Family-Based SPL Model Checking Using Parity Games with Variability

(FASE 2020 paper presented at FASE 2021)

Maurice ter Beek Sjef van Loo Erik de Vink Tim Willemse

QAVS 2022

3 April 2022

FMT, ISTI–CNR, Pisa, Italy Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Context

- Software Product Lines:
  - collection of similar software systems
  - products obtained by configuring variation points

Context

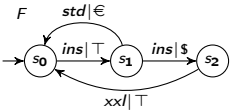
- Software Product Lines:
  - collection of similar software systems
  - products obtained by configuring variation points
- Model checking of SPL:
  - product based: analyse each configured system individually
  - family based: exploit commonalities to analyse families of systems

T. Thüm, S. Apel, C. Kästner, I. Schaefer & G. Saake, Survey of Analysis Strategies for Software Product Lines @ ACM Comput. Surv. 47, 2014
A.S. Dimovski, A.S. Al-Sibahi, C. Brabrand & A. Wąsowski, Family-based model checking without a family-based model checker @ SPIN'15
P. Chrszon, C. Dubslaff, S. Klüppelholz & C. Baier, Family-based modeling and analysis for probabilistic systems: featuring ProFeat @ FASE'16
M.H. ter Beek, E.P. de Vink & T.A.C. Willemse, Family-Based Model Checking with mCRL2 @ FASE'17
A.S. Dimovski & A. Wąsowski, Variability-specific Abstraction Refinement for Family-based Model Checking @ FASE'17
A.S. Dimovski, Abstract Family-Based Model Checking Using Modal Featured Transition Systems: Preservation of CTL* @ FASE'18

Example

A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay & J.-F. Raskin, Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and their Application to LTL Model Checking @ IEEE Trans. Softw. Eng. 39, 2013

FTS representing a product line of (four) coffee machines

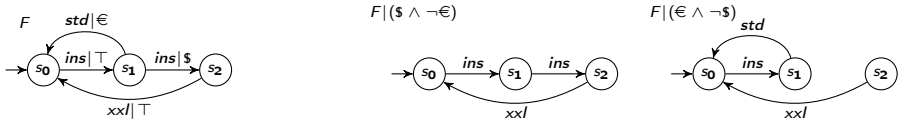


- independent features $, € represent the presence of a $/€-coin slot; product ≈ set of features
- *ins*: insert coin; *std*: standard coffee; *xxl*: extra large coffee
- feature expressions on edges indicate which products can take transition
  - e.g. $s_1 \xrightarrow{ins|\$} s_2$ is possible for all products with $-feature

Example

A. Classen, M. Cordy, P.-Y. Schobbens, P. Heymans, A. Legay & J.-F. Raskin, Featured Transition Systems: Foundations for Verifying Variability-Intensive Systems and their Application to LTL Model Checking @ IEEE Trans. Softw. Eng. 39, 2013

FTS representing a product line of (four) coffee machines



- independent features $\$, €$ represent the presence of a $/€-coin slot; product $\approx$ set of features
- *ins*: insert coin; *std*: standard coffee; *xxl*: extra large coffee
- feature expressions on edges indicate which products can take transition
  - e.g. $s_1 \xrightarrow{ins|\$} s_2$ is possible for all products with $-feature
- projection on products yield LTSs; e.g. LTSs $F|(\$ \wedge \neg €)$ and $F|(€ \wedge \neg \$)$

Problem Statement

The SPL model checking problem

Given: an FTS F, a set of products $\mathbb{P}$, and a μ-calculus formula ϕ

Problem: compute partition (P^+, P^-) of $\mathbb{P}$ with

- P^+: products satisfying ϕ, i.e., for all $p \in P^+$, $F|p \models \phi$
- P^-: products violating ϕ, i.e., for all $p \in P^-$, $F|p \not\models \phi$

Observations

- FTSs encode many LTSs
- Model checking of LTS = Parity Game solving
- Parity Games, like FTSs, are essentially graphs

Observations

- FTSs encode many LTSs
- Model checking of LTS = Parity Game solving
- Parity Games, like FTSs, are essentially graphs

Main idea:

- Add 'features' to Parity Games ..... Variability Parity Games
- VPGs encode many PGs
- Focus on 'family-based' solving VPGs

Observations

- FTSs encode many LTSs
- Model checking of LTS = Parity Game solving
- Parity Games, like FTSs, are essentially graphs

Main idea:

- Add 'features' to Parity Games ..... Variability Parity Games
- VPGs encode many PGs
- Focus on 'family-based' solving VPGs

Main results:

- Model checking of FTS reduces to VPG solving
- 'Family-based' algorithm for recursively solving VPGs
- 'Family-based' VPG solving outperforms 'product-based' VPG solving

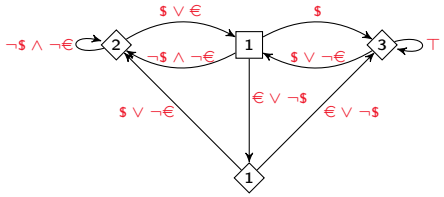
Variability Parity Games

VPGs

- Game arena: a graph (V, E) with 'featured' edges
- vertices are assigned a (natural number) priority
- Two players: $\Diamond$ (Even) and $\Box$ (Odd), each owning a set of vertices

VPGs

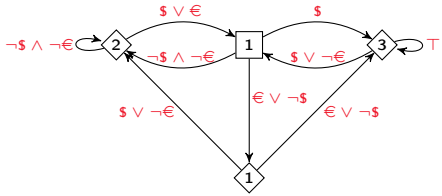
- Game arena: a graph (V, E) with 'featured' edges
- vertices are assigned a (natural number) priority
- Two players: $\Diamond$ (Even) and $\Box$ (Odd), each owning a set of vertices



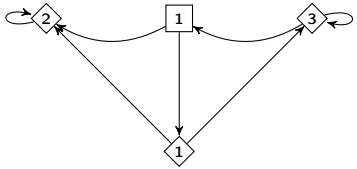
VPG for features $ and €

VPGs

- Game arena: a graph (V, E) with 'featured' edges
- vertices are assigned a (natural number) priority
- Two players: $\Diamond$ (Even) and $\Box$ (Odd), each owning a set of vertices



VPG for features $ and €



projection onto ¬$ ∧ ¬€

Variability Parity Games

Rules of the game:

1. place a token (for a product) on some vertex v
2. owner of the vertex v moves token to successor vertex v' via an edge accepting the product
3. Repeat step 2

Variability Parity Games

Rules of the game:

1. place a token (for a product) on some vertex v
2. owner of the vertex v moves token to successor vertex v' via an edge accepting the product
3. Repeat step 2

(p-)Play: maximal sequence of vertices visited by token (for product p)

Variability Parity Games

Rules of the game:

1. place a token (for a product) on some vertex v
2. owner of the vertex v moves token to successor vertex v' via an edge accepting the product
3. Repeat step 2

(p-)Play: maximal sequence of vertices visited by token (for product p)

Player $\Diamond$ wins $W \subseteq V$ for p iff from all $v \in W$ she has a strategy so that all p-plays are winning for $\Diamond$

Variability Parity Games

Rules of the game:

1. place a token (for a product) on some vertex v
2. owner of the vertex v moves token to successor vertex v' via an edge accepting the product
3. Repeat step 2

(p-)Play: maximal sequence of vertices visited by token (for product p)

Player $\Diamond$ wins $W \subseteq V$ for p iff from all $v \in W$ she has a strategy so that all p-plays are winning for $\Diamond$

where winning condition is the usual parity condition

- for infinite p-plays $v_1 v_2 \ldots$ player $\Diamond$ wins for p iff $\max(\inf(v_1 v_2 \ldots))$ is even
  - where $\inf(v_1 v_2 \ldots)$ is the set of priorities occurring infinitely often in $v_1 v_2 \ldots$
- for finite p-plays, the player that gets stuck loses

Reducing SPL Model Checking to VPG Solving

VPG encoding SPL model checking problem:

- Vertices: pairs of a state and (sub)formula
- Edges, priorities, owners and edge labels defined structurally:

Reducing SPL Model Checking to VPG Solving

VPG encoding SPL model checking problem:

- Vertices: pairs of a state and (sub)formula
- Edges, priorities, owners and edge labels defined structurally:

Table: For a given vertex v (1st column), its owner α (2nd column), successors $w \in vE$ (3rd column) and edge condition (3rd column), and priority (4th column) are given.

Vertex	Owner	Successor(s) | Configurations	Priority
$(s, \psi_1 \vee \psi_2)$	$\Diamond$	$(s, \psi_1) \mid \top$ and $(s, \psi_2) \mid \top$	0
$(s, \psi_1 \wedge \psi_2)$	$\Box$	$(s, \psi_1) \mid \top$ and $(s, \psi_2) \mid \top$	0
$(s, \langle a \rangle \psi)$	$\Diamond$	$(t, \psi) \mid \gamma$ for every $s \xrightarrow{a \mid \gamma}_F t$	0
$(s, [a]\psi)$	$\Box$	$(t, \psi) \mid \gamma$ for every $s \xrightarrow{a \mid \gamma}_F t$	0
$(s, \nu X.\psi)$	$\Box$	$(s, \psi[X := \nu X.\psi]) \mid \top$	$2\lfloor AD_\phi(X)/2 \rfloor$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Reducing SPL Model Checking to VPG Solving

VPG encoding SPL model checking problem:

- Vertices: pairs of a state and (sub)formula
- Edges, priorities, owners and edge labels defined structurally:

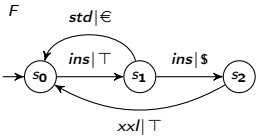
Table: For a given vertex v (1st column), its owner α (2nd column), successors $w \in vE$ (3rd column) and edge condition (3rd column), and priority (4th column) are given.

Vertex	Owner	Successor(s) | Configurations	Priority
$(s, \psi_1 \vee \psi_2)$	$\Diamond$	$(s, \psi_1) \mid \top$ and $(s, \psi_2) \mid \top$	0
$(s, \psi_1 \wedge \psi_2)$	$\Box$	$(s, \psi_1) \mid \top$ and $(s, \psi_2) \mid \top$	0
$(s, \langle a \rangle \psi)$	$\Diamond$	$(t, \psi) \mid \gamma$ for every $s \xrightarrow{a \mid \gamma}_F t$	0
$(s, [a]\psi)$	$\Box$	$(t, \psi) \mid \gamma$ for every $s \xrightarrow{a \mid \gamma}_F t$	0
$(s, \nu X.\psi)$	$\Box$	$(s, \psi[X := \nu X.\psi]) \mid \top$	$2\lfloor AD_\phi(X)/2 \rfloor$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

- Encoding satisfies: (s, ϕ) is won by $\Diamond$ for product p iff $s \models_{F \mid p} \phi$

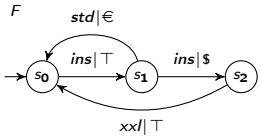
Example

Which products satisfy that all infinite paths contain an infinite number of *std* actions?



Example

Which products satisfy that all infinite paths contain an infinite number of *std* actions?

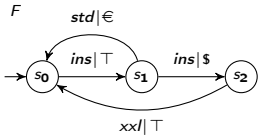


$$\overbrace{\nu X.\,\mu Y.\,\underbrace{\big(\underbrace{[ins]Y \wedge [xxl]Y}_{\phi_3} \wedge [std]X\big)}_{\phi_2}}^{\phi_1}$$

Verifying Families of Coffee Machines

Example

Which products satisfy that all infinite paths contain an infinite number of *std* actions?

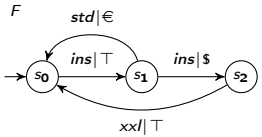


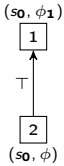
$\boxed{2}$
(s_0, ϕ)

$$\overbrace{\nu X.\,\mu Y.\,\underbrace{\underbrace{([ins]Y \wedge [xxl]Y}_{\phi_3} \wedge [std]X)}_{\phi_2}}^{\phi_1}}^{\phi}$$

Example

Which products satisfy that all infinite paths contain an infinite number of *std* actions?

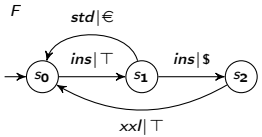


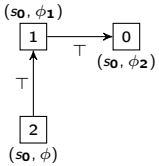


$$\overbrace{\nu X.\,\mu Y.\,\underbrace{\big(\underbrace{[ins]Y \wedge [xxl]Y}_{\phi_3} \wedge [std]X\big)}_{\phi_2}}^{\phi_1}$$

Example

Which products satisfy that all infinite paths contain an infinite number of *std* actions?





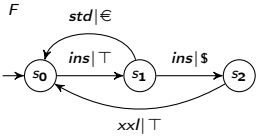
$$\overbrace{\nu X.\,\mu Y.\,\big(\underbrace{\underbrace{[ins]Y \wedge [xxl]Y}_{\phi_3} \wedge [std]X}_{\phi_2}\big)}^{\phi_1}$$

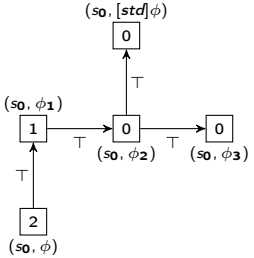
ϕ

Verifying Families of Coffee Machines

Example

Which products satisfy that all infinite paths contain an infinite number of *std* actions?



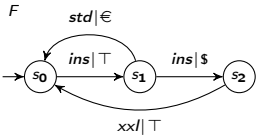


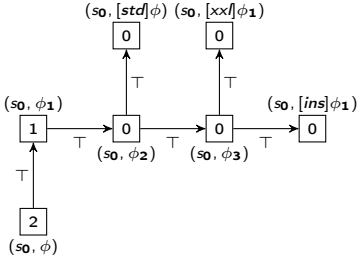
$$\overbrace{\nu X.\,\mu Y.\,\big(\underbrace{\overbrace{[ins]Y \wedge [xxl]Y}^{\phi_3} \wedge [std]X}_{\phi_2}\big)}^{\phi_1}$$

Verifying Families of Coffee Machines

Example

Which products satisfy that all infinite paths contain an infinite number of *std* actions?



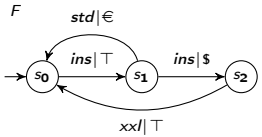


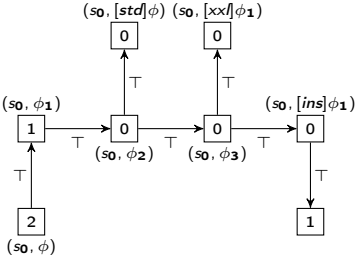
$$\underbrace{\nu X.\,\mu Y.\,\Big(\underbrace{\underbrace{[ins]Y \wedge [xxl]Y}_{\phi_3} \wedge [std]X}_{\phi_2}\Big)}_{\phi_1}\Big)}_{\phi}$$

Verifying Families of Coffee Machines

Example

Which products satisfy that all infinite paths contain an infinite number of *std* actions?



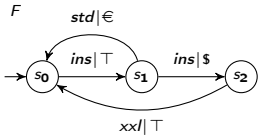


$$\overbrace{\nu X.\,\mu Y.\,\Big(\underbrace{\overbrace{[ins]Y \wedge [xxl]Y}^{\phi_3} \wedge [std]X}_{\phi_1}\Big)}^{\phi}$$
$$\phi_2$$

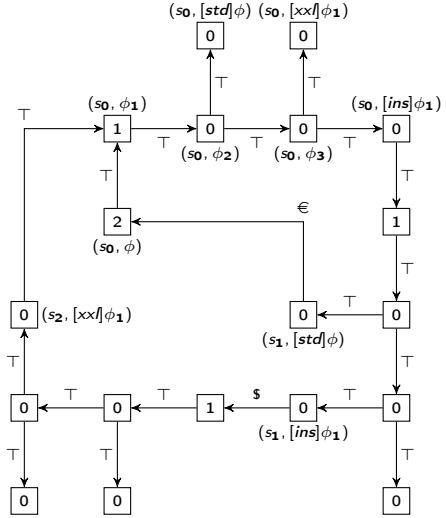
Verifying Families of Coffee Machines

Example

Which products satisfy that all infinite paths contain an infinite number of *std* actions?



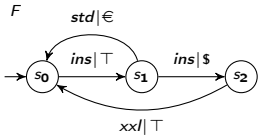
$$\underbrace{\nu X.\,\mu Y.\,\Big(\underbrace{\underbrace{[ins]Y \wedge [xxl]Y}_{\phi_3} \wedge [std]X}_{\phi_2}\Big)}_{\phi_1}\quad\phi$$



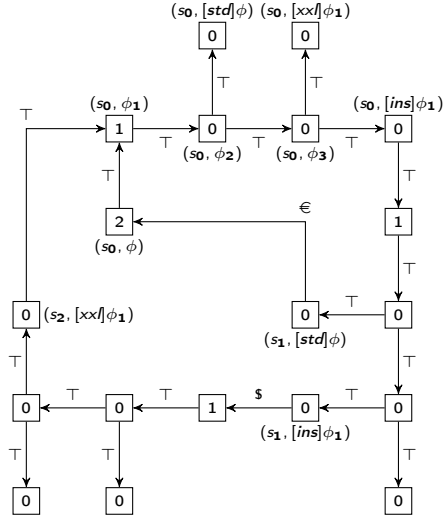
Verifying Families of Coffee Machines

Example

Which products satisfy that all infinite paths contain an infinite number of *std* actions?



$$\underbrace{\nu X.\,\mu Y.\,\overbrace{\big(\underbrace{[ins]Y}_{\phi_3}\wedge[xxl]Y\wedge[std]X\big)}^{\phi_2}}_{\phi_1}\qquad \phi$$



Answer: $(\neg\$, \$)$

Solving Variability Parity Games

- product based:
  - project VPG onto PGs
  - solve each PG indivually using Zielonka's recursive algorithm defined in
    Infinite games on finitely coloured graphs with applications to automata on infinite trees (TCS 1998)

Solving Variability Parity Games

- product based:
  - project VPG onto PGs
  - solve each PG indivually using Zielonka's recursive algorithm defined in *Infinite games on finitely coloured graphs with applications to automata on infinite trees* (TCS 1998)
- family based:
  - generalise Zielonka's recursive algorithm to VPGs
  - along recursions, associate each vertex to a family of relevant products
  - implementation: represent families of products as BDDs

Case Studies FASE 2020 Paper

- Minepump SPL: 582 states, 1376 transitions, 11 features, 128 products
  A. Classen, P. Heymans, P.-Y. Schobbens, A. Legay & J.-F. Raskin, Model Checking Lots of Systems: Efficient Verification of Temporal Properties in Software Product Lines @ ICSE 2010
- Elevator SPL: 95591 states, 622265 transitions, 5 features, 32 products
  M. Plath & M. Ryan, Feature integration using a feature construct @ Sci. Comput. Program. 41, 2001

Table: Running times (**in ms**) for experiments for the product-based and family-based SPL model checking of the minepump and elevator case studies. Note: purely solving times

Minepump SPL				Elevator SPL			
Property	product	family	$\lvert P^+\rvert$ / $\lvert P^-\rvert$	Property	product	family	$\lvert P^+\rvert$ / $\lvert P^-\rvert$
φ_1	28.88	3.92	128/0	ψ_1	14335	5409	2/30
φ_2	54.79	6.76	0/128	ψ_2	14988	5744	4/28
φ_3	184.7	24.70	0/128	ψ_3	16045	5020	4/28
φ_4	145.0	37.46	96/32	ψ_4	16865	5272	4/28
φ_5	144.5	12.19	96/32	ψ_5	8954	3013	16/16
φ_6	242.9	42.79	112/16	ψ_6	4252	772	32/0
φ_7	134.3	11.71	128/0	ψ_7	4171	765	32/0
φ_8	17.44	1.058	128/0	TOTAL	**79610**	**25995**	
φ_9	110.0	6.853	0/128				
TOTAL	**1061**	**147**					

Closing FASE 2021 Presentation

Summary:

- SPL model checking can be solved by VPG solving
- Family-based SPL model checking via VPGs outperforms product-based SPL model checking via VPGs

Closing FASE 2021 Presentation

Summary:

- SPL model checking can be solved by VPG solving
- Family-based SPL model checking via VPGs outperforms product-based SPL model checking via VPGs

Future work:

- study other algorithms for solving VPGs
- simplification of VPGs
- SPLs with feature upgrades (DSPL)

Ongoing Work

1. Lifted other algorithms to the SPL setting
   - the fixpoint algorithm defined in *The Fixpoint-Iteration Algorithm for Parity Games* (GandALF 2014) by Florian Bruse, Michael Falk & Martin Lange
   - Zielonka's algorithm + SCC decomposition defined in *Solving Parity Games in Practice* (ATVA 2009) by Oliver Friedmann & Martin Lange
   - the priority promotion algorithm defined in *Solving parity games via priority promotion* (FMSD 2018) by Massimo Benerecetti, Daniele Dell'Erba & Fabio Mogavero

- Sjef van Loo's M.Sc. thesis: *Verifying SPLs using parity games expressing variability* (TU/e 2019)
- Koen Degeling's M.Sc. thesis: *New algorithms and heuristics for solving Variability Parity Games* (TU/e 2021)

Ongoing Work

1. Lifted other algorithms to the SPL setting
   - the fixpoint algorithm defined in *The Fixpoint-Iteration Algorithm for Parity Games* (GandALF 2014) by Florian Bruse, Michael Falk & Martin Lange
   - Zielonka's algorithm + SCC decomposition defined in *Solving Parity Games in Practice* (ATVA 2009) by Oliver Friedmann & Martin Lange
   - the priority promotion algorithm defined in *Solving parity games via priority promotion* (FMSD 2018) by Massimo Benerecetti, Daniele Dell'Erba & Fabio Mogavero
2. Proved the correctness of these algorithm for VPGs and implemented these algorithms in mCRL2
   + an algorithm based on Jurdziński's small progress measures defined in *Featured Games* (TASE 2021) by Uli Fahrenberg & Axel Legay

- Sjef van Loo's M.Sc. thesis: *Verifying SPLs using parity games expressing variability* (TU/e 2019)
- Koen Degeling's M.Sc. thesis: *New algorithms and heuristics for solving Variability Parity Games* (TU/e 2021)

Ongoing Work

1. Lifted other algorithms to the SPL setting
   - the fixpoint algorithm defined in *The Fixpoint-Iteration Algorithm for Parity Games* (GandALF 2014) by Florian Bruse, Michael Falk & Martin Lange
   - Zielonka's algorithm + SCC decomposition defined in *Solving Parity Games in Practice* (ATVA 2009) by Oliver Friedmann & Martin Lange
   - the priority promotion algorithm defined in *Solving parity games via priority promotion* (FMSD 2018) by Massimo Benerecetti, Daniele Dell'Erba & Fabio Mogavero
2. Proved the correctness of these algorithm for VPGs and implemented these algorithms in mCRL2
   + an algorithm based on Jurdziński's small progress measures defined in *Featured Games* (TASE 2021) by Uli Fahrenberg & Axel Legay
3. Implemented several optimisations (e.g. self-loop elimination, early termination)

- Sjef van Loo's M.Sc. thesis: *Verifying SPLs using parity games expressing variability* (TU/e 2019)
- Koen Degeling's M.Sc. thesis: *New algorithms and heuristics for solving Variability Parity Games* (TU/e 2021)

Ongoing Work

1. Lifted other algorithms to the SPL setting
   - the fixpoint algorithm defined in *The Fixpoint-Iteration Algorithm for Parity Games* (GandALF 2014) by Florian Bruse, Michael Falk & Martin Lange
   - Zielonka's algorithm + SCC decomposition defined in *Solving Parity Games in Practice* (ATVA 2009) by Oliver Friedmann & Martin Lange
   - the priority promotion algorithm defined in *Solving parity games via priority promotion* (FMSD 2018) by Massimo Benerecetti, Daniele Dell'Erba & Fabio Mogavero
2. Proved the correctness of these algorithm for VPGs and implemented these algorithms in mCRL2
   + an algorithm based on Jurdziński's small progress measures defined in *Featured Games* (TASE 2021) by Uli Fahrenberg & Axel Legay
3. Implemented several optimisations (e.g. self-loop elimination, early termination)

- Sjef van Loo's M.Sc. thesis: *Verifying SPLs using parity games expressing variability* (TU/e 2019)
- Koen Degeling's M.Sc. thesis: *New algorithms and heuristics for solving Variability Parity Games* (TU/e 2021)

To do:

- evaluate the efficiency of these algorithms and the optimisations on SPL/FTS benchmark models