Tailoring behavioural equivalences to parity games

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

1. Introduction
2. Motivation
3. Equivalences for parity games
4. Experimental results
5. Outlook
Parity game

Definition (Parity game)

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

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

Play:

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

A strategy for player *Odd* is a function $\sigma_{Odd}:V_{Odd} \to V$, with $\sigma \subseteq \to$. 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 $v_2 \to v_2$. 
Example (Parity game)
Example (Parity game)

- Can player *Odd* win every play from *v₀*?
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 (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 (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 $v_0$?
- Yes, given the strategy in red!
Example (Parity game)

- Can player \textit{Odd} win every play from $v_0$?
- Yes, given the strategy in red!
- Player \textit{Even} can win every play from $v_2$. 

![Diagram](image-url)
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.
Model checking

\( \mu \)-calculus model checking problem: answer \( L \models f \), where

- \( L \) is a labelled transition system;
- \( f \) a \( \mu \)-calculus formula.

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

- Known algorithms for solving \( \mu \)-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 \( \mathcal{O}(mn^3) \).
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:

\[ v \sim_w v' \text{ iff } v \in W_{Even} \iff v' \in W_{Even} \]

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. $vRv'$. 
Example (Encoded model checking problem)

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

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

LTS $L$

Parity game encoding $L \models \phi$
Example (Strong bisimulation)

Parity game
Example (continued)

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 $G = (V_{Even}, V_{Odd}, \rightarrow, \Omega)$ 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 $v \rightarrow w$ and $v' \rightarrow w'$: $wRw'$;
- if $v \rightarrow w$, then there should be $w'$ s.t. $v' \rightarrow w'$ and $wRw'$.

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

Two parity games minimal using strong bisimulation:

Both forced-move identifying bisimilar to the following parity game:
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 \( O(n \log n) \) time;

Reduction is effective in practice.

Observation

After reduction, often parity games with the following structure:

\[
\begin{array}{ccccccccc}
0 & \rightarrow & 0 & \rightarrow & \cdots & \rightarrow & 0 & \rightarrow & 1 \\
\end{array}
\]

All nodes have the same winner (Odd)
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:

```
0 0 ... 0 1
v0 v1 ... vn w
```
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.
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.
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.
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.
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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Orig.</th>
<th>$\sim_{sb}$</th>
<th>$\sim_{fb}$</th>
<th>$\sim_{st}$</th>
<th>$\sim_{st}^{fb}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABP (16)</td>
<td>88,307</td>
<td>223</td>
<td>210</td>
<td>116</td>
<td>$\leq 71$</td>
</tr>
<tr>
<td>ABP (32)</td>
<td>352,739</td>
<td>223</td>
<td>210</td>
<td>116</td>
<td>$\leq 71$</td>
</tr>
<tr>
<td>CABP (16)</td>
<td>1,732,627</td>
<td>1,238</td>
<td>1,237</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>SWP (4)</td>
<td>8,957,959</td>
<td>25,354</td>
<td>25,353</td>
<td>13</td>
<td>11</td>
</tr>
</tbody>
</table>
Experiments (continued)

Property II: All messages can be sent infinitely often.

<table>
<thead>
<tr>
<th>Model</th>
<th>Orig.</th>
<th>$\sim_{sb}$</th>
<th>$\sim_{fb}$</th>
<th>$\sim_{st}$</th>
<th>$\sim_{st_{fb}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CABP (16)</td>
<td>579,379</td>
<td>406</td>
<td>405</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>SWP (4)</td>
<td>3,095,565</td>
<td>9,100</td>
<td>9,099</td>
<td>10</td>
<td>5</td>
</tr>
</tbody>
</table>

Property III: Branching bisimilarity of two specifications.

<table>
<thead>
<tr>
<th>Model 1</th>
<th>Model 2</th>
<th>Orig.</th>
<th>$\sim_{sb}$</th>
<th>$\sim_{fb}$</th>
<th>$\sim_{st}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABP (16)</td>
<td>CABP (16)</td>
<td>4,054,706</td>
<td>21,329</td>
<td>21,311</td>
<td>17,483</td>
</tr>
<tr>
<td>ABP (2)</td>
<td>SWP (2)</td>
<td>1,864,138</td>
<td>434,820</td>
<td>434,037</td>
<td>361,721</td>
</tr>
</tbody>
</table>
Conclusions

- Lattice of equivalences for minimising parity games;
- Potentially more effective than reduction of LTS;
- Efficiently computable ($O(n \log n)$, $O(mn)$);
- 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*)