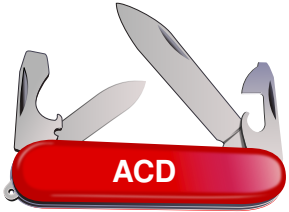
Practical Applications of the Alternating Cycle Decomposition

Antonio Casares Alexandre Duret-Lutz
Klara J. Meyer Florian Renkin Salomon Sickert



TACAS'22 — April 7

Alternating Cycle Decomposition

[ICALP'21]

Optimal Transformations of Games and Automata Using Muller Conditions

Antonio Casares
LaBRI, Université de Bordeaux, France

Thomas Colcombet
CNRS, IRIF, Université de Paris, France

Nathanaël Fijalkow
CNRS, LaBRI, Université de Bordeaux, France
The Alan Turing Institute of Data Science, London, UK

Abstract

We consider the following question: given an automaton or a game with a Muller condition, how can we efficiently construct an equivalent one with a parity condition? There are several examples of such transformations in the literature, including in the determinisation of Büchi automata.

We define a new transformation called the alternating cycle decomposition, inspired and extending Zielonka's construction. Our transformation operates on transition systems, encompassing both automata and games, and preserves semantic properties through the existence of a locally bijective morphism. We show a strong optimality result: the obtained parity transition system is minimal both in number of states and number of priorities with respect to locally bijective morphisms.

We give two applications: the first is related to the determinisation of Büchi automata, and the second is to give crisp characterisations on the possibility of relabelling automata with different acceptance conditions.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects

Keywords and phrases Automata over infinite words, Omega regular languages, Determinisation of

- Defines the ACD structure of Muller automata.
- How to use ACD to paritize an automaton. (With an optimality result.)
- How to use ACD to check automaton types.
- Purely theoretical (no implementation).

doi

Alternating Cycle Decomposition

[ICALP'21]

Optimal Transformations of Games and Automata Using Muller Conditions

Antonio Casares
LaBRI, Université de Bordeaux, France

Thomas Colcombet
CNRS, IRIF, Université de Paris, France

Nathanaël Fijalkow
CNRS, LaBRI, Université de Bordeaux, France
The Alan Turing Institute of Data Science, London, UK

Abstract

We consider the following question: given an automaton or a game with a Muller condition, how can we efficiently construct an equivalent one with a parity condition? There are several examples of such transformations in the literature, including in the determinisation of Büchi automata.

We define a new transformation called the alternating cycle decomposition, inspired and extending Zielonka's construction. Our transformation operates on transition systems, encompassing both automata and games, and preserves semantic properties through the existence of a locally bijective morphism. We show a strong optimality result: the obtained parity transition system is minimal both in number of states and number of priorities with respect to locally bijective morphisms.

We give two applications: the first is related to the determinisation of Büchi automata, and the second is to give crisp characterisations on the possibility of relabelling automata with different acceptance conditions.

2012 ACM Subject Classification Theory of computation → Automata over infinite objects

Keywords and phrases Automata over infinite words, Omega regular languages, Determinisation of

doi

- Defines the ACD structure of Muller automata.
- How to use ACD to paritize an automaton. (With an optimality result.)
- How to use ACD to check automaton types.
- Purely theoretical (no implementation).

But is this ACD construction practical?

This Paper

- Adaptation of the definition of ACD and `acd_transform()` to TELA (Transition-based Emerson-Lei Automata, as supported by the HOA format.)
- Implementation in two tools:

Owl 21.0
`owl.model.in.tum.de`

Spot 2.10
`spot.lrde.epita.fr`

Motivation is LTL synthesis with: LTL→DTELA→ DPA→game→controller

paritization

This Paper

- Adaptation of the definition of ACD and `acd_transform()` to TELA (Transition-based Emerson-Lei Automata, as supported by the HOA format.)
- Implementation in two tools:

Owl 21.0
`owl.model.in.tum.de`

Spot 2.10
`spot.lrde.epita.fr`

Motivation is LTL synthesis with: LTL→DTELA→ DPA→game→controller

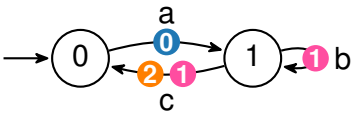
- Comparison to other existing paritization procedures
- State-based version of `acd_transform()`
- Comparison to degeneralization procedures (transition-based generalized Büchi → state-based Büchi)

Emerson-Lei Automata (Using the HOA Syntax)

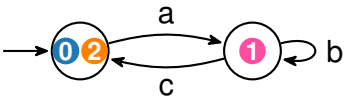
Acceptance condition = any positive Boolean combination of Inf(n) or Fin(n) terms.

Büchi	Inf(0)
generalized Büchi	Inf(0) ∧ Inf(1) ∧ Inf(2) ∧ ...
co-Büchi	Fin(0)
generalized co-Büchi	Fin(0) ∨ Fin(1) ∨ Fin(2) ∨ ...
Rabin	(Fin(0) ∧ Inf(1)) ∨ (Fin(2) ∧ Inf(3)) ∨ ...
Streett	(Inf(0) ∨ Fin(1)) ∧ (Inf(2) ∨ Fin(3)) ∧ ...
parity min even	Inf(0) ∨ (Fin(1) ∧ (Inf(2) ∨ (Fin(3) ∧ ...)))
parity min odd	Fin(0) ∧ (Inf(1) ∨ (Fin(2) ∧ (Inf(3) ∨ ...)))

transition-based acceptance (TELA):

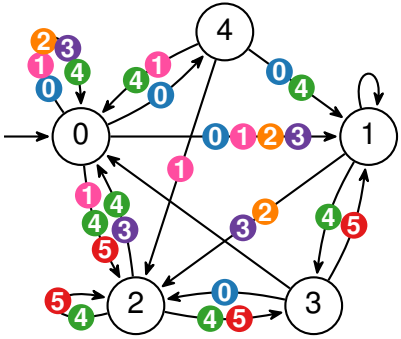


state-based acceptance:



TELA

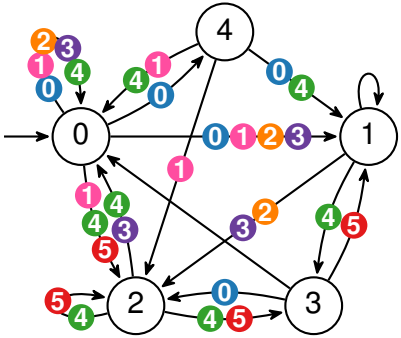
$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$

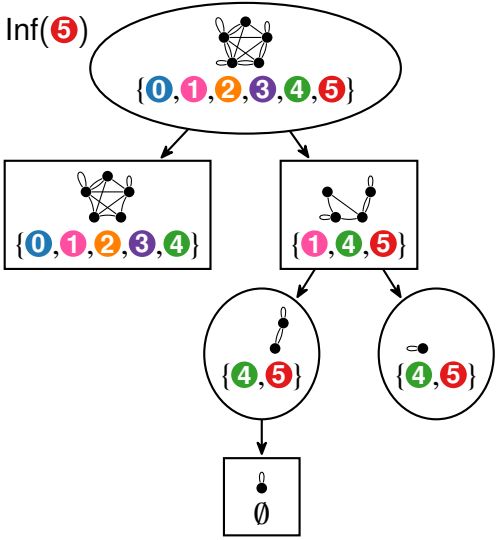


Automaton labels/alphabet hidden for simplicity.

ACD for TELA

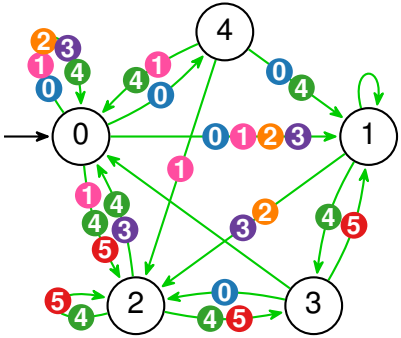
$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$

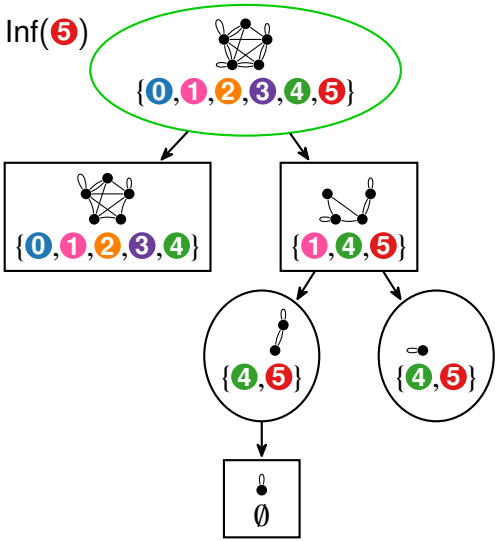




ACD for TELA

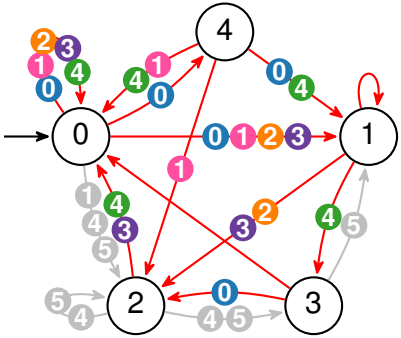
$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$

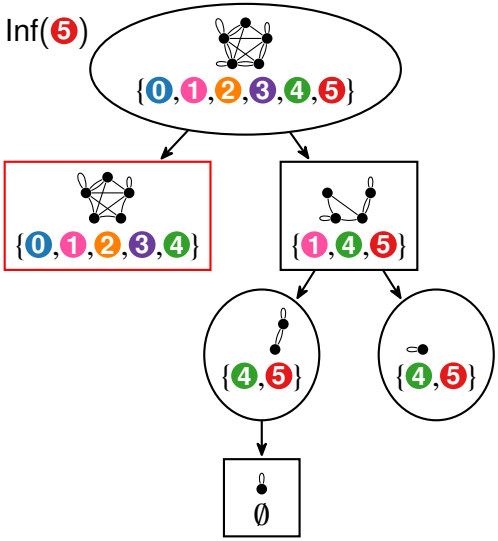




ACD for TELA

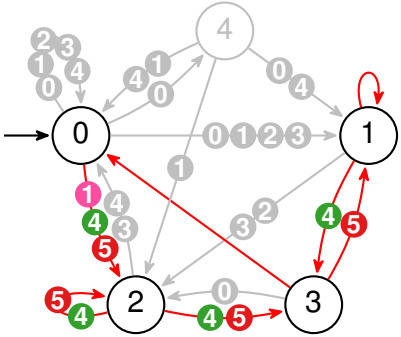
$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$

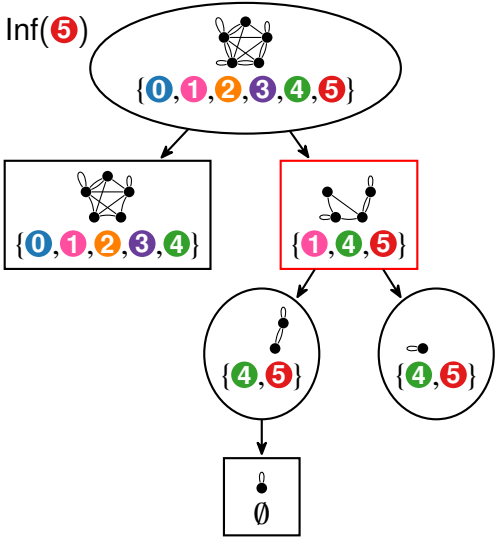




ACD for TELA

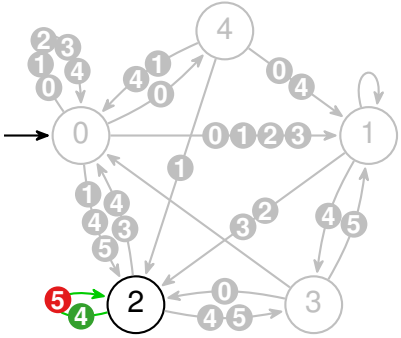
$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$

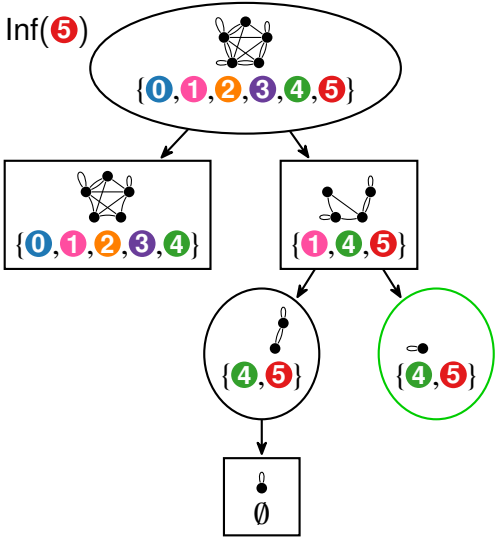




ACD for TELA

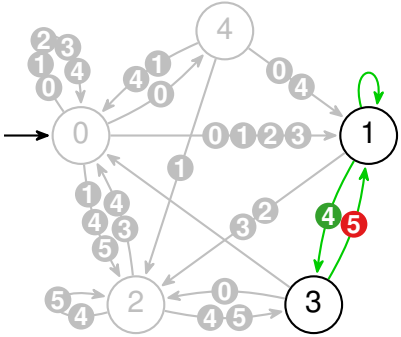
$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$

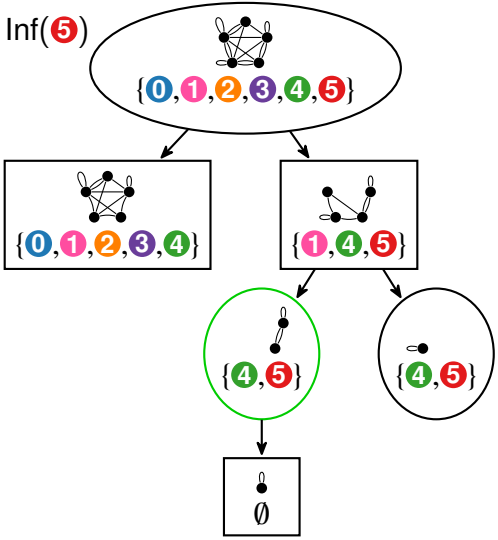




ACD for TELA

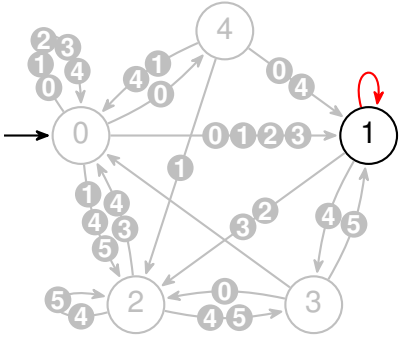
$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$

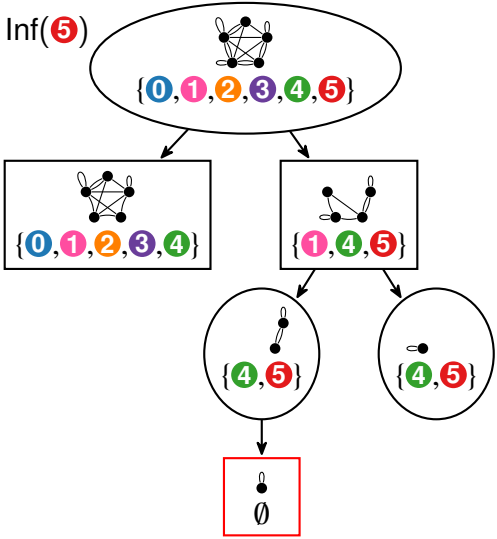




ACD for TELA

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$

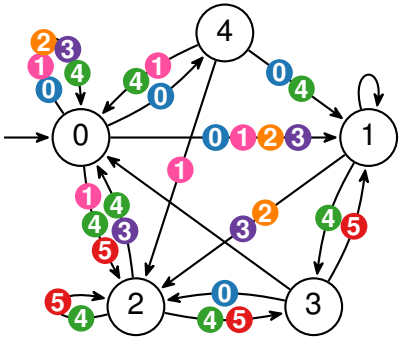


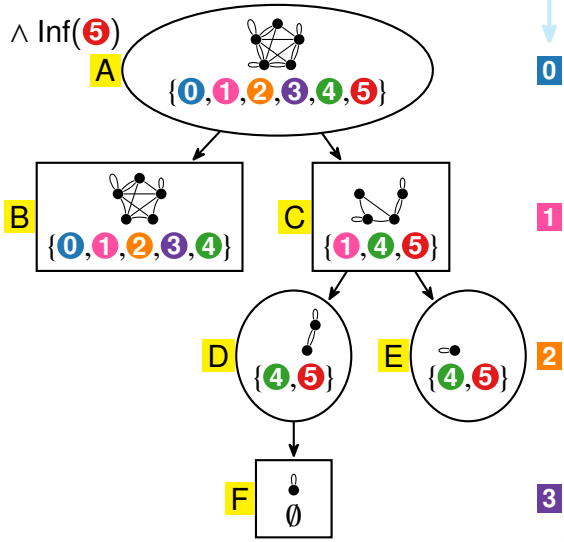


Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$

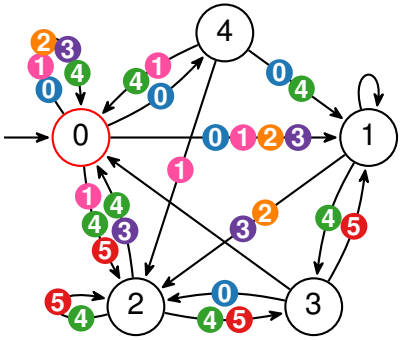


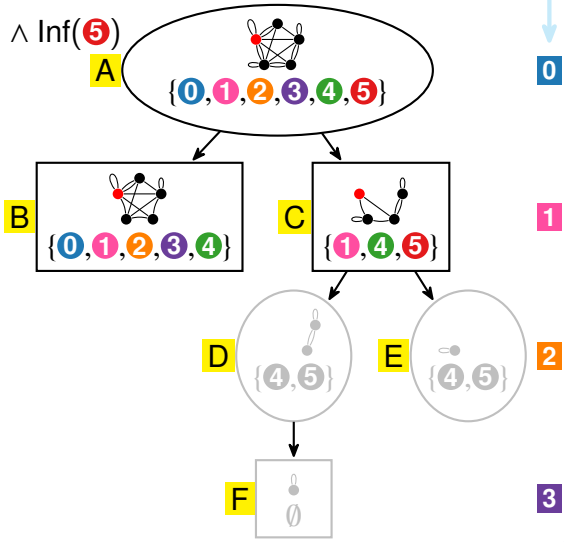


Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$



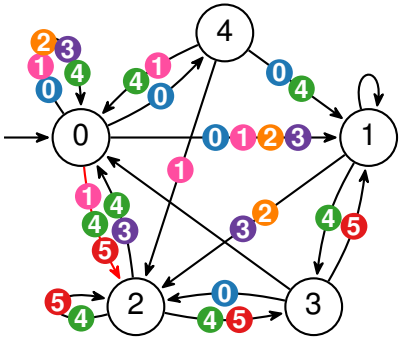


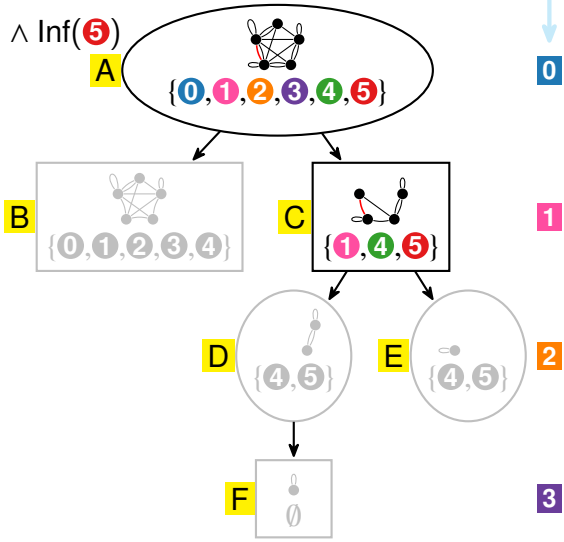
→ 0,B

Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$



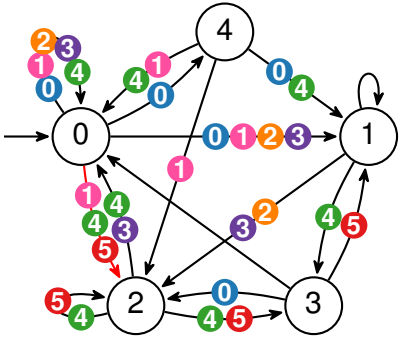


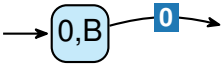
→ 0,B

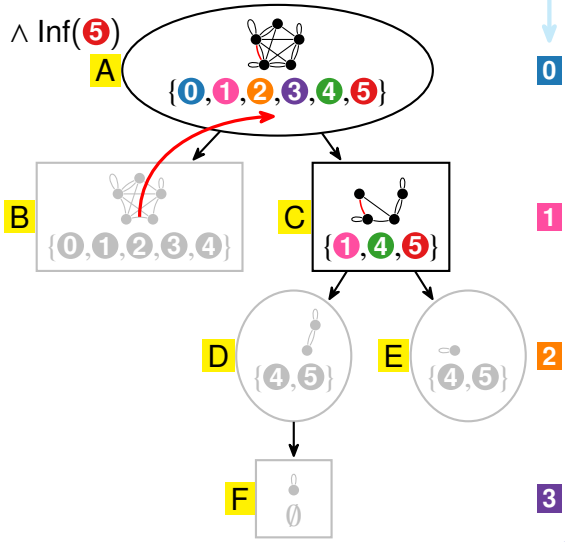
Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$



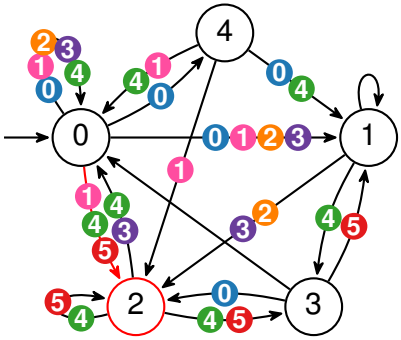


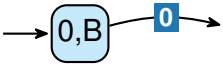


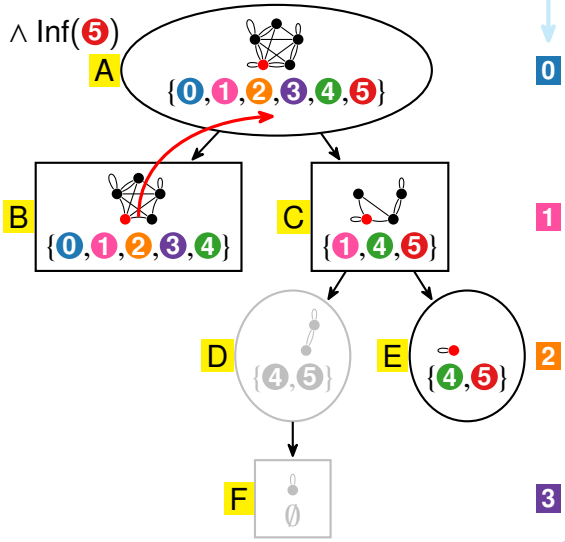
Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$



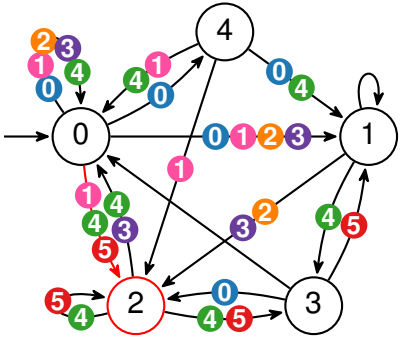


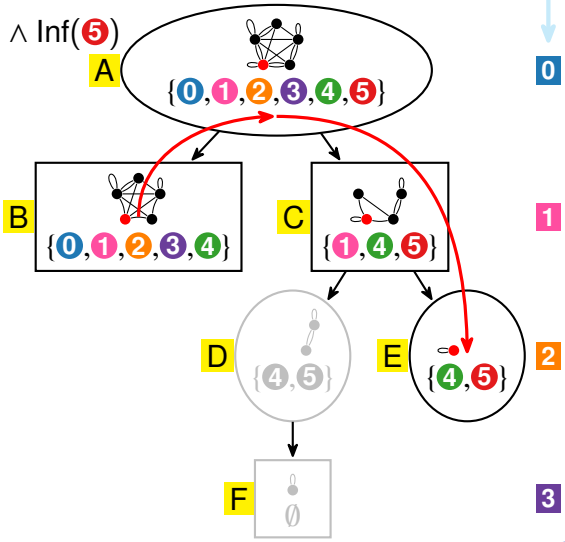


Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$

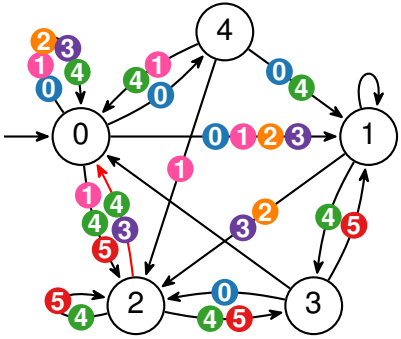


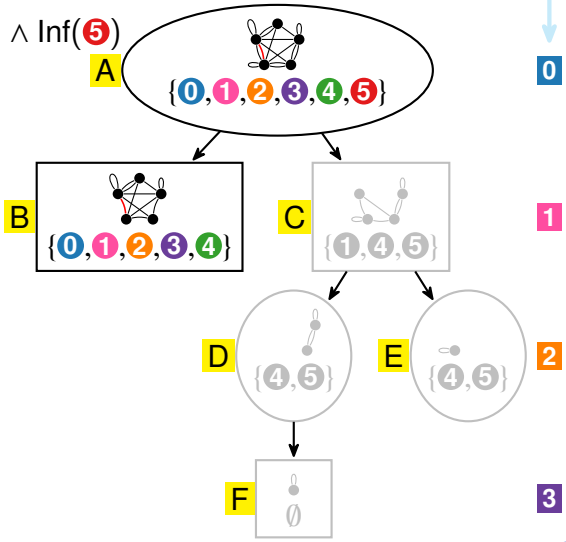


Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$

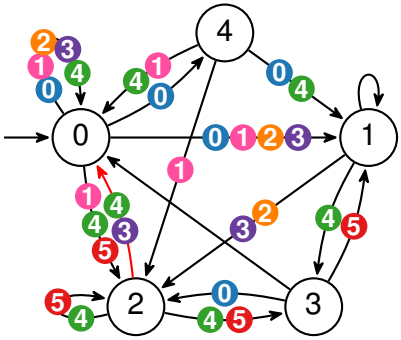


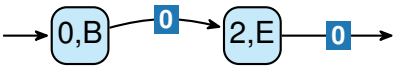


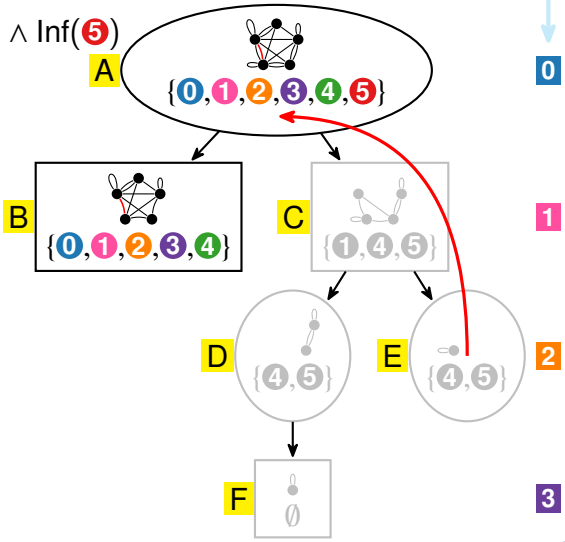
Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$



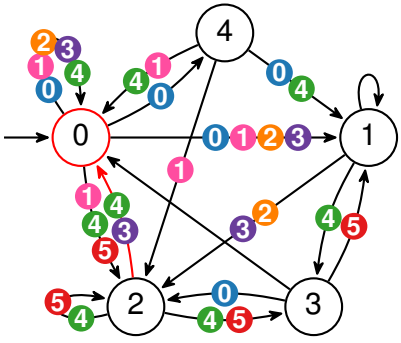


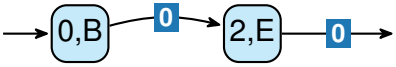


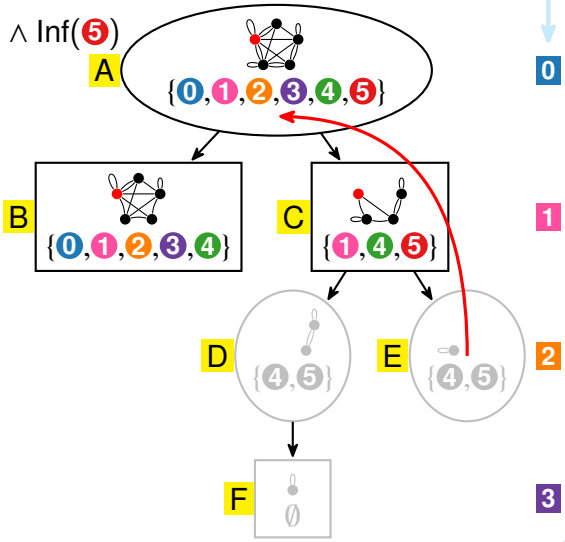
Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$



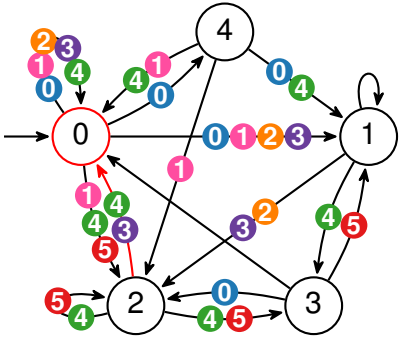


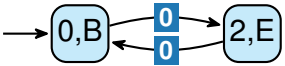


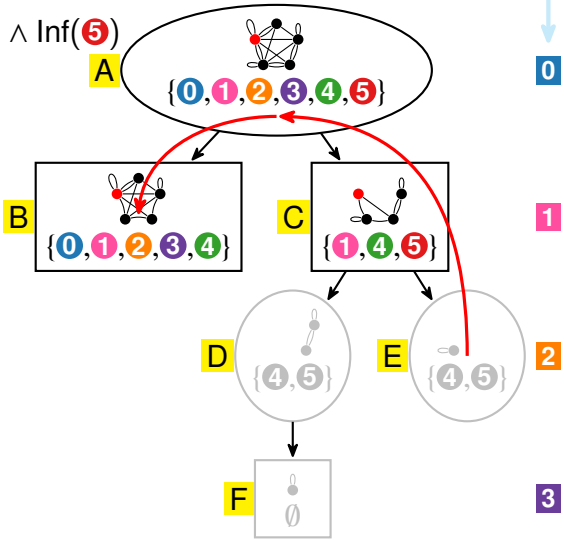
Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$



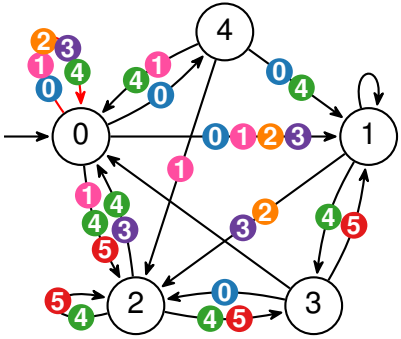


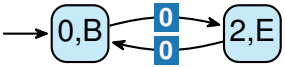


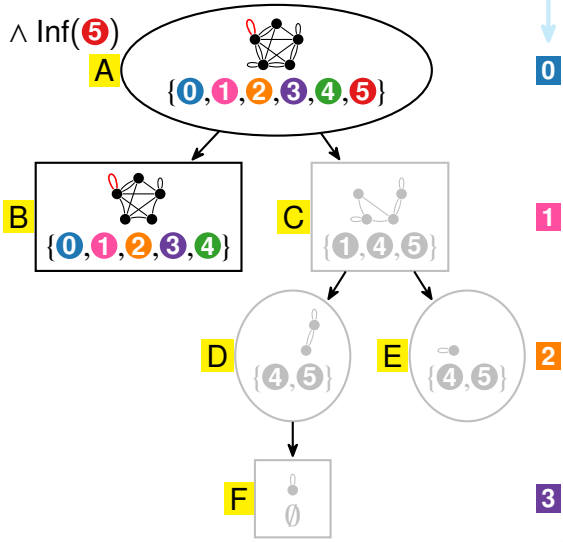
Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$



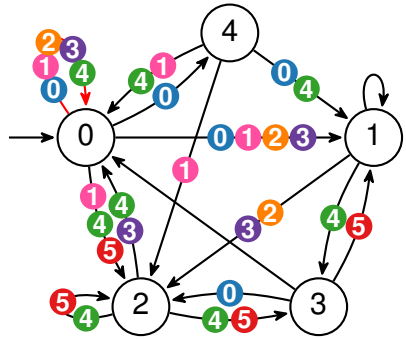


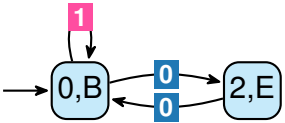


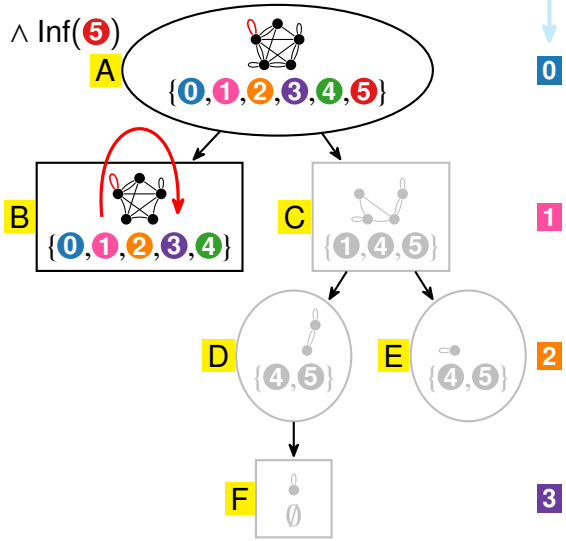
Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$



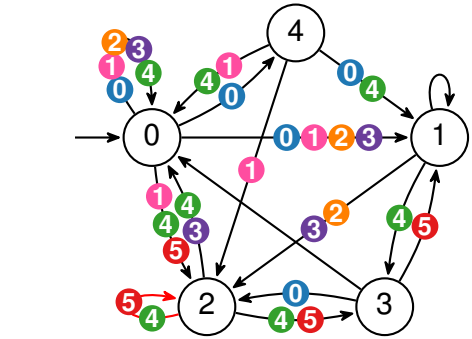


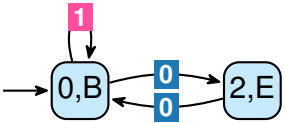


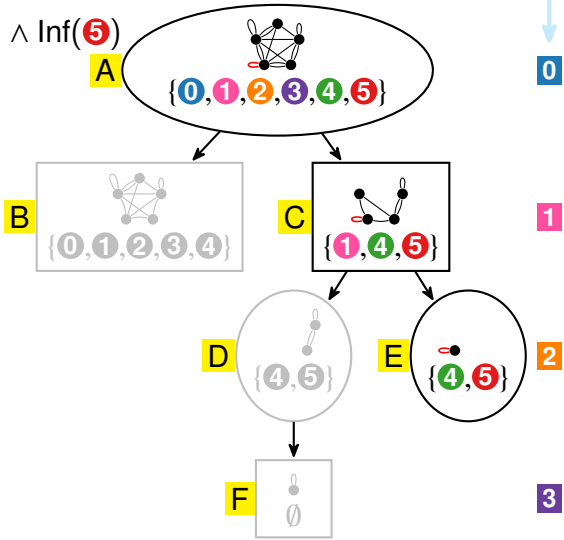
Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$



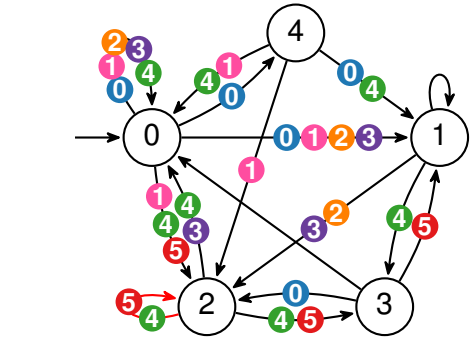


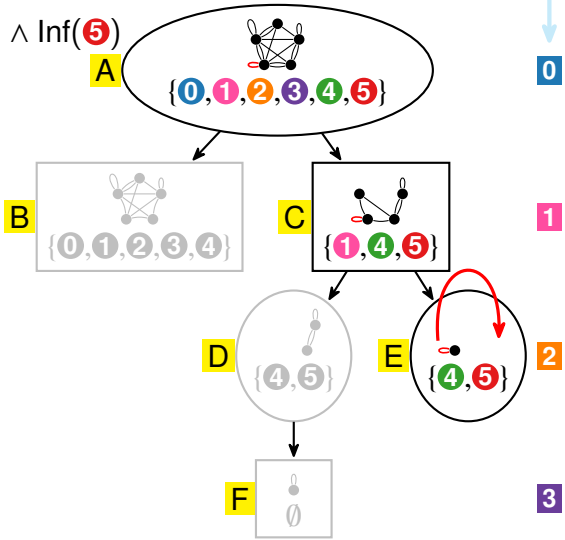


Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$

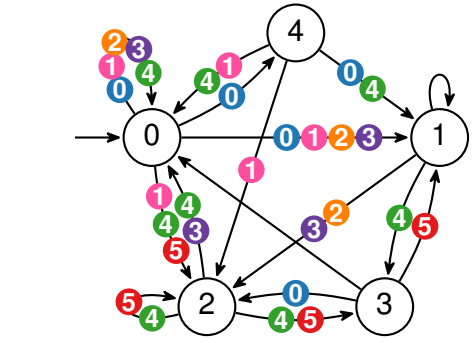


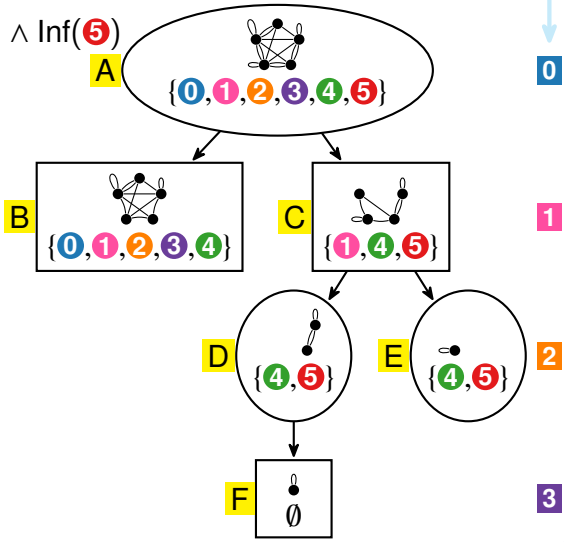


Paritization of a TELA

priorities for **min even** acceptance

$(\mathrm{Inf}(0) \vee \mathrm{Fin}(1) \vee \mathrm{Inf}(2) \vee \mathrm{Inf}(3)) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5)$





Other Paritization Procedures

Many paritization procedures create states (s, m) where $\begin{cases} s\text{: original state} \\ m\text{: memory} \end{cases}$

Latest Appearance Record (LAR) works on any TELA,
m is an order of all colors

Löding. Optimal bounds for transformations of ω-automata. *FSTTCS'99*. doi

Other Paritization Procedures

Many paritization procedures create states (s, m) where $\begin{cases} s\text{: original state} \\ m\text{: memory} \end{cases}$

Latest Appearance Record (LAR) works on any TELA,
m is an order of all colors

Index Appearance Record (IAR) take Rabin or Streett as input,
m is an order of the acceptance pairs


Křetínský, Meggendorfer, Waldmann, and Weininger. Index appearance record with preorders. *Acta Informatica*, 2021. doi

Other Paritization Procedures

Many paritization procedures create states (s, m) where $\begin{cases} s\text{: original state} \\ m\text{: memory} \end{cases}$

Latest Appearance Record (LAR)	works on any TELA,
	m is an order of all colors
Index Appearance Record (IAR)	take Rabin or Streett as input,
	m is an order of the acceptance pairs
Degeneralization	takes generalized Büchi as input,
	m is a color number

Other Paritization Procedures

Many paritization procedures create states (s, m) where $\begin{cases} s\text{: original state} \\ m\text{: memory} \end{cases}$

Latest Appearance Record (LAR)	works on any TELA, m is an order of all colors
Index Appearance Record (IAR)	take Rabin or Streett as input, m is an order of the acceptance pairs
Degeneralization	takes generalized Büchi as input, m is a color number
Spot's `to_parity()`	works on any TELA, combines all the above + optimizations

Other Paritization Procedures

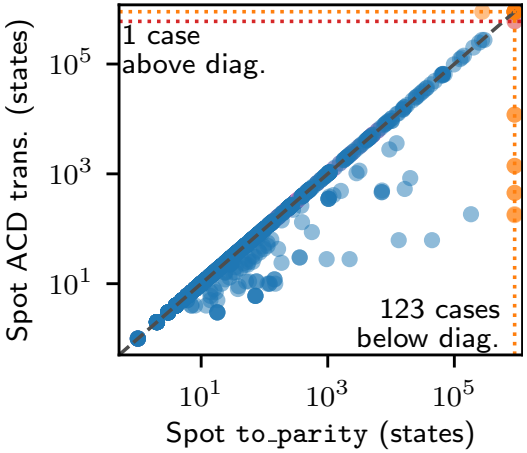
Many paritization procedures create states (s, m) where $\begin{cases} s\text{: original state} \\ m\text{: memory} \end{cases}$

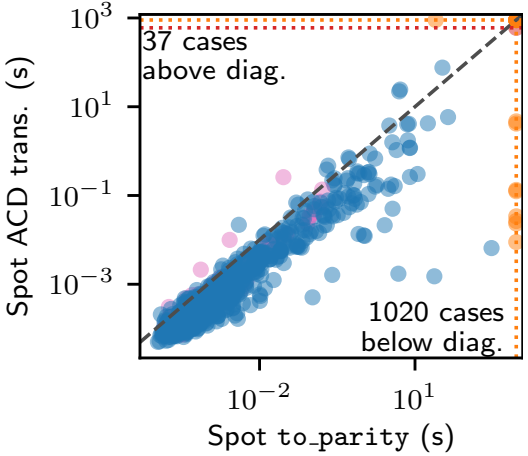
Latest Appearance Record (LAR)	works on any TELA, m is an order of all colors
Index Appearance Record (IAR)	take Rabin or Streett as input, m is an order of the acceptance pairs
Degeneralization	takes generalized Büchi as input, m is a color number
Spot's `to_parity()`	works on any TELA, combines all the above + optimizations
ACD-transform	works on any TELA, m denotes a node of the ACD

ACD-transform is optimal among algorithms that build states of shape (s, m).

Casares, Colcombet, and Fijalkow. Optimal transformations of games and automata using Muller conditions. *ICALP'21*. doi

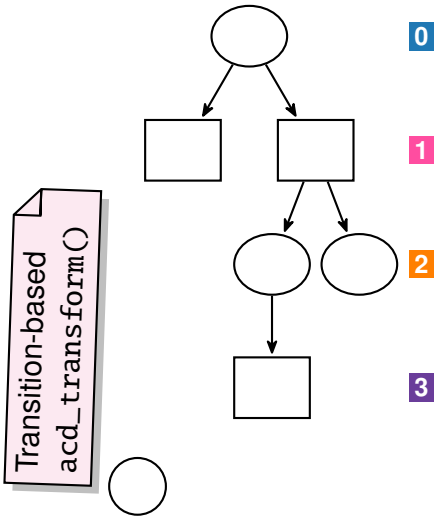
Comparison between `to_parity()` and `acd_transform()`



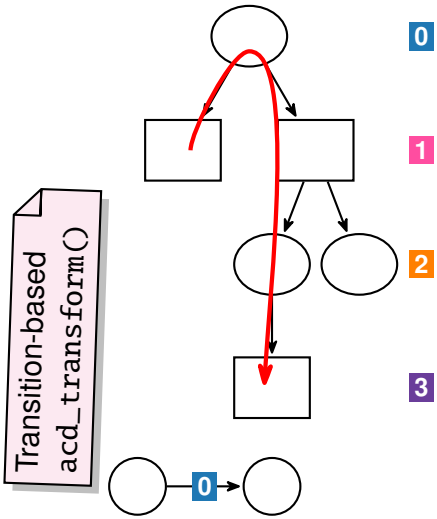


Benchmarck of 1065 TELA generated from LTL formulas from SyntComp. These TELA have between 2 and 55 colors (median 5) and up to 245761 states (median 20).

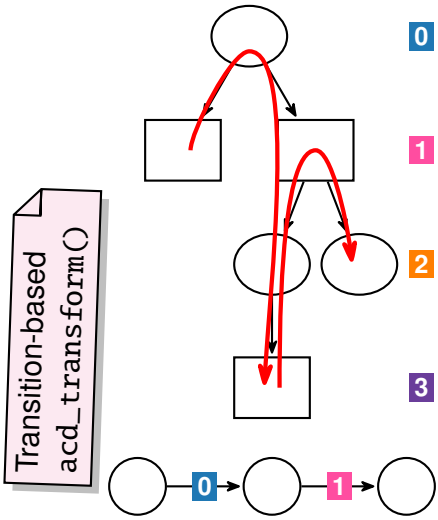
Producing State-Based Parity Automata (Intuition)



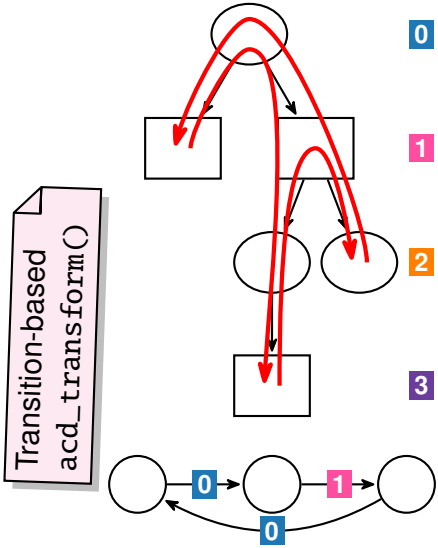
Producing State-Based Parity Automata (Intuition)



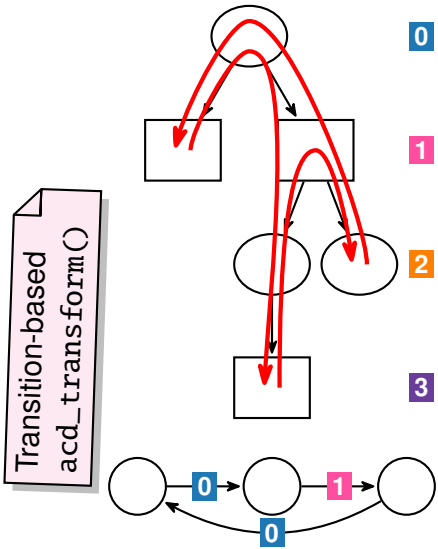
Producing State-Based Parity Automata (Intuition)



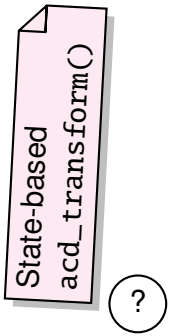
Producing State-Based Parity Automata (Intuition)

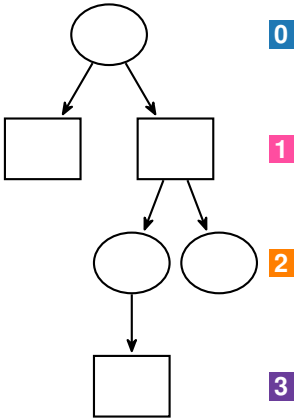


Producing State-Based Parity Automata (Intuition)

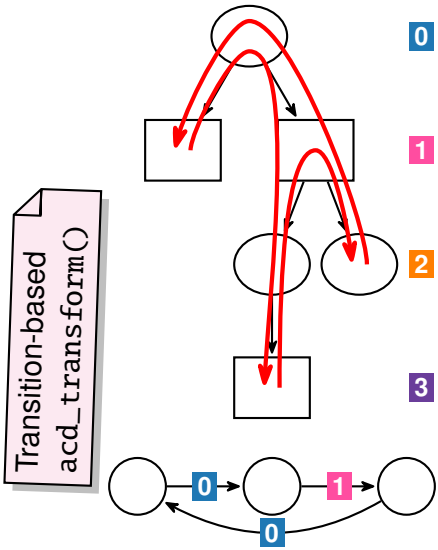


Every cycle has at least one leftward jump in the ACD.

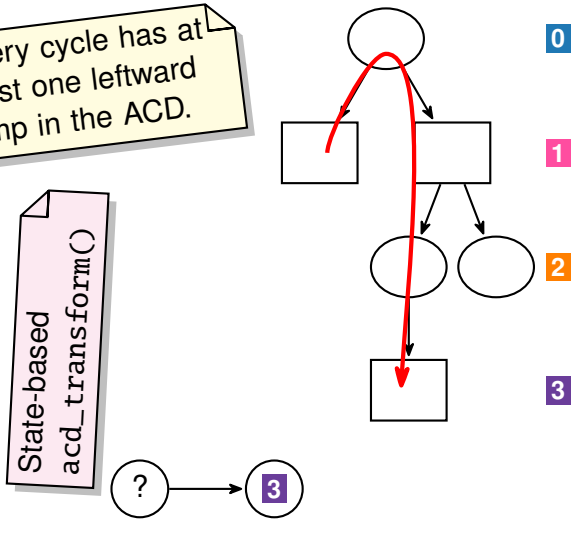




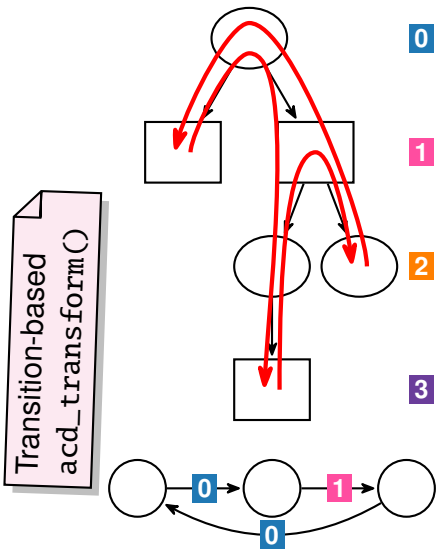
Producing State-Based Parity Automata (Intuition)



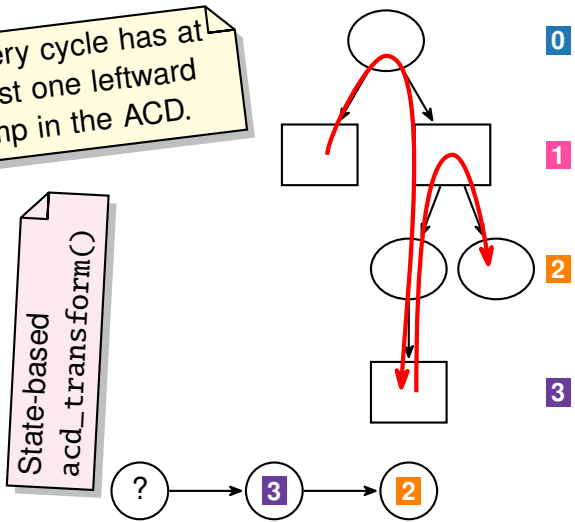
Every cycle has at least one leftward jump in the ACD.



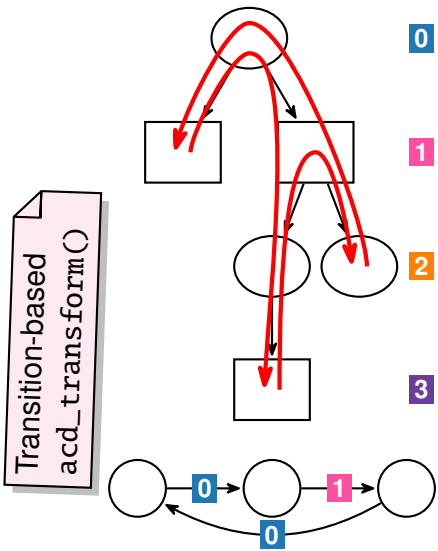
Producing State-Based Parity Automata (Intuition)



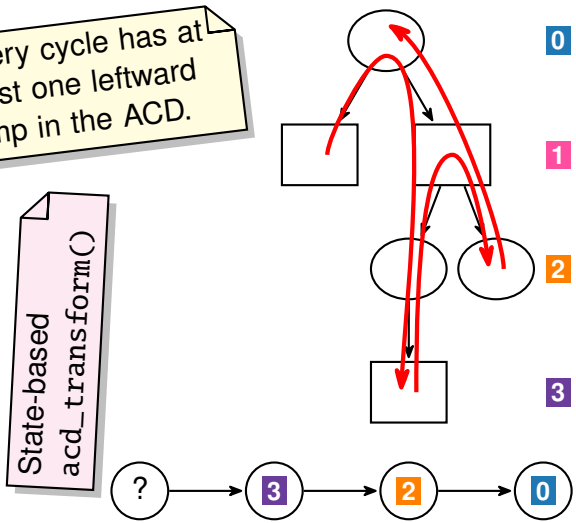
Every cycle has at least one leftward jump in the ACD.



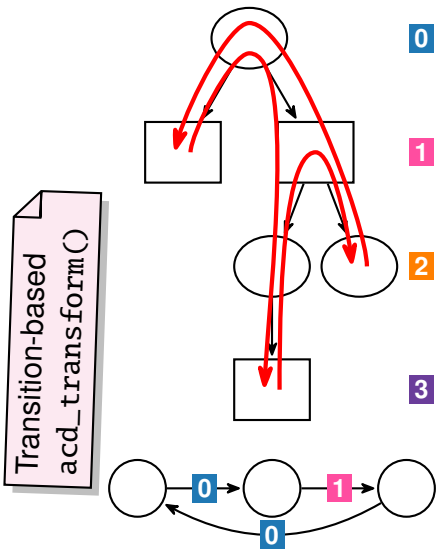
Producing State-Based Parity Automata (Intuition)



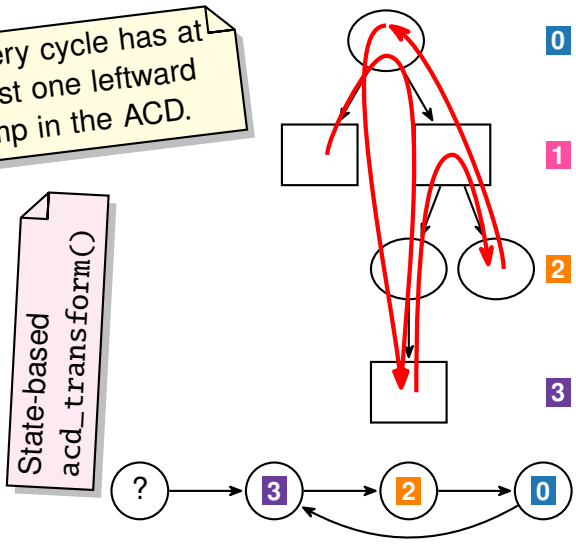
Every cycle has at least one leftward jump in the ACD.



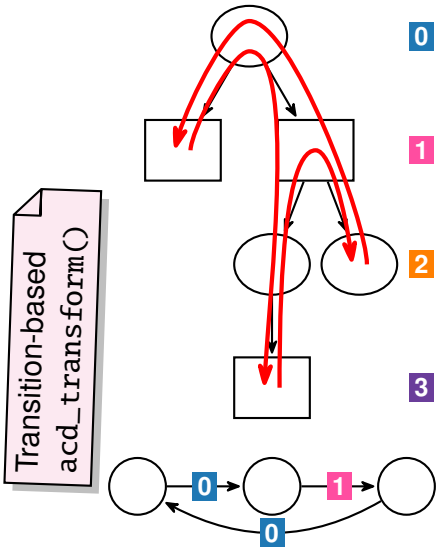
Producing State-Based Parity Automata (Intuition)



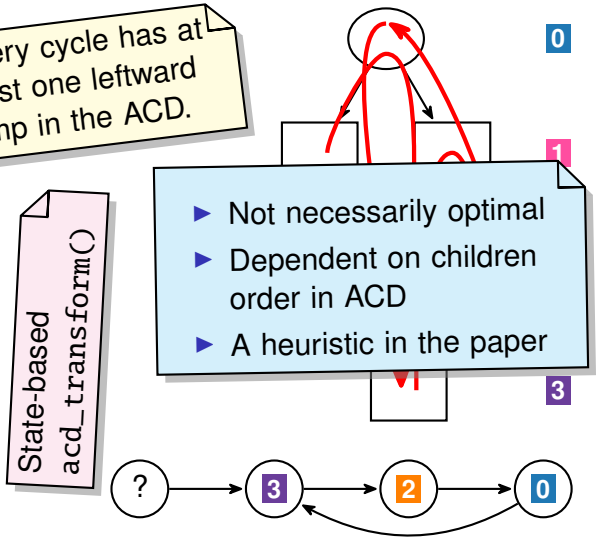
Every cycle has at least one leftward jump in the ACD.



Producing State-Based Parity Automata (Intuition)



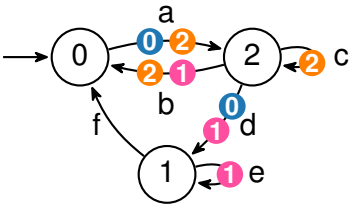
Every cycle has at least one leftward jump in the ACD.



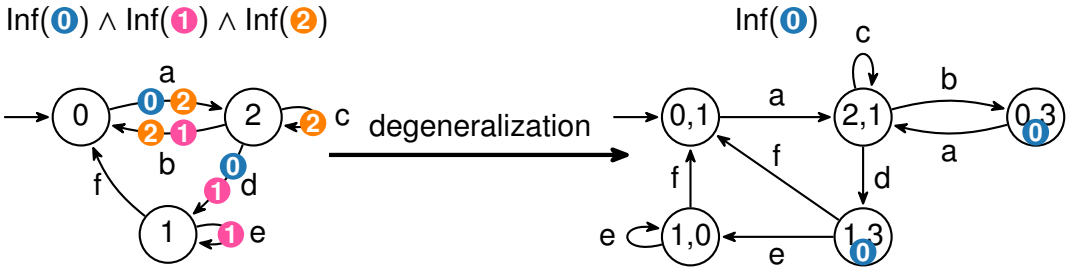
- Not necessarily optimal
- Dependent on children order in ACD
- A heuristic in the paper

Application to Degeneralization (TGBA → SBA)

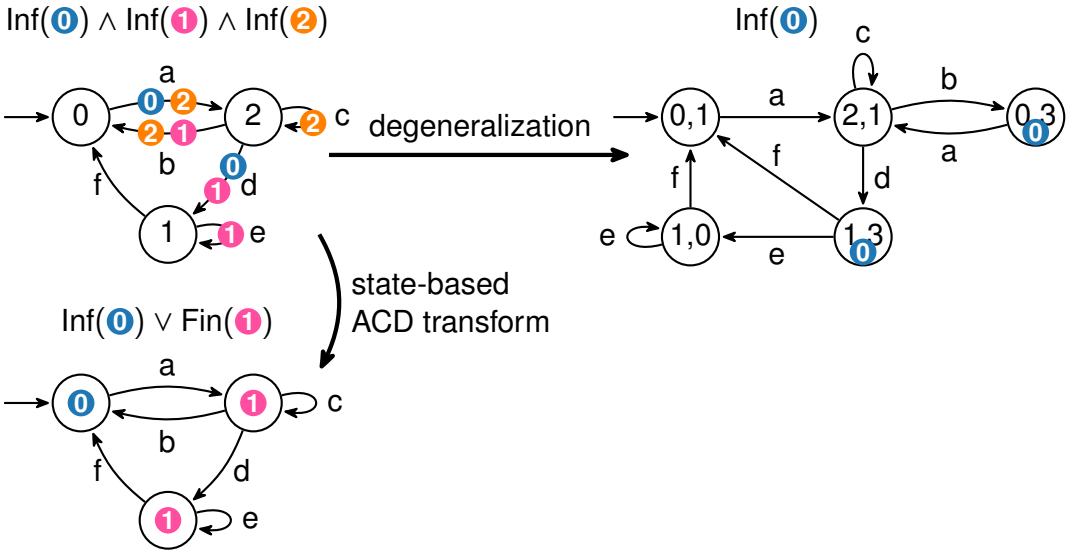
$\mathrm{Inf}(0) \wedge \mathrm{Inf}(1) \wedge \mathrm{Inf}(2)$



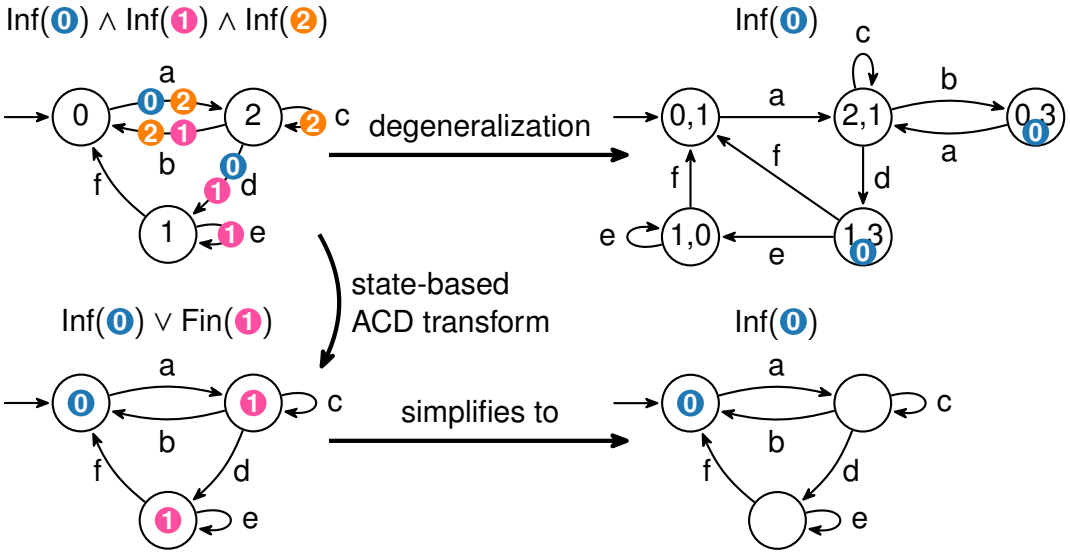
Application to Degeneralization (TGBA → SBA)



Application to Degeneralization (TGBA → SBA)



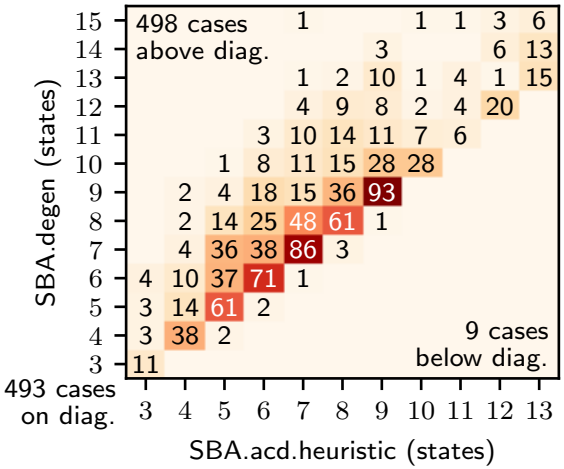
Application to Degeneralization (TGBA → SBA)



Application to Degeneralization: Benchmark

Application of the state-based version of `acd_transform()` on 1000 random transition-based generalized Büchi automata, with 3–4 states and 2–3 colors.

Comparison with Spot 2.10's best degeneralization routine.

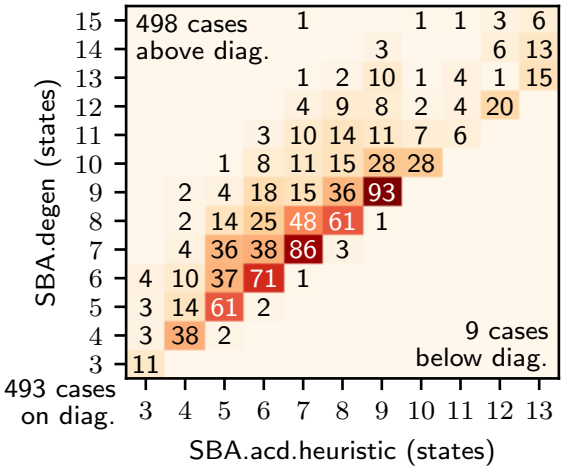


Application to Degeneralization: Benchmark

Application of the state-based version of `acd_transform()` on 1000 random transition-based generalized Büchi automata, with 3–4 states and 2–3 colors.

Comparison with Spot 2.10's best degeneralization routine.

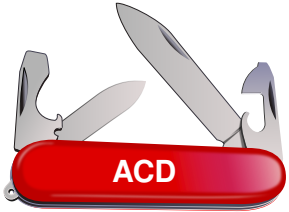
Compared to min. sizes (found via SAT), ACD gives **+0.17** states on avg., and Spot's degen. gives **+1.21** states on avg.



Conclusion

ACD is versatile, and can replace several existing algorithms:

- paritization of automata with arbitrary acceptance
  (always better than LAR, IAR, and their combinations)
- degeneralization (better than traditional algorithms)
- minimization of the number of priorities in a parity automaton
- several typeness checks (looking at the shape of the ACD)



Two implementations can be a basis for further experiments:

Owl 21.0
`owl.model.in.tum.de`

Spot 2.10 ▸ demo
`spot.lrde.epita.fr`