

Partial Order Reduction for State/Event LTL

N. Beneš, L. Brim, I. Černá, J. Sochor, P. Vařeková, B. Zimmerova

ParaDiSe laboratory, Faculty of Informatics, Masaryk University, Brno, Czech republic

MEMICS 2009

originally presented at iFM 2009
extended version to appear in SCP

Partial Order Reduction for State/Event LTL

N. Beneš, L. Brim, I. Černá, J. Sochor, P. Vařeková, B. Bührenová

ParaDiSe laboratory, Faculty of Informatics, Masaryk University, Brno, Czech republic

MEMICS 2009

originally presented at iFM 2009
extended version to appear in SCP

Component-based system development

- (prefabricated) autonomous components
- component interaction – critical correctness issue
- state/event specifications suitable for modelling
- State/Event LTL

Verification

- automated verification methods (model checking)
- state-space explosion problem
- techniques to reduce the state space
- Partial Order Reduction

State-based LTL

- model – Kripke structures

Basic operators

- p



- $X\varphi$



- $\psi U \varphi$



Derived operators

- $F\varphi$



- $G\varphi$



Action-based LTL

- model – labelled transitions systems

Basic operators

- $\mathcal{P}(a)$



- $\mathbf{X} \varphi$



- $\psi \mathbf{U} \varphi$



Derived operators

- $\mathbf{F} \varphi$



- $\mathbf{G} \varphi$



State/Event LTL

- model – labelled Kripke structures

Basic operators

- p



- $\mathcal{P}(a)$



- $\mathbf{X} \varphi$



- $\psi \mathbf{U} \varphi$



Derived operators

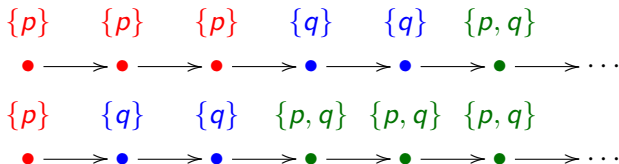
- $\mathbf{F} \varphi$



- $\mathbf{G} \varphi$



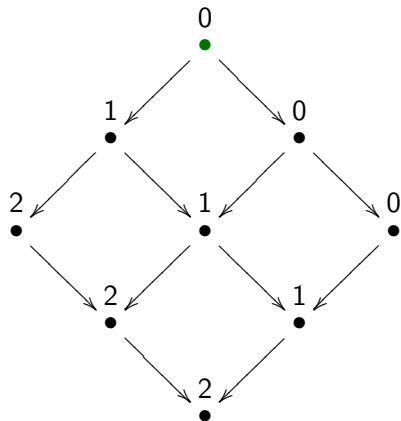
- exploiting commutativity of concurrent transitions
- based on the concept of *stuttering equivalence*



- all LTL-X properties preserved
- reduction: preserve at least one run of each equivalence class

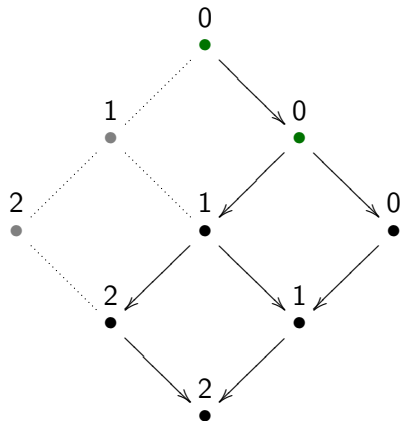
Partial Order Reduction – Example

- two concurrent processes
 $(\mathbf{inc\ }x; \mathbf{inc\ }x) \parallel (\mathbf{inc\ }y; \mathbf{inc\ }y)$
- only interested in value of x
- independence
 - $\mathbf{inc\ }x$ and $\mathbf{inc\ }y$ independent
- invisibility
 - transitions that do not change the labelling

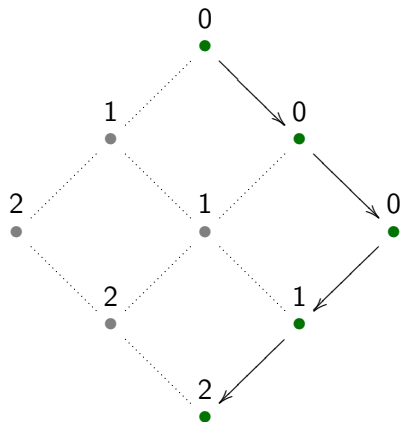


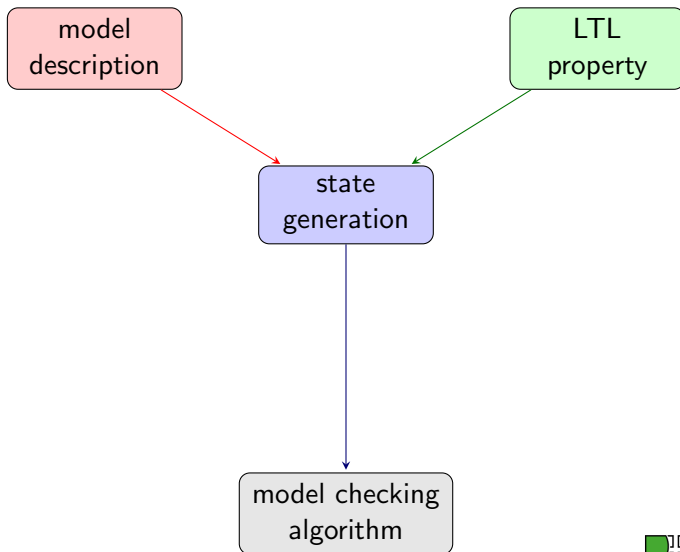
Partial Order Reduction – Example

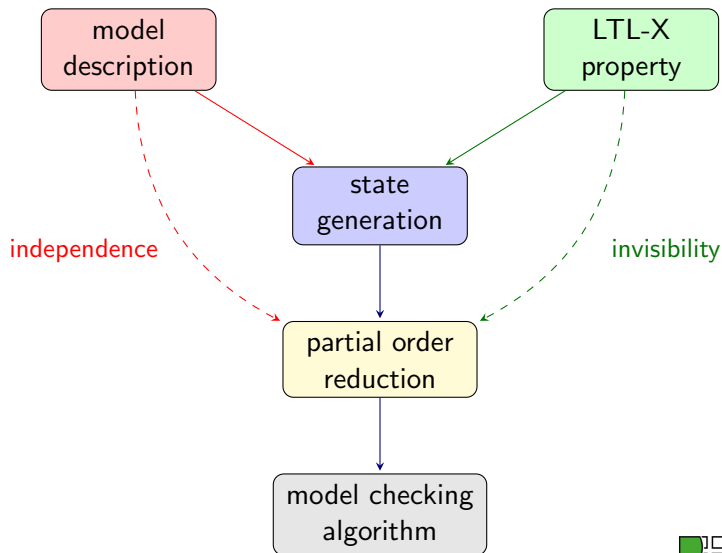
- two concurrent processes
 $(\mathbf{inc\ }x; \mathbf{inc\ }x) \parallel (\mathbf{inc\ }y; \mathbf{inc\ }y)$
- only interested in value of x
- independence
 - $\mathbf{inc\ }x$ and $\mathbf{inc\ }y$ independent
- invisibility
 - transitions that do not change the labelling



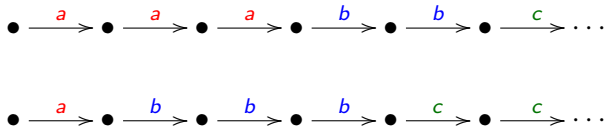
- two concurrent processes
 $(\mathbf{inc\ }x; \mathbf{inc\ }x) \parallel (\mathbf{inc\ }y; \mathbf{inc\ }y)$
- only interested in value of x
- independence
 - $\mathbf{inc\ }x$ and $\mathbf{inc\ }y$ independent
- invisibility
 - transitions that do not change the labelling







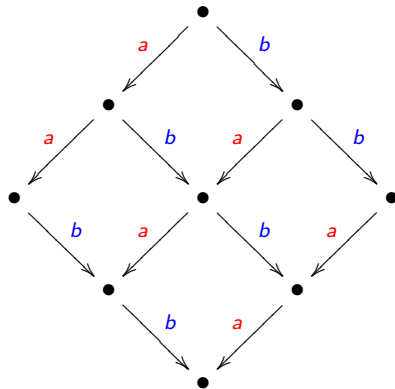
- need to find a concept similar to stuttering
- naive first idea: action stuttering



- preserves action-based LTL-X
- however, it is not the right choice

Why Does Action Stuttering Not Work?

- two concurrent processes
 $(a; a) \parallel (b; b)$
- only interested in action a
- partial order reduction would not preserve action stuttering
- six runs, none of them equivalent



Main idea

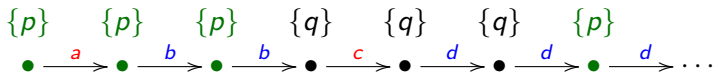
- retain the stuttering approach to state propositions
- different approach to actions
- some actions regarded as *interesting*
- transitions with noninteresting actions “overlooked”

Preserved SE-LTL properties

- no syntactic characterization
- new logic, weak SE-LTL

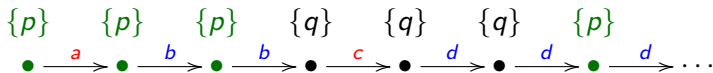
State/Event Stuttering Equivalence

- let interesting actions be $\{a, c\}$

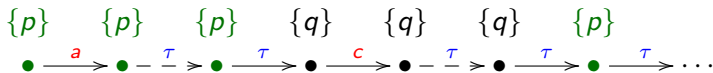


State/Event Stuttering Equivalence

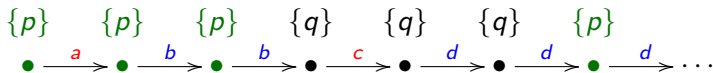
- let interesting actions be $\{a, c\}$



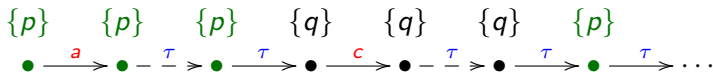
- projection** – change noninteresting actions into τ



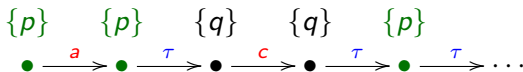
- let interesting actions be $\{a, c\}$



- projection** – change noninteresting actions into τ



- signature** – remove τ transitions with no state change



- two runs are equivalent if they have same signature



Weak State/Event LTL

● p





Weak State/Event LTL

- p



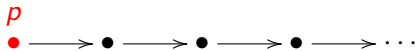
- $\mathcal{P}(a)$





Weak State/Event LTL

- p



- $\mathcal{P}(a)$



- $\tilde{\mathcal{P}}(a)$



- $\tilde{\mathcal{P}}(a)$ means “first *interesting* action is a ”



Weak State/Event LTL

- p



- $\mathcal{P}(a)$

- $\tilde{\mathcal{P}}(a)$



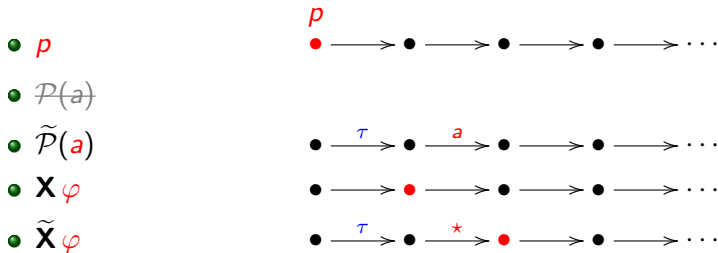
- $\mathbf{X} \varphi$



- $\tilde{\mathcal{P}}(a)$ means “first *interesting* action is a ”



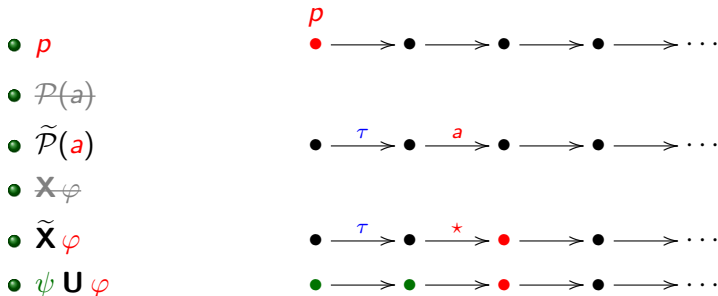
Weak State/Event LTL



- $\tilde{\mathcal{P}}(a)$ means “first *interesting* action is a ”
- $\tilde{\mathbf{X}}\varphi$ means “after next *interesting* action, φ holds”



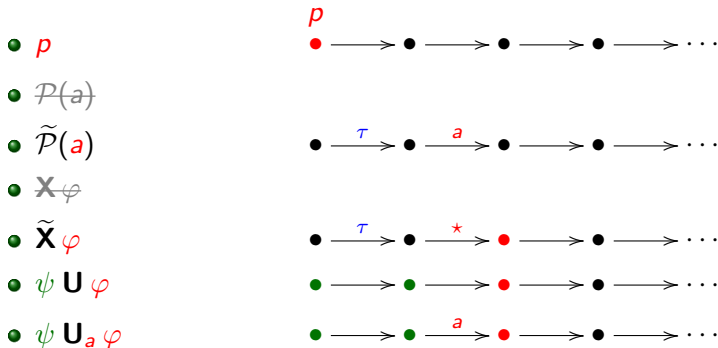
Weak State/Event LTL



- $\tilde{\mathcal{P}}(a)$ means “first *interesting* action is a ”
- $\tilde{\mathbf{X}}\varphi$ means “after next *interesting* action, φ holds”



Weak State/Event LTL



- $\tilde{\mathcal{P}}(a)$ means “first *interesting* action is a ”
- ~~$\tilde{\mathbf{X}}\varphi$~~ means “after next *interesting* action, φ holds”



Theorem

State/event stuttering equivalence preserves wSE-LTL properties.



Theorem

State/event stuttering equivalence preserves wSE-LTL properties.

Theorem

Every formula of wSE-LTL can be translated into a formula of SE-LTL.



Theorem

State/event stuttering equivalence preserves wSE-LTL properties.

Theorem

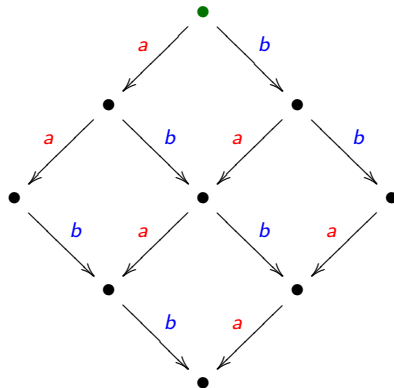
Every formula of wSE-LTL can be translated into a formula of SE-LTL.

Theorem

Every state/event stutter-invariant property expressible in SE-LTL is expressible in wSE-LTL.

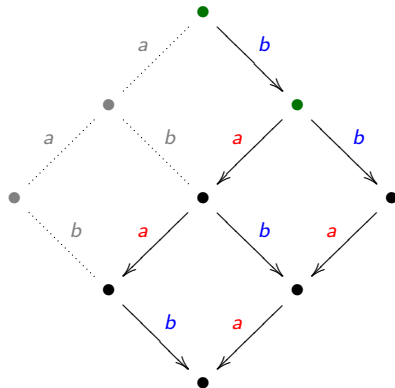


- two concurrent processes
 $(a; a) \parallel (b; b)$
- only interested in action a
- independence
 - a and b independent
- invisibility
 - τ transitions that do not change the labelling



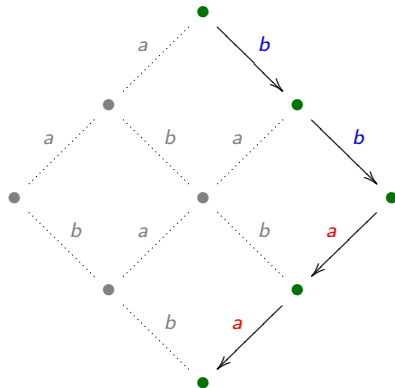


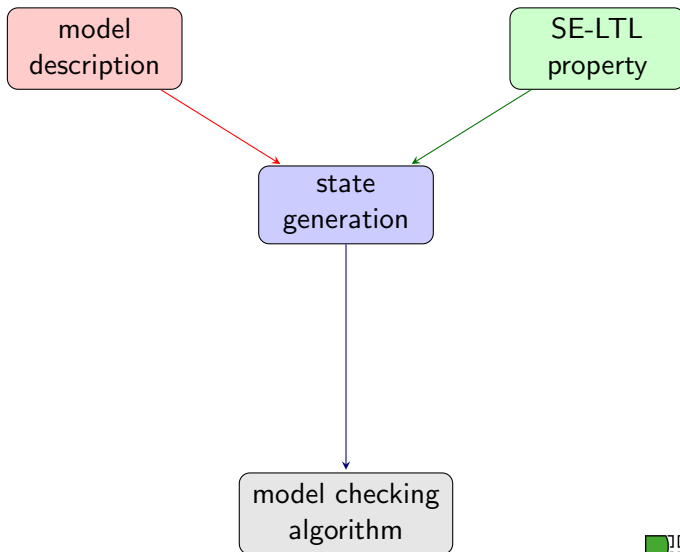
- two concurrent processes
 $(a; a) \parallel (b; b)$
- only interested in action a
- independence
 - a and b independent
- invisibility
 - τ transitions that do not change the labelling

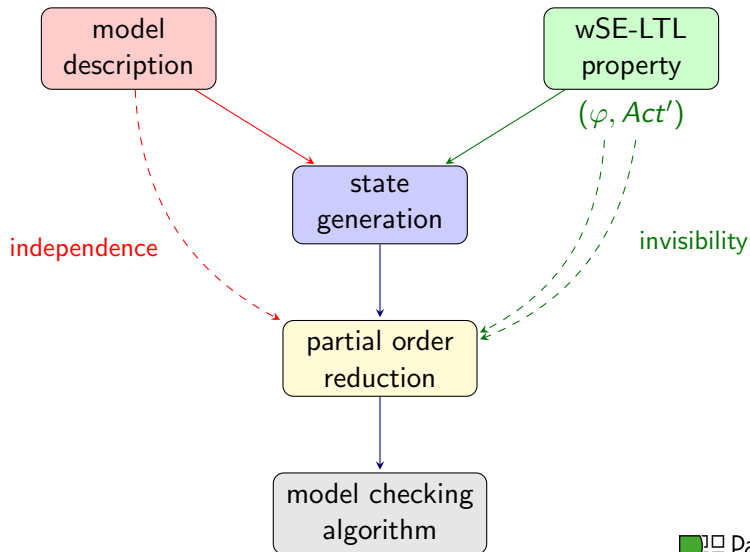




- two concurrent processes
 $(a; a) \parallel (b; b)$
- only interested in action a
- independence
 - a and b independent
- invisibility
 - τ transitions that do not change the labelling









Partial order reduction technique for SE-LTL

- state/event stuttering equivalence
- weak SE-LTL
- reduction task can be done the same way as in state-based case
 - known algorithms may be used
 - known efficiency

Further work

- employ the method in our verification framework
 - model checking of Component Interaction automata
 - tool based on DiVinE
- experimental evaluation on a case study

Frequently Asked Question No. 1

Q: Why don't you encode actions into states and then perform the standard method?

Frequently Asked Question No. 1

Q: Why don't you encode actions into states and then perform the standard method?

A: For several reasons:

- Encoding the actions into states makes the state space larger.
- LTL over such system would be weaker than wSE-LTL.
- Our approach works with no overhead.