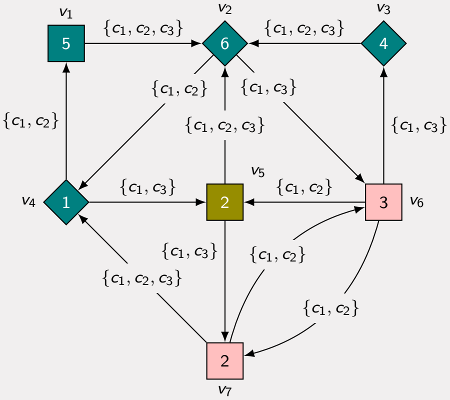
Verifying SPLs using parity games expressing variability



Sjef van Loo

6 November, 2019

Msc Thesis
Computer Science and Engineering

Supervised by T.A.C. Willemse

Outline

- Verification & SPLs
- Problem statement
- Variability Parity Games
- VPG algorithms
- Experimental results

Verification

- Creating correct software is difficult
- Even when testing is done rigorously errors can slip in
- Mathematically *model the behaviour* of software (LTS)
- Mathematically *specify a requirement* (modal μ-calculus)
- Check if the model satisfies the requirement

LTS M Modal μ-calculus formula φ

M satisfies φ?

Verification

- Creating correct software is difficult
- Even when testing is done rigorously errors can slip in
- Mathematically *model the behaviour* of software (LTS)
- Mathematically *specify a requirement* (modal μ-calculus)
- Check if the model satisfies the requirement

LTS M Modal μ-calculus formula φ

M satisfies φ?

Verification

- Creating correct software is difficult
- Even when testing is done rigorously errors can slip in
- Mathematically *model the behaviour* of software (LTS)
- Mathematically *specify a requirement* (modal μ-calculus)
- Check if the model satisfies the requirement

LTS M Modal μ-calculus formula φ

M satisfies φ?

Software product lines

- Software product lines are configurable systems
- Many variants of the same system, i.e. *software products*
- FTSs express multiple LTSs using *features*

Problem statement

- Find all the products in an SPL that satisfy a requirement

FTS M Modal μ-calculus formula φ

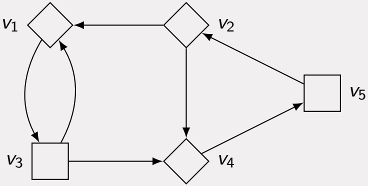
Which products in M satisfy φ?

- Do so more efficiently than verifying every product independently

Variability parity game

Parity game.

Players 0 (even,$\Diamond$) and 1 (odd,$\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.

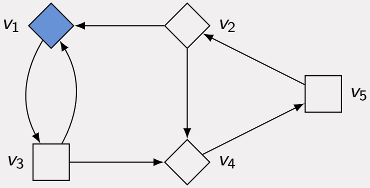
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \dots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

Players 0 (even, $\Diamond$) and 1 (odd, $\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

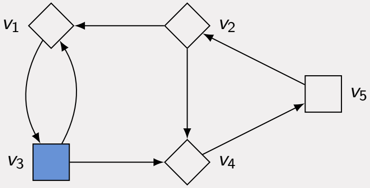
Variability parity game.

Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \dots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.
Players 0 (even, $\Diamond$) and 1 (odd, $\Box$)



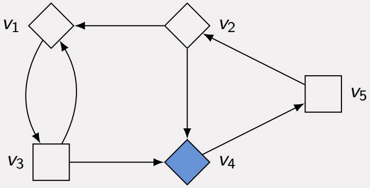
- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.
Players 0 (even, $\Diamond$) and 1 (odd, $\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

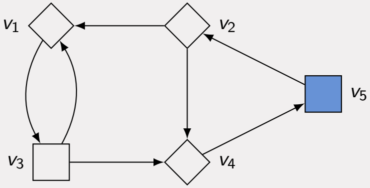
Variability parity game.
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

Players 0 (even,$\Diamond$) and 1 (odd,$\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.

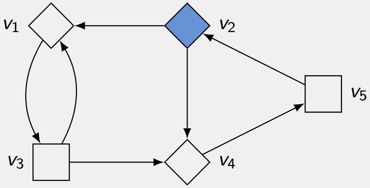
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

Players 0 (even,◇) and 1 (odd,□)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.

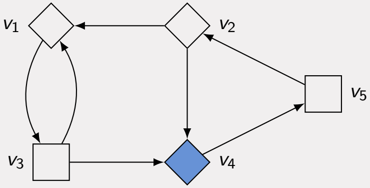
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

Players 0 (even,$\Diamond$) and 1 (odd,$\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.

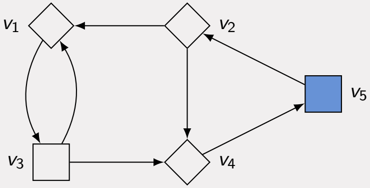
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

Players 0 (even,◇) and 1 (odd,□)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

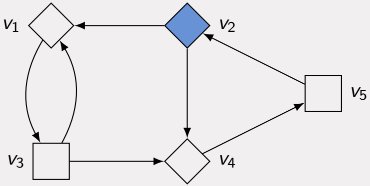
Variability parity game.

Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \dots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.
Players 0 (even,$\Diamond$) and 1 (odd,$\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

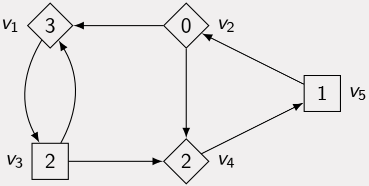
Variability parity game.
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

Players 0 (even,$\Diamond$) and 1 (odd,$\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.

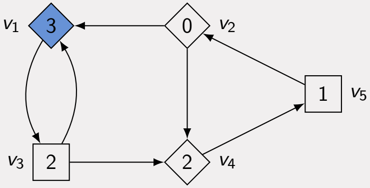
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

Players 0 (even,$\Diamond$) and 1 (odd,$\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.

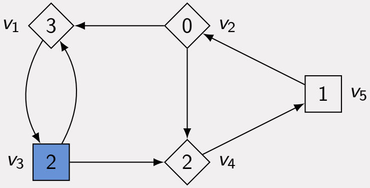
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

Players 0 (even,$\Diamond$) and 1 (odd,$\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.

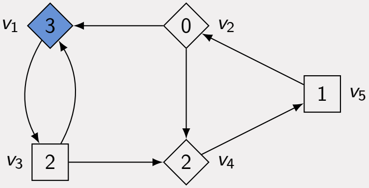
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

Players 0 (even,$\Diamond$) and 1 (odd,$\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.

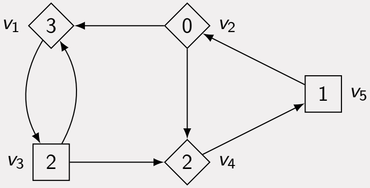
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

Players 0 (even,$\Diamond$) and 1 (odd,$\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.

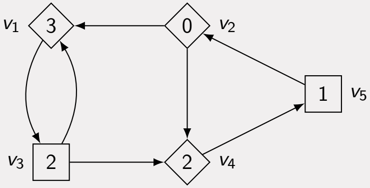
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

Players 0 (even,$\Diamond$) and 1 (odd,$\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.

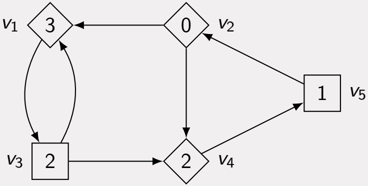
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

Players 0 (even,$\Diamond$) and 1 (odd,$\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.

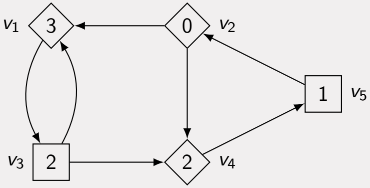
Configurations $\mathfrak{C} = \{c_1, c_2\}$

- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

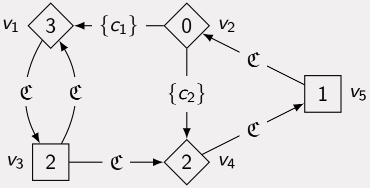
Players 0 (even,$\Diamond$) and 1 (odd,$\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.

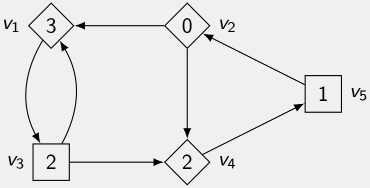
Configurations $\mathfrak{C} = \{c_1, c_2\}$



- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \dots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

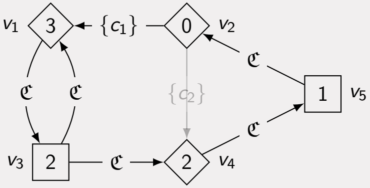
Variability parity game

Parity game.
Players 0 (even, $\Diamond$) and 1 (odd, $\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.
Configurations $\mathfrak{C} = \{c_1, c_2\}$

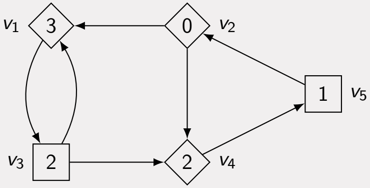


- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

Parity game.

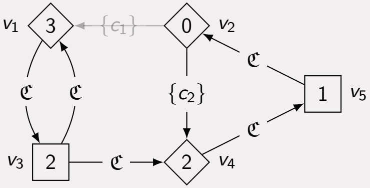
Players 0 (even, $\Diamond$) and 1 (odd, $\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.

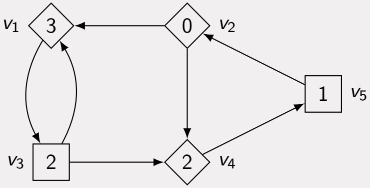
Configurations $\mathfrak{C} = \{c_1, c_2\}$



- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

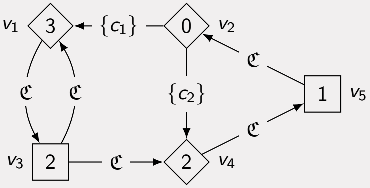
Variability parity game

Parity game.
Players 0 (even, $\Diamond$) and 1 (odd, $\Box$)



- The winner is determined by the parity of the priority occurring infinitely often
- Player 1 wins $\{v_1, v_3\}$, using $v_3 \mapsto v_1$
- Player 0 wins $\{v_2, v_4, v_5\}$, using $v_2 \mapsto v_4$
- *Solving*: Partition the vertices in W_0, W_1

Variability parity game.
Configurations $\mathfrak{C} = \{c_1, c_2\}$



- $W_0^{c_1} = \emptyset, W_1^{c_1} = \{v_1, \ldots, v_5\}$
- $W_0^{c_2} = \{v_1, v_3\}, W_1^{c_2} = \{v_2, v_4, v_5\}$
- *Solving*: Partition the vertices in W_0^c, W_1^c, for every $c \in \mathfrak{C}$

Variability parity game

LTS M — Modal μ-calculus formula φ

$\downarrow$

M satisfies φ?

Theorem

A parity game can be constructed from an LTS and a modal μ-calculus formula φ such that M satisfies φ iff special vertex $v_0 \in W_0$ in the resulting parity game.

Theorem

A VPG can be constructed from an FTS and a modal μ-calculus formula φ such that M satisfies φ for product p iff special vertex $v_0 \in W_0^p$ in the resulting VPG.

Variability parity game

LTS M — Modal μ-calculus formula φ

PG

$\downarrow$

M satisfies φ?

Theorem

A parity game can be constructed from an LTS and a modal μ-calculus formula φ such that M satisfies φ iff special vertex $v_0 \in W_0$ in the resulting parity game.

Theorem

A VPG can be constructed from an FTS and a modal μ-calculus formula φ such that M satisfies φ for product p iff special vertex $v_0 \in W_0^p$ in the resulting VPG.

Variability parity game

LTS M Modal μ-calculus formula φ

PG

M satisfies φ?

FTS M Modal μ-calculus formula φ

Which products in M satisfy φ?

Theorem

A parity game can be constructed from an LTS and a modal μ-calculus formula φ such that M satisfies φ iff special vertex $v_0 \in W_0$ in the resulting parity game.

Theorem

A VPG can be constructed from an FTS and a modal μ-calculus formula φ such that M satisfies φ for product p iff special vertex $v_0 \in W_0^p$ in the resulting VPG.

Variability parity game

LTS M — Modal μ-calculus formula φ

PG

$\downarrow$

M satisfies φ?

FTS M — Modal μ-calculus formula φ

VPG

$\downarrow$

Which products in M satisfy φ?

Theorem

A parity game can be constructed from an LTS and a modal μ-calculus formula φ such that M satisfies φ iff special vertex $v_0 \in W_0$ in the resulting parity game.

Theorem

A VPG can be constructed from an FTS and a modal μ-calculus formula φ such that M satisfies φ for product p iff special vertex $v_0 \in W_0^p$ in the resulting VPG.

Variability parity game

Theorem

A parity game can be constructed from an LTS and a modal μ-calculus formula φ such that M satisfies φ iff special vertex $v_0 \in W_0$ in the resulting parity game.

Theorem

A VPG can be constructed from an FTS and a modal μ-calculus formula φ such that M satisfies φ for product p iff special vertex $v_0 \in W_0^p$ in the resulting VPG.

$$\begin{tikzcd}
\text{FTS} \arrow[d, dashed, "\Pi_p"'] & \\
\text{LTS} \arrow[r, "\varphi"] & \text{PG}
\end{tikzcd}$$

Variability parity game

Theorem

A parity game can be constructed from an LTS and a modal μ-calculus formula φ such that M satisfies φ iff special vertex $v_0 \in W_0$ in the resulting parity game.

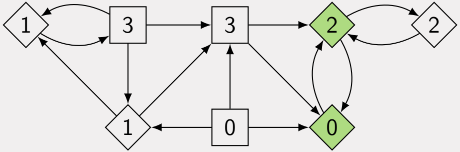
Theorem

A VPG can be constructed from an FTS and a modal μ-calculus formula φ such that M satisfies φ for product p iff special vertex $v_0 \in W_0^p$ in the resulting VPG.

$$\begin{tikzcd}
\text{FTS} \arrow[r, "\varphi"] \arrow[d, dashed, "\Pi_p"'] & \text{VPG} \arrow[d, dashed, "\Pi_p"] \\
\text{LTS} \arrow[r, "\varphi"] & \text{PG}
\end{tikzcd}$$

VPG algorithms - Recursive algorithm

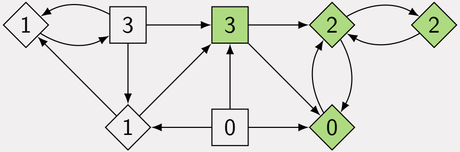
- The recursive algorithm reasons about sets of vertices



- the recursive algorithm for VPGs reasons about sets of vertex configuration pairs
- Attractor calculation example on VPG
- Function-wise representation to efficiently perform attractor calcs
- Short explanation of symbolic representation
- Time complexities

VPG algorithms - Recursive algorithm

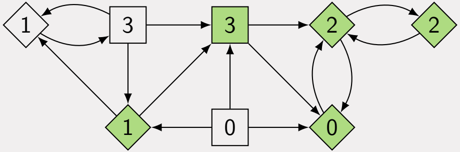
- The recursive algorithm reasons about sets of vertices



- the recursive algorithm for VPGs reasons about sets of vertex configuration pairs
- Attractor calculation example on VPG
- Function-wise representation to efficiently perform attractor calcs
- Short explanation of symbolic representation
- Time complexities

VPG algorithms - Recursive algorithm

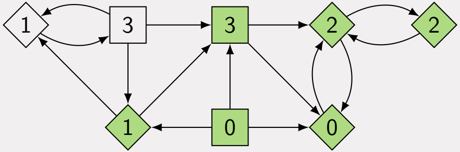
- The recursive algorithm reasons about sets of vertices



- the recursive algorithm for VPGs reasons about sets of vertex configuration pairs
- Attractor calculation example on VPG
- Function-wise representation to efficiently perform attractor calcs
- Short explanation of symbolic representation
- Time complexities

VPG algorithms - Recursive algorithm

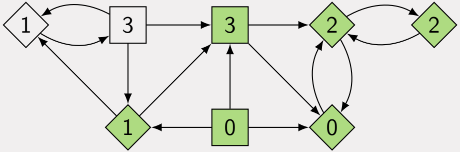
- The recursive algorithm reasons about sets of vertices



- the recursive algorithm for VPGs reasons about sets of vertex configuration pairs
- Attractor calculation example on VPG
- Function-wise representation to efficiently perform attractor calcs
- Short explanation of symbolic representation
- Time complexities

VPG algorithms - Recursive algorithm

- The recursive algorithm reasons about sets of vertices



- the recursive algorithm for VPGs reasons about sets of vertex configuration pairs
- Attractor calculation example on VPG
- Function-wise representation to efficiently perform attractor calcs
- Short explanation of symbolic representation
- Time complexities

VPG algorithms - Incremental pre-solve algorithm

- Introduce algorithm;, idea of pre-solving
- Introduce pessimistic PGs
- We need an alg to solve PGs using pre-solved vertices for efficiency

VPG algorithms - Incremental pre-solve algorithm

- FPIte, show FP formula
- Show modified FP formula
- Explain the efficiency gained

- Very short explanation of a fixed-point

VPG algorithms - Local solving

- explain local solving
- introduced local algs for the novel VPG algs and existing PG algs.

Experimental results - SPL games

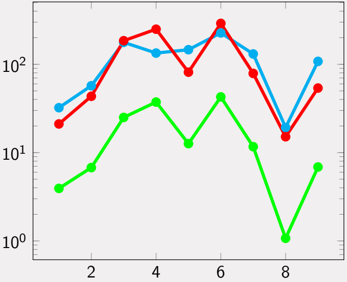


Figure: Running times, in ms, on the minepump games.

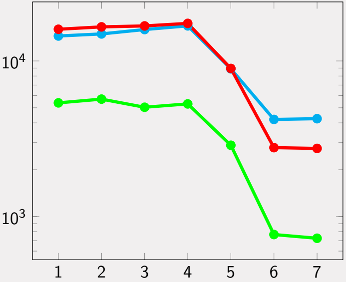


Figure: Running times, in ms, on the elevator games.

- Recursive algorithm for parity games (blue)
- Recursive algorithm for VPGs with a symbolic representation of configurations (green)
- Recursive algorithm for VPGs with an explicit representation of configurations (red)

Experimental results - SPL games

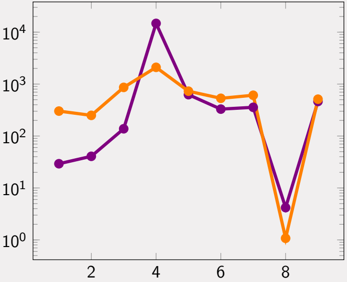


Figure: Running times, in ms, on the minepump games.

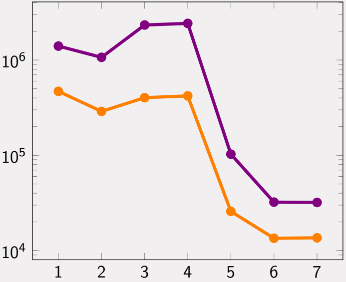


Figure: Running times, in ms, on the elevator games.

Fixed-point iteration algorithm for parity games
Incremental pre-solve algorithm

Experimental results - Random games

- Show the type of games where recursive symbolic fails and the explicit does not.

Experimental results - Local solving

- Show the same graphs but with local solving as well

Conclusions

- Collective approach can improve SPLs verifying performance
- The symbolic recursive can do this well
- The explicit recursive is "robust"
- Local solving can increase performance, however very dependent on alg & type of VPG