

Computing safe winning regions of parity games in polynomial time

Adam Antonik, Nathaniel Charlton, and **Michael Huth**

Department of Computing
Imperial College London

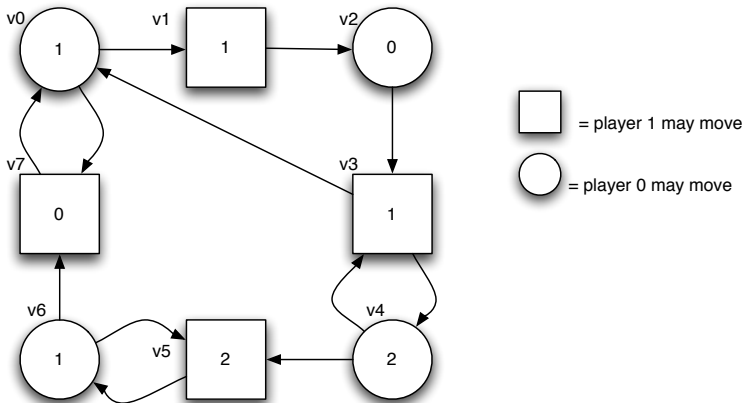
Talk for MFCSIT'06, 2 August 2006
Cork, Ireland

- 1 Introduction
- 2 Our approach
- 3 Conclusions and future work

Parity games G

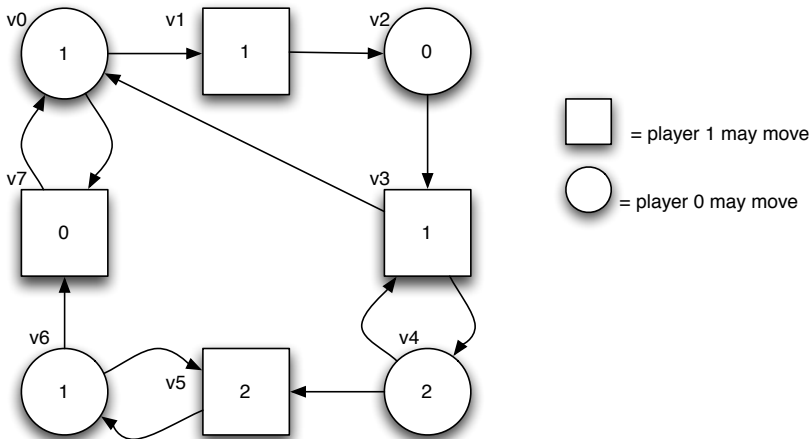
- parity games specify sets of finite and infinite plays between two players 0 and 1 on directed graph (V, E)
- each node v labelled with priority $\chi(v)$ in $\{0, 1, \dots, d - 1\}$
- winning condition for infinite plays derived from priorities
- finite plays end in dead end, node v with no $(v, v') \in E$; lost by player who owns v
- parity games are determined:
 - V disjoint union of Win_0 and Win_1
 - player 0 (resp. 1) has memoryless winning strategy with which she wins all plays starting in Win_0 (resp. Win_1)

Example [Küsters'02]: an infinite play



- infinite play: $v_7 v_0 v_7 v_0 \dots = (v_7 v_0)^\omega$
- $\{0, 1\}$ set of priorities that occur infinitely often in that play
- play won by player 1 as $\max\{0, 1\}$ is odd

Example [Küsters'02]: winning regions



- $\text{Win}_0 = \{v_4, v_5, v_6\}$ and $\text{Win}_1 = \{v_0, v_1, v_2, v_3, v_7\}$
- winning strategy for player 0 plays $v_6 \rightsquigarrow v_5$ and $v_4 \rightsquigarrow v_5$

Motivation

- complexity: “ $v \in \text{Win}_0$?” in $\text{UP} \cap \text{coUP}$, not known to be in P
- algorithms that compute Win_0 : either not in P (exponential worst-case inputs exist) or have unknown complexity

Our **complementary approach**:

Present design pattern for algorithms that are in P by construction but under-approximate Win_0 and Win_1 through efficient and sound static analysis.

Subsequent convention: σ either 0 or 1 and $\bar{\sigma} = 1 - \sigma$.

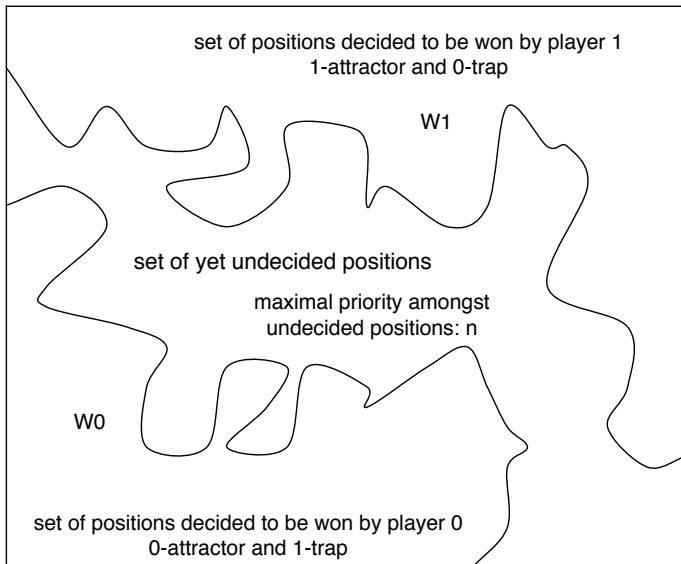
Design pattern

```
W0 = all positions at which 0 can force dead end for 1;  
W1 = all positions at which 1 can force dead end for 0;  
cache = 0;  
while (rank(G) != cache) {  
    cache = rank(G);  
    for (i=1; i++; i <= k) { A#i; }  
}  
return (W0,W1);
```

- Instantiation of pattern: **rank** function and **aspects** $A\#i$ in P
- Aspects **change state of G** , basic invariant of pattern:

For all $\sigma \in \{0, 1\}$: set Win_σ won't change; set W_σ is $\bar{\sigma}$ -trap, σ -attractor, and subset of Win_σ .

Picture of basic invariant



Four aspects, part 1

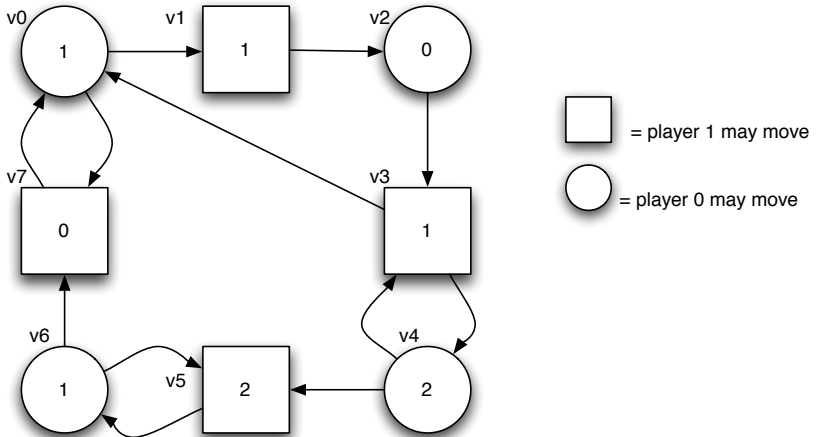
- **A1:** $v \in W \setminus (W_0 \cup W_1)$, $2 \leq \chi(v)$, and no cycle in (V, E) through v with node w satisfying $\chi(w) = \chi(v) - 1$
 $\Rightarrow \chi(v) := \chi(v) - 2$
- **A2:** $v \in W \setminus (W_0 \cup W_1)$, all cycles in (V, E) through v have node w with $\chi(v) \leq \chi(w)$
 $\Rightarrow \chi(v) := 0$
- **A4:** $v \in W \setminus (W_0 \cup W_1)$, all cycles in (V, E) through v have node w_C with $\chi(v) < \chi(w_C)$
 $\Rightarrow \chi(v) := \min_C \chi(w_C) \quad // \text{ increases } \chi(v)$

Four aspects: A3

Abstract interval of odd (resp. even) priorities with single odd (resp. even) priority:

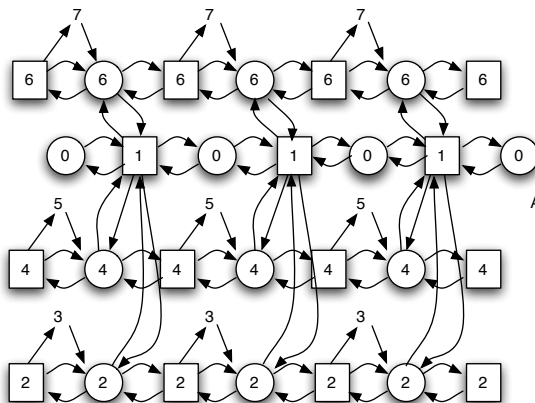
- $n = \max\{\chi(v) \mid v \in V \setminus (W_0 \cup W_1)\}$
- $\sigma = n \bmod 2$
- add to W_σ all $v \in V \setminus (W_0 \cup W_1)$ from which σ can force priority n infinitely often in **sub-game** $V \setminus W_\sigma$
- add to $W_{\bar{\sigma}}$ all $v \in V \setminus (W_0 \cup W_1)$ from which $\bar{\sigma}$ can force priority $n - 1$ infinitely often while restricting n to occur finitely often in sub-game $V \setminus W_{\bar{\sigma}}$
- add to W_σ all $v \in V \setminus (W_0 \cup W_1)$ from which σ can force priority n or $n - 2$ infinitely often while restricting $n - 1$ to occur finitely often in sub-game $V \setminus W_\sigma$
- etc

Example [Küsters'02]



- Design pattern instantiated with $k = 1$ and **A3** alone computes $W_\sigma = \text{Win}_\sigma \Rightarrow$ **game solved completely.**

Example [Jurdziński'00]

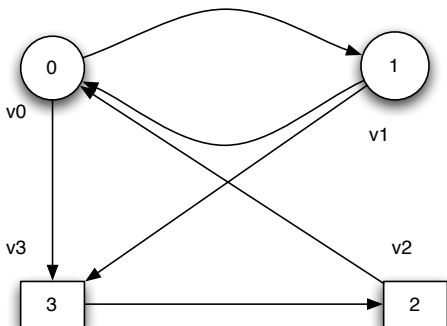


A3: abstractions $\{7,5,3,1\} = 1$
 and $\{6,4,2\} = 0$ reveal that
 all positions with priorities
 0 or 1 are won by player 1

- Algorithm in [Jurdziński'00] **exponential** for this game.¹
- Pattern instantiated with $k = 1$ and **A3** solves entire game.

¹Shown in loc. cit. for **min** parity acceptance condition.

Interaction of aspects



- Running **A3** alone here returns $W_\sigma = \{\}$.
- Pattern instantiated with $k = 2$, $A\#1 = \mathbf{A2}$ and $A\#1 = \mathbf{A3}$:
 - **A2** resets $\chi(v_2)$ from 2 to 0
 - eventually **A3** abstracts $\{1, 3\}$ to 1 and $\{0, 2\}$ to 0
 - this abstraction determines $W_1 = V$, solves game completely.

Future work

- Explore additional aspects and rank functions.
- Systematic account of interaction of these aspects:
 - connections to complexity measures:
 - **DAG-Width** [Berwanger et al.'06]
 - **Entanglement** [Berwanger & Grädel'04]
 - recommended pattern instantiations
- Experimental results and their comparison to reducing parity games to SAT [Lange'05].

Conclusions

- Proposed design pattern for **under-approximating winning regions** of parity games **in polynomial time**.
- Pattern instantiated with k many aspects, presented four.
- Demonstrated that **aspect interaction increases precision**.
- Utility of approach:
 - **local model checking**: only interested in whether initial state (designated node) won by player 0
 - our algorithms can be seen as **pre-processors** as W_σ can be abstracted to single node due to basic invariants.