Partial solvers for parity games: effective polynomial-time composition

Patrick Ah-Fat, Michael Huth

Imperial College London

Parity game

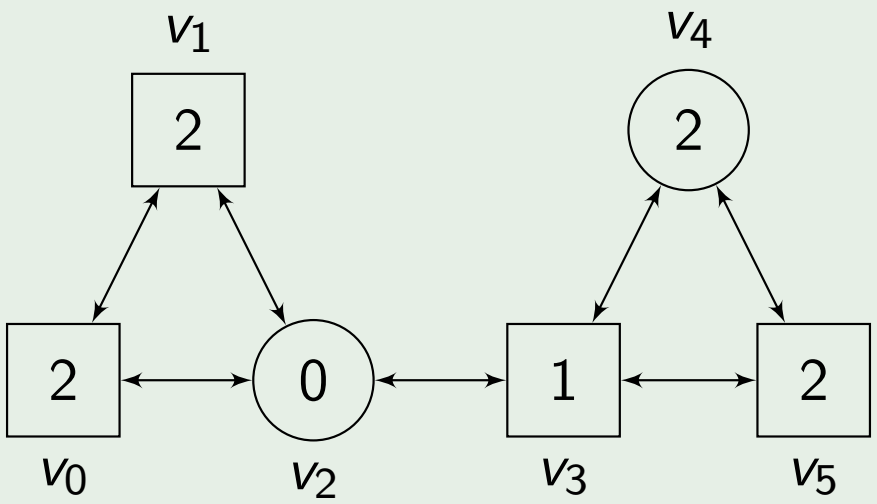


Figure 1 : Player 0 wins v_0, v_1 and v_2 while player 1 wins v_3, v_4 and v_5.

- Parity games are determined (Zielonka's algorithm ([Zie98]), ...)
- Is there a polynomial time solver ?

Background

Partial solvers and static analyses

- Partial solvers decide the winners of some nodes:
  - Fatal attractors removal ($psol_B$, [HKP13])
  - Alternating reachability under parity ($psol_C$, [HKP16])
- Static analyses reduce the complexity of a game:
  - Static color compression (*scc*)
  - Priority propagation (*pp*)
  - Abstract Rabin index reduction (*ari*, [HKP15])

Effectiveness VS efficiency (of $psol_B$)

- $psol_B$ solves *ladder* and *strategy improvement* benchmark games more *efficiently* than Zielonka's algorithm.
- $psol_B$ is also *effective* on *model checker ladder* benchmark games (i.e. runs in PTIME)

Example: Fatal attractors and $psol_B$

A set of nodes X of color c and parity p has a fatal attractor for color c iff: p can force from X to get to X again through nodes of color $\geq c$. Such an X is won by player p.

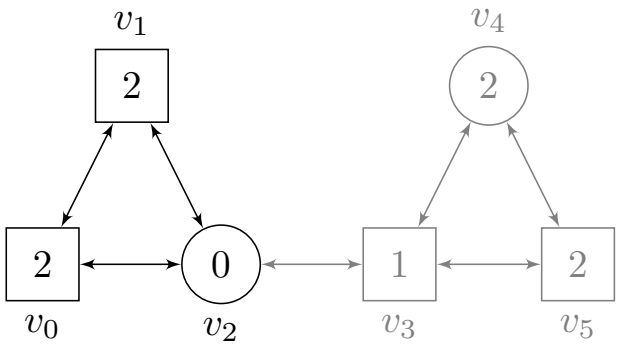
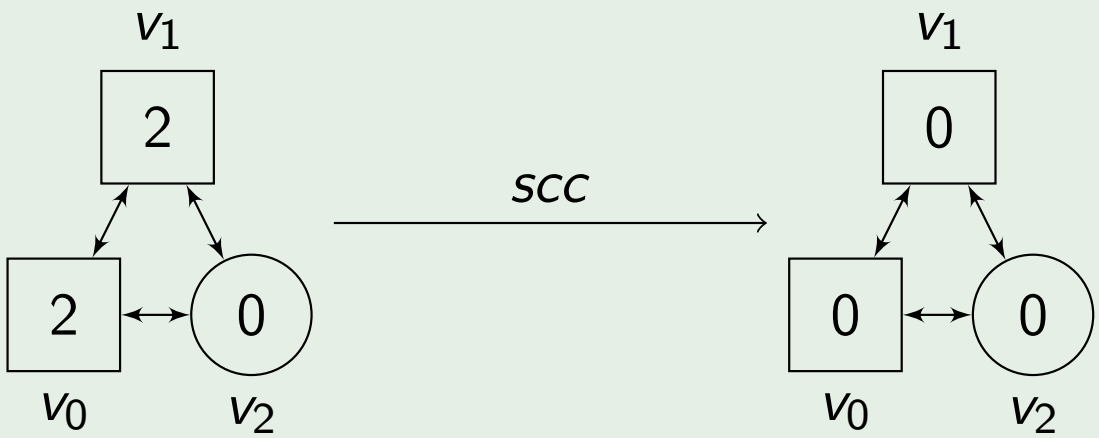


Figure 2 : Nodes v_3, v_4 and v_5 form a fatal attractor for color 1. We call the game composed of $\{v_0, v_1, v_2\}$ a *residual game* for *psolB*.

Residual game

A game G is called *residual* for partial solver *ps* if *ps* cannot simplify G.

Example: Composition effectiveness



Residual game for psol_B Game solved by psol_B

Figure 3 : Composition of *scc* and $psol_B$.

- $\{psol_B\}$ alone cannot solve game G whereas $\{scc, psol_B\}$ can
- Composition is effective
- Residual games can help designing new static analyses

Procedure and objectives

Our first partial solver was $ps_1 = while(f_1, \cdots, f_5)$ as presented below.

```
ps1(G){
  (flag, G') = f1(G); if (flag){ps1(G');}
  (flag, G') = f2(G); if (flag){ps1(G');}
  ...
  (flag, G') = f5(G); if (flag){ps1(G');}
  return }
```

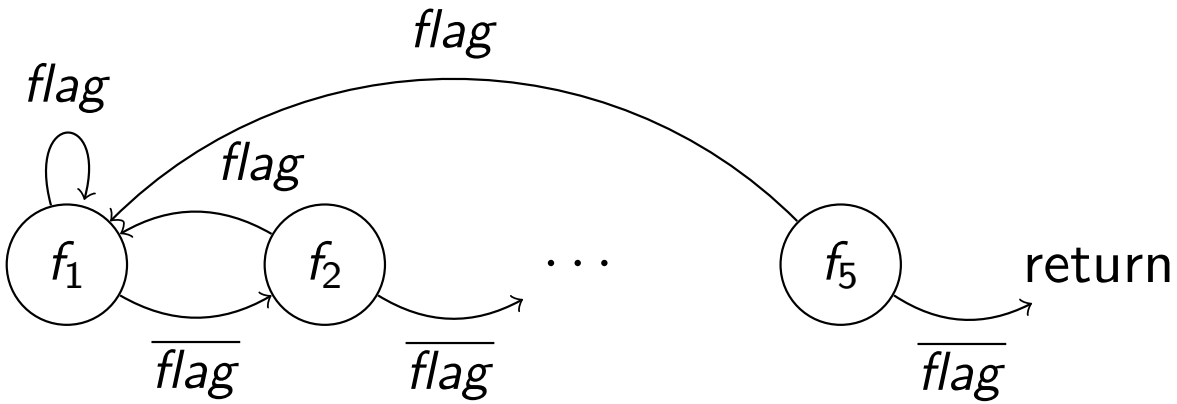


Figure 4 : Composition pattern between analyses.

Procedure and objectives

Methodology

- Consider partial solver $ps_1 = while(f_1, \cdots, f_5)$
- Generate and manually study residual games for ps_1 (the *data*)
- Invent static analysis f_6
- Restart with $ps_2 = chain(ps_1, f_6)$

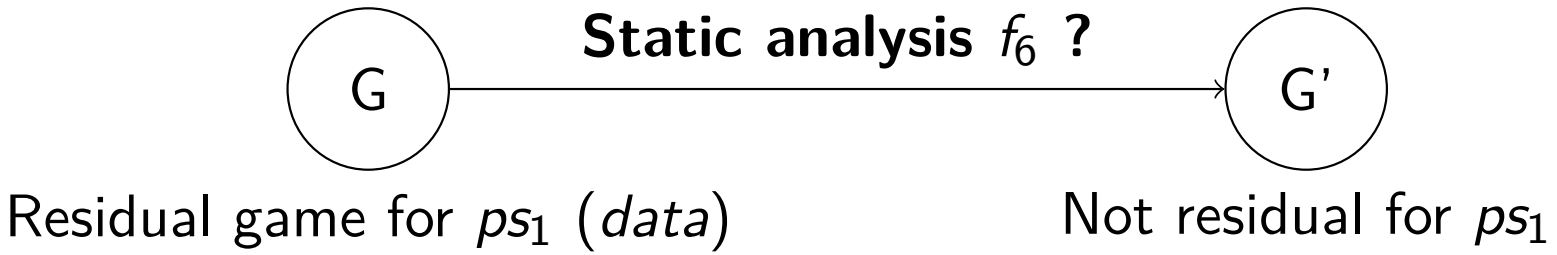


Figure 5 : Data-driven approach given partial solver ps_1 and residual game G.

Procedure and objectives

New static analyses

	Node merging	**Corresponding partial solver**
m_{ss}	Sole successors node merging	$ps_2 = while(ps_1, m_{ss})$
m_{scc}	SCC node merging	$ps_3 = while(ps_2, m_{scc})$
	Edge removal	
er_{fa}	Edge removal based on conditional fatal attractors	$ps_4 = while(ps_3, er_{fa})$
er_{sd}	Edge removal based on shared descendant	$ps_5 = while(ps_4, er_{sd})$

Figure 6 : Notations of the different new static analyses we produced.

Note

We also studied variants and generalisations of those static analyses.

Example: SCC nodes merging (m_{scc})

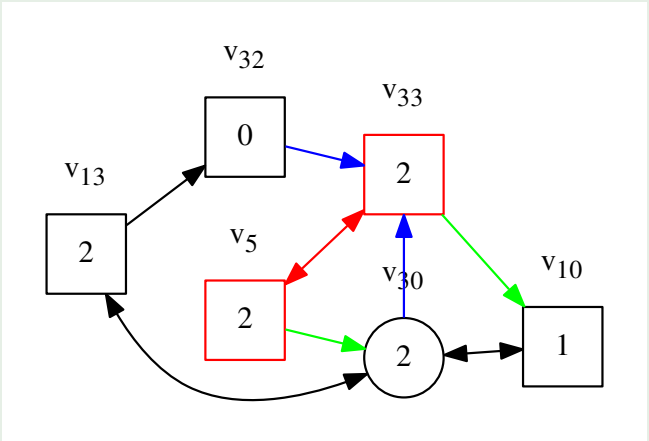
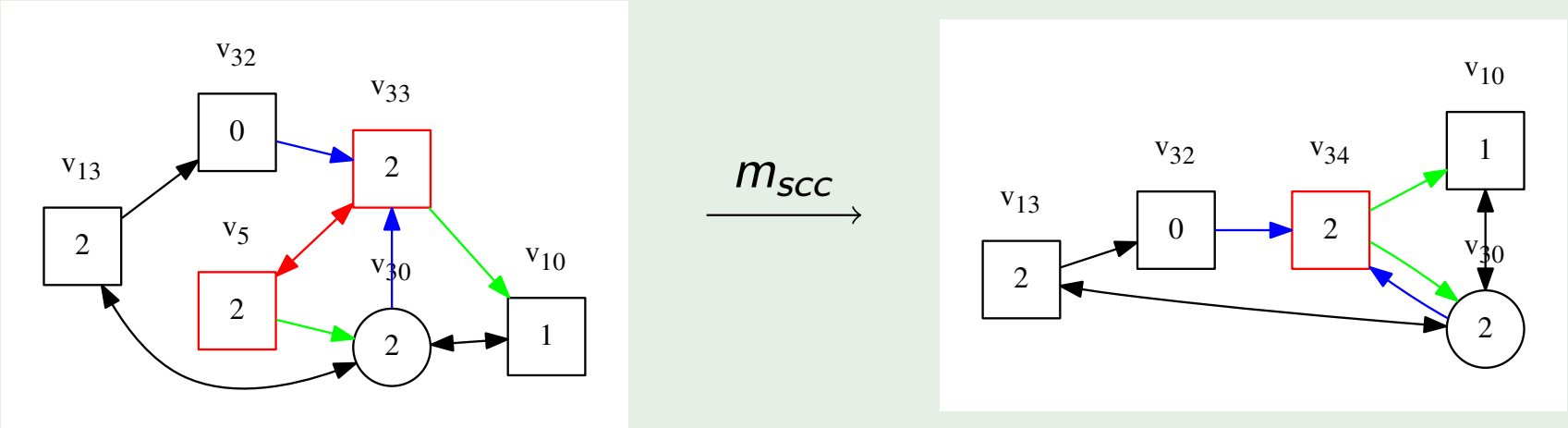


Figure 7 : Example of residual game for ps_2.

- Observation: Nodes v_5 and v_{33} have the same winner and same color.
- Conclusion (m_{scc}): Nodes v_5 and v_{33} can be merged together.

Example: SCC nodes merging (m_{scc})



G: Residual game for ps_2

G': Game solved by ps_2

Figure 8 : Effectiveness of node merging m_{scc}.

Example: Edge removal based on conditional fatal attractors (er_{fa})

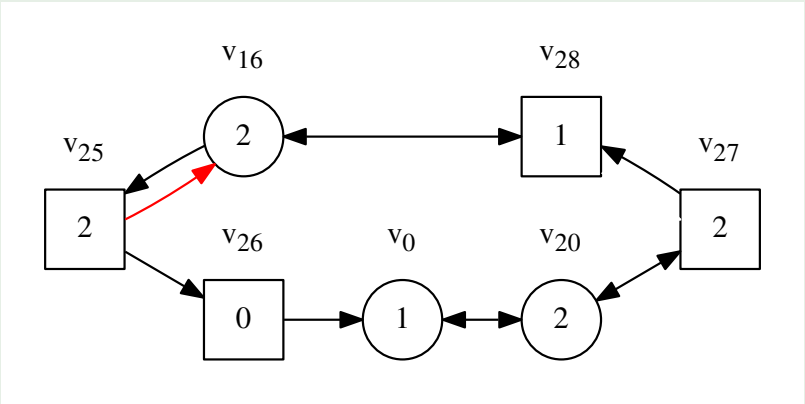
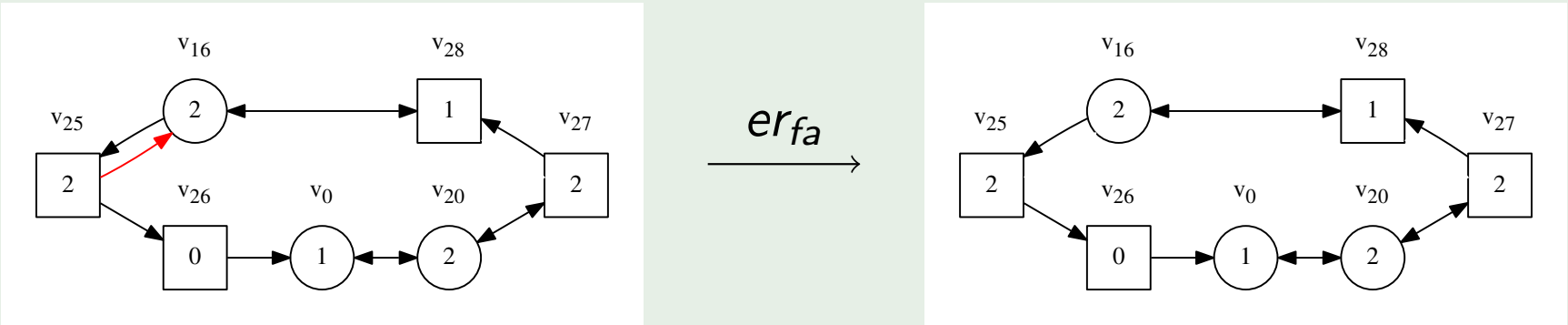


Figure 9 : Example of residual game for ps_3.

- Observation: Committing to edge (v_{25}, v_{16}) creates a fatal attractor for player 0 and color 2.
- Conclusion (er_{fa}): Edge (v_{25}, v_{16}) can be removed.

Example: Edge removal based on conditional fatal attractors (er_{fa})



G: Residual game for ps_3 G': Game solved by ps_3

Figure 10 : Effectiveness of edge removal er_{fa}.

Experimental Results

A theoretical result

Theorem 1

Partial solver $\mathit{while}(\mathit{scc}', \mathit{ari}, \mathit{fa})$ solves all 1-player games.

- scc': Static color compression on SCC's
- ari: Abstract Rabin index reduction
- fa: Fatal attractors removal

Experimental Results

Effectiveness tests

Model Checking	MLSolver
Alternating Bit Protocol (ABP) Concurrent Alternating Bit Protocol (CABP) Positive Acknowledgement with Retransmission Protocol (PAR) Bounded Retransmission Protocol (BRP) Sliding Window Protocol (SWP) Cache Coherence Protocol (CCP) Leader Election Protocol Domineering Snake Lift Elevator FIFO Elevator LIFO Hanoi	CTL Star Binary Counter Demri Killer Formula Fair Scheduler FLCTL Limit Closure Include LT Mucalc Binary Counter Mu Calc Limit Closure Nester Parity And Buechi PDL Binary Counter Petri Star Nester
Equivalence checking	**PGSolver**
Alternating Bit Protocol (ABP) Concurrent Alternating Bit Protocol (CABP) Onebit Sliding Window Protocol Positive Acknowledgement with Retransmission Protocol (PAR) Sliding Window Protocol (SWP)	Random Games Clustered Random Games Steady Random Games Clique Games Ladder Games Jurdzinski Games Recursive Ladder Games Model Checker Ladder Games Tower of Hanoi Fairness Verification of an Elevator System

Figure 11 : Keiren's benchmark suite ([Kei15]): types of games that ps_5 solved.

Experimental Results

Effectiveness tests

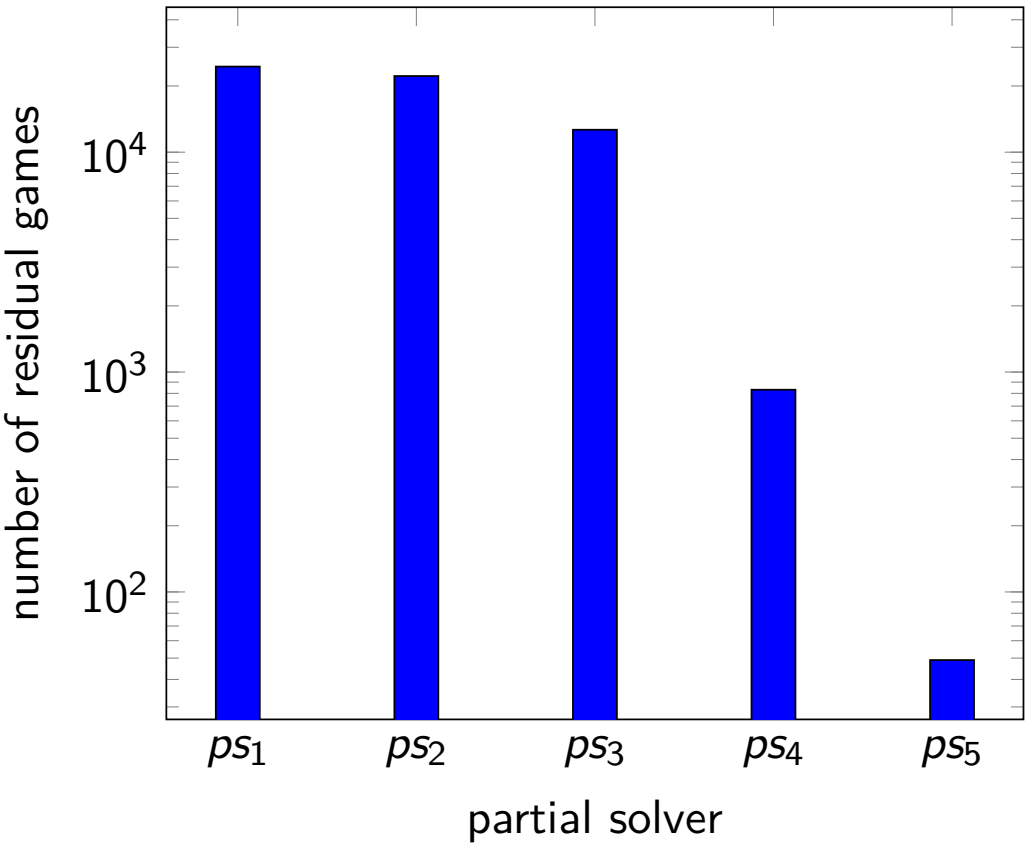


Figure 12 : Frequency of residual games for our partial solvers. We have generated 13 800 347 games under the configuration $50 - 25 - 2 - 4$.

Experimental Results

Results for ps_5

configuration	time (s)	games generated	residual games for ps_5	residual games for $lift(ps_5)$
1000-300-2-4	1 879 881	3 079 306	0	0
200-50-1-2	214 621	406 768	22	0
200-50-1-3	214 691	507 032	44	0
200-50-2-3	214 734	361 253	99	0
200-50-2-4	214 771	4 600 227	43	0
30-15-2-4	2 117 308	2 723 701 031	2 023	0
40-20-2-4	2 117 288	1 466 037 789	3 040	0
50-25-2-4	2 117 273	862 630 173	3 328	0
55-27-2-4	2 117 217	658 895 399	3 119	0
60-30-2-4	2 117 250	528 227 035	3 084	0

Future Works

The following questions might be worth studying in more details:

- What types of games are solved by a given partial solver ?
- What properties do the residual games for a given partial solver have ?
- Can we extend Theorem 1 to interesting classes of 2-player games ?

Conlusion

- We studied and applied composition between known partial solvers.
- We followed a data-driven approach and proposed new static analyses.
- We built effective polynomial time partial solvers that appear to solve all structured benchmarks.
- We still seek further mathematical insights into partial solvers composition.

Michael Huth, Jim Huan-Pu Kuo, and Nir Piterman.

Fatal attractors in parity games.

In *International Conference on Foundations of Software Science and Computational Structures*, pages 34–49. Springer, 2013.

Michael Huth, Jim Huan-Pu Kuo, and Nir Piterman.

The rabin index of parity games: Its complexity and approximation.

Information and Computation, 2015.

Michael Huth, Jim Huan-Pu Kuo, and Nir Piterman.

Static analysis of parity games: Alternating reachability under parity.

In *Semantics, Logics, and Calculi*, pages 159–177. Springer, 2016.

Jeroen JA Keiren.

Benchmarks for parity games.

In *International Conference on Fundamentals of Software Engineering*, pages 127–142. Springer, 2015.

Wieslaw Zielonka.

Infinite games on finitely coloured graphs with applications to automata on infinite trees.

Theoretical Computer Science, 200(1):135–183, 1998.