decision maker games
  2. CDM parity in CDM games

References I

Bernd Finkbeiner and Ernst-Rüdiger Olderog, *Petri games: Synthesis of distributed systems with causal memory*, Information and Computation **253** (2017), 181–203.

Blaise Genest, Hugo Gimbert, Anca Muscholl, and Igor Walukiewicz, *Asynchronous games over tree architectures*, Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II (Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, eds.), Lecture Notes in Computer Science, vol. 7966, Springer, 2013, pp. 275–286.

Hugo Gimbert, *Distributed asynchronous games with causal memory are undecidable*, Logical Methods in Computer Science **Volume 18, Issue 3** (2022).

References II

Madhavan Mukund and Milind A. Sohoni, *Keeping track of the latest gossip in a distributed system*, Distributed Computing **10** (1997), no. 3, 137–148.

Thank you

Lower memory bound

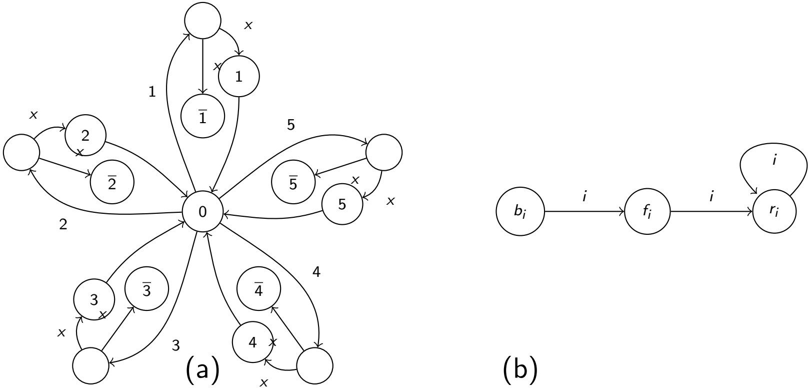
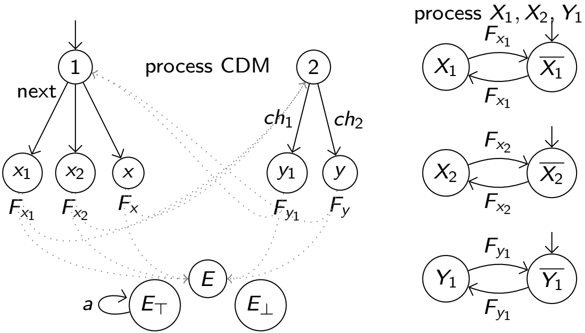


Figure 1: Memory lower bound: The pairing $(\bar{i}, f_i)$ and (i, r_i) in any global state makes it unsafe

EXP lower bound for CDM global safety and local parity

G_5: A position is a triple $(\tau, F(X, Y), \alpha)$ where $\tau \in 1, 2$, F is a formula in CNF whose variables have been partitioned into disjoint sets X, Y and a is n assignment for the set of variables $V(F)$ in F. Player $I(II)$ moves by changing at most one variable in $X(Y)$: passing is allowed. Player I wins if the formula F is ever true. we define game $\overline{G_5}$ when Player I wins if the formula F is never true.



$$(E, x_1, y_1, \vee_0) \xrightarrow{\vee} (E, x_1, y_1, \vee)$$
$$(E, \overline{x_1}, y_1, \vee_0) \xrightarrow{\vee} (E, \overline{x_1}, y_1, \vee)$$
$$(E, x_1, \overline{y_1}, \vee_0) \xrightarrow{\vee} (E, x_1, \overline{y_1}, \vee)$$
$$(E, \overline{x_1}, \overline{y_1}, \vee_0) \xrightarrow{\vee} (E, \overline{x_1}, \overline{y_1}, \overline{\vee})$$

$$(E, \vee, x_2, \wedge_0) \xrightarrow{\wedge} (E_\top, \vee, x_2, \wedge)$$
$$(E, \overline{\vee}, x_2, \wedge_0) \xrightarrow{\wedge} (E_\bot, \overline{\vee}, x_2, \overline{\wedge})$$
$$(E, \vee, \overline{x_2}, \wedge_0) \xrightarrow{\wedge} (E_\bot, \vee, \overline{x_2}, \overline{\wedge})$$
$$(E, \overline{\vee}, \overline{x_2}, \wedge_0) \xrightarrow{\wedge} (E_\bot, \overline{\vee}, \overline{x_2}, \overline{\wedge})$$

Figure 2: *$\overline{G_5}$ reduces to global safety game with local unsafe state $E_\top$ and G_5 reduces to parity game with coloring $\chi(E_\top) = 2$ and if $s_1 \neq E_\top$ then $\chi(s_1) = 1$. Formula is $(x_1 \vee y_1) \wedge x_2$.*