Boundedness problems in games: a roadmap

GT Jeux'2012

Nathanaël Fijalkow
(based on a joint work with Martin Zimmermann)

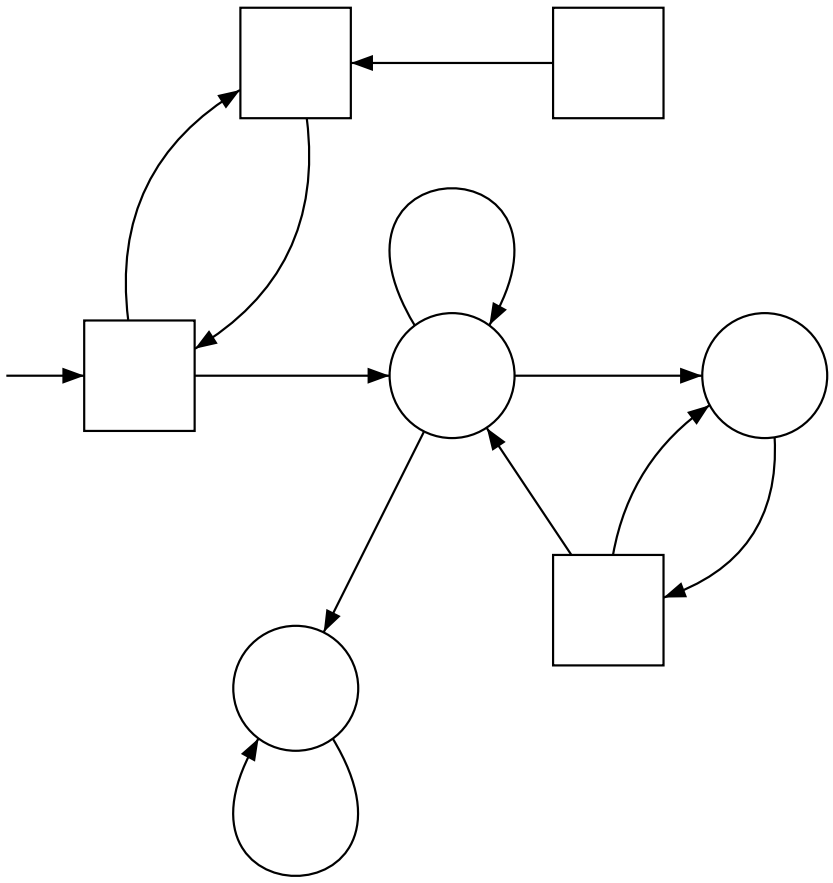
Institute of Informatics, Warsaw University – Poland

LIAFA, Université Paris 7 Denis Diderot – France

September 21st, 2012

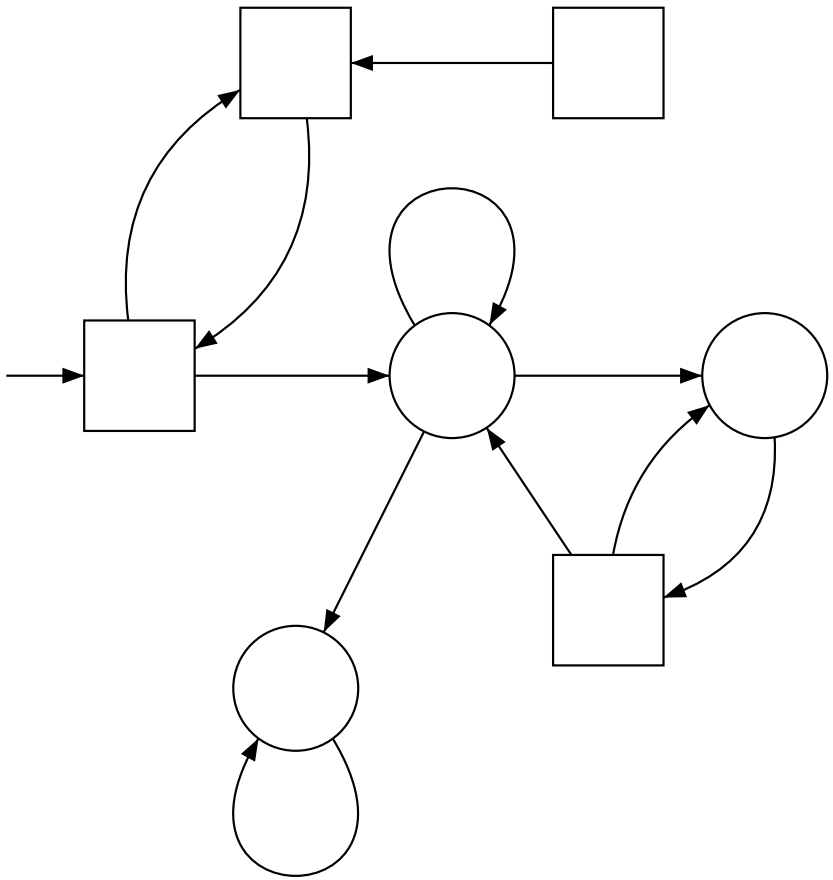
Games with counter

Two-player turn-based games over **finite** graphs



Games with counter

Two-player turn-based games over **finite** graphs



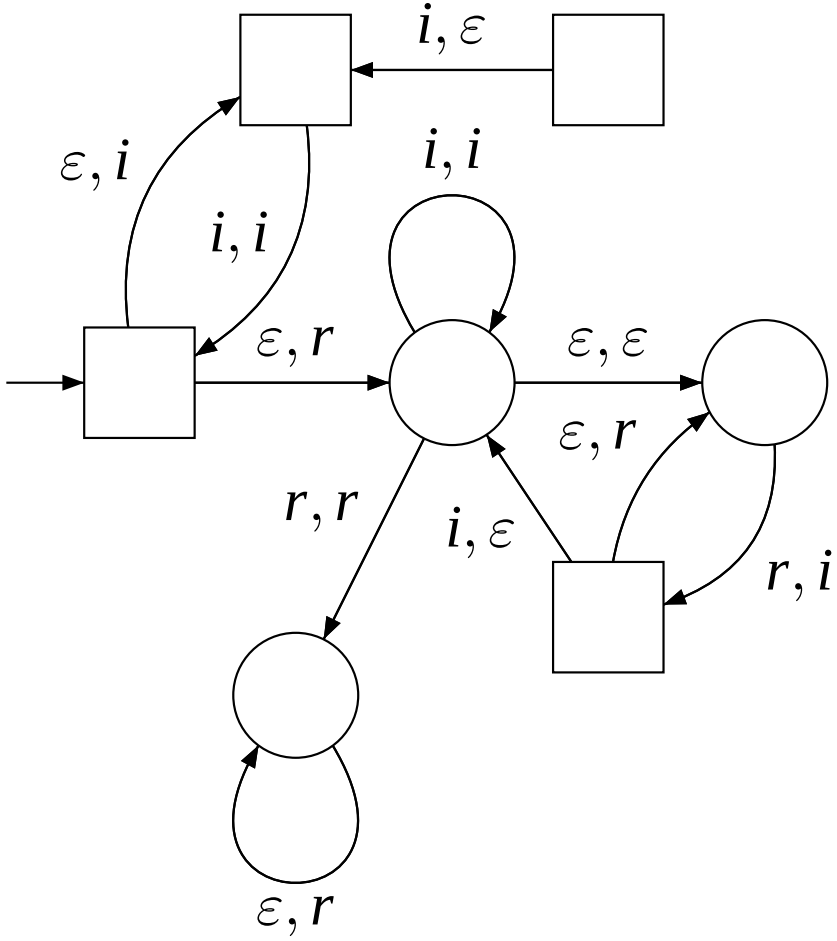
$$c_1 \doteq 0$$

$$c_2 \doteq 0$$

Winning condition: “there exists B, such that . . .”

Games with counter

Two-player turn-based games over **finite** graphs



$$c_1 \doteq 0$$

$$c_2 \doteq 0$$

Winning condition: “there exists B, such that . . .”

Actions on the counters

increment: i or "$c \longleftarrow c + 1$"

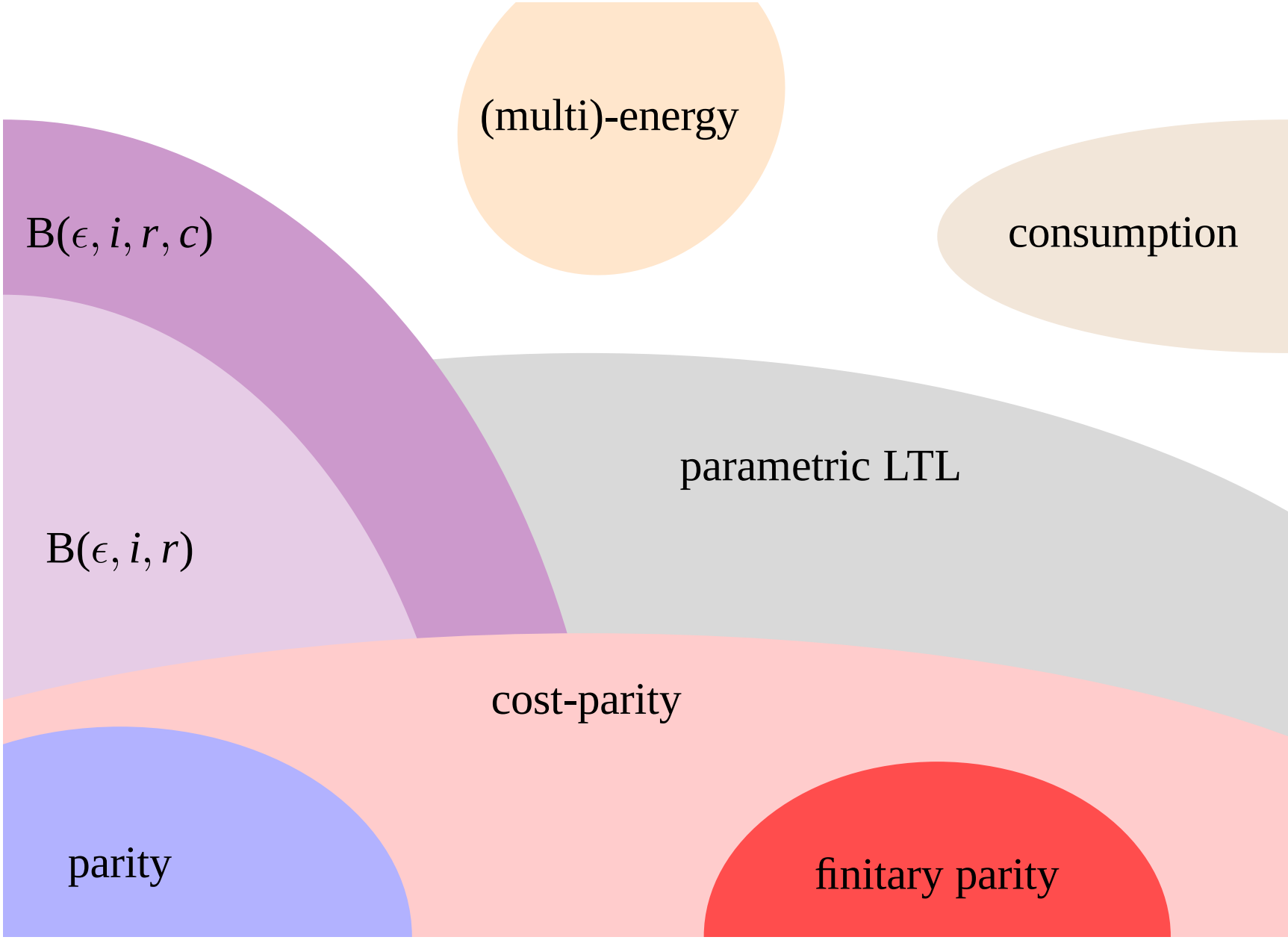
no action: ε

reset: r or "$c \longleftarrow 0$"

decrement: "$c \longleftarrow c - 1$"

refill: "$c \longleftarrow c + \omega$"

Roadmap


(multi)-energy
consumption
B(ϵ, i, r, c)
parametric LTL
B(ϵ, i, r)
cost-parity
parity
finitary parity

The robust: B-conditions (Bojańczyk, Colcombet)

Actions: ε, i, r.

B-conditions are (positive) boolean combination of:

- "counter c is bounded"
- Büchi(F): "F appears infinitely often"
- CoBüchi(G): "G appears finitely often"

The robust: B-conditions (Bojańczyk, Colcombet)

Actions: ε, i, r.

B-conditions are (positive) boolean combination of:

- "counter c is bounded"
- Büchi(F): "F appears infinitely often"
- CoBüchi(G): "G appears finitely often"

Observation

Eve wins "counter c is bounded"
if and only if
she wins "Büchi(i) $\Rightarrow$ Büchi(r)".

The robust: B-conditions (Bojańczyk, Colcombet)

Actions: ε, i, r.

B-conditions are (positive) boolean combination of:

- "counter c is bounded"
- Büchi(F): "F appears infinitely often"
- CoBüchi(G): "G appears finitely often"

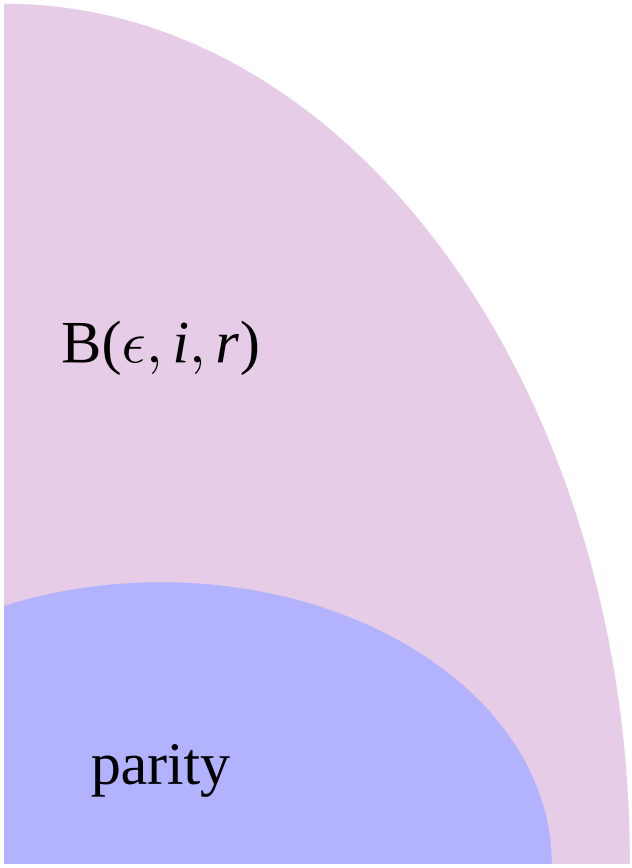
Observation

Eve wins "counter c is bounded"
if and only if
she wins "Büchi(i) $\Rightarrow$ Büchi(r)".

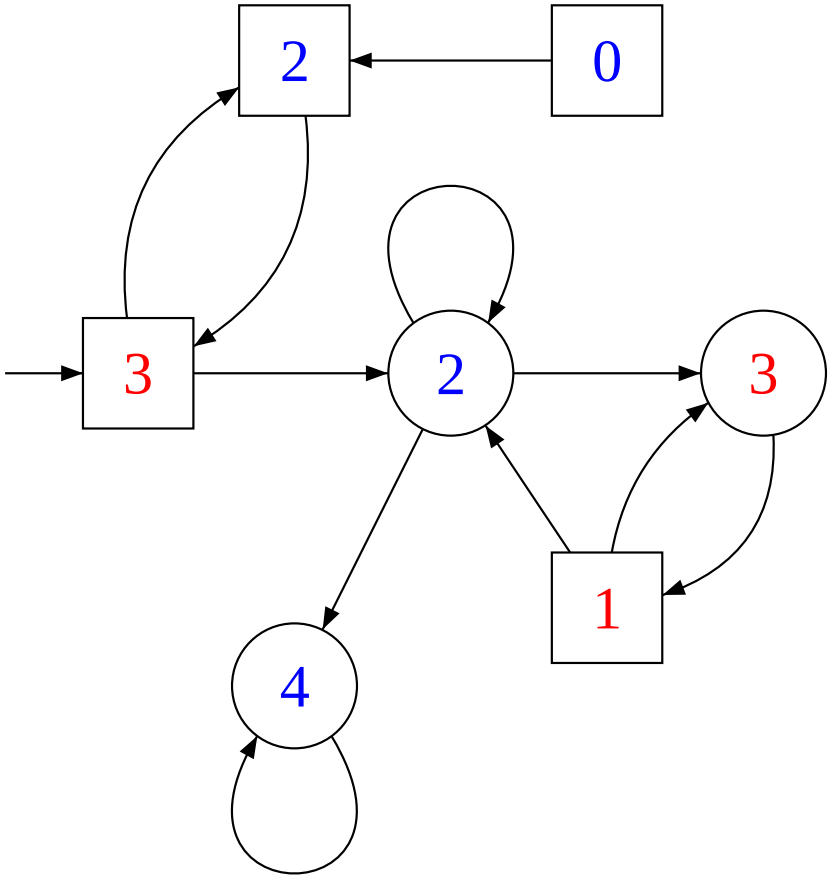
Corollary

One can decide who wins a B-game and Eve needs finite memory.

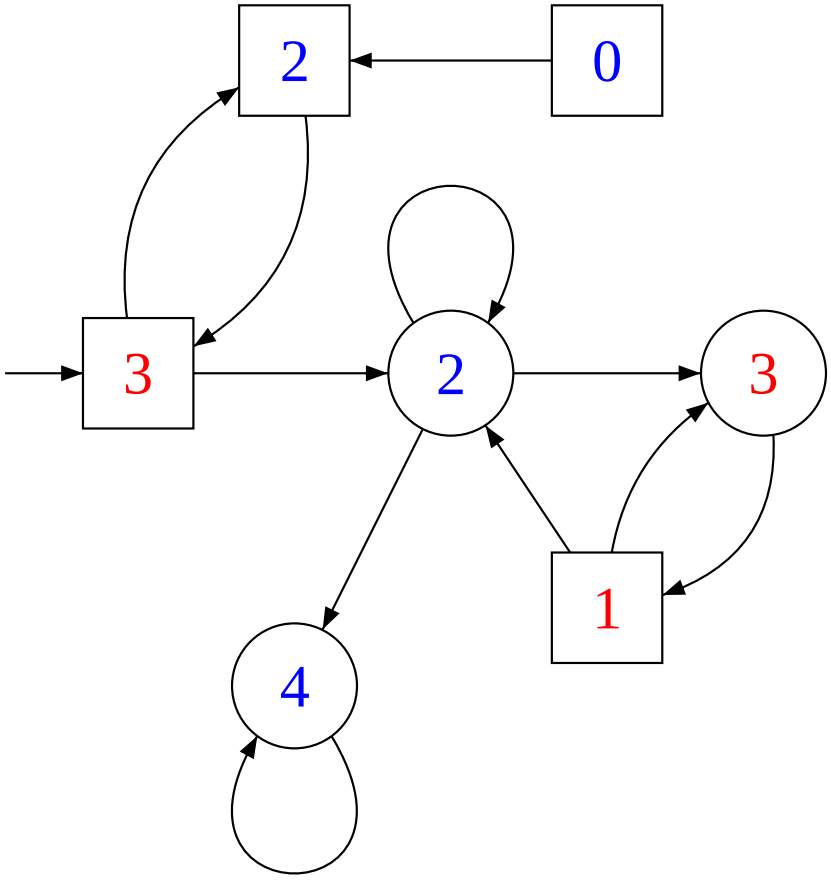
Roadmap

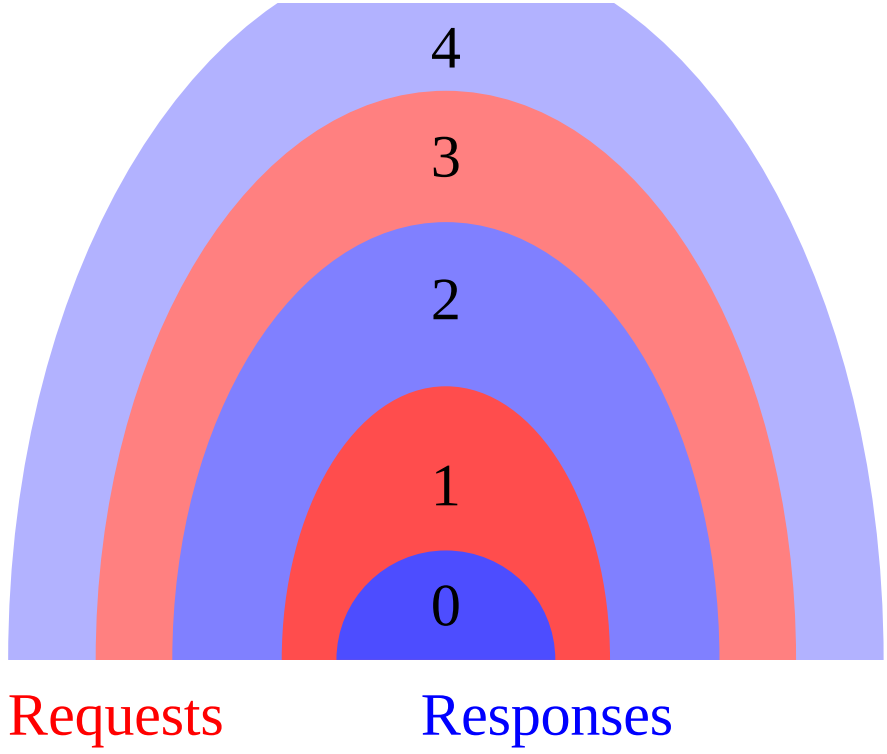


The surprising: finitary parity (Alur,Henzinger)

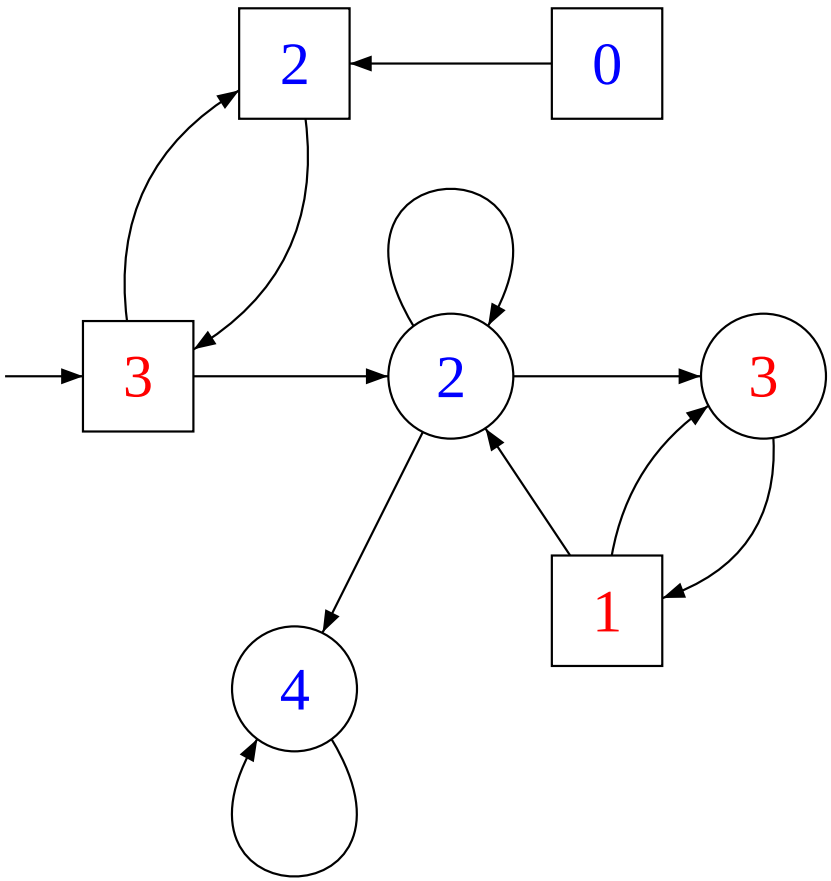


The surprising: finitary parity (Alur,Henzinger)





The surprising: finitary parity (Alur,Henzinger)



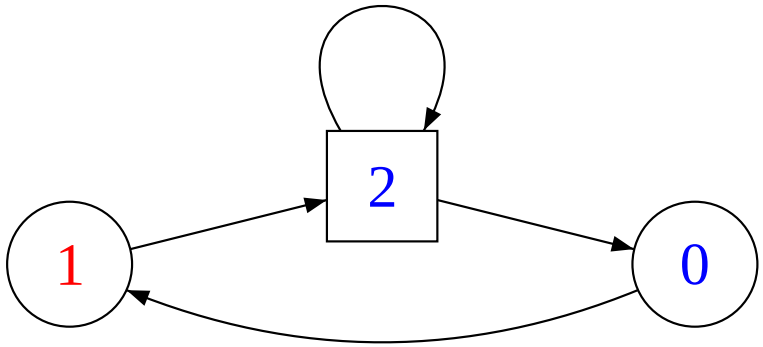
Parity:
Almost all requests are answered.

Finitary parity:
There exists a bound *b*, s.t. almost all requests are answered *within b steps*.

Parity games and finitary parity games are different

Parity:
Almost all requests are answered.

Finitary parity:
There exists a bound *b*, s.t. almost all requests are answered *within b steps*.



Eve wins for the parity condition,

but loses for the finitary parity condition!

Parity games and finitary parity games are different

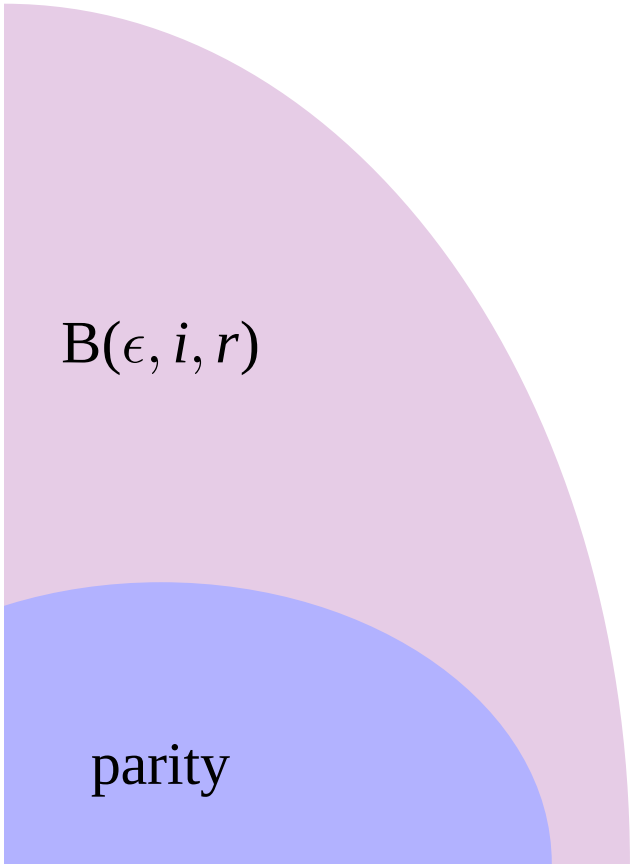
Parity:
Almost all requests are answered.

Finitary parity:
There exists a bound *b*, s.t. almost all requests are answered *within b steps*.

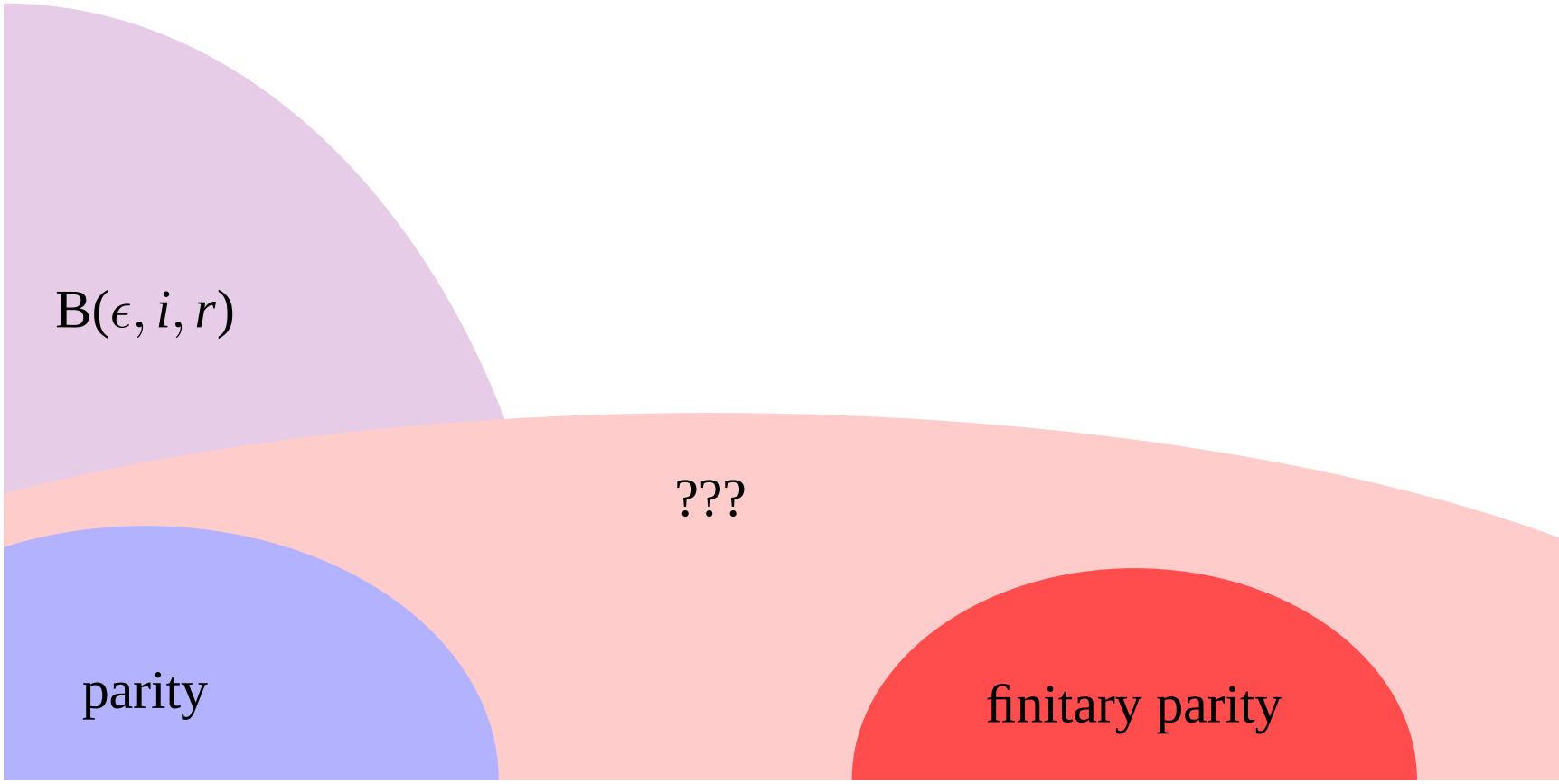
- Both players have positional winning strategies.
- Deciding the winner is in NP ∩ coNP.

- Eve has positional winning strategies.
- Adam needs infinite memory.
- Deciding the winner is in PTIME.

Roadmap



Roadmap



Cost-parity conditions

Parity:
Almost all requests are answered.

Finitary parity:
There exists a bound *b*, s.t. almost all requests are answered *within b steps*.

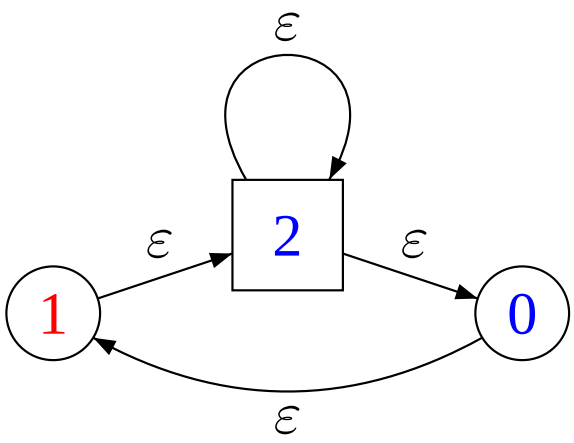
Cost-parity conditions

Parity:
Almost all requests are answered.

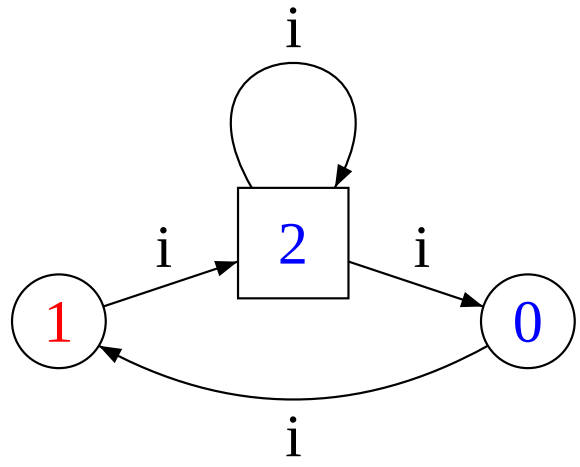
Finitary parity:
There exists a bound *b*, s.t. almost all requests are answered *within b steps*.

Cost-parity:
There exists a bound *b*, s.t.
almost all requests are answered *with cost at most b*.

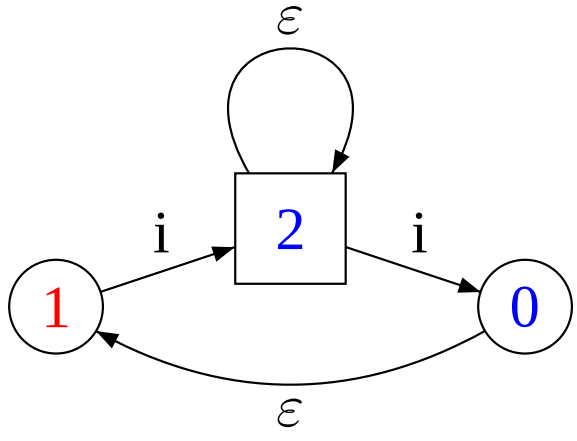
Costs



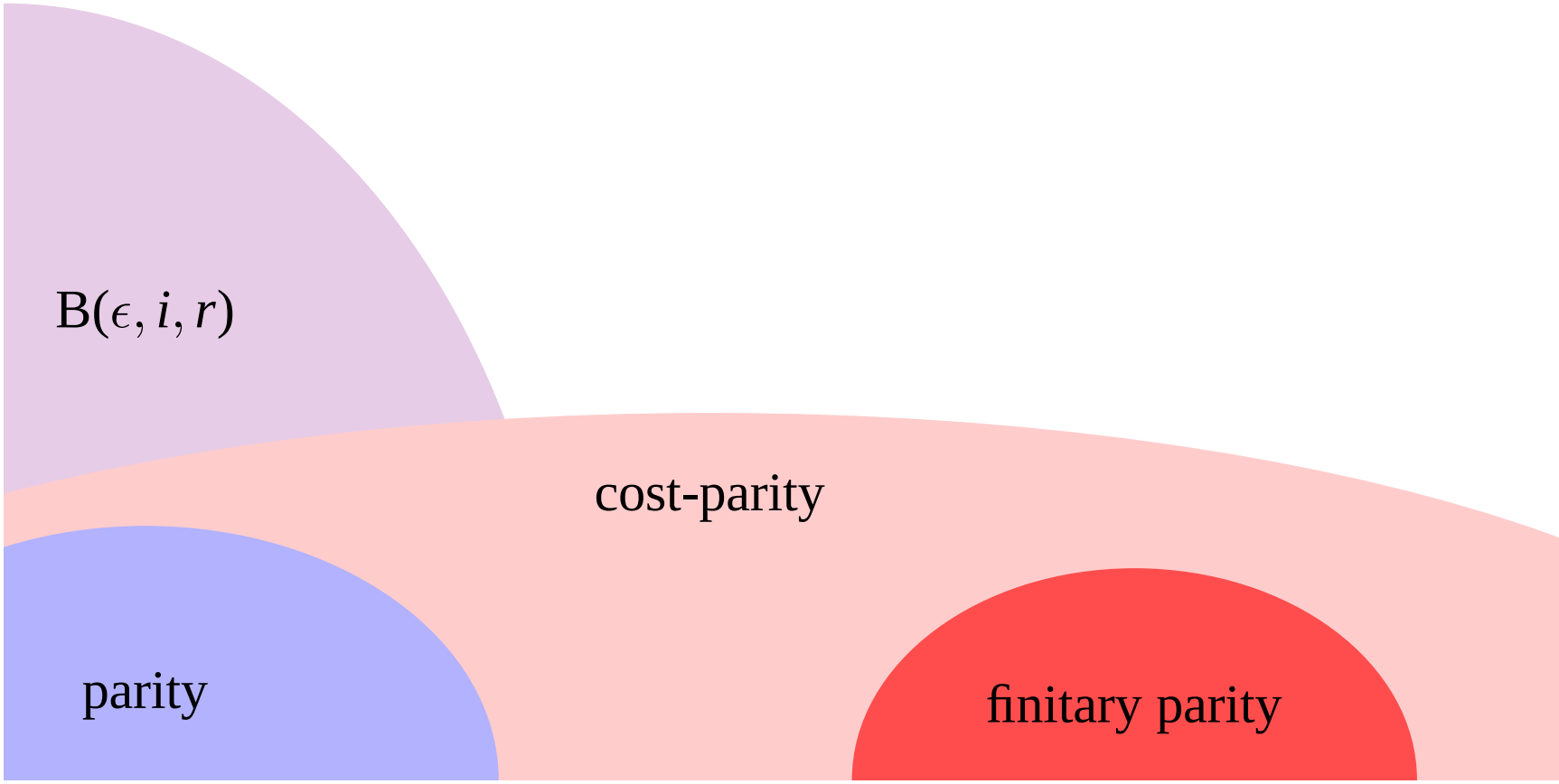
Parity game



Finitary parity game



Roadmap



Deciding the winner in cost-parity games

n: number of vertices
m: number of edges
d: number of colors

Theorem (F., Zimmermann 2012)

Given a parity games solver of complexity $T(n, m, d)$,
we construct a cost-parity games solver of complexity

$$O(n \cdot T(n \cdot d, m \cdot d, d + 2)) \ .$$

Positional determinacy for Eve

Objective: **strategy optimization**

Assume σ is a winning strategy.
How to construct a memoryless winning strategy σ' from σ?

Positional determinacy for Eve

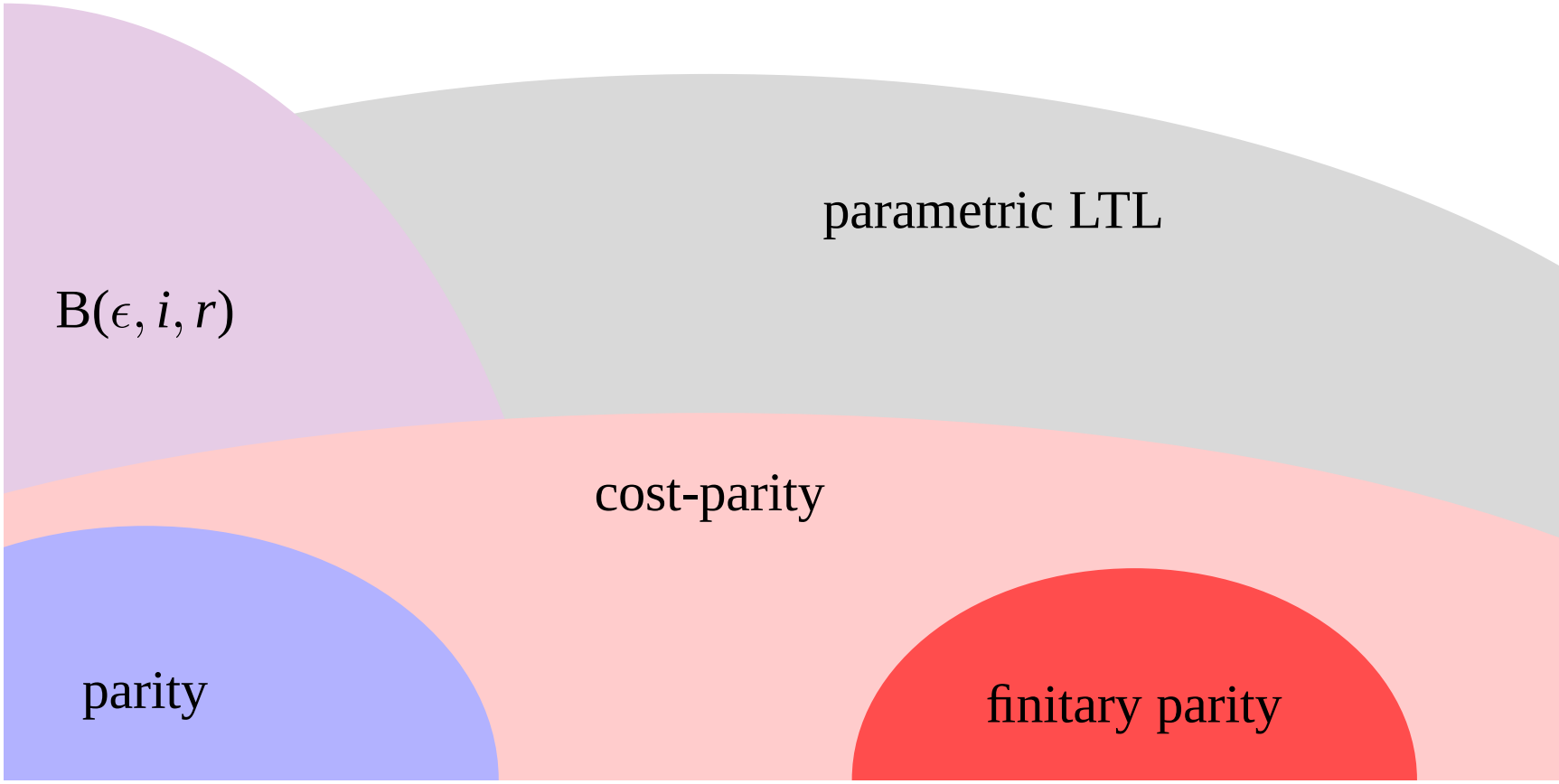
Objective: **strategy optimization**

Assume σ is a winning strategy.
How to construct a memoryless winning strategy σ' from σ?

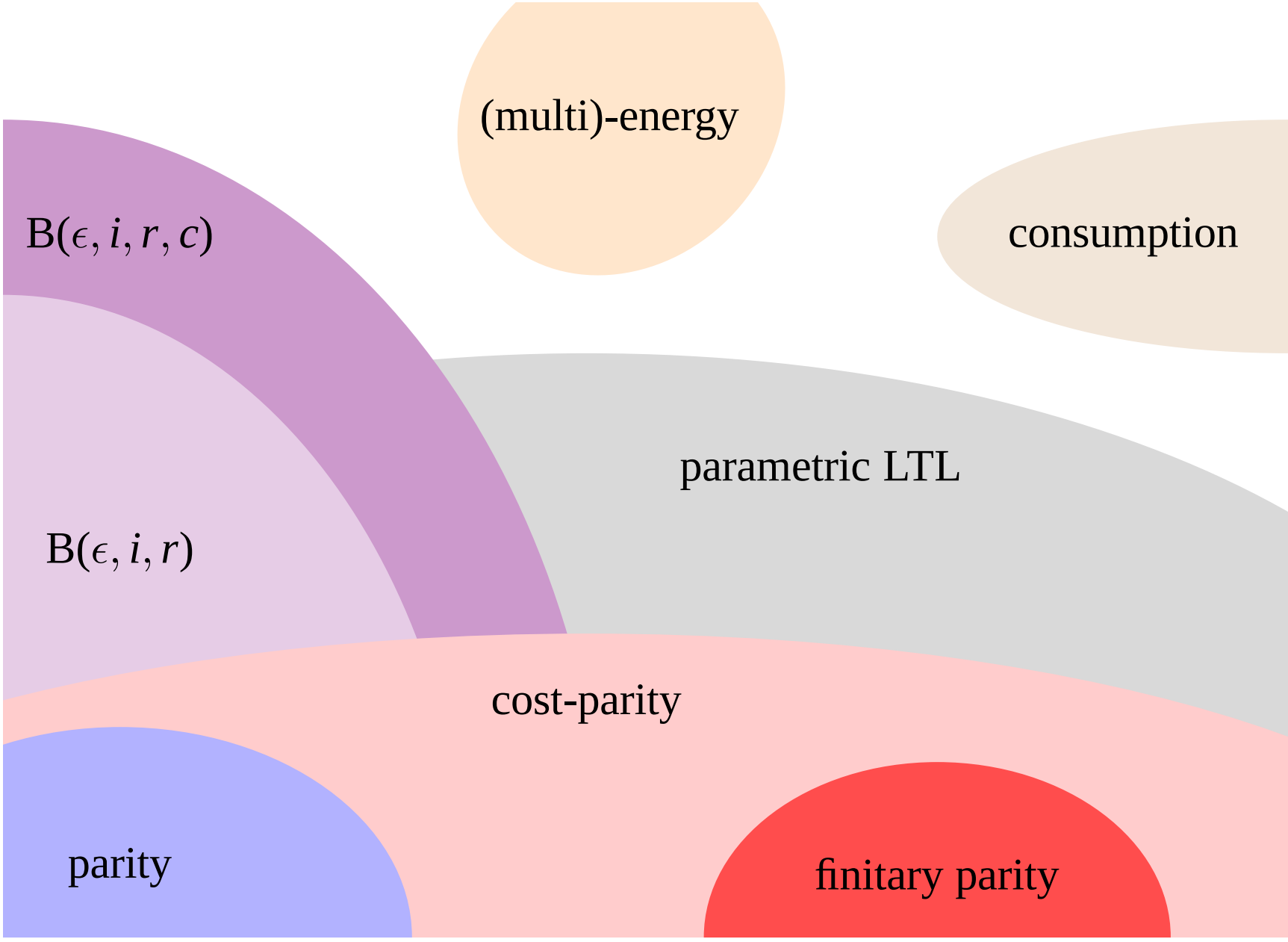
Tool: **scoring functions**

“à la Müller and Schupp” past-oriented proof

Roadmap



Roadmap


(multi)-energy
B(ϵ, i, r, c)
consumption
parametric LTL
B(ϵ, i, r)
cost-parity
parity
finitary parity

Multi-energy

increment: i or "$c \longleftarrow c + 1$"

no action: ε

decrement: "$c \longleftarrow c - 1$"

(Tomáš Brázdil, Jakub Chaloupka, Krishnendu Chatterjee, Laurent Doyen, Uli Fahrenberg, Petr Jančar, Line Juhl, Antonín Kučera, Kim G. Larsen, Jean-François Raskin, Mickael Randour, Jiři Srba,. . .)

Consumption

~~**increment:** i or "$c \longleftarrow c + 1$"~~

no action: ε

decrement: "$c \longleftarrow c - 1$"

refill: "$c \longleftarrow c + \omega$"

(Tomáš Brázdil, Krishnendu Chatterjee, Antonín Kučera, Petr Novotný, CAV'2012)

B-conditions with checks

check: "counter c is bounded on **checked** values"

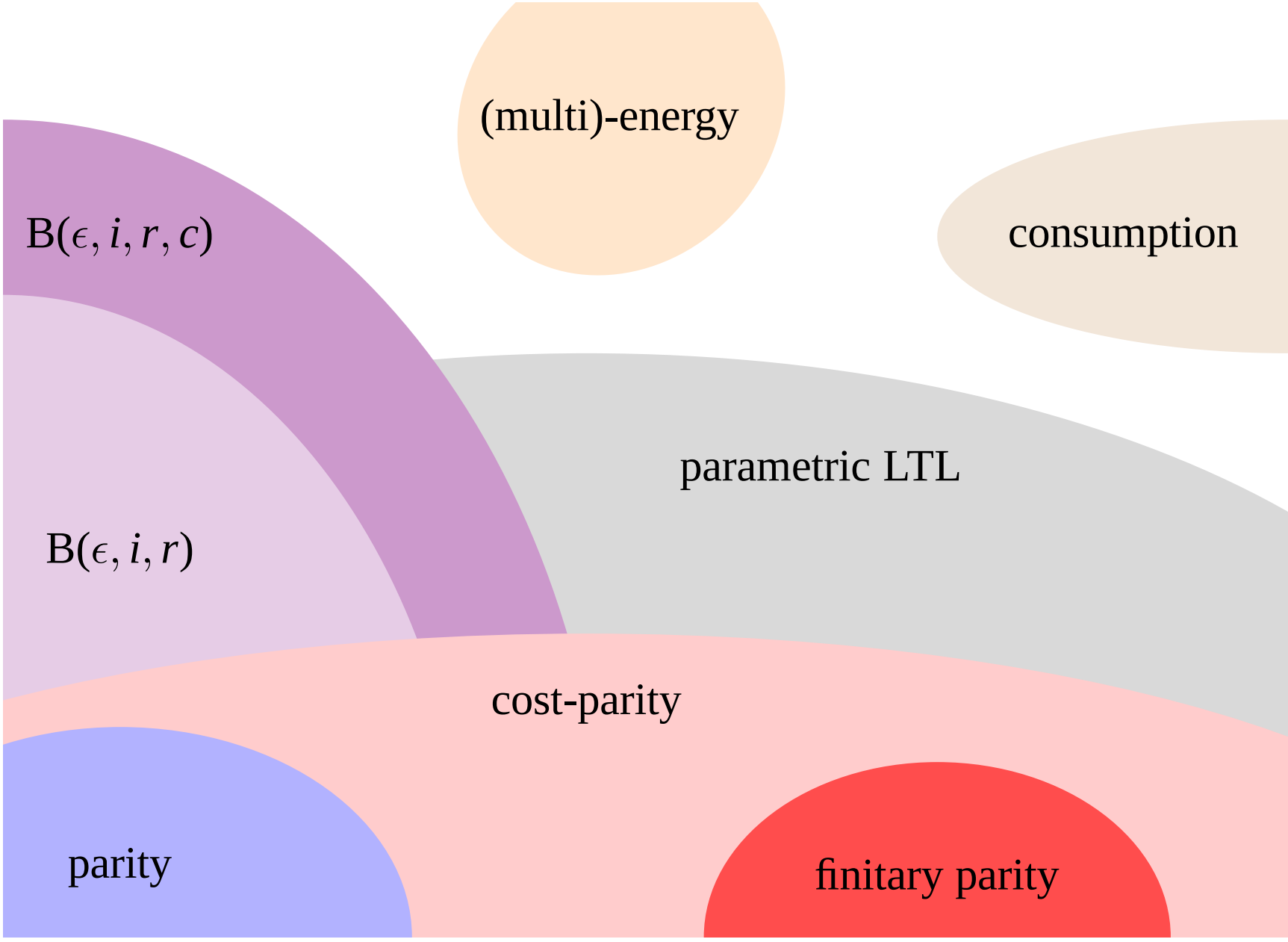
increment: i or "$c \longleftarrow c + 1$"

no action: ε

reset: "$c \longleftarrow 0$"

max: "$c \longleftarrow \max(c', c'')$"

Roadmap


(multi)-energy
consumption
B(ϵ, i, r, c)
parametric LTL
B(ϵ, i, r)
cost-parity
parity
finitary parity