A Quasi-Polynomial Black-Box Algorithm for Fixed Point Evaluation

André Arnold, Damian Niwiński, **Paweł Parys**

CSL 2021

Plan

- parity games ≈ modal μ-calculus

  $\in \cap$

  (black-box) fixed point evaluation

- quasi-polynomial algorithms for parity games

  ⇩

  quasi-polynomial black-box algorithms for fixed point evaluation

- Our algorithm is an abstract version of recent quasi-polynomial algorithms solving parity games

- We unify two kinds of parity-games algorithms (asymmetric, symmetric) in a common framework

- Some lower bounds for the method (universal trees are needed)

Considered problem: fixed point evaluation

Compute: $\nu x_d.\mu x_{d-1}\ldots\nu x_2.\mu x_1.f(x_1,x_2,\ldots,x_{d-1},x_d)$

where $x_i \in \{0,1\}^n$

$f : (\{0,1\}^n)^d \to \{0,1\}^n$ monotone

access to f: only evaluation for given arguments (f is a black-box)

Considered problem: fixed point evaluation

Compute: $\nu x_d.\mu x_{d-1}\ldots\nu x_2.\mu x_1.f(x_1,x_2,\ldots,x_{d-1},x_d)$

where $x_i \in \{0,1\}^n$

$f : (\{0,1\}^n)^d \to \{0,1\}^n$ monotone

access to f: only evaluation for given arguments (f is a black-box)

Relation to parity games

parity game
(n nodes, d priorities)

fixed point evaluation
(n bits, d arguments)
f of a special form

parity game
($exp(n)$ nodes, d priorities)

fixed point evaluation
(n bits, d arguments)
arbitrary f

Parity games vs. fixed point evaluation

$$\nu x_d.\mu x_{d-1}\ldots\nu x_2.\mu x_1.f(x_1,x_2,\ldots,x_{d-1},x_d)$$

For parity games:

- f is of a special form: every output bit is either AND or OR of some input bits
- the game graph can be accessed also in other ways, not only by evaluating f

Recent quasipolynomial algorithms for parity games:

- access the game graph only by evaluating f
- work for arbitrary f, not only for f coming from parity games

⇩

After a careful analysis, they give black-box algorithms for fixed point evaluation

This paper / this talk:

- Why?
- How to prove this in a nice way?

Recent results on parity games

- Calude, Jain, Khoussainov, Li, Stephan 2017
- Fearnley, Jain, Schewe, Stephan, Wojtczak 2017
- Jurdziński, Lazić 2017
- Lehtinen 2018

} asymmetric algo. (separator approach)

complexity: $n^{\lg(d/\lg n)+O(1)} \approx |U_{n,d}|$

- Bojańczyk, Czerwiński 2018
- Czerwiński, Daviaud, Fijalkow, Jurdziński, Lazić, Parys 2019

- Parys 2019
- Lehtinen, Schewe, Wojtczak 2019
- Jurdziński, Morvan 2020

} symmetric algo. (recursive)

complexity: $n^{2\lg(d/\lg n)+O(1)} \approx |U_{n,d}|^2$

- Jurdziński, Morvan, Ohlmann, Thejaswini 2020 – symmetric, in $n^{\lg(d/\lg n)+O(1)} \approx |U_{n,d}|$

fixed point evaluation:

- Hausmann, Schröder 2019
- Hausmann, Schröder 2020

(blue = after writing this paper)

Standard exponential algorithm

Notation: $|(\Theta,f,(\mathbf{0},\mathbf{1}))|=\nu x_d.\mu x_{d-1}\ldots\nu x_2.\mu x_1.f(x_1,x_2,\ldots,x_{d-1},x_d)$ for $\Theta=\nu\mu\ldots\nu\mu$

$f^{\hookrightarrow A}(x_1,x_2,\ldots,x_{d-1}) = f\,(x_1,x_2,\ldots,x_{d-1},A)$

Algorithm evaluating $|(\Theta,f,(\mathbf{0},\mathbf{1}))|$ (where $j=1,2,\ldots,n$)

for $\Theta=\mu\,\Theta$':

$A_0 = \mathbf{0}$

$A_j = |(\Theta',f^{\hookrightarrow A_{j-1}},(\mathbf{0},\mathbf{1}))|$

return A_n

for $\Theta=\nu\,\Theta$':

$B_0 = \mathbf{1}$

$B_j = |(\Theta',f^{\hookrightarrow B_{j-1}},(\mathbf{0},\mathbf{1}))|$

return B_n

Standard exponential algorithm

Notation: $|(\Theta,f,(\mathbf{0},\mathbf{1}))|=\nu x_d.\mu x_{d-1}\ldots\nu x_2.\mu x_1.f(x_1,x_2,\ldots,x_{d-1},x_d)$ for $\Theta=\nu\mu\ldots\nu\mu$

$f^{\hookrightarrow A}(x_1,x_2,\ldots,x_{d-1}) = f(x_1,x_2,\ldots,x_{d-1},A)$

Algorithm evaluating $|(\Theta,f,(\mathbf{0},\mathbf{1}))|$ (where $j=1,2,\ldots,n$)

for $\Theta=\mu\Theta'$:

$A_0 = \mathbf{0}$

$A_j = |(\Theta',f^{\hookrightarrow A_{j-1}},(\mathbf{0},\mathbf{1}))|$

return A_n

for $\Theta=\nu\Theta'$:

$B_0 = \mathbf{1}$

$B_j = |(\Theta',f^{\hookrightarrow B_{j-1}},(\mathbf{0},\mathbf{1}))|$

return B_n

How to make it quasipolynomial?

- do not start from **0** / **1**, but from some intermediate values (restrictions)
- perform less iterations (follow a structure of some universal trees)

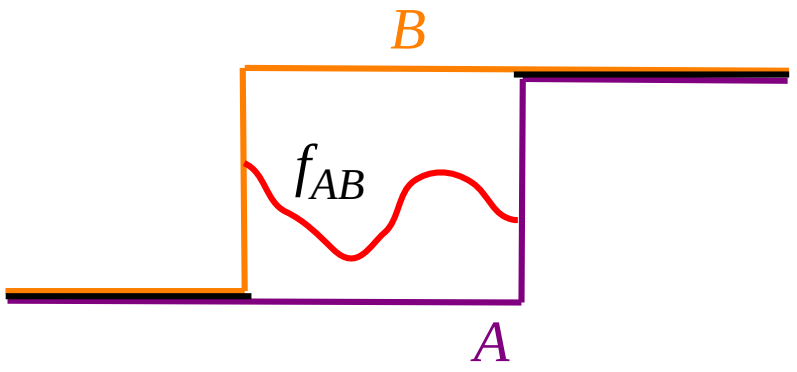
Restrictions

$A \leq B$

Notation: $f_{AB}(x_1,x_2,\ldots,x_d) = A + B * f(x_1,x_2,\ldots,x_d)$

= sup
= bitwise OR

= inf
= bitwise AND



Restrictions

Notation: $f_{AB}(x_1,x_2,\ldots,x_d) = A + B * f(x_1,x_2,\ldots,x_d)$ $\quad A \le B$

$|(\Theta,f,(A,B))| = \nu x_d.\mu x_{d-1}\ldots\nu x_2.\mu x_1.f_{AB}(x_1,x_2,\ldots,x_{d-1},x_d)$ for $\Theta = \nu\mu\ldots\nu\mu$

Algorithm evaluating $|(\Theta,f,(A,B))|$ (where $j=1,2,\ldots,n$)

for $\Theta = \mu\,\Theta'$:

$A_0 = A$

$A_j = |(\Theta', f^{\hookrightarrow A_{j-1}}, (A_{j-1},B))|$

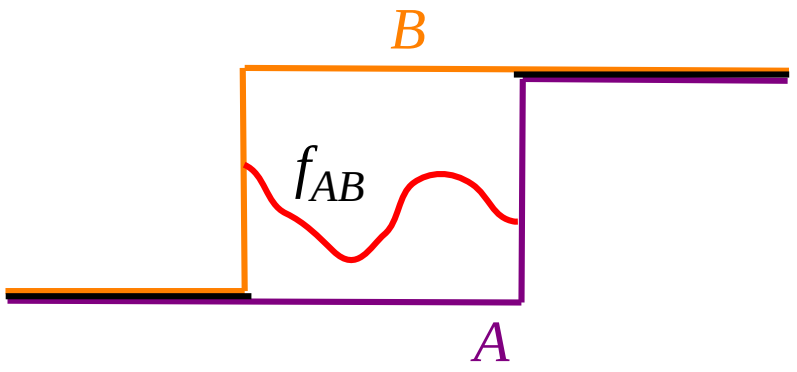
return A_n

for $\Theta = \nu\,\Theta'$:

$B_0 = B$

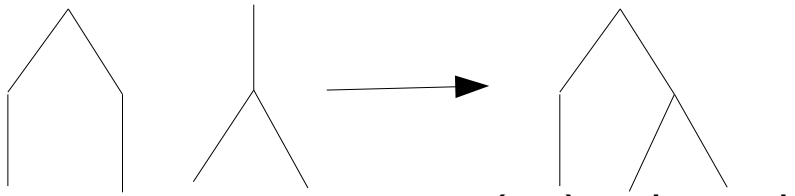
$B_j = |(\Theta', f^{\hookrightarrow B_{j-1}}, (A,B_{j-1}))|$

return B_n



Universal trees

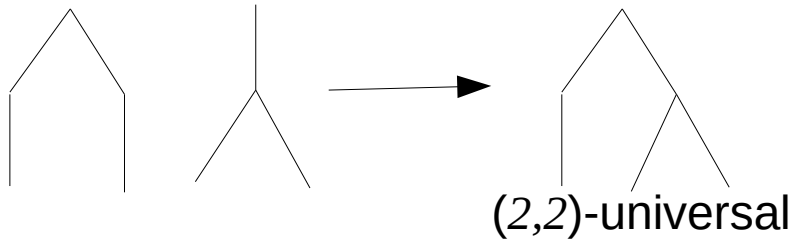
A tree U (of height h) is (n,h)-universal if every tree of height h with n leaves embeds in U.



(2,2)-universal

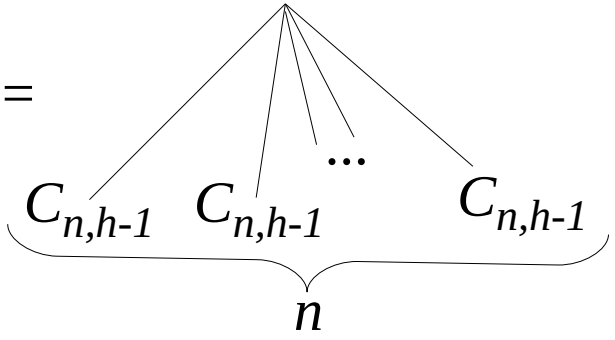
Universal trees

A tree U (of height h) is (n,h)-universal if every tree of height h with n leaves embeds in U.

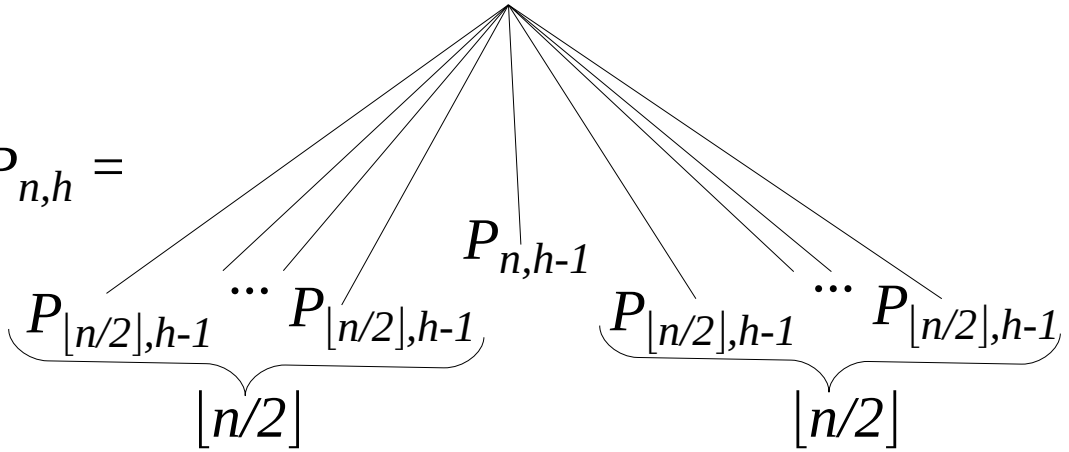


Examples:

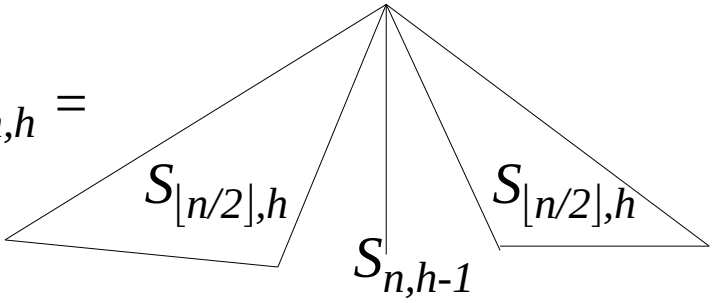
$C_{n,h} =$



$P_{n,h} =$

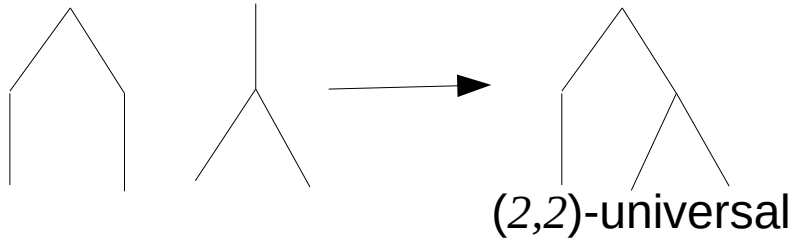


$S_{n,h} =$



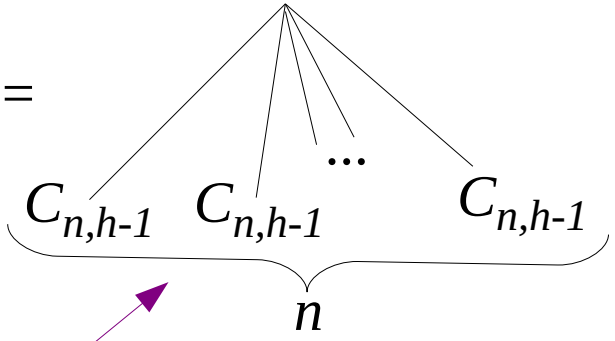
Universal trees

A tree U (of height h) is (n,h)-universal if every tree of height h with n leaves embeds in U.



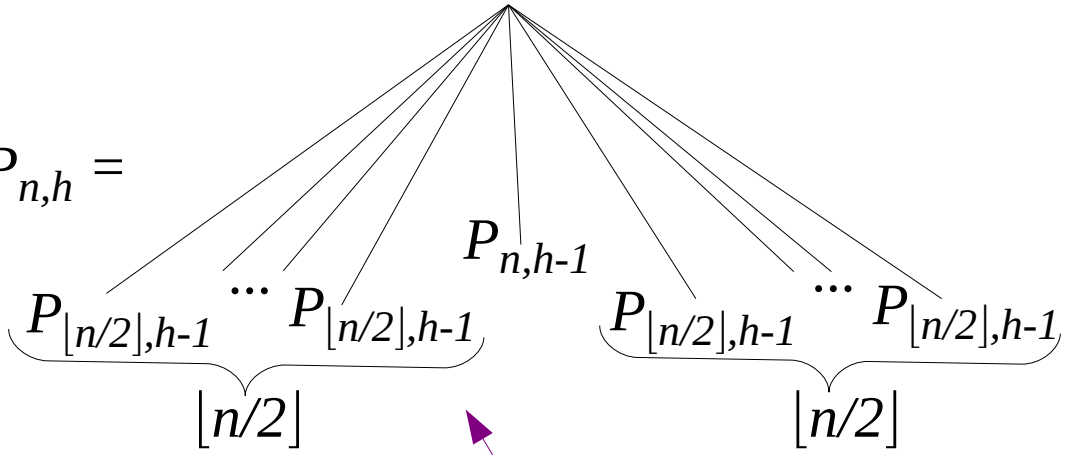
Examples:

$C_{n,h} =$



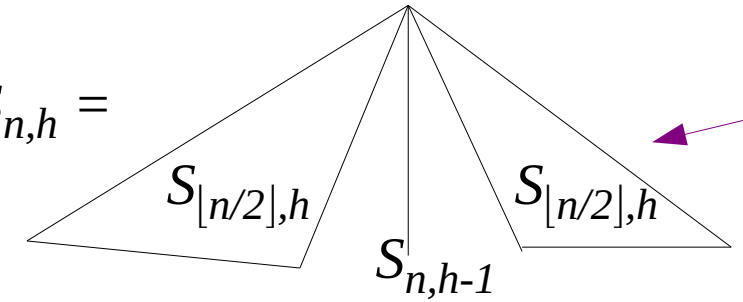
size n^h

$P_{n,h} =$



size $n^{\lg n + \lg(h/\lg n)+O(1)}$

$S_{n,h} =$



size $n^{\lg(h/\lg n)+O(1)}$

Symmetric algorithm based on universal trees

≈ the symmetric algorithm for parity games

U,V – (universal) trees

Definition / Algorithm evaluating $|(\Theta,f,(A,B))|_{U,V}$ (where $j=1,2,\ldots,p$)

for $\Theta=\mu\Theta'$, $U=\langle U_1,\ldots,U_p\rangle$

$A_0 = A$

$A_j = |(\Theta',f^{\hookrightarrow A_{j-1}},(A_{j-1},B))|_{U_j,V}$

return A_p

for $\Theta=\nu\Theta'$, $V=\langle V_1,\ldots,V_p\rangle$

$B_0 = B$

$B_j = |(\Theta',f^{\hookrightarrow B_{j-1}},(A,B_{j-1}))|_{U,V_j}$

return B_p

Symmetric algorithm based on universal trees

≈ the symmetric algorithm for parity games

U,V – (universal) trees

Definition / Algorithm evaluating $|(\Theta,f,(A,B))|_{U,V}$ (where $j=1,2,\ldots,p$)

for $\Theta=\mu\Theta'$, $U=\langle U_1,\ldots,U_p\rangle$

$A_0 = A$

$A_j = |(\Theta',f^{\hookrightarrow A_{j-1}},(A_{j-1},B))|_{U_j,V}$

return A_p

for $\Theta=\nu\Theta'$, $V=\langle V_1,\ldots,V_p\rangle$

$B_0 = B$

$B_j = |(\Theta',f^{\hookrightarrow B_{j-1}},(A,B_{j-1}))|_{U,V_j}$

return B_p

Correctness

If U,V are $(n,d/2)$-universal then $|(\Theta,f,(\mathbf{0},\mathbf{1}))|_{U,V} = |(\Theta,f,(\mathbf{0},\mathbf{1}))|$.

Proof is based on:

- dominions
- dominion decomposition

} adapted from parity games [Jurdziński, Morvan 2020]

Symmetric algorithm based on universal trees

≈ the symmetric algorithm for parity games

U,V – (universal) trees

Definition / Algorithm evaluating $|(\Theta,f,(A,B))|_{U,V}$ (where $j=1,2,\ldots,p$)

for $\Theta=\mu\Theta'$, $U=\langle U_1,\ldots,U_p\rangle$

$A_0 = A$

$A_j = |(\Theta',f^{\hookrightarrow A_{j-1}},(A_{j-1},B))|_{U_j,V}$

return A_p

for $\Theta=\nu\Theta'$, $V=\langle V_1,\ldots,V_p\rangle$

$B_0 = B$

$B_j = |(\Theta',f^{\hookrightarrow B_{j-1}},(A,B_{j-1}))|_{U,V_j}$

return B_p

Correctness

If U,V are $(n,d/2)$-universal then $|(\Theta,f,(\mathbf{0},\mathbf{1}))|_{U,V} = |(\Theta,f,(\mathbf{0},\mathbf{1}))|$.

Proof is based on:

- dominions
- dominion decomposition

} adapted from parity games [Jurdziński, Morvan 2020]

Time complexity: $|U|\cdot|V| = n^{2\lg(d/\lg n)+O(1)}$ (two universal trees)

Asymmetric algorithm (Seidl's idea, 1996)

- evaluate recursively
  time: n^d

Seidl '96 →

- create a system of **least** fixed point equations, and solve it
  time: $n^{d/2+1}$

- universal trees
- restrictions
- evaluate recursively
  time: $n^{2\lg(d/\lg n)+O(1)}$

≀

symmetric algorithm
for parity games

→

- universal trees
- restrictions
- create a system of **least** fixed point equations, and solve it
  time: $n^{2\lg(d/\lg n)/2+O(1)}$

≀

asymmetric algorithm
for parity games

A lower bound (for our method)

Theorem: Fix n,d.
If $|(\Theta,f,(\mathbf{0},\mathbf{1}))| = |(\Theta,f,(\mathbf{0},\mathbf{1}))|_{U,V}$ for all f, then U,V are $(n,d/2)$-universal.

Corollary:
It is known that every universal tree has size at least $n^{\lg(h/\lg n)+\Omega(1)}$
Thus our algorithm cannot work faster (using potentially some smaller tree).

A lower bound (for our method)

Theorem: Fix n,d.
If $|(\Theta,f,(\mathbf{0},\mathbf{1}))| = |(\Theta,f,(\mathbf{0},\mathbf{1}))|_{U,V}$ for all f, then U,V are $(n,d/2)$-universal.

Corollary:
It is known that every universal tree has size at least $n^{\lg(d/\lg n)+\Omega(1)}$
Thus our algorithm cannot work faster (using potentially some smaller tree).

Remark:
It is enough to assume equality for functions f defined by parity games
(so the lower bound applies also to parity games)

Conclusions

- quasi-polynomial algorithms for fixed-point evaluation
- an abstract formulation using universal trees
- unified treatment of symmetric / asymmetric variants
- a lower bound for the method

Open problem:

- prove a (quasi-polynomial?) lower bound for the number of queries for black-box fixed point evaluation
  [we only have $\Omega(n^2/\log n)$ – Parys 2009]

Thank you!