Model Checking Product Lines

Martin Leucker

partially joint work with Alarico
Campetelli, Alexander Gruler and
Daniel Thoma

University of Lübeck

Dagstuhl, February 25th, 2013

Outline

Presentation outline

Building a family of products

Building a family of products

family of products = product line

Software Product Family

How to deal with software product lines?

- how to model software product lines?
- how to verify software product lines?
- how to model software product lines to allow their verification?

Software Product Family

How to deal with software product lines?

- how to model software product lines?
- how to verify software product lines?
- how to model software product lines to allow their verification?
- *one system model incorporating all products*
- PL-CCS: product line extension of Milner's CCS [FMOODS'08]

Software Product Family

Dijstra'72

If a program has to exist in two different versions, I would rather not regard (the text of) the one program as a modification of (the text of) the other. It would be much more attractive if the two different programs could, in some sense or another, be viewed as, say, different children from a common ancestor, where the ancestor represents a more or less abstract program, embodying what the two versions have in common.

Software Product Line

Definition [Clements&Northrop]

A *software product line* is a set of software intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets.

Software Product Line

Definition [Clements&Northrop]

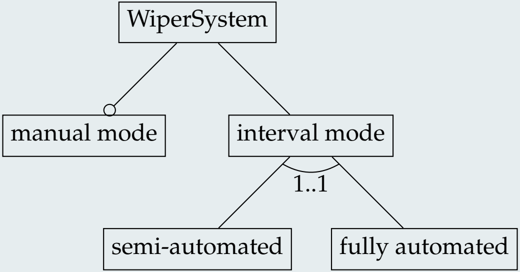
A *software product line* is a set of software intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets.

Software Product Line

Definition (Feature)

A *feature* is the ability of a product to cover a certain use case or meet a certain customer need.

Feature Diagram



Feature versus Product Line

Different views

- Feature: Customer view
- SPL: Technical view
- It is frequently impossible to map features independently to certain technical properties (=core assets).
- Mapping features combinations to products is no homomorphism!

Definition (Features to Products)

$\mathcal{F} : \mathbb{P} \to 2^{\mathbb{F}}$ is a *feature function* mapping products $p \in \mathbb{P}$ to features $f \in \mathbb{F}$ they have.

Definition (Feasible Feature Combinations)

The set $F \subseteq \mathbb{F}$ is a *feasible feature combination* if $\exists p \in \mathbb{P} : F \subseteq \mathcal{F}(p)$.

The core (of PL-CCS)

Variability = Choice Points

$$\text{wiper} := \text{wiper}_1 \oplus_1 \text{wiper}_2; \quad \text{sensor} := \text{sensor}_1 \oplus_2 \text{sensor}_2$$

Composition of assets

$$\text{wiper} || \text{sensor}$$

PL-CCS Semantics

Three semantics

- flat semantics

PL-CCS Semantics

Three semantics

- flat semantics
- unfolded semantics

PL-CCS Semantics

Three semantics

- flat semantics
- unfolded semantics
- configured-transitions semantics

Flat Semantics

Definition (fully configured)

Given a well-formed PL-CCS program with N variants operators, we call a corresponding configuration vector

$$\theta \in \{R, L, ?\}^N$$

fully configured if

$$\theta \in \{R, L\}^N$$

From a PL-CCS program to a set of CCS programs

$$\mathit{config} : \mathcal{P} \times \{R, L, ?\}^N \mapsto \mathcal{R}$$

Definition (flat semantics)

$$[\![\mathit{Prog}]\!]_{\mathit{Flat}} = \left\{ [\![V]\!]_{\mathrm{CCS}} \;\mid\; \exists\theta : \mathit{config}(\mathit{Prog}, \theta) = V \right\}$$

Unfolded Semantics

Definition (PL-LTS)

A *product-line transition system* (PL-LTS) with N variants operators is a tuple $(\mathcal{S}, \mathcal{A}, \Delta, \sigma)$, where

- $\mathcal{S}$ is a (countably, possibly infinite) set of states,
- $\mathcal{A}$ is a set of actions, and
- Δ is a finite set of transition relations of the form $\xrightarrow{\alpha,\ \nu} \subseteq \mathcal{S} \times \mathcal{S}$, where $\alpha \in \mathcal{A}$, $\nu \in \times\{R, L, ?\}^N$,
- and $\sigma \in \mathcal{S}$ is the start state.

From a PL-CCS program to a PL-LTS

SOS rules

$$\frac{P, \nu \xrightarrow{\alpha,\ \nu} P', \nu}{C, \nu \xrightarrow{\alpha,\ \nu} P', \nu}\ ,\ C \stackrel{def}{=} P \qquad (\textit{constant definition})$$

$$\frac{}{\alpha.P, \nu \xrightarrow{\alpha,\ \nu} P, \nu}\ ,\ \text{for arbitrary } \nu \in \{R, L, ?\}^N \qquad (\textit{prefix})$$

$$\frac{P_j, \nu \xrightarrow{\alpha,\ \nu} P'_j, \nu}{P_1 + P_2, \nu \xrightarrow{\alpha,\ \nu} P'_j, \nu}\ ,\ j \in \{1, 2\} \qquad (\textit{summation})$$

$$\frac{P, \nu \xrightarrow{\alpha,\ \nu} P', \nu}{(P \parallel Q), \nu \xrightarrow{\alpha,\ \nu} (P' \parallel Q), \nu} \qquad (\textit{parallel composition (1)})$$

$$\vdots$$

Presentation outline

Model Checking

Definition (Model Checking)

- Specification of system

Model Checking

Definition (Model Checking)

- Specification of system
- Implementation of system

Model Checking

Definition (Model Checking)

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

Model Checking

Definition (Model Checking)

- Specification of system given by logical formula φ
- Implementation of system
- Question: Does the system meet its specification??

Model Checking

Definition (Model Checking)

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

Model Checking

Definition (Model Checking)

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

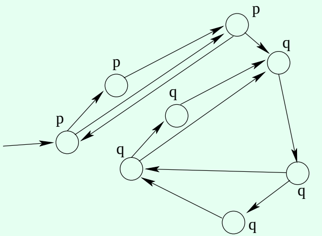
$$\mathcal{K} \models \varphi$$

Model Checking

Definition (Model Checking)

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

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



$\models$ AG(EX*true*)

Model Checking

Definition (Model Checking)

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

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

Practical Definition

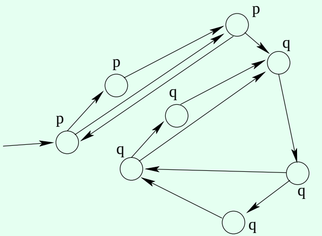
Model Checking is a powerful analysis tool
parameterized via a logical specification

Model Checking

Definition (Model Checking)

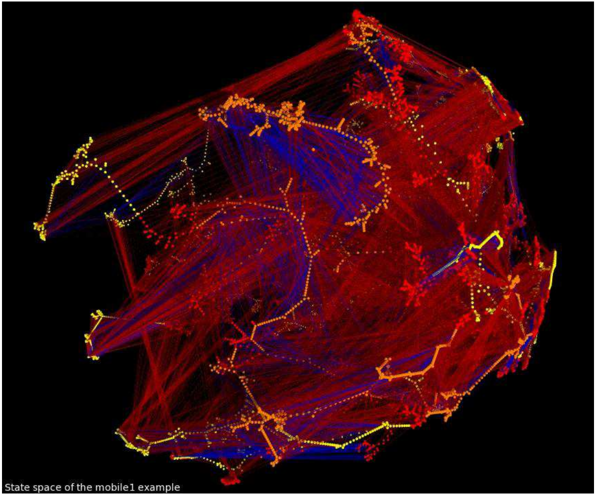
- Specification of system given by logical formula φ
- Implementation of system given by Kripke structure $\mathcal{K}$
- Question: Does the system meet its specification??

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



$\models$ AG(EX*true*)

State Space



Multi-valued (mv) Model Checking

Definition (Multi-valued Model Checking)

- Specification of a property

Multi-valued (mv) Model Checking

Definition (Multi-valued Model Checking)

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

Multi-valued (mv) Model Checking

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

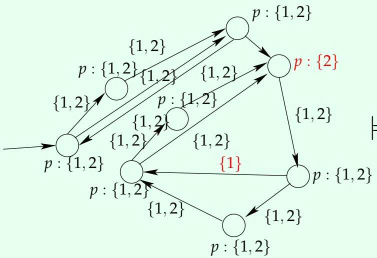
Definition (Multi-valued Model Checking)

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

Multi-valued (mv) Model Checking

Definition (Multi-valued Model Checking)

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



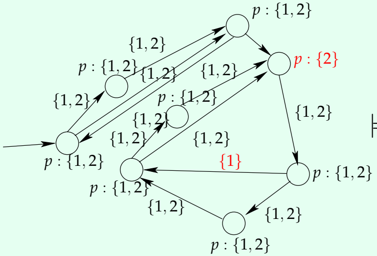
$\models$ $\mathrm{AG}(\mathrm{EX}\mathit{true})$

Multi-valued (mv) Model Checking

Definition (Multi-valued Model Checking)

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

$$[\![\varphi]\!]_{\mathcal{K}} = v$$



$\models$ AG(EX*true*)

Thesis

Rational

Model Checking Product Lines is Multi-valued Model Checking

However. . .

. . . there are different approaches

based on open system's verification:

`http://cs.brown.edu/~sk/Publications/Papers/Published/lkf-verif-cc-features-open-sys/`

and

`http://cs.brown.edu/~sk/Publications/Papers/Published/bfkv-param-int-open-sys-verif-prod-line/`

but this is not considered here.

Presentation outline

Lattices

Lattices

- *lattice* is a partially ordered set $(\mathcal{L}, \sqsubseteq)$
- where for each $x, y \in \mathcal{L}$, there exists
  - a unique *greatest lower bound* (glb) $x \sqcap y$, and
  - a unique *least upper bound* (lub) $x \sqcup y$.
- *bottom* $\bot$ *top* $\top$
- *distributive* iff

$$x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$$

$$x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z)$$

- *DeMorgan*

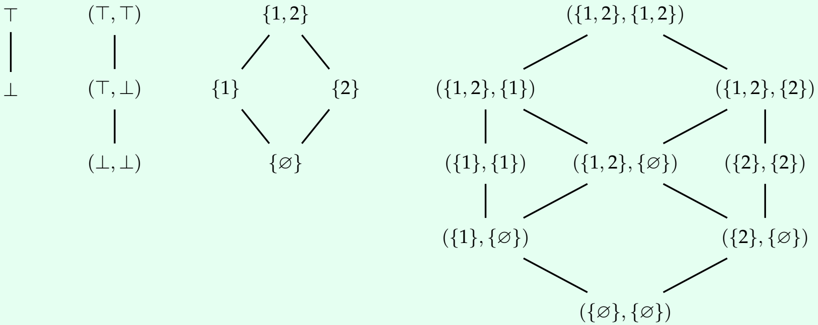
$$\neg\neg x = x$$

- *Boolean* iff complete, distributive, and

$$x \sqcup \neg x = \top \qquad x \sqcap \neg x = \bot$$

Examples

Lattices



Product Lines

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

Multi-valued Modal Kripke Structure

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

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

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

Multi-valued μ-Calculus

Definition (mv-$\mathfrak{L}_\mu$—Syntax)

$$\begin{aligned}\varphi \quad ::= \quad & \mathit{true} \mid \mathit{false} \mid q \mid \neg q \mid Z \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \\ & \Diamond\varphi \mid \Box\varphi \mid \\ & \mu Z.\varphi \mid \nu Z.\varphi\end{aligned}$$

Multi-valued Modal μ-Calculus

Definition (mv-$\mathfrak{L}_\mu$—Semantics)

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

Multi-valued Modal μ-Calculus

Definition (mv-$\mathfrak{L}_\mu$—Semantics)

$$
\begin{array}{rcl}
[\![true]\!]_\rho & := & \lambda s.\top \\
[\![false]\!]_\rho & := & \lambda s.\bot \\
[\![q]\!]_\rho & := & \lambda s.L(s)(q) \\
[\![\neg q]\!]_\rho & := & \lambda s.\neg L(s)(q) \\
[\![Z]\!]_\rho & := & \rho(Z) \\
[\![\varphi \vee \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcup [\![\psi]\!]_\rho \\
[\![\varphi \wedge \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcap [\![\psi]\!]_\rho \\
[\![\Diamond\varphi]\!]_\rho & := & \lambda s. \bigsqcup\{\mathcal{R}(s,s') \sqcap [\![\varphi]\!]_\rho(s')\} \\
[\![\Box\varphi]\!]_\rho & := & \lambda s. \bigsqcap\{\neg\mathcal{R}(s,s') \sqcup [\![\varphi]\!]_\rho(s')\} \\
[\![\mu Z.\varphi]\!]_\rho & := & \bigsqcap \{f \mid [\![\varphi]\!]_{\rho[Z \mapsto f]} \sqsubseteq f\} \\
[\![\nu Z.\varphi]\!]_\rho & := & \bigsqcup \{f \mid f \sqsubseteq [\![\varphi]\!]_{\rho[Z \mapsto f]}\}
\end{array}
$$

Multi-valued Modal μ-Calculus

Definition (mv-$\mathfrak{L}_\mu$—Semantics)

$$
\begin{array}{rcl}
[\![true]\!]_\rho & := & \lambda s.\top \\
[\![false]\!]_\rho & := & \lambda s.\bot \\
[\![q]\!]_\rho & := & \lambda s.L(s)(q) \\
[\![\neg q]\!]_\rho & := & \lambda s.\neg L(s)(q) \\
[\![Z]\!]_\rho & := & \rho(Z) \\
[\![\varphi \vee \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcup [\![\psi]\!]_\rho \\
[\![\varphi \wedge \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcap [\![\psi]\!]_\rho \\
[\![\Diamond\varphi]\!]_\rho & := & \lambda s. \bigsqcup\{\mathcal{R}(s,s') \sqcap [\![\varphi]\!]_\rho(s')\} \\
[\![\Box\varphi]\!]_\rho & := & \lambda s. \bigsqcap\{\neg\mathcal{R}(s,s') \sqcup [\![\varphi]\!]_\rho(s')\} \\
[\![\mu Z.\varphi]\!]_\rho & := & \bigsqcap \{f \mid [\![\varphi]\!]_{\rho[Z\mapsto f]} \sqsubseteq f\} \\
[\![\nu Z.\varphi]\!]_\rho & := & \bigsqcup \{f \mid f \sqsubseteq [\![\varphi]\!]_{\rho[Z\mapsto f]}\}
\end{array}
$$

Multi-valued Modal μ-Calculus

Definition (mv-$\mathfrak{L}_\mu$—Semantics)

$$
\begin{array}{rcl}
[\![true]\!]_\rho & := & \lambda s.\top \\
[\![false]\!]_\rho & := & \lambda s.\bot \\
[\![q]\!]_\rho & := & \lambda s.L(s)(q) \\
[\![\neg q]\!]_\rho & := & \lambda s.\neg L(s)(q) \\
[\![Z]\!]_\rho & := & \rho(Z) \\
[\![\varphi \vee \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcup [\![\psi]\!]_\rho \\
[\![\varphi \wedge \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcap [\![\psi]\!]_\rho \\
[\![\Diamond\varphi]\!]_\rho & := & \lambda s.\bigsqcup\{\mathcal{R}(s,s') \sqcap [\![\varphi]\!]_\rho(s')\} \\
[\![\Box\varphi]\!]_\rho & := & \lambda s.\bigsqcap\{\neg\mathcal{R}(s,s') \sqcup [\![\varphi]\!]_\rho(s')\} \\
[\![\mu Z.\varphi]\!]_\rho & := & \bigsqcap\{f \mid [\![\varphi]\!]_{\rho[Z\mapsto f]} \sqsubseteq f\} \\
[\![\nu Z.\varphi]\!]_\rho & := & \bigsqcup\{f \mid f \sqsubseteq [\![\varphi]\!]_{\rho[Z\mapsto f]}\}
\end{array}
$$

Multi-valued Modal μ-Calculus

Definition (mv-$\mathfrak{L}_\mu$—Semantics)

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

Multi-valued Modal μ-Calculus

Definition (mv-$\mathfrak{L}_\mu$—Semantics)

$$
\begin{array}{rcl}
[\![true]\!]_\rho & := & \lambda s.\top \\
[\![false]\!]_\rho & := & \lambda s.\bot \\
[\![q]\!]_\rho & := & \lambda s.L(s)(q) \\
[\![\neg q]\!]_\rho & := & \lambda s.\neg L(s)(q) \\
[\![Z]\!]_\rho & := & \rho(Z) \\
[\![\varphi \vee \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcup [\![\psi]\!]_\rho \\
[\![\varphi \wedge \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcap [\![\psi]\!]_\rho \\
[\![\Diamond\varphi]\!]_\rho & := & \lambda s. \bigsqcup\{\mathcal{R}(s,s') \sqcap [\![\varphi]\!]_\rho(s')\} \\
[\![\Box\varphi]\!]_\rho & := & \lambda s. \bigsqcap\{\neg\mathcal{R}(s,s') \sqcup [\![\varphi]\!]_\rho(s')\} \\
[\![\mu Z.\varphi]\!]_\rho & := & \bigsqcap\{f \mid [\![\varphi]\!]_{\rho[Z\mapsto f]} \sqsubseteq f\} \\
[\![\nu Z.\varphi]\!]_\rho & := & \bigsqcup\{f \mid f \sqsubseteq [\![\varphi]\!]_{\rho[Z\mapsto f]}\}
\end{array}
$$

Multi-valued Modal μ-Calculus

Definition (mv-$\mathfrak{L}_\mu$—Semantics)

$$
\begin{array}{rcl}
[\![true]\!]_\rho & := & \lambda s.\top \\
[\![false]\!]_\rho & := & \lambda s.\bot \\
[\![q]\!]_\rho & := & \lambda s.L(s)(q) \\
[\![\neg q]\!]_\rho & := & \lambda s.\neg L(s)(q) \\
[\![Z]\!]_\rho & := & \rho(Z) \\
[\![\varphi \vee \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcup [\![\psi]\!]_\rho \\
[\![\varphi \wedge \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcap [\![\psi]\!]_\rho \\
[\![\Diamond \varphi]\!]_\rho & := & \lambda s. \bigsqcup \{ \mathcal{R}(s, s') \sqcap [\![\varphi]\!]_\rho (s') \} \\
[\![\Box \varphi]\!]_\rho & := & \lambda s. \bigsqcap \{ \neg \mathcal{R}(s, s') \sqcup [\![\varphi]\!]_\rho (s') \} \\
[\![\mu Z.\varphi]\!]_\rho & := & \bigsqcap \{ f \mid [\![\varphi]\!]_{\rho[Z \mapsto f]} \sqsubseteq f \} \\
[\![\nu Z.\varphi]\!]_\rho & := & \bigsqcup \{ f \mid f \sqsubseteq [\![\varphi]\!]_{\rho[Z \mapsto f]} \}
\end{array}
$$

Multi-valued Modal μ-Calculus

Definition (mv-$\mathfrak{L}_\mu$—Semantics)

$$
\begin{array}{rcl}
[\![true]\!]_\rho & := & \lambda s.\top \\
[\![false]\!]_\rho & := & \lambda s.\bot \\
[\![q]\!]_\rho & := & \lambda s.L(s)(q) \\
[\![\neg q]\!]_\rho & := & \lambda s.\neg L(s)(q) \\
[\![Z]\!]_\rho & := & \rho(Z) \\
[\![\varphi \vee \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcup [\![\psi]\!]_\rho \\
[\![\varphi \wedge \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcap [\![\psi]\!]_\rho \\
[\![\Diamond\varphi]\!]_\rho & := & \lambda s.\bigsqcup\{\mathcal{R}(s,s') \sqcap [\![\varphi]\!]_\rho(s')\} \\
[\![\Box\varphi]\!]_\rho & := & \lambda s.\bigsqcap\{\neg\mathcal{R}(s,s') \sqcup [\![\varphi]\!]_\rho(s')\} \\
[\![\mu Z.\varphi]\!]_\rho & := & \bigsqcap\{f \mid [\![\varphi]\!]_{\rho[Z\mapsto f]} \sqsubseteq f\} \\
[\![\nu Z.\varphi]\!]_\rho & := & \bigsqcup\{f \mid f \sqsubseteq [\![\varphi]\!]_{\rho[Z\mapsto f]}\}
\end{array}
$$

Multi-valued Modal μ-Calculus

Definition (mv-$\mathfrak{L}_\mu$—Semantics)

$$
\begin{array}{rcl}
[\![true]\!]_\rho & := & \lambda s.\top \\
[\![false]\!]_\rho & := & \lambda s.\bot \\
[\![q]\!]_\rho & := & \lambda s.L(s)(q) \\
[\![\neg q]\!]_\rho & := & \lambda s.\neg L(s)(q) \\
[\![Z]\!]_\rho & := & \rho(Z) \\
[\![\varphi \vee \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcup [\![\psi]\!]_\rho \\
[\![\varphi \wedge \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcap [\![\psi]\!]_\rho \\
[\![\Diamond\varphi]\!]_\rho & := & \lambda s.\bigsqcup\{\mathcal{R}(s,s') \sqcap [\![\varphi]\!]_\rho(s')\} \\
[\![\Box\varphi]\!]_\rho & := & \lambda s.\bigsqcap\{\neg\mathcal{R}(s,s') \sqcup [\![\varphi]\!]_\rho(s')\} \\
[\![\mu Z.\varphi]\!]_\rho & := & \bigsqcap\{f \mid [\![\varphi]\!]_{\rho[Z\mapsto f]} \sqsubseteq f\} \\
[\![\nu Z.\varphi]\!]_\rho & := & \bigsqcup\{f \mid f \sqsubseteq [\![\varphi]\!]_{\rho[Z\mapsto f]}\}
\end{array}
$$

Multi-valued Modal μ-Calculus

Definition (mv-$\mathfrak{L}_\mu$—Semantics)

$$
\begin{array}{rcl}
[\![true]\!]_\rho & := & \lambda s.\top \\
[\![false]\!]_\rho & := & \lambda s.\bot \\
[\![q]\!]_\rho & := & \lambda s.L(s)(q) \\
[\![\neg q]\!]_\rho & := & \lambda s.\neg L(s)(q) \\
[\![Z]\!]_\rho & := & \rho(Z) \\
[\![\varphi \vee \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcup [\![\psi]\!]_\rho \\
[\![\varphi \wedge \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcap [\![\psi]\!]_\rho \\
[\![\Diamond\varphi]\!]_\rho & := & \lambda s.\bigsqcup\{\mathcal{R}(s,s') \sqcap [\![\varphi]\!]_\rho(s')\} \\
[\![\Box\varphi]\!]_\rho & := & \lambda s.\bigsqcap\{\neg\mathcal{R}(s,s') \sqcup [\![\varphi]\!]_\rho(s')\} \\
[\![\mu Z.\varphi]\!]_\rho & := & \bigsqcap\{f \mid [\![\varphi]\!]_{\rho[Z\mapsto f]} \sqsubseteq f\} \\
[\![\nu Z.\varphi]\!]_\rho & := & \bigsqcup\{f \mid f \sqsubseteq [\![\varphi]\!]_{\rho[Z\mapsto f]}\}
\end{array}
$$

Multi-valued Modal μ-Calculus

Definition (mv-$\mathfrak{L}_\mu$—Semantics)

$$
\begin{array}{rcl}
[\![true]\!]_\rho & := & \lambda s.\top \\
[\![false]\!]_\rho & := & \lambda s.\bot \\
[\![q]\!]_\rho & := & \lambda s.L(s)(q) \\
[\![\neg q]\!]_\rho & := & \lambda s.\neg L(s)(q) \\
[\![Z]\!]_\rho & := & \rho(Z) \\
[\![\varphi \vee \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcup [\![\psi]\!]_\rho \\
[\![\varphi \wedge \psi]\!]_\rho & := & [\![\varphi]\!]_\rho \sqcap [\![\psi]\!]_\rho \\
[\![\Diamond\varphi]\!]_\rho & := & \lambda s. \bigsqcup\{\mathcal{R}(s,s') \sqcap [\![\varphi]\!]_\rho(s')\} \\
[\![\Box\varphi]\!]_\rho & := & \lambda s. \bigsqcap\{\neg\mathcal{R}(s,s') \sqcup [\![\varphi]\!]_\rho(s')\} \\
[\![\mu Z.\varphi]\!]_\rho & := & \bigsqcap\{f \mid [\![\varphi]\!]_{\rho[Z\mapsto f]} \sqsubseteq f\} \\
[\![\nu Z.\varphi]\!]_\rho & := & \bigsqcup\{f \mid f \sqsubseteq [\![\varphi]\!]_{\rho[Z\mapsto f]}\}
\end{array}
$$

Multi-valued Modal μ-Calculus

Theorem (Computation of Fixpoints, Tarski'55)

For all MMKS $\mathcal{T}$ with state set $\mathcal{S}$ there is an $\alpha \in \mathbb{O}\mathrm{rd}$ s.t. for all $s \in \mathcal{S}$ we have: if $[\![\eta Z.\varphi]\!]_\rho(s) = x$ then $Z^\alpha(s) = x$.

Model Checking

Theorem (Correctness of Model Checking)

For all PL-CCS programs $Prog = (\mathcal{E}, P_1)$*, every configuration vector* ν*, and formulae* $\varphi \in$ *mv-*$\mathfrak{L}_\mu$*, we have*

$$[\![config(Prog, \nu)]\!]_{CCS} \models \varphi \text{ iff } \nu \in ([\![Prog]\!]_{CT} \models \varphi)(P_1)$$

Practical Model Checking?

Similar stories...

- On-the-fly: Adapt Shoham&Grumberg's game-based approach
- Symbolic MC: ...
- CTL: As restrictions of μ-calculus, Chechik et al.
- Automata-based for mv-LTL: Checkik et al.
- More specific integration of notion of features in on-the-fly mc: Legay et al.
- Bounded MC: ...
- Abstraction: *see next*

Presentation outline

Two-valued Abstraction

Idea

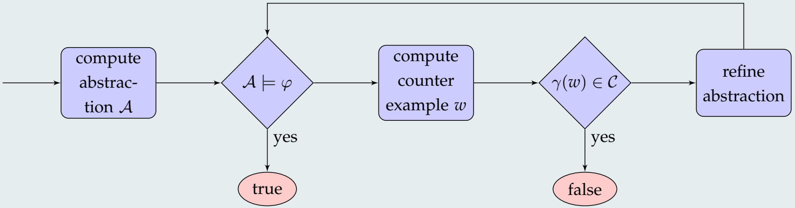
Check smaller over-approximation of the system

Two-valued Abstraction

Idea

Check smaller over-approximation of the system

CEGAR

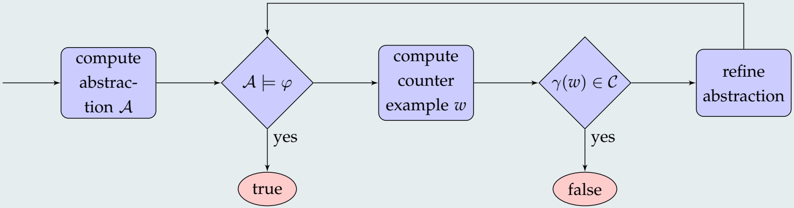


Two-valued Abstraction

Idea

Check smaller over-approximation of the system

CEGAR



[Clarke, Grumberg, Jha, Lu, Veith'03] [Lakhnech, Bensalem, Berezin, Owre:'01] [. . .]

Three-valued Abstraction

Idea

- Yields conservative results for both, TRUE and FALSE

Three-valued Abstraction

Idea

- Yields conservative results for both, TRUE and FALSE
- Requires third value: *Don't know*

Three-valued Abstraction

Idea

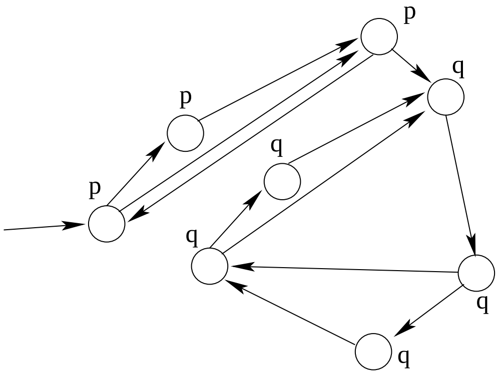
- Yields conservative results for both, TRUE and FALSE
- Requires third value: *Don't know*
- Check over-approximation and under-approximation of the system

Three-valued Abstraction

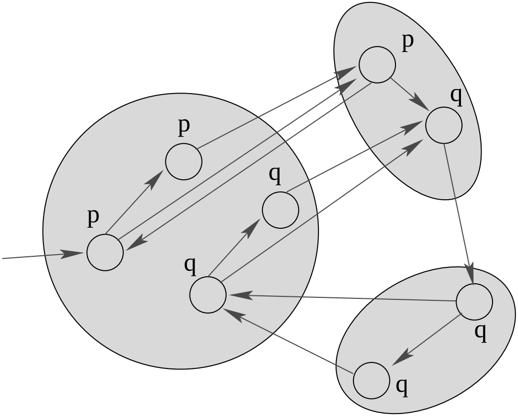
Idea

- Yields conservative results for both, TRUE and FALSE
- Requires third value: *Don't know*
- Check over-approximation and under-approximation of the system
- carried out for the μ-calculus in [Bruns, P. Godefroid'99]

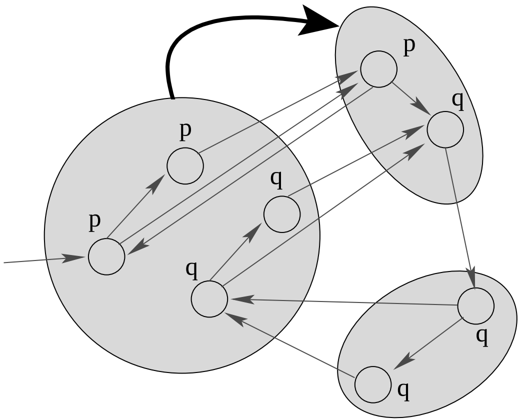
Three-valued Abstraction



Three-valued Abstraction

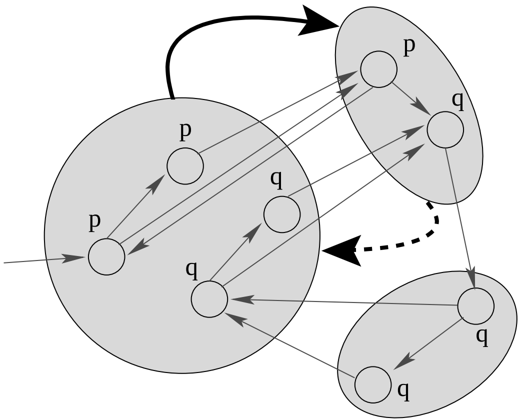


Three-valued Abstraction

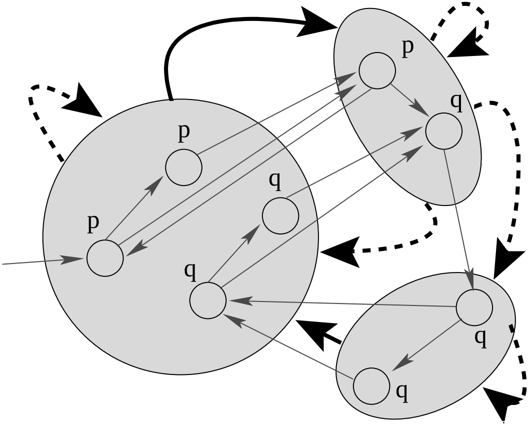


Three-valued Abstraction

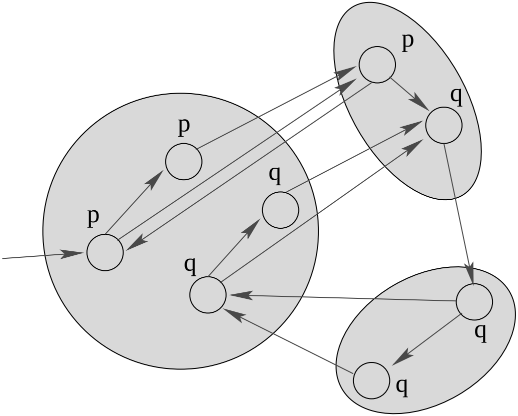
must/may transitions



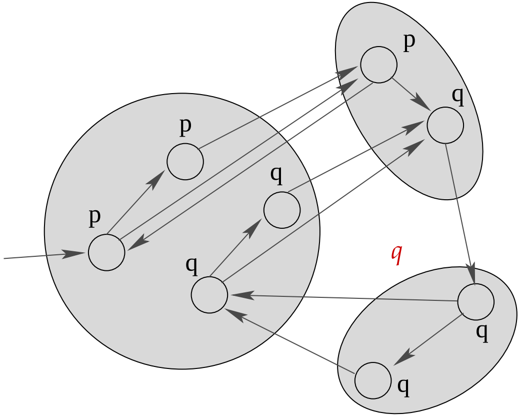
Three-valued Abstraction



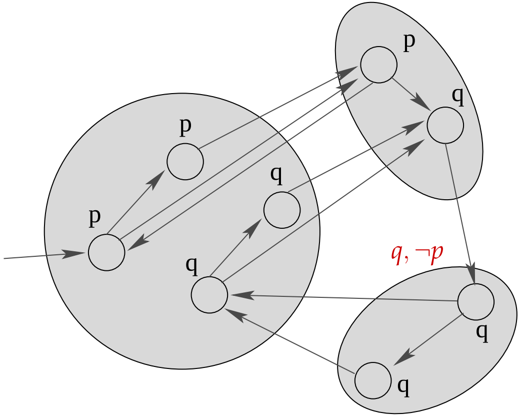
Three-valued Abstraction



Three-valued Abstraction

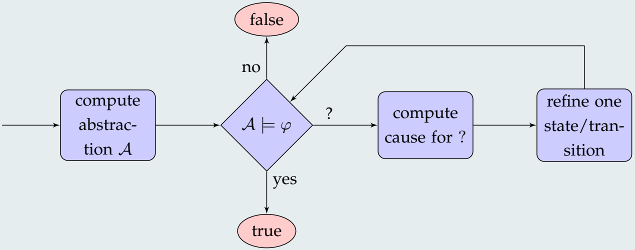


Three-valued Abstraction



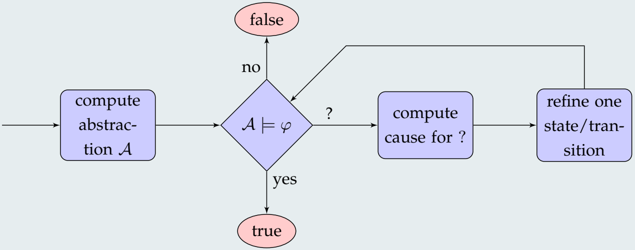
CBAR

CBAR—Cause-based Abstraction Refinement



CBAR

CBAR—Cause-based Abstraction Refinement

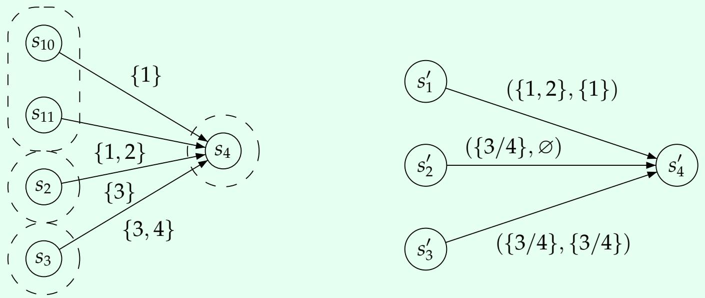


[Grumberg, Lange, L_, Shoham'07]

Presentation outline

Abstraction by joining states

Idea



The abstract lattice

Definition (*op-lattice*)

Let $\mathcal{L}$ be a de Morgan lattice. The lattice

$$\mathcal{L}_{op} = (\{(m_1, m_2) \in \mathcal{L} \times \mathcal{L} \mid m_1 \sqsupseteq m_2\}, \sqcap_{op}, \sqcup_{op}, \neg_{op})$$

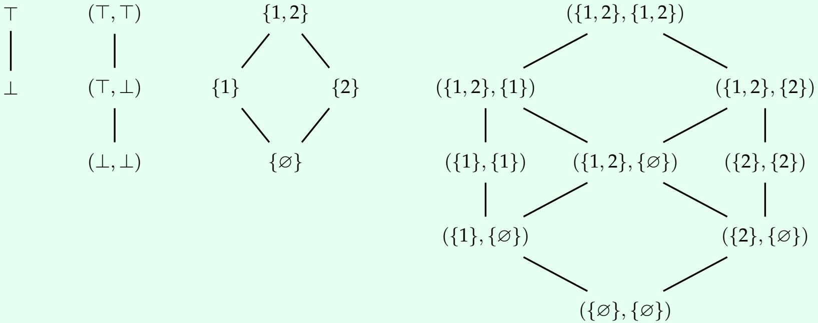
with the operations $\sqcap_{op}$, $\sqcup_{op}$, $\neg_{op}$ given by

$$\begin{aligned} (m_1, m_2) \sqcap_{op} (m_1', m_2') &:= (m_1 \sqcap m_1', m_2 \sqcap m_2') \\ (m_1, m_2) \sqcup_{op} (m_1', m_2') &:= (m_1 \sqcup m_1', m_2 \sqcup m_2') \\ \neg_{op}(m_1, m_2) &:= (\neg m_2, \neg m_1) \end{aligned}$$

is called the *optimistic-pessimistic lattice (op-lattice)* for $\mathcal{L}$.

Examples

Lattices and op-Lattices



Abstraction by Joining States

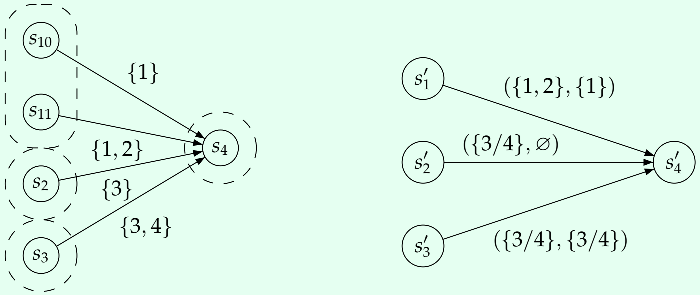
Definition (State Abstraction Operator)

We call the function abs_S [...] by joining states according to the abstraction complete function γ the *state abstraction operator*, where the set S_A of abstract states is implicitly given by γ, the lattice $\mathcal{L}_A$ is the op-lattice of $\mathcal{L}_C$ and

$$\begin{aligned}
\mathcal{R}_A(s_A, s'_A) &= \Bigg(\bigsqcup_{s_C \in \gamma(s_A)} \bigsqcup_{s'_C \in \gamma(s'_A)} \mathcal{R}_C(s_C, s'_C) \,, \\
&\qquad \bigsqcap_{s_C \in \gamma(s_A)} \bigsqcup_{s'_C \in \gamma(s'_A)} \mathcal{R}_C(s_C, s'_C) \Bigg) \\
L_A(s_A, p) &= \Bigg(\bigsqcup_{s_C \in \gamma(s_A)} L_C(s_C, p) \,, \bigsqcap_{s_C \in \gamma(s_A)} L(s_C, p) \Bigg)
\end{aligned}$$

Abstraction by joining lattice elements

Idea



Abstraction of lattices

Definition (Galois Connection)

Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be lattices. A pair $(\uparrow, \downarrow)$ of monotone functions $\uparrow : \mathcal{L}_1 \rightarrow \mathcal{L}_2$ and $\downarrow : \mathcal{L}_2 \rightarrow \mathcal{L}_1$ is a *Galois connection* from $\mathcal{L}_1$ to $\mathcal{L}_2$, if

$$\forall l \in \mathcal{L}_1 : l \sqsubseteq \downarrow(\uparrow(l))$$

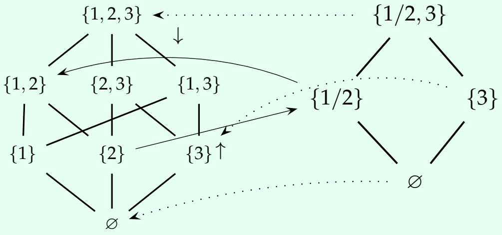
and

$$\forall a \in \mathcal{L}_2 : \uparrow(\downarrow(a)) \sqsubseteq a$$

.

Abstraction by joining lattice elements

Galois connection



Abstraction by joining lattice elements

Definition (aop-lattice)

Let $\mathcal{L}_C$, $\mathcal{L}_{\mathsf{o}}$, and $\mathcal{L}_{\mathsf{p}}$ be de Morgan lattices.
Let $\uparrow_{\mathsf{o}} : \mathcal{L}_C \to \mathcal{L}_{\mathsf{o}}$ and $\downarrow_{\mathsf{o}} : \mathcal{L}_{\mathsf{o}} \to \mathcal{L}_C$ and
$\uparrow_{\mathsf{p}} : \mathcal{L}_{\mathsf{p}} \to \mathcal{L}_C$ and $\downarrow_{\mathsf{p}} : \mathcal{L}_C \to \mathcal{L}_{\mathsf{p}}$ be Galois connections.
We call the lattice

$$\mathcal{L}_{aop} = \left(\{(m_{\mathsf{o}}, m_{\mathsf{p}}) \in \mathcal{L}_{\mathsf{o}} \times \mathcal{L}_{\mathsf{p}} \mid \downarrow_{\mathsf{o}}(m_{\mathsf{o}}) \sqsupseteq \uparrow_{\mathsf{p}}(m_{\mathsf{p}})\}, \sqcap_{aop}, \sqcup_{aop}, \neg_{aop}\right)$$

with the operations given by

$$\begin{aligned}(m_{\mathsf{o}}, m_{\mathsf{p}}) \sqcap_{aop} (m'_{\mathsf{o}}, m'_{\mathsf{p}}) &:= (m_{\mathsf{o}} \sqcap m'_{\mathsf{o}}\ ,\ m_{\mathsf{p}} \sqcap m'_{\mathsf{p}})\\ (m_{\mathsf{o}}, m_{\mathsf{p}}) \sqcup_{aop} (m'_{\mathsf{o}}, m'_{\mathsf{p}}) &:= (m_{\mathsf{o}} \sqcup m'_{\mathsf{o}}\ ,\ m_{\mathsf{p}} \sqcup m'_{\mathsf{p}})\\ \neg_{aop}(m_{\mathsf{o}}, m_{\mathsf{p}}) &:= (\neg_{\mathsf{p}} m_{\mathsf{p}}\ ,\ \neg_{\mathsf{o}} m_{\mathsf{o}})\end{aligned}$$

the *abstract optimistic-pessimistic lattice (aop-lattice)* for the lattice $\mathcal{L}_C$.

Abstraction by joining lattice elements

Definition (aop-lattice)

Let $\mathcal{L}_C$, $\mathcal{L}_\mathsf{o}$, and $\mathcal{L}_\mathsf{p}$ be de Morgan lattices.
Let $\uparrow_\mathsf{o} : \mathcal{L}_C \to \mathcal{L}_\mathsf{o}$ and $\downarrow_\mathsf{o} : \mathcal{L}_\mathsf{o} \to \mathcal{L}_C$ and
$\uparrow_\mathsf{p} : \mathcal{L}_\mathsf{p} \to \mathcal{L}_C$ and $\downarrow_\mathsf{p} : \mathcal{L}_C \to \mathcal{L}_\mathsf{p}$ be Galois connections.
We call the lattice

$$\mathcal{L}_{aop} = \left(\{(m_\mathsf{o}, m_\mathsf{p}) \in \mathcal{L}_\mathsf{o} \times \mathcal{L}_\mathsf{p} \mid \downarrow_\mathsf{o}(m_\mathsf{o}) \sqsupseteq \uparrow_\mathsf{p}(m_\mathsf{p})\}, \sqcap_{aop}, \sqcup_{aop}, \neg_{aop}\right)$$

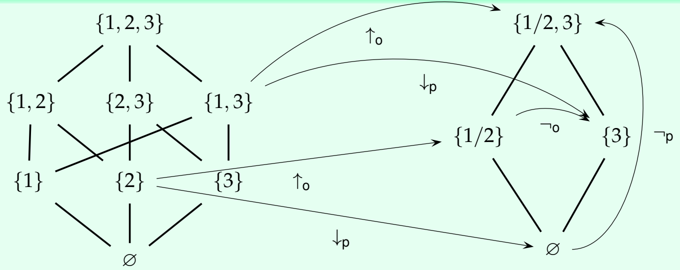
with the operations given by

$$\begin{aligned} (m_\mathsf{o}, m_\mathsf{p}) \sqcap_{aop} (m'_\mathsf{o}, m'_\mathsf{p}) &:= (m_\mathsf{o} \sqcap m'_\mathsf{o} \,,\, m_\mathsf{p} \sqcap m'_\mathsf{p}) \\ (m_\mathsf{o}, m_\mathsf{p}) \sqcup_{aop} (m'_\mathsf{o}, m'_\mathsf{p}) &:= (m_\mathsf{o} \sqcup m'_\mathsf{o} \,,\, m_\mathsf{p} \sqcup m'_\mathsf{p}) \\ \neg_{aop}(m_\mathsf{o}, m_\mathsf{p}) &:= (\neg_\mathsf{p} m_\mathsf{p} \,,\, \neg_\mathsf{o} m_\mathsf{o}) \end{aligned}$$

the *abstract optimistic-pessimistic lattice (aop-lattice)* for the lattice $\mathcal{L}_C$.
Furthermore, let $\mathcal{L}_\mathsf{o}$ and $\mathcal{L}_\mathsf{p}$ be connected by two anti-monotone negation functions $\neg_\mathsf{o} : \mathcal{L}_\mathsf{o} \to \mathcal{L}_\mathsf{p}$ and $\neg_\mathsf{p} : \mathcal{L}_\mathsf{p} \to \mathcal{L}_\mathsf{o}$ with $\neg_\mathsf{o}\uparrow_\mathsf{o}(x) \sqsubseteq \downarrow_\mathsf{p}(\neg x)$ and $\uparrow_\mathsf{o}(\neg x) \sqsubseteq \neg_\mathsf{p}\downarrow_\mathsf{p}(x)$.

Abstraction by joining lattice elements

aop-lattice



Abstraction by joining lattice elements

Definition (Lattice Abstraction Operator)

Let $(S_A, \mathcal{L}_A, \mathcal{R}_A, L_A)$ be a mv-KS, and $\uparrow_o, \downarrow_p$ be two Galois connections with corresponding negation functions $\neg_o, \neg_p$. Then, the *lattice abstraction operator* abs_L yields an abstracted mv-KS

$$abs_L\left((S_A, \mathcal{L}_A, \mathcal{R}_A, L_A), \uparrow_o, \downarrow_p, \neg_o, \neg_p\right) = (S'_A, \mathcal{L}'_A, \mathcal{R}'_A, L'_A)$$

labeled with an aop-lattice $\mathcal{L}'_A$, where $S'_A = S_A$ and

$$\begin{aligned} \mathcal{R}'_A(s,s') &= \left(\uparrow_o\left((\mathcal{R}_A(s,s'))_1\right), \downarrow_p\left((\mathcal{R}_A(s,s'))_2\right)\right) \\ L'_A(s,p) &= \left(\uparrow_o\left((L_A(s,p))_1\right), \downarrow_p\left((L_A(s,p))_2\right)\right) \end{aligned}$$

Conservative Abstraction

Theorem (Correctness of abstraction)

[. . .]

$$\uparrow_{\mathsf{p}}(m_p) \sqsubseteq [\![\varphi]\!]_{\varnothing}^{\mathcal{K}_C}(s_C) \sqsubseteq \downarrow_{\mathsf{o}}(m_o)$$

where $(m_o, m_p) = [\![\varphi]\!]_{\varnothing}^{\mathcal{K}_A}(s_A)$ *is the result of the evaluation of* φ *on* $\mathcal{K}_A$.

Presentation outline

Towards Refinement

Question?

Why do optimistic and pessimistic assessment differ?

Relevant cases

(i) the evaluation of the labeling function L for some atomic proposition p and state s

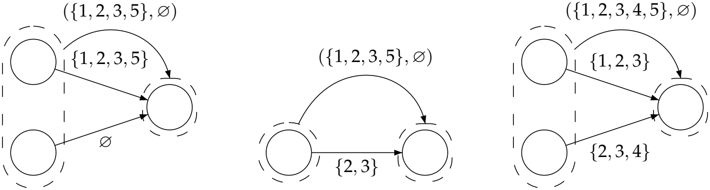
(ii) the evaluation of the transition relation function $\mathcal{R}$ for two states s and s',

(iii) the computation of negation, or

(iv) the computation of meet and join.

$$\Phi := \neg\Phi \mid \Phi \sqcap \Phi \mid \Phi \sqcup \Phi \mid \bigsqcap_{s_i} \Phi \mid \bigsqcup_{s_i} \Phi \mid L(s_i, p) \mid \mathcal{R}(s_i, s_j)$$

Sources of Imprecision



Atomic propositions

$$causes(p(s), m_{\mathsf{o}}, m_{\mathsf{p}}, \xi_{\mathsf{o}}, \xi_{\mathsf{p}}, \zeta) = \{(s, p, (\downarrow_{\mathsf{o}}(m_{\mathsf{o}}), \uparrow_{\mathsf{p}}(m_{\mathsf{p}})))\}$$

p evaluates to $\{1, 2, 3, 4, 5\}$ in the optimistic and to $\{2, 3, 4, 5\}$ in the pessimistic account:
the cause is $(s, p, (\{1, 2, 3, 4, 5\}, \{2, 3, 4, 5\}))$.

Meet

- Imprecision due to lattice abstraction
- Precision due to meet: $(\top, \bot) \sqcap (\bot, \bot) = (\bot, \bot)$

$$\begin{aligned} &\mathit{causes}((\varphi_1 \sqcap \varphi_2)(s), m_o, m_p, \xi_o, \xi_p, \zeta) = \\ &\qquad \{(\downarrow_o(\xi_o(\varphi_1(s))) \sqcap \downarrow_o(\xi_o(\varphi_2(s))), \uparrow_p(m_p))\} \text{ if components differ} \\ &\quad \cup \bigcup_{c \in \zeta(\varphi_1(s)) \cup \zeta(\varphi_(s))} \mathit{fil}(m_o, m_p, c) \end{aligned}$$

$$\mathit{fil}(m_o, m_p, (k, (l_o, l_p))) = (k, l_o \sqcap \downarrow_o(m_o), (l_p \sqcup \uparrow_p(m_p)) \sqcap (l_o \sqcap \downarrow_o(m_o)))$$

Presentation outline

Conclusions

We have shown

- product famililiy verification is multi-valued model-checking

Conclusions

We have shown

- product famililly verification is multi-valued model-checking
- abstractions for multi-valued systems

Conclusions

We have shown

- product famililly verification is multi-valued model-checking
- abstractions for multi-valued systems
- by joining states

Conclusions

We have shown

- product famililly verification is multi-valued model-checking
- abstractions for multi-valued systems
- by joining states
- by abstraction of truth values

Conclusions

We have shown

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

Conclusions

We have shown

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

Conclusions

We have shown

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

Conclusions

We have shown

- product famililly verification is multi-valued model-checking
- 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

- abstractions for compact representations

Conclusions

We have shown

- product famililly verification is multi-valued model-checking
- 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

- abstractions for compact representations
- implementation?

Conclusions

We have shown

- product famililly verification is multi-valued model-checking
- 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

- abstractions for compact representations
- implementation?
- . . .

Conclusions

We have shown

- product famililyverification is multi-valued model-checking
- 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

- abstractions for compact representations
- implementation?
- ...
- feature-based verification – Is it compositional (multi-valued) model checking?