

# *Don't know* for Multi-valued Systems

Alarico Competelli

joint work with Alexander Gruler,  
Martin Leucker and  
Daniel Thoma

Technische Universität München

Znojmo, November 14, 2009



## Definition (Model Checking)

- Specification of a property

## Definition (Model Checking)

- Specification of a property
- Implementation of system

## Definition (Model Checking)

- Specification of a property
- Implementation of system
- Question: Does the system meet its specification??

## Definition (Model Checking)

- Specification of a property **given by logical formula  $\varphi$**
- Implementation of system
- Question: Does the system meet its specification??

## Definition (Model Checking)

- Specification of a property given by logical formula  $\varphi$
- Implementation of system given by Kripke structure  $\mathcal{K}$
- Question: Does the system meet its specification??

## Definition (Model Checking)

- Specification of a property given by logical formula  $\varphi$
- Implementation of system given by Kripke structure  $\mathcal{K}$
- Question: Does the system meet its specification??

$$\mathcal{K} \models \varphi$$

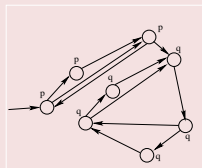
## Definition (Model Checking)

- Specification of a property given by logical formula  $\varphi$
- Implementation of system given by Kripke structure  $\mathcal{K}$
- Question: Does the system meet its specification??

$$\mathcal{K} \models \varphi$$

## Practical Definition

*Model Checking is a powerful and automatic analysis tool for verifying specifications. The answer is **yes** or **no** typically with a counterexample.*



# Multi-valued (mv) Model Checking

Multi-valued logic doesn't evaluate true or false but to many **truth values**



# Multi-valued (mv) Model Checking

Multi-valued logic doesn't evaluate true or false but to many **truth values**

## Definition (Multi-valued Model Checking)

- Specification of a property



# Multi-valued (mv) Model Checking

Multi-valued logic doesn't evaluate true or false but to many **truth values**

## Definition (Multi-valued Model Checking)

- Specification of a property
- Multi-valued model of system(s)



# Multi-valued (mv) Model Checking

Multi-valued logic doesn't evaluate true or false but to many **truth values**

## Definition (Multi-valued Model Checking)

- Specification of a property
- Multi-valued model of system(s)
- Question: To which extent does system meet its specification??



# Multi-valued (mv) Model Checking

Multi-valued logic doesn't evaluate true or false but to many **truth values**

## Definition (Multi-valued Model Checking)

- Specification of a property **given by logical formula  $\varphi$**
- Multi-valued model of system(s)
- Question: To which extent does system meet its specification??

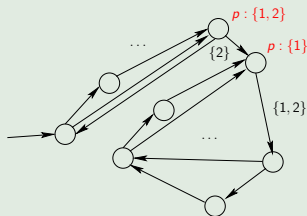


# Multi-valued (mv) Model Checking

Multi-valued logic doesn't evaluate true or false but to many **truth values**

## Definition (Multi-valued Model Checking)

- Specification of a property **given by logical formula  $\varphi$**
- Multi-valued model of system(s) **given by mv-Kripke structure  $\mathcal{K}$**
- Question: To which extent does system meet its specification??



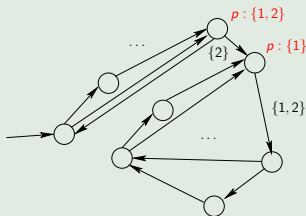
# Multi-valued (mv) Model Checking

Multi-valued logic doesn't evaluate true or false but to many **truth values**

## Definition (Multi-valued Model Checking)

- Specification of a property given by logical formula  $\varphi$
- Multi-valued model of system(s) given by mv-Kripke structure  $\mathcal{K}$
- Question: To which extent does system meet its specification??

$$\llbracket \varphi \rrbracket_{\mathcal{K}} = v$$



- 1 Traditional Abstractions
- 2 Multi-valued Logic
- 3 Optimistic-Pessimistic Abstractions
- 4 Causes for Indefinite Results
- 5 Conclusions

# Presentation outline

- 1 Traditional Abstractions
- 2 Multi-valued Logic
- 3 Optimistic-Pessimistic Abstractions
- 4 Causes for Indefinite Results
- 5 Conclusions



## State explosion problem

Major problem of model checking: state space explosion



## State explosion problem

Major problem of model checking: state space explosion

Approaches

- Optimizing time and space e.g. Symmetry Reduction

## State explosion problem

Major problem of model checking: state space explosion

Approaches

- Optimizing time and space e.g. Symmetry Reduction
- Partial Order Reduction

## State explosion problem

Major problem of model checking: state space explosion

Approaches

- Optimizing time and space e.g. Symmetry Reduction
- Partial Order Reduction
- *Abstraction*

## State explosion problem

Major problem of model checking: state space explosion

Approaches

- Optimizing time and space e.g. Symmetry Reduction
- Partial Order Reduction
- *Abstraction*

## Abstraction

- The model representation is too complex

## State explosion problem

Major problem of model checking: state space explosion

Approaches

- Optimizing time and space e.g. Symmetry Reduction
- Partial Order Reduction
- *Abstraction*

## Abstraction

- The model representation is too complex
- To abstract means remove informations from the model

## State explosion problem

Major problem of model checking: state space explosion

Approaches

- Optimizing time and space e.g. Symmetry Reduction
- Partial Order Reduction
- *Abstraction*

## Abstraction

- The model representation is too complex
- To abstract means remove informations from the model
- The verification is applied on the abstraction



## State explosion problem

Major problem of model checking: state space explosion

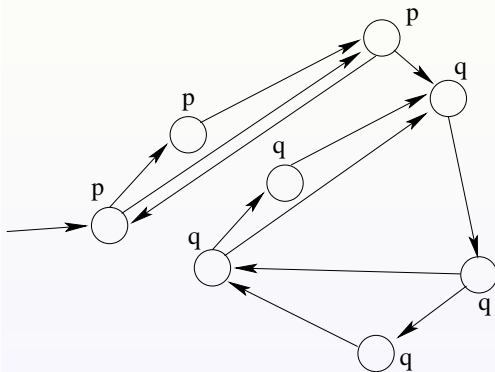
Approaches

- Optimizing time and space e.g. Symmetry Reduction
- Partial Order Reduction
- *Abstraction*

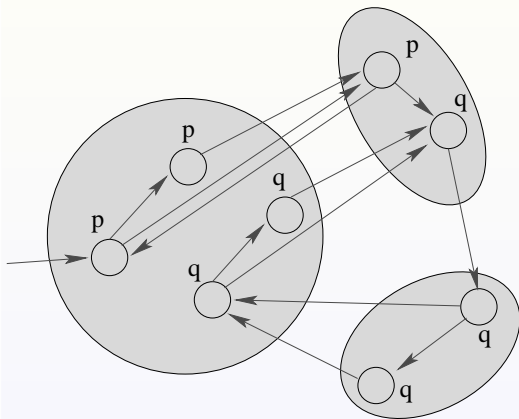
## Abstraction

- The model representation is too complex
- To abstract means remove informations from the model
- The verification is applied on the abstraction

# State Abstraction



# State Abstraction



# Presentation outline

- 1 Traditional Abstractions
- 2 Multi-valued Logic**
- 3 Optimistic-Pessimistic Abstractions
- 4 Causes for Indefinite Results
- 5 Conclusions



# Building a family of products



# Building a family of products



## Software product lines

- family of products = product line
- family of software systems = software product line
- one system model incorporating all products

## Definition (*Multi-valued Kripke structure (mv-KS)*)

$$\mathcal{T} = (\mathcal{S}, \mathcal{R}, L)$$

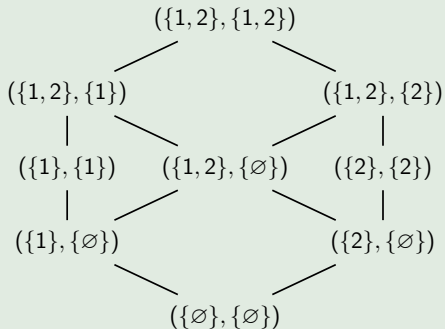
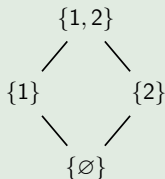
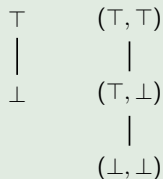
- $\mathcal{S}$  states
- $\mathcal{R}(\cdot, \cdot) : \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{L}$  valuation function
- $L : \mathcal{S} \rightarrow \mathcal{L}^{\mathcal{P}}$  value of proposition

## Motivation

Lattices specify in which truth value the property holds and the transition is defined.

# Examples

## Lattices



## Product Lines

$(2^N, \subseteq)$  – The powerset of all products

# Multi-valued Modal $\mu$ -Calculus

$\mu$ -calculus is used for specifying the properties to verify



# Multi-valued Modal $\mu$ -Calculus

$\mu$ -calculus is used for specifying the properties to verify

## Definition ( $mv\text{-}\mathcal{L}_\mu$ —Semantics)

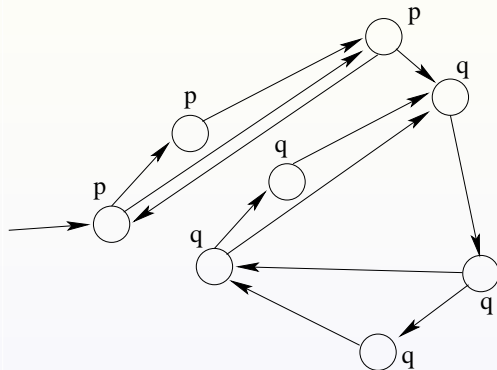
$\llbracket true \rrbracket_\rho$	$:=$	$\lambda s. \top$
$\llbracket false \rrbracket_\rho$	$:=$	$\lambda s. \perp$
$\llbracket q \rrbracket_\rho$	$:=$	$\lambda s. L(s)(q)$
$\llbracket \neg q \rrbracket_\rho$	$:=$	$\lambda s. \neg L(s)(q)$
$\llbracket Z \rrbracket_\rho$	$:=$	$\rho(Z)$
$\llbracket \varphi \vee \psi \rrbracket_\rho$	$:=$	$\llbracket \varphi \rrbracket_\rho \sqcup \llbracket \psi \rrbracket_\rho$
$\llbracket \varphi \wedge \psi \rrbracket_\rho$	$:=$	$\llbracket \varphi \rrbracket_\rho \sqcap \llbracket \psi \rrbracket_\rho$
$\llbracket \Diamond \varphi \rrbracket_\rho$	$:=$	$\lambda s. \bigsqcup \{ \mathcal{R}(s, s') \sqcap \llbracket \varphi \rrbracket_\rho(s') \}$
$\llbracket \Box \varphi \rrbracket_\rho$	$:=$	$\lambda s. \bigsqcap \{ \neg \mathcal{R}(s, s') \sqcup \llbracket \varphi \rrbracket_\rho(s') \}$
$\llbracket \mu Z. \varphi \rrbracket_\rho$	$:=$	$\bigsqcap \{ f \mid \llbracket \varphi \rrbracket_{\rho[Z \mapsto f]} \sqsubseteq f \}$
$\llbracket \nu Z. \varphi \rrbracket_\rho$	$:=$	$\bigsqcup \{ f \mid f \sqsubseteq \llbracket \varphi \rrbracket_{\rho[Z \mapsto f]} \}$

# Presentation outline

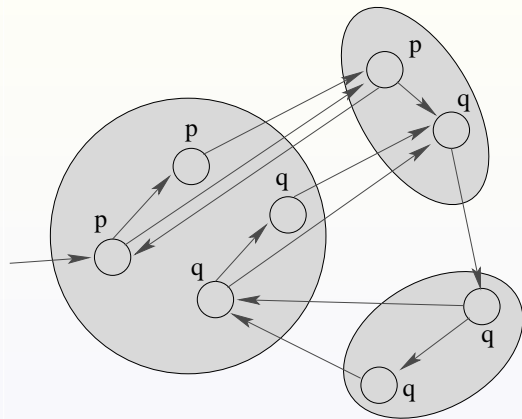
- 1 Traditional Abstractions
- 2 Multi-valued Logic
- 3 Optimistic-Pessimistic Abstractions**
- 4 Causes for Indefinite Results
- 5 Conclusions



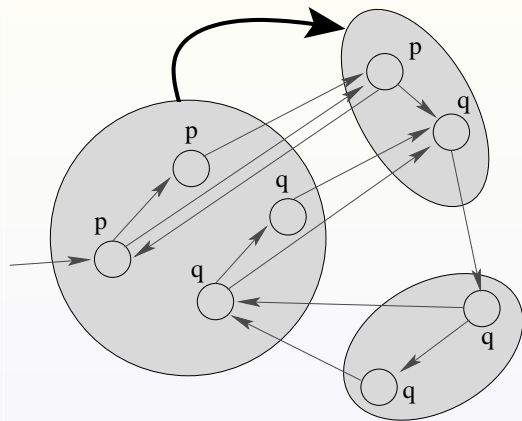
# Abstraction by joining states



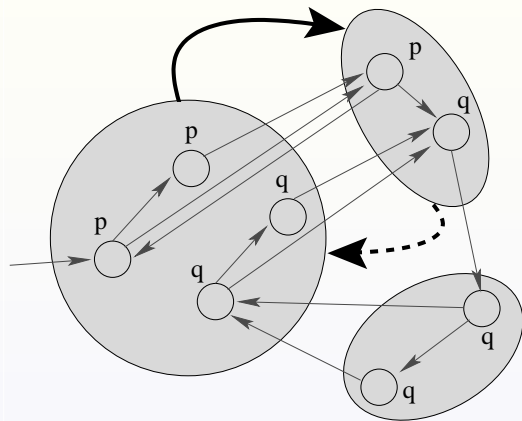
# Abstraction by joining states



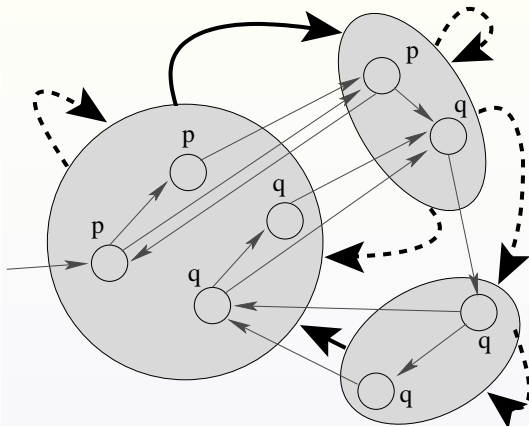
# Abstraction by joining states



# Abstraction by joining states



# Abstraction by joining states



must/may transitions

Abstraction by joining states induce an over- and under-approximation.

## Definition (*op-lattice*)

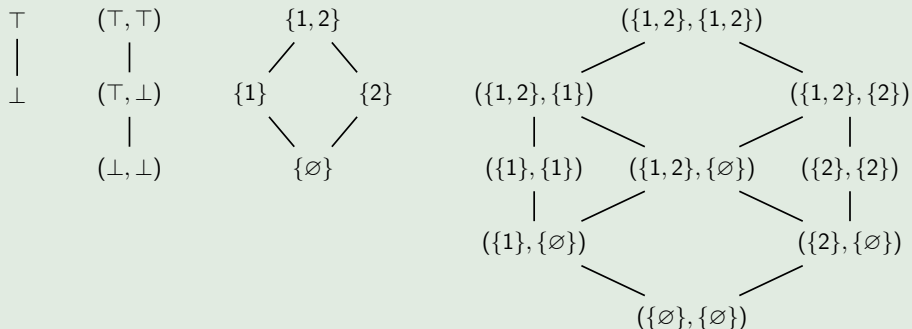
The optimistic-pessimistic lattice has two lattice components.

# Optimistic-pessimistic lattice

## Definition (*op-lattice*)

The optimistic-pessimistic lattice has two lattice components.

## Lattices and op-Lattices



# Abstraction by joining lattice elements

## Motivation

The complete lattice for a product line has  $2^n$  elements. It is convenient to also abstract lattice elements, identifying different products by reducing  $n$ .

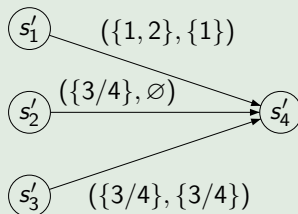
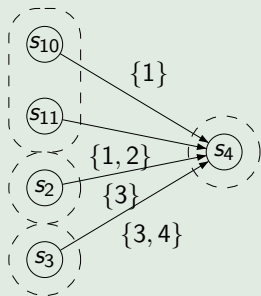


# Abstraction by joining lattice elements

## Motivation

The complete lattice for a product line has  $2^n$  elements. It is convenient to also abstract lattice elements, identifying different products by reducing  $n$ .

## Example



## Galois connection

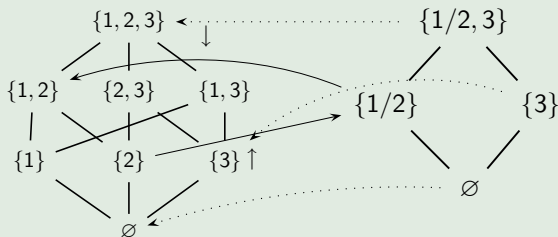
We use Galois connections as **concretization** and **abstraction** functions, between the concrete and the two abstract lattices.

# Abstraction by joining lattice elements

## Galois connection

We use Galois connections as **concretization** and **abstraction** functions, between the concrete and the two abstract lattices.

## Concrete and Abstract Lattices



## Conservative Abstraction

An abstraction is **conservative** with respect to evaluation if the results are correct in the concrete system.

# Correctness of Abstraction

## Conservative Abstraction

An abstraction is **conservative** with respect to evaluation if the results are correct in the concrete system.

## Theorem (Correctness of abstraction)

$$\uparrow_p(m_p) \sqsubseteq \llbracket \varphi \rrbracket_{\emptyset}^{\mathcal{K}^C}(s_C) \sqsubseteq \downarrow_o(m_o) \quad (m_o, m_p) = \llbracket \varphi \rrbracket_{\emptyset}^{\mathcal{K}^A}(s_A)$$



# Correctness of Abstraction

## Conservative Abstraction

An abstraction is **conservative** with respect to evaluation if the results are correct in the concrete system.

## Theorem (Correctness of abstraction)

$$\uparrow_p(m_p) \sqsubseteq \llbracket \varphi \rrbracket_{\emptyset}^{\mathcal{K}^C}(s_C) \sqsubseteq \downarrow_o(m_o) \quad (m_o, m_p) = \llbracket \varphi \rrbracket_{\emptyset}^{\mathcal{K}^A}(s_A)$$

## Abstraction and Refinement

- If optimistic and pessimistic values are **equal** the result is correct
- Otherwise the result is **indefinite**, one needs to **refine** the abstraction
- Notion of “**causes**” for guiding the refinement



# Presentation outline

- 1 Traditional Abstractions
- 2 Multi-valued Logic
- 3 Optimistic-Pessimistic Abstractions
- 4 Causes for Indefinite Results**
- 5 Conclusions



## Indefine Result

- We define **causes** from indefinite results for identifying the reasons for such result

## Indefinite Result

- We define **causes** from indefinite results for identifying the reasons for such result
- Causes provide a **criterion** for refining the abstraction, by concretizing states and lattice elements

## Indefine Result

- We define **causes** from indefinite results for identifying the reasons for such result
- Causes provide a **criterion** for refining the abstraction, by concretizing states and lattice elements
- The refinement by causes prevents the occurrence of the same indefinite value

## Indefine Result

- We define **causes** from indefinite results for identifying the reasons for such result
- Causes provide a **criterion** for refining the abstraction, by concretizing states and lattice elements
- The refinement by causes prevents the occurrence of the same indefinite value

## Relevant cases

- (i) the evaluation of an atomic proposition in a state
- (ii) the evaluation of a transition relation between two states
- (iii) the computation of negation, meet or join operation on lattices



## Imprecision in atomic proposition

**p** evaluates in **s** to the tuple  $(\{1, 2, 3/4, 5\}, \{2, 3/4/5\})$

## Imprecision in atomic proposition

$p$  evaluates in  $s$  to the tuple  $(\{1, 2, 3/4, 5\}, \{2, 3/4/5\})$

The resulting cause is  $(s, p, (\{1, 2, 3, 4, 5\}, \{2, 3, 4, 5\}))$

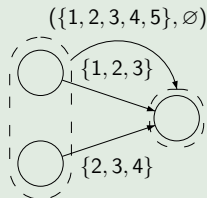
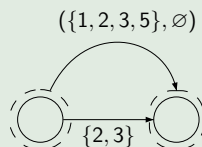
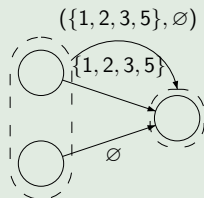
# Examples

## Imprecision in atomic proposition

$p$  evaluates in  $s$  to the tuple  $(\{1, 2, 3/4, 5\}, \{2, 3/4/5\})$

The resulting cause is  $(s, p, (\{1, 2, 3, 4, 5\}, \{2, 3, 4, 5\}))$

## Imprecision in transition relation



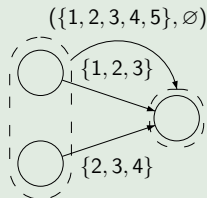
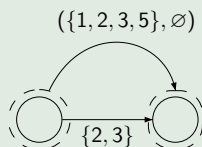
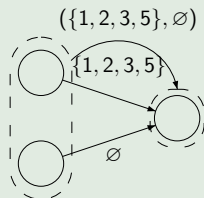
- Imprecision due to state abstraction

## Imprecision in atomic proposition

$p$  evaluates in  $s$  to the tuple  $(\{1, 2, 3/4, 5\}, \{2, 3/4/5\})$

The resulting cause is  $(s, p, (\{1, 2, 3, 4, 5\}, \{2, 3, 4, 5\}))$

## Imprecision in transition relation



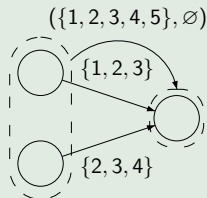
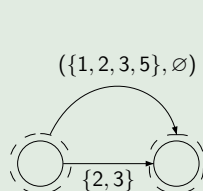
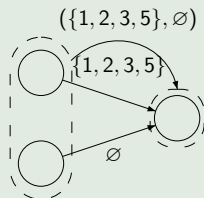
- Imprecision due to state abstraction
- Imprecision due to lattice abstraction

## Imprecision in atomic proposition

$p$  evaluates in  $s$  to the tuple  $(\{1, 2, 3/4, 5\}, \{2, 3/4/5\})$

The resulting cause is  $(s, p, (\{1, 2, 3, 4, 5\}, \{2, 3, 4, 5\}))$

## Imprecision in transition relation



- Imprecision due to state abstraction
- Imprecision due to lattice abstraction
- Combination of both

# Presentation outline

- 1 Traditional Abstractions
- 2 Multi-valued Logic
- 3 Optimistic-Pessimistic Abstractions
- 4 Causes for Indefinite Results
- 5 Conclusions



## We have shown

- abstractions for multi-valued systems

## We have shown

- abstractions for multi-valued systems
- by joining states

## We have shown

- abstractions for multi-valued systems
- by joining states
- by abstraction of truth values

## We have shown

- abstractions for multi-valued systems
- by joining states
- by abstraction of truth values
- as multi-valued model checking problem

## We have shown

- abstractions for multi-valued systems
- by joining states
- by abstraction of truth values
- as multi-valued model checking problem
- identified causes for indefinite results

## We have shown

- abstractions for multi-valued systems
- by joining states
- by abstraction of truth values
- as multi-valued model checking problem
- identified causes for indefinite results

## We have shown

- abstractions for multi-valued systems
- by joining states
- by abstraction of truth values
- as multi-valued model checking problem
- identified causes for indefinite results

## Future work

- multi-valued abstraction for other models

## We have shown

- abstractions for multi-valued systems
- by joining states
- by abstraction of truth values
- as multi-valued model checking problem
- identified causes for indefinite results

## Future work

- multi-valued abstraction for other models
- implementation?

## We have shown

- abstractions for multi-valued systems
- by joining states
- by abstraction of truth values
- as multi-valued model checking problem
- identified causes for indefinite results

## Future work

- multi-valued abstraction for other models
- implementation?

## We have shown

- abstractions for multi-valued systems
- by joining states
- by abstraction of truth values
- as multi-valued model checking problem
- identified causes for indefinite results

## Future work

- multi-valued abstraction for other models
- implementation?

Thank You!

