Tailoring behavioural equivalences to parity games

Jeroen Keiren
j.j.a.keiren@tue.nl
http://www.win.tue.nl/~jkeiren

Technische Universiteit Eindhoven
University of Technology

Where innovation starts

Outline

TU/e Technische Universiteit Eindhoven University of Technology

Parity game

Definition (Parity game)

A parity game $\mathcal{G}$ is a tuple $(V_{Even}, V_{Odd}, \rightarrow, \Omega)$, where:

- $V = V_{Even} \cup V_{Odd}$; $V_{Even} \cap V_{Odd} = \emptyset$;
- $(V, \rightarrow)$ is a total, directed graph;
- $\Omega : V \rightarrow \mathbb{N}$ assigns priority to vertices.

Play:

- Place token on vertex;
- Player *Even* does step if token is on $v \in V_{Even}$ (likewise for *Odd*);
- A play is an infinite sequence of such steps;
- Player *Even* wins play iff least priority that occurs infinitely often is even.

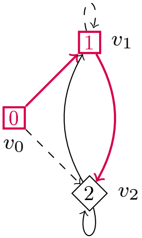
Technische Universiteit Eindhoven University of Technology

Strategy

Definition (Stategy & Winning)

A strategy for player *Odd* is a function σ_{Odd}:$V_{Odd} \rightarrow V$, with $\sigma \subseteq \rightarrow$. Player *Odd* wins from vertex $v \in V$ if he has a strategy such that he wins every play starting in v, regardless of his opponent's strategy.

Example (Strategy)

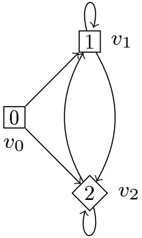


- Strategy for player *Odd* (in red);
- *Odd* does not win if *Even* plays $v_2 \rightarrow v_2$.

TU/e Technische Universiteit Eindhoven University of Technology

Example

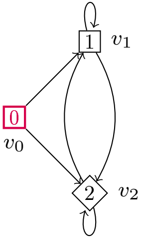
Example (Parity game)



Technische Universiteit Eindhoven University of Technology

Example

Example (Parity game)

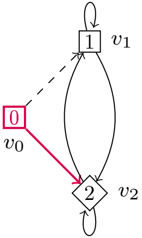


- Can player *Odd* win every play from v_0?

Technische Universiteit Eindhoven University of Technology

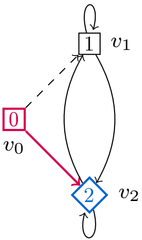
Example

Example (Parity game)



- Can player *Odd* win every play from v_0?

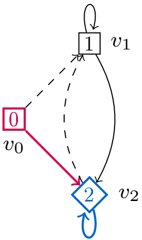
Example (Parity game)



- Can player *Odd* win every play from v_0?

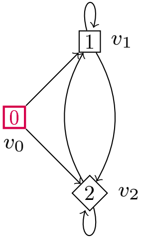
Example

Example (Parity game)



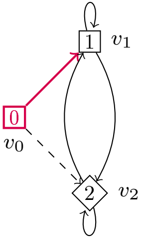
- Can player *Odd* win every play from v_0?

Example (Parity game)



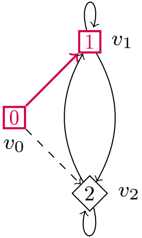
- Can player *Odd* win every play from v_0?

Example (Parity game)



- Can player *Odd* win every play from v_0?

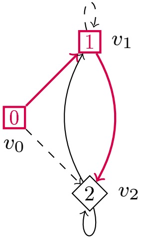
Example (Parity game)



- Can player *Odd* win every play from v_0?

Example

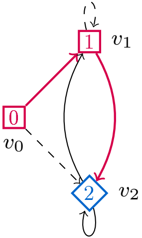
Example (Parity game)



- Can player *Odd* win every play from v_0?

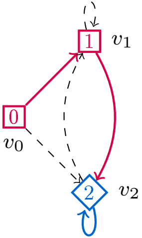
Example

Example (Parity game)



- Can player *Odd* win every play from v_0?

Example (Parity game)

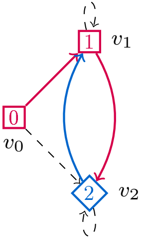


- Can player *Odd* win every play from v_0?

TU/e Technische Universiteit Eindhoven University of Technology

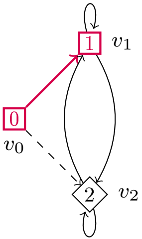
Example

Example (Parity game)



- Can player *Odd* win every play from v_0?

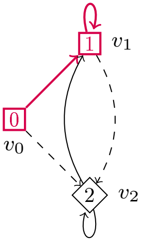
Example (Parity game)



- Can player *Odd* win every play from v_0?

Example

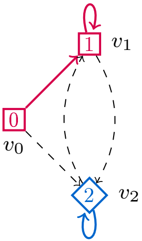
Example (Parity game)



- Can player *Odd* win every play from v_0?
- Yes, given the strategy in red!

Example

Example (Parity game)



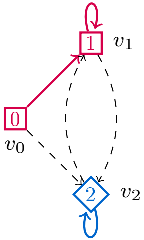
- Can player *Odd* win every play from v_0?
- Yes, given the strategy in red!
- Player *Even* can win every play from v_2.

Determining winning sets

Goal

Partition V into W_{Even}, W_{Odd}, the vertices won by player *Even* or *Odd*.

Example (Parity game)



Winning sets:

- $W_{Even} = \{v_2\}$
- $W_{Odd} = \{v_0, v_1\}$

Coloured edges provide winning strategy.

TU/e Technische Universiteit Eindhoven University of Technology

Parity games and model checking

Model checking

μ-calculus model checking problem: answer $L \models f$, where

- L is a labelled transition system;
- f a μ-calculus formula.

Solving model checking problem $\equiv$ finding winner in parity game.

- Known algorithms for solving μ-calculus model checking problem, and for finding winner in parity game are exponential;
- Best known algorithm for solving parity games (bigstep) has worst case complexity of $\mathcal{O}(mn^{\frac{c}{3}})$.

Technische Universiteit Eindhoven University of Technology

Motivation

Typical model checking approach:

1. Find abstraction of model;
2. Minimise abstract model using behavioural equivalence;
3. Model check minimised model.

Drawbacks:

- Requires human intellect;
- Requires different abstractions for verifying different properties.

Proposed approach

1. Encode model checking problem as parity game;
2. Minimise parity game;
3. Solve minimised parity game.

Observation

Parity games are similar to Kripke structures, but have a restricted set of labels.

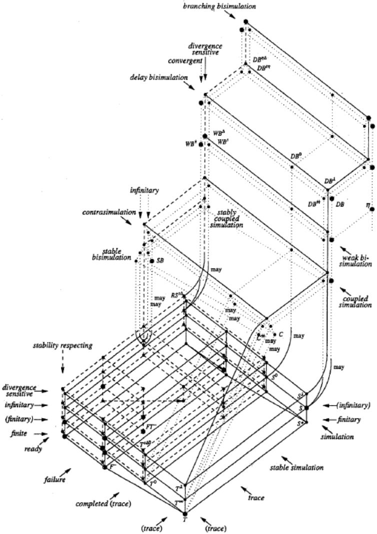
The coarsest meaningful equivalence for parity games is winner equivalence:

$$v \sim_w v' \text{ iff } v \in W_{Even} \Leftrightarrow v' \in W_{Even}$$

Question

Can we use (known) behavioural equivalences to reduce parity games, while preserving the winning sets?

Van Glabbeek's lattice



Strong bisimulation

Requirements

Equivalence for parity games should

- be efficiently (polynomial time) computable;
- preserve winning sets.

Definition (Strong bisimulation)

Let $\mathcal{G} = (V_{Even}, V_{Odd}, \rightarrow, \Omega)$ be a parity game. Symmetric relation R is a strong bisimulation relation if vRv' implies:

- $\Omega(v) = \Omega(v')$;
- v and v' are owned by the same player;
- if $v \rightarrow w$, then there should be w' s.t. $v' \rightarrow w'$ and wRw'.

$v \sim_{sb} v'$ iff there exists a strong bisimulation relation R s.t. vRv'.

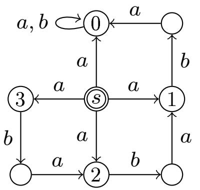
Technische Universiteit Eindhoven University of Technology

Example

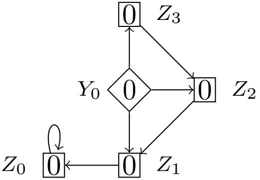
Example (Encoded model checking problem)

Consider property $\phi = \nu Y.\langle a\rangle([a]\mathit{false} \wedge \nu Z.[b]\langle a\rangle Z)$

Informally: after an a action, following a b action will always enable an a action.



LTS L

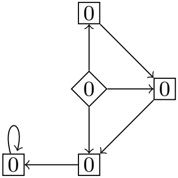


Parity game encoding $L \models \phi$

Technische Universiteit Eindhoven University of Technology

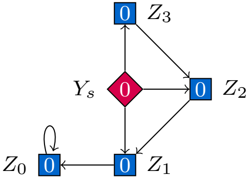
Example (continued)

Example (Strong bisimulation)

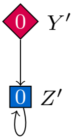


Parity game

Example (Strong bisimulation)



Parity game with two bisimulation equivalence classes



Minimal bisimulation equivalent parity game

Observe: Y' and Z' both won by player *Even*, but not equivalent.

Question

Can we do better than this?

Definition (Forced-move identifying bisimulation)

Let $\mathcal{G} = (V_{Even}, V_{Odd}, \to, \Omega)$ be a parity game. Symmetric relation R is a forced-move identifying bisimulation relation if vRv' implies:

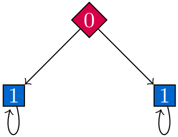
- $\Omega(v) = \Omega(v')$;
- v and v' are not owned by the same player implies for all w, w' such that $v \to w$ and $v' \to w'$: wRw';
- if $v \to w$, then there should be w' s.t. $v' \to w'$ and wRw'.

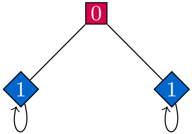
$v \sim_{fb} v'$ iff there exists a forced-move identifying bisimulation relation R s.t. vRv'.

Example

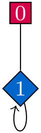
Example (Forced-move identifying bisimulation)

Two parity games minimal using strong bisimulation:





Both forced-move identifying bisimilar to the following parity game:



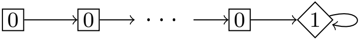
Technische Universiteit Eindhoven University of Technology

$\sim_{sb}$, $\sim_{fb}$ and winning games

- Strong bisimulation is strictly finer than forced-move identifying bisimulation;
- $v \sim_{fb} v'$ implies $v \sim_w v'$;
- Both relations can be decided in $\mathcal{O}(n \log n)$ time;
- Reduction is effective in practice.

Observation

After reduction, often parity games with the following structure:

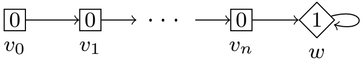


All nodes have the same winner (Odd)

Stuttering equivalence

- Stuttering equivalence is state based counterpart of divergence sensitive branching bisimulation;
- Allows relating sequences of nodes with same potential;
- Requires same priority and same player.

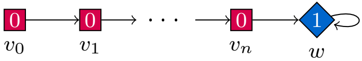
Example



TU/e Technische Universiteit Eindhoven University of Technology

- Stuttering equivalence is state based counterpart of divergence sensitive branching bisimulation;
- Allows relating sequences of nodes with same potential;
- Requires same priority and same player.

Example



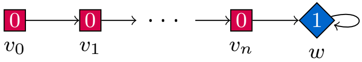
All priority 0 nodes can be related.

TU/e Technische Universiteit Eindhoven University of Technology

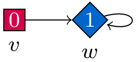
Stuttering equivalence

- Stuttering equivalence is state based counterpart of divergence sensitive branching bisimulation;
- Allows relating sequences of nodes with same potential;
- Requires same priority and same player.

Example



All priority 0 nodes can be related.



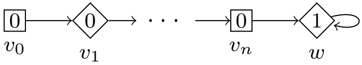
Technische Universiteit Eindhoven University of Technology

Stuttering equivalence with forced moves

- Stuttering equivalence does not relate vertices of different players;
- Combine stuttering equivalence and forced-move identifying bisimulation.

Example

Parity game minimal w.r.t stuttering equivalence.



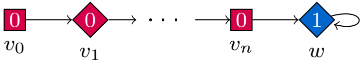
Technische Universiteit Eindhoven University of Technology

Stuttering equivalence with forced moves

- Stuttering equivalence does not relate vertices of different players;
- Combine stuttering equivalence and forced-move identifying bisimulation.

Example

Parity game minimal w.r.t stuttering equivalence.



Using forced-move identifying stuttering equivalence all priority 0 nodes can be related.

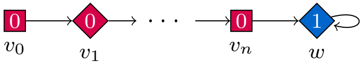
TU/e Technische Universiteit Eindhoven University of Technology

Stuttering equivalence with forced moves

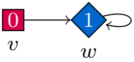
- Stuttering equivalence does not relate vertices of different players;
- Combine stuttering equivalence and forced-move identifying bisimulation.

Example

Parity game minimal w.r.t stuttering equivalence.



Using forced-move identifying stuttering equivalence all priority 0 nodes can be related.



Experiments

- Strong- and forced-move identifying bisimulation reduction is useful in practice [KW09];
- Preliminary experiments: stuttering equivalence even more effective.

Property I: If sending some message is enabled infinitely often, then it is sent infinitely often.

Model	Orig.	$\sim_{sb}$	$\sim_{fb}$	$\sim^{st}$	$\sim^{st}_{fb}$
ABP (16)	88,307	223	210	116	$\leq$71
ABP (32)	352,739	223	210	116	$\leq$71
CABP (16)	1,732,627	1,238	1,237	13	11
SWP (4)	8,957,959	25,354	25,353	13	11

Technische Universiteit Eindhoven University of Technology

Experiments (continued)

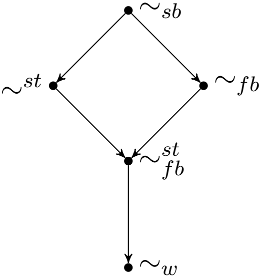
Property II: All messages can be sent infinitely often.

Model	Orig.	$\sim_{sb}$	$\sim_{fb}$	$\sim^{st}$	$\sim^{st}_{fb}$
CABP (16)	579,379	406	405	10	5
SWP (4)	3,095,565	9,100	9,099	10	5

Property III: Branching bisimilarity of two specifications.

Model 1	Model 2	Orig.	$\sim_{sb}$	$\sim_{fb}$	$\sim^{st}$
ABP (16)	CABP (16)	4,054,706	21,329	21,311	17,483
ABP (2)	SWP (2)	1,864,138	434,820	434,037	361,721

Conclusions



- Lattice of equivalences for minimising parity games;
- Potentially more effective than reduction of LTS;
- Efficiently computable ($\mathcal{O}(n \log n)$, $\mathcal{O}(mn)$);
- Rewarding in practice: solving original game greatly exceeds time for reduction + solving reduced game.

Future work/Open problems

- Give algorithm for forced-move identifying bisimulation;
- Investigate applicability of other relations, e.g. simulation equivalence;
- Determine classes of parity games characterised by our equivalences;
- Find equivalences that characterise parity game encodings of logics (e.g. LTL, CTL*)

Technische Universiteit Eindhoven University of Technology