Small Progress Measures for Solving Parity Games

Marcin Jurdziński
BRICS
University of Aarhus
Denmark

Lille, France, 18 February 2000

Plan

1. Verification of (finite-state systems) by model checking
2. Solving parity games
3. Summary of results on the complexity of solving parity games
4. The new algorithm for solving parity games
   (a) Progress measures: witnesses for winning strategies
   (b) Least progress measures: existence and computing
   (c) Worst-case behaviour

Model checking for the modal μ-calculus

Verification of (finite-state) systems by model checking
[Clarke, Emerson 1982; Queille, Sifakis 1982; Emerson, Lei 1986]

Given:

- (a model of) a finite-state system: a Kripke structure $\mathcal{K}$
- (a description of) a property: a modal μ-calculus formula φ

Decide: whether $\mathcal{K} \models \varphi$ holds

Modal μ-calculus [Kozen 1983]

- very expressive
- good compromise between succinctness and complexity
- low-level formalism of many automatic model checking tools

Model checking games

Theorem [Emerson, Jutla, Sistla 1993; Stirling 1995]
The following problems are linear-time equivalent:

- modal μ-calculus model checking
- non-emptiness of parity automata on infinite trees
- solving parity games

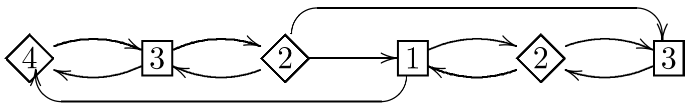
Model checking: does $\mathcal{K} \models \varphi$ hold?	$\longmapsto$	Solving a parity game: who is the winner in $\mathcal{G}_{\mathcal{K},\varphi}$?

reduction in time $O\big(|\mathcal{K}| \cdot |\varphi|\big)$

Parity games

Parity game $G = (V, (V_\diamond, V_\Box), E, p)$ $\quad p : V \to \{1, 2, \ldots, d\}$

Example



Play: path $\pi = \langle v_1, v_2, \ldots, v_\ell \rangle$ "closing" a cycle ($v_k = v_\ell$ for $k < \ell$)

Value of a play: $\mathtt{Val}(\pi) = \min \{ p(v_i) : k < i \leq \ell \}$

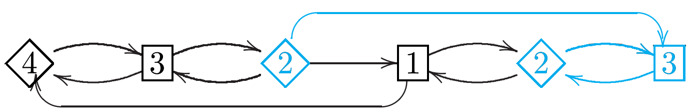
Winning play for player $\diamond$: $\mathtt{Val}(\pi)$ is even

Strategies

(Memoryless) strategy for $\diamond$: a subgraph (W, F) of (V, E), s.t.

- for all $v \in W \cap V_\diamond$, there is some $(v, w) \in F$
- for all $v \in W \cap V_\Box$, for all $(v, w) \in E$ we have $(v, w) \in F$

Example

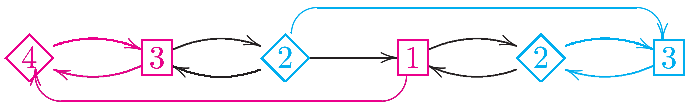


Winning strategy for $\diamond$: all cycles in (W, F) are "even"

Solving parity games

Memoryless Determinacy Thm [Emerson, Jutla; Mostowski 1991]
For every parity game $G = \big(V, (V_\Diamond, V_\Box), E, p\big)$,
there is a unique partition $(W_\Diamond, W_\Box)$ of V, and some $F_\Diamond, F_\Box$, s.t.
$(W_\Diamond, F_\Diamond)$ and $(W_\Box, F_\Box)$ are winning strategies for $\Diamond$ and $\Box$, resp.

Example



Solving a parity game: finding the winning sets, i.e., $W_\Diamond$ and $W_\Box$

Complexity of solving parity games

Corollary [EJS 1993; Zwick, Paterson 1996; J 1998]
Solving parity games is in **NP** $\cap$ **co-NP**, and even in **UP** $\cap$ **co-UP**

An **NP** procedure

1. Guess a strategy $(W_\Diamond, F_\Diamond)$
2. Check that $(W_\Diamond, F_\Diamond)$ is a winning strategy for $\Diamond$

Complexity of solving parity games

Algorithms

	TIME	SPACE
[EL′86, ..., McN′93, Zie′98]	$O\left(m \cdot \left(\frac{n}{d}\right)^{d}\right)$	$O(d \cdot n)$
[BCJLM′97, Sei′96]	$O\left(m \cdot \left(\frac{n}{d/2}\right)^{d/2}\right)$	$O\left(m \cdot \left(\frac{n}{d/2}\right)^{d/2}\right)$
This talk	$O\left(m \cdot \left(\frac{n}{d/2}\right)^{d/2}\right)$	$O(d \cdot n)$

where $n = |V|$, $m = |E|$, and d is the number of priorities

Towards the new algorithm

An alternative **NP** procedure

1. Guess a strategy $(W_\diamond, F_\diamond)$ and a function $\alpha : W_\diamond \to \mathbb{N}^d$
2. Check that α is a progress measure

Progress measures are witnesses for winning strategies

Theorem [... ; Klarlund, Kozen 1991; EJ 1991; Walukiewicz 1996]
There exists a winning strategy (W, F) for $\diamond$ if and only if there exists a progress measure $\varrho : W \to \mathbb{N}^d$

Main ideas behind the new algorithm

1. Existence of "small" progress measures
2. Progress measures as pre-fixed points of monotone maps in a complete lattice of "small" height:
   (a) yields existence of least progress measures [Walukiewicz 1996]
   (b) guides the way to efficiently compute them

Progress measures (1)

Notation: $(a_1, \ldots, a_d) \geq_i (b_1, \ldots, b_d)$ iff $(a_1, \ldots, a_i) \geq_{\text{lex}} (b_1, \ldots, b_i)$

Let $\varrho : V \to \big(\mathbb{N}^d \cup \{\top\}\big)$, and $(v, w) \in E$

Define a predicate $\texttt{Prog}\big(\varrho, (v, w)\big)$ to hold iff

- $\varrho(v) \geq_{p(v)} \varrho(w)$, and
- if $p(v)$ is odd then $\varrho(v) >_{p(v)} \varrho(w)$

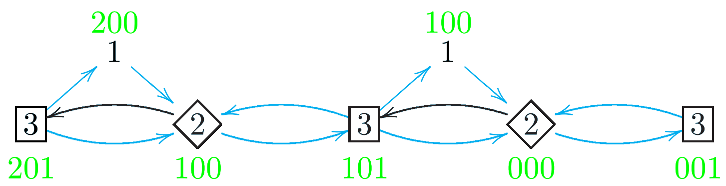
or $\varrho(v) = \varrho(w) = \top$

Progress measures (2)

Progress measure is a function $\varrho : V \to (\mathbb{N}^d \cup \{\top\})$, such that for all $v \in V$, we have:

- if $v \in V_\diamond$ then $\mathtt{Prog}(\varrho, (v, w))$ holds for some $(v, w) \in E$
- if $v \in V_\Box$ then $\mathtt{Prog}(\varrho, (v, w))$ holds for all $(v, w) \in E$

Example



Progress measures with small co-domains

Theorem (Small progress measure)
If $\diamond$ has a winning strategy (W, F) in game $G = (V, (V_\diamond, V_\Box), E, p)$ then there exists a progress measure $\varrho : V \to (M_G \cup \{\top\})$ where

$$M_G = ([n_1] \times [0] \times [n_3] \times \cdots \times [0] \times [n_{d-1}] \times [0])$$

and $n_i = |p^{-1}(i)|$, and $[i] = \{0, 1, \ldots, i\}$, and

$$\varrho(w) \neq \top \quad \text{for all } w \in W$$

Progress measures with small co-domains

Theorem

If all cycles in (W, F) are even then there exists a progress measure $\varrho : W \to M_G$

Small progress measures: proof

- $\boxed{p^{-1}(1) \neq \emptyset}$

  Claim There are $U_1 \uplus U_2 = V$, such that $(U_1 \times U_2) \cap E = \emptyset$

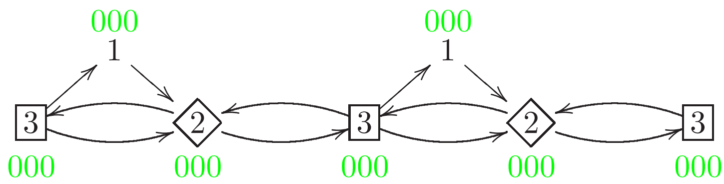
$$\varrho_1 \cup \left(\varrho_2 + (n'_1, 0, n'_3, 0, \dots)\right)$$

- $\boxed{p^{-1}(1) = \emptyset,\ p^{-1}(2) \neq \emptyset}$

$$\varrho \cup \left(\lambda v.(0, \dots, 0)\right)$$

How to compute (small) progress measures?

Example



$\texttt{MinProg}\big(\varrho, (v, w)\big)$:

the least $m \in \big(M_G \cup \{\top\}\big)$ which makes $\texttt{Prog}\big(\varrho[v \mapsto m], (v, w)\big)$ hold

How to compute (small) progress measures?

$\texttt{Lift}(\cdot, v) : (V \to M_G^{\top}) \to (V \to M_G^{\top})$ operators:

$$\texttt{Lift}\big(\varrho, v\big)(u) = \begin{cases} \varrho(u) & \text{if } u \neq v \\ \min_{(v,w) \in E} \texttt{MinProg}\big(\varrho, (v, w)\big) & \text{if } u = v \in V_{\Diamond} \\ \max_{(v,w) \in E} \texttt{MinProg}\big(\varrho, (v, w)\big) & \text{if } u = v \in V_{\Box} \end{cases}$$

Progress measures as pre-fixed points

Point-wise order $\sqsubseteq$ on $(V \to M_G^\top)$:

$\varrho \sqsubseteq \varrho'$ iff $\varrho(v) \leq_{\text{lex}} \varrho'(v)$ for all $v \in V$

Fact The operator $\texttt{Lift}(\cdot, v)$ is $\sqsubseteq$-monotone, for all $v \in V$

Fact A function $\varrho : V \to M_G^\top$ is a progress measure if and only if ϱ is a pre-fixed point of $\texttt{Lift}(\cdot, v)$ operators, for all $v \in V$

Corollary (by Knaster-Tarski Theorem)

- there exists the $\sqsubseteq$-least progress measure
- it can be computed by iterating $\texttt{Lift}(\cdot, v)$ operators for all $v \in V$

The algorithm

ProgressMeasureLifting

$\mu := \lambda v \in V.(0, \ldots, 0)$

while $\mu \sqsubset \texttt{Lift}(\mu, v)$ for some $v \in V$ **do** $\mu := \texttt{Lift}(\mu, v)$

Space: $O(d \cdot n)$

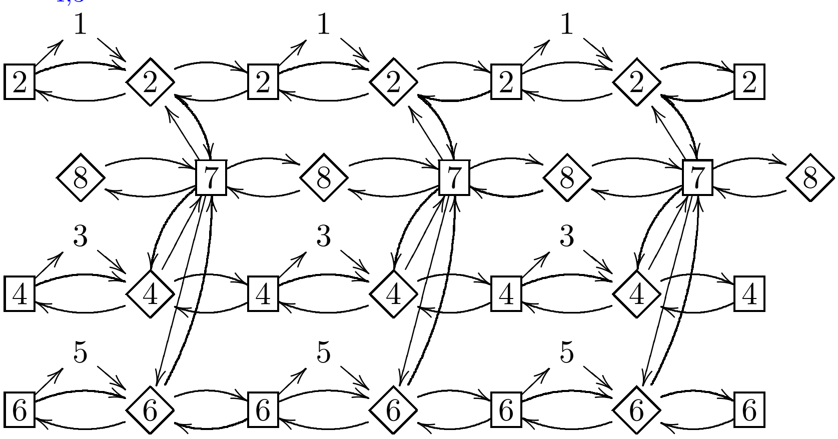
Time: $O\Big(\sum_{v \in V} d \cdot \deg(v) \cdot |M_G|\Big) = O\big(d \cdot m \cdot |M_G|\big)$

$$|M_G| = \prod_{i=1}^{d/2} (n_{2i-1} + 1) \leq \left(\frac{n}{d/2}\right)^{d/2}$$

Worst-case performance

Theorem For all $d, n \in \mathbb{N}$, there is a game $(H_{d/2,n/d})$ of size $O(n)$ with priorities bounded by d, on which the algorithm performs at least $(n/d)^{d/2}$ many lifts, for all lifting policies

Game $H_{4,3}$:



Conclusion

Main points

- Solving parity games: the algorithmic essence of the modal μ-calculus model checking
- Progress measures: witnesses for winning strategies
- Progress measures as pre-fixed points
  - existence of least progress measures
  - a guide for efficient computation of witnesses

Questions

- Is it a local model checking algorithm?
- Can it be refined to a **P**-time algorithm?