Parity to Safety in Polynomial Time for Pushdown and Collapsible Pushdown Games

Matthew Hague
Royal Holloway, University of London

with Roland Meyer, Sebastian Muskalla, and Martin Zimmermann
Abstract

- Safety games for pushdown systems: n-EXPTIME-complete
- Parity games for pushdown systems: n-EXPTIME-complete
  (for order-n collapsible pushdown)
Abstract

- Safety games for pushdown systems: n-EXPTIME-complete
- Parity games for pushdown systems: n-EXPTIME-complete
  (for order-n collapsible pushdown)

We give a "natural" parity->safety reduction
Abstract

- Safety games for pushdown systems: n-EXPTIME-complete
- Parity games for pushdown systems: n-EXPTIME-complete

(for order-n collapsible pushdown)

We give a "natural" parity->safety reduction
(i.e. not parity algorithm -> TM -> safety)
Motivation

- Safety is easy to reason about
  (Parity is hard)

- Parity is more expressive

- To know the relationship
Ideas We Start With

*Finite-state parity to safety using counters*  
(Bernet et al., 2002)
Ideas We Start With

- Finite-state parity to safety using counters (Bernet et al, 2002)
- Pushdown parity to safety with large counters (Fridman and Zimmermann, 2012)
Ideas We Start With

- Finite-state parity to safety using counters
  (Bernet et al., 2002)
- Pushdown parity to safety with large counters
  (Fridman and Zimmermann, 2012)
- Reduction order-$n$ -> order-$(n-1)$
  - Rank awareness
  (H, Murawski, Ong, Serre, 2008)
Ideas We Start With

- Finite-state parity to safety using counters (Bernet et al, 2002)
- Pushdown parity to safety with large counters (Fridman and Zimmermann, 2012)
- Reduction order-n -> order-(n-1)
  - Rank awareness (H, Murawski, Ong, Serre, 2008)
- Encoding large counters in a pushdown stack (Cachat and Walukiewicz, 2007)
Contributions

- Generalise counter encoding to collapse
- Direct proof based on commutativity of counters encoding
  - Stack removal
- Counters behave like a stack
Parity Game

Diagram:

- Node 4 (red square) is connected to Node 1 and Node 4 (red square).
- Node 1 (green circle) is connected to Node 2 (green circle).
- Node 2 (green circle) is connected to Node 4 (red square).

Nodes:
- Circle: Elvis
- Box: Anarchist
Parity Game

Play: 4
Parity Game

Play: 4 1
Parity Game

Play: 4 1 2
Parity Game

Play: 4 1 2 4
Parity Game

Play: 4 1 2 4 4

Elvis

Anarchist
Parity Game

Play: 4 1 2 4 4 1 4 4 1 2 4 ...
Parity Game

Play: 4 1 2 4 4 1 4 4 1 2 4...
Winner: Elvis if least infinitely occurring rank is even
Parity with Counters

Count: number of each rank
(without seeing anything smaller)
Parity with Counters

Count: number of each rank
(without seeing anything smaller)
Parity with Counters

Count: number of each rank
(without seeing anything smaller)

Counters:
1:
2:
3:
4: |
Parity with Counters

Counters:
1: |
2: 
3: 
4: 

Count: number of each rank (without seeing anything smaller)
Parity with Counters

Count: number of each rank
(without seeing anything smaller)
Parity with Counters

Count: number of each rank
(without seeing anything smaller)
Parity with Counters

Count: number of each rank
(without seeing anything smaller)

Counters:
1: |
2: |
3: |
4: ||
Parity with Counters

Count: number of each rank (without seeing anything smaller)
Parity with Counters

Count: number of each rank
(without seeing anything smaller)

Counters:
1: ||
2: 
3: 
4: |
Count: number of each rank
(without seeing anything smaller)
Parity with Counters

Counters:
1: ||||
2: 
3: 
4: 

Count: number of each rank
(without seeing anything smaller)
Parity with Counters

Count: number of each rank
(without seeing anything smaller)

Counters:
1: ||||
2:
3:
4: |
Parity with Counters

Count: number of each rank
(without seeing anything smaller)
Parity with Counters

Count: number of each rank
(without seeing anything smaller)
Parity with Counters

Count: number of each rank
(without seeing anything smaller)
Parity with Counters

Count: number of each rank
(without seeing anything smaller)

Counters:
1: ||||
2: 
3: 
4: |
Parity with Counters

Count: number of each rank
(without seeing anything smaller)

Counters:
1: |||||
2: 
3: 
4: ||
Parity with Counters

Count: number of each rank (without seeing anything smaller)
There is a loop!
(Hit 1 five times over 4 states)
There is a loop!
(Hit 1 five times over 4 states)
The smallest rank is odd: Elvis loses!
Parity to Safety

Reduce parity to safety game (Bernet et al):

- Keep counters in state \((n \times n^k)\) states
- Elvis loses if odd counter is high
Parity to Safety

Reduce parity to safety game (Bernet et al):

- Keep counters in state \((n \times n^k)\) states
- Elvis loses if odd counter is high

Exponential blow up!
Each state has:
Control state (from finite set)
Stack of characters

Model recursive programs
Pushdown Parity to Safety

Pushdown Parity

Finite-state Parity

(Walukiewicz 1996)

Finite-State Safety

Finite-State Safety

(Bernet et al 2002)
Pushdown Parity to Safety

Pushdown Parity → Pushdown Safety

(Fridman and Zimmermann 2012)

 Finite-state Parity → Finite-State Safety

(Walukiewicz 1996)

(Bernet et al 2002)
Pushdown Parity to Safety

Pushdown Parity ➔ Pushdown Safety

(Walukiewicz 1996) ➔ (Fridman and Zimmermann 2012)

Finite-state Parity ➔ Finite-State Safety

(Bernet et al 2002)

Commutes!
Pushdown Parity to Safety

From:

\[ \begin{array}{c}
\text{p1} \\
\text{a} \\
\text{b} \\
\text{c}
\end{array} \rightarrow
\begin{array}{c}
\text{p2} \\
\text{b} \\
\text{c}
\end{array} \rightarrow
\begin{array}{c}
\text{p3} \\
\text{b} \\
\text{b} \\
\text{c}
\end{array} \rightarrow
\begin{array}{c}
\text{p4} \\
\text{a} \\
\text{b} \\
\text{b} \\
\text{c}
\end{array} \]

To:

\[ \begin{array}{c}
\text{p1} \\
\text{(a, c1, c2)} \\
\text{(b, c1, c2)} \\
\text{(c, c1, c2)}
\end{array} \rightarrow
\begin{array}{c}
\text{p2} \\
\text{(b, c1, c2')} \\
\text{(c, c1, c2)}
\end{array} \]
From:

\[ p_1 \xrightarrow{a} p_2 \xrightarrow{b} p_3 \xrightarrow{b} p_4 \]

To:

\[ p_1 \xrightarrow{(a, c_1, c_2)} \xrightarrow{(b, c_1, c_2)} \xrightarrow{(b, c_1, c_2')} \xrightarrow{(c, c_1, c_2)} \]

Counters
Exponential Blow Up?

Counter values: can be exponential
(Pushdown->finite-state has $2^n$ states)

Number of stack characters: $(2^n)^k$
(For $k$ ranks)
Exponential Blow Up?

Counter values: can be exponential
(Pushdown->finite-state has $2^n$ states)

Number of stack characters: $(2^n)^k$
(For k ranks)
Reducing Size

From:

\[
\begin{align*}
\text{p} & \quad (a, c_1, c'_2) \\
(b, c_1, c_2)
\end{align*}
\]

To:

\[
\begin{align*}
\text{p} & \quad c_1 \\
c_2' & \quad a \\
b & \\
c_2 & \\
c_1
\end{align*}
\]
Reducing Size

From:

\[(a, c_1, c_2')\]
\[(b, c_1, c_2)\]

To:

\[a\]
\[c_2'\]
\[c_1\]
\[b\]
\[c_2\]
\[c_1\]

Number of characters: \(2^n\)
Increment c1
Increment $c_1$
Updating Counters

Increment $c_1$

$(p, a)$
Updating Counters

Increment $c_1$

Lost $c_2'$(It will be reset)
Updating Counters

Increment \( c_1 \)

\[ (p, a) \]

\( c_1' \)

\( b \)

\( c_2 \)

\( c_1 \)
Updating Counters

Increment c1

0

(p, a)

Lost c2'

(c1', b, c2, c1)

(It will be reset)
Updating Counters

Increment c1

\[
\begin{align*}
\text{a} & : 0 \\
\text{b} & : \text{c1}' \\
\text{c2} & : \text{c2} \\
\text{c1} & : \text{c1}
\end{align*}
\]
Counters up to $2^n$
Alphabet exponential
Polynomial No. of Characters

Counters up to $2^n$
Alphabet exponential

c2' in binary

$\begin{align*}
  &a \\
  &c_2' \\
  &c_1 \\
  &b \\
  &c_2 \\
  &c_1 \\
  \end{align*}$

$\begin{align*}
  &a_0 \\
  &1 \\
  &1 \\
  &0 \\
  &1 \\
  &0 \\
  \end{align*}$

$c_1$ in binary
Counters up to $2^n$

Alphabet exponential

Polynomial No. of Characters

$\text{c2'}$ in binary

$c1$ in binary

Pushdowns handle length-$n$ binary

Alphabet polynomial!
Pushdown Parity to Safety

- Pushdown parity game $\rightarrow$ pushdown safety game
- Naively exponential
- Use stack discipline of counters
- Binary counter encoding of pushdowns
- Polynomial time reduction
Collapsible Pushdown Systems

Pushdown Systems = First-Order Recursion

Collapsible Pushdown Systems

= Higher-Order Recursion
Higher-Order Programming: Niche?

Almost all modern languages support it

- Scala, Go, JavaScript, Python, ...

- Retro-fitted to C++ and Java

- Asynchronous programs/callbacks

- Map/Reduce
Collapsible Pushdown Systems

Higher-order program: functions of functions
Higher-order pushdown: stack of stacks
Collapsible Pushdown Systems

Higher-order program: functions of functions
Higher-order pushdown: stack of stacks
Collapsible Pushdown Systems

Higher-order program: functions of functions
Higher-order pushdown: stack of stacks
Collapsible Pushdown Systems

Higher-order program: functions of functions
Higher-order pushdown: stack of stacks (+links)
Generalising Parity $\rightarrow$ Safety

- Collapsible Pushdown $\rightarrow$ Finite-State
  (n-Exponential blow up)

- n-Exponential Counters

- Can encode in binary on order-n stack!
Summary

- Parity $\rightarrow$ Safety
- Counters look out for loops
  (Even with infinite states)
- Polynomial-time encoding
  - Reduction to finite-state
  - Counters behave like stacks
  - Counter values match limits of system
Future and Related Work

Can these ideas lead to implementations?
- Direct implementation
- Abstraction/refinement of counters
- Counter size vs. structure of game
  - Backend: HorSat, Preface, &c.

Parity->Safety:
- Berwanger and Doyen, 2008
- Sohail and Somenzi, 2009
- Biere et al, 2002
- Podelski and Rybalchenko, 2011
- Konnov et al, 2017