Lower bound for evaluation of $\mu\nu$ fixpoint

Paweł Parys
University of Warsaw

Paper presented by Damian Niwiński

Big open problem

Find a polynomial time algorithm which:

- finds winning regions in parity games
- evaluates modal μ -calculus formulas in a Kripke structure
- checks non-emptiness of automata on infinite trees with the parity acceptance condition

These three problems are equivalent.

Big open problem

Find a polynomial time algorithm which:

- finds winning regions in parity games
- evaluates modal μ -calculus formulas in a Kripke structure
- checks non-emptiness of automata on infinite trees with the parity acceptance condition

These three problems are equivalent.

- Easy $O(n^d)$ algorithm.
- The complexity was slightly improved several times.
- No polynomial time algorithm known.
- The problem is in NP$\cap$co-NP, so there is no hope for lower bound.

Lower bound for μ-calculus: another approach

The problem is in NP$\cap$co-NP, so there is no hope for lower bound.

The only possibility is to reformulate the problem slightly, so that it becomes combinatorial.

We use a black-box model (an oracle model) defined in:

Browne, Clarke, Jha, Long, Marrero. An improved algorithm for the evaluation of fixpoint expressions. TCS, 1997.

The black-box model

Consider the following form of expressions:

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

over the lattice $\{0,1\}^n$ (with the order $a_1\ldots a_n \leq b_1\ldots b_n$ iff $a_i \leq b_i$ for each i)

(every expression may be converted to a polynomially bigger expression of this form)

The black-box model

Consider the following form of expressions:

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

over the lattice $\{0,1\}^n$ (with the order $a_1...a_n \leq b_1...b_n$ iff $a_i \leq b_i$ for each i)

(every expression may be converted to a polynomially bigger expression of this form)

Assume the only way how F is accessed is evaluating its value for given arguments (the algorithm does not analyse the expression defining F).

The black-box model

Consider the following form of expressions:

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

over the lattice $\{0,1\}^n$ (with the order $a_1\ldots a_n \leq b_1\ldots b_n$ iff $a_i \leq b_i$ for each i)

(every expression may be converted to a polynomially bigger expression of this form)

Assume the only way how F is accessed is evaluating its value for given arguments (the algorithm does not analyse the expression defining F).

Additionally F is an arbitrary monotone function
(not necessarily given by a short formula).

The black-box model

Consider the following form of expressions:

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

over the lattice $\{0,1\}^n$ (with the order $a_1...a_n \leq b_1...b_n$ iff $a_i \leq b_i$ for each i)

(every expression may be converted to a polynomially bigger expression of this form)

Assume the only way how F is accessed is evaluating its value for given arguments (the algorithm does not analyse the expression defining F).

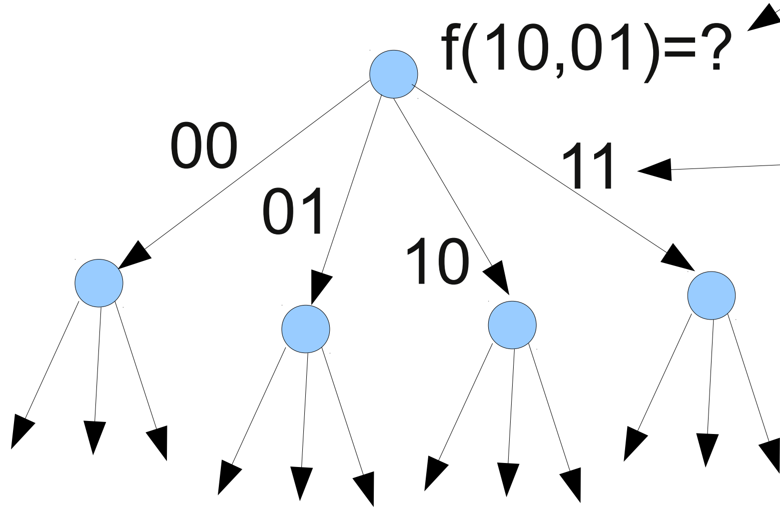
Additionally F is an arbitrary monotone function
(not necessarily given by a short formula).

Moreover we are not interested in the exact complexity,
we count only the number of queries to F.

The black-box model

In other words we consider decision trees:

Each internal node is labeled by an argument, for which the function F is checked



Edge corresponds to a possible value of F for that argument

For each path from the root to a leaf there is at most one possible value of the fixpoint expression for all monotone functions F consistent with the answers on that path.

We are interested in the (minimal) height of such decision tree.

The black-box model

Another view - a game:

- two players: an algorithm and an oracle
- the algorithm gives arguments for F
- the oracle gives an value of F for that arguments
- the algorithm wins when there is only one value of the fixpoint expression compatible with the answers of the oracle

How many steps needs the algorithm to win?

Comparison: classic approach vs black-box model

If the needed number of queries in the black-box model is high:

- There may still exist a fast algorithm, but it has to use the expression defining F in some other, better way!!
- It is possible that the lower bound requires functions F defined by a very long expressions, while distinguishing only the functions defined by short (polynomial) formulas may be done faster (this is not the case for d=2; we use only functions of polynomial size to obtain the lower bound).

If the needed number of queries in the black-box model is low:

- It may give fast algorithm!! (but this is not automatic - the decision tree with small number of queries may be very irregular and it may take a lot of time to compute what the next query should be)

Comparison: classic approach vs black-box model

Known algorithms for μ -calculus / parity games:

- n^d - the direct evaluation
- $n^{(d/2)}$ - Browne, Clarke, Jha, Long, Marrero, 1997
- $n^{(d/3)}$ - Schewe, 2007
- $n^{\sqrt{n}}$ - Jurdzinski, Paterson, Zwick, 2008

The first two algorithms (immediately) translate to the black-box model. The last two use parity games framework and do not translate to the black-box model.

This paper

We solve only the case d=2.

We get the bound $\Omega(n^2/\log(n))$ queries.

This paper

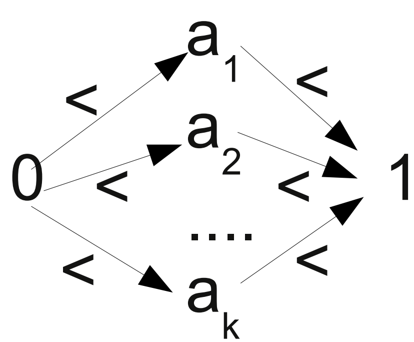
We solve only the case d=2.

We get the bound Ω ($n^2/\log(n)$) queries.

- Possibly it is a first step to giving a lower bound in a general case.
- It shows that alternation of quantifiers μ and ν is more complex that just one type of quantifiers (although it is highly believed that alternation should be a source of algorithmic complexity, results of that type are very rare).

<u>Proof ideas</u>

We take a more convenient lattice $(\Sigma_k)^m$,

where $\Sigma_k =$ 

The problem may be converted to this lattice,
loosing only log(n) factor.

Proof ideas

First trick:

- fixpoint contains m unknown letters
- each evaluation of f gives only one letter
- so m queries are needed to find all letters

$$f(0\ 0\ 0 \ldots 0\ 0) = v_1 0\ 0 \ldots 0\ 0$$
$$f(v_1 0\ 0 \ldots 0\ 0) = v_1 v_2 0 \ldots 0\ 0$$
$$f(v_1 v_2 0 \ldots 0\ 0) = v_1 v_2 v_3 \ldots 0\ 0$$
$$\ldots$$
$$f(v_1 v_2 v_3 \ldots v_{m-1} 0) = v_1 v_2 v_3 \ldots v_{m-1} v_m = \mu\, y.f(y)$$

(for any other arguments the values of f are chosen so that no information is given by asking for them)

Proof ideas

Second trick:

- fixpoint contains only 1 unknown letter, but on unknown position
- evaluation of g gives a letter on requested position
- the oracle may be malicious, so m queries are needed to find the position of the unknown letter

First 1 is replaced by 0 (and by v in the m-th step). If the algorithm asks in different order, v will be in a different place.

$$g(1\ 1\ 1 \ldots 1\ 1) = 0\ 1\ 1 \ldots 1\ 1$$
$$g(0\ 1\ 1 \ldots 1\ 1) = 0\ 0\ 1 \ldots 1\ 1$$
$$g(0\ 0\ 1 \ldots 1\ 1) = 0\ 0\ 0 \ldots 1\ 1$$
$$\ldots$$
$$g(0\ 0\ 0 \ldots 0\ 1) = 0\ 0\ 0 \ldots 0\ v \overset{}{=} \nu\, x.g(x)$$

(for any other arguments the values of g are chosen so that no information is given by asking for them)

<u>Proof ideas</u>

The second trick is used m times - to find each one letter in the first trick the algorithm needs to solve a copy of the second trick.

$$\mu\ \underbrace{y.\nu\ \overbrace{x.F(x,y)}^{g_y(x)}}_{f(y)}$$

Summary

- A black-box model is presented - a combinatorial version of μ -calculus in which some lower bounds may be proven.
- For d=2 we show that almost n^2 queries are needed.
- The case of general d is left for future work.