OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

# Potential Synergies of Theorem Proving and Model Checking for Software Product Lines

Thomas Thüm[1], Jens Meinicke[1], Fabian Benduhn[1], Martin Hentschel[2], Alexander von Rhein[3], Gunter Saake[1]

May 7th, 2014

[1] University of Magdeburg, Germany
[2] University of Darmstadt, Germany
[3] University of Passau, Germany

# Potential Synergies of Theorem Proving and Model Checking for Software Product Lines

Thomas Thüm[1], Jens Meinicke[1], Fabian Benduhn[1],
Martin Hentschel[2], Alexander von Rhein[3], Gunter Saake[1]

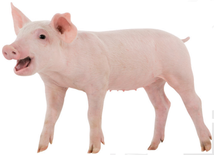
May 7th, 2014

[1] University of Magdeburg, Germany
[2] University of Darmstadt, Germany
[3] University of Passau, Germany

# Variability in Single-System Engineering

1. Strategy: clone-and-own, copy-and-modify, branching, ...

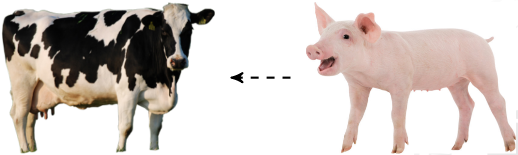
**David W. Stefan Tassio**



# Variability in Single-System Engineering

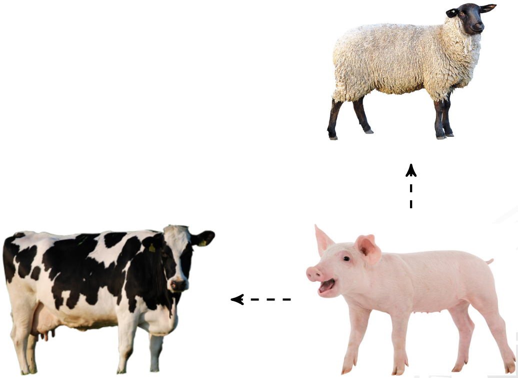
1. Strategy: clone-and-own, copy-and-modify, branching, ...





# Variability in Single-System Engineering

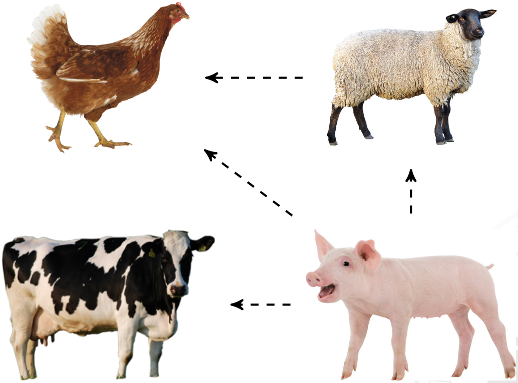
1. Strategy: clone-and-own, copy-and-modify, branching, ...



David W. Stefan Tassio

# Variability in Single-System Engineering

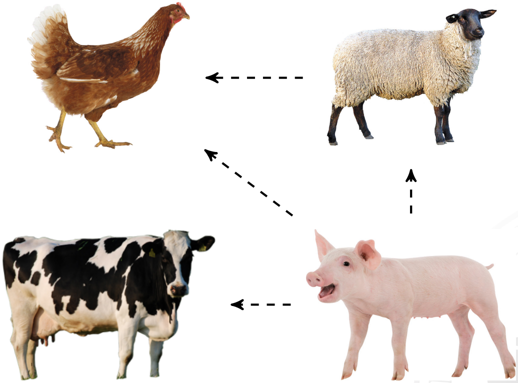
1. Strategy: clone-and-own, copy-and-modify, branching, ...



David W. Stefan Tassio

# Variability in Single-System Engineering

1. Strategy: clone-and-own, copy-and-modify, branching, ...



Problems: creation, bug fixes, extension, ... [code-clones problems]

# Variability in Single-System Engineering

2. Strategy: runtime variability/parameters, all-in-one-solution, swiss army knife (German: Eierlegende Wollmilchsau), . . .

Max

# Variability in Single-System Engineering

2. Strategy: runtime variability/parameters, all-in-one-solution, swiss army knife (German: Eierlegende Wollmilchsau), ...

Problems: footprint, performance, safety, security, ... [unused functionality]

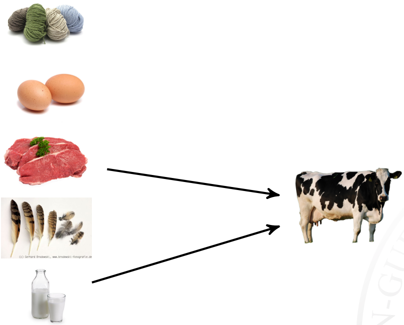
Max

# Variability in Product-Line Engineering

Compile-time variability: components, plug-ins, feature modules, aspects, build scripts, preprocessors, virtual separation, . . .

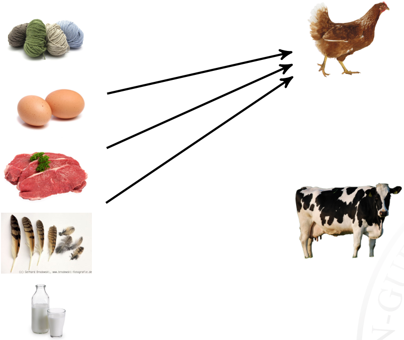
# Variability in Product-Line Engineering

Compile-time variability: components, plug-ins, feature modules, aspects, build scripts, preprocessors, virtual separation, . . .



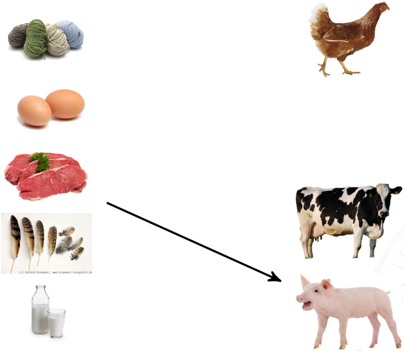
# Variability in Product-Line Engineering

Compile-time variability: components, plug-ins, feature modules, aspects, build scripts, preprocessors, virtual separation, . . .



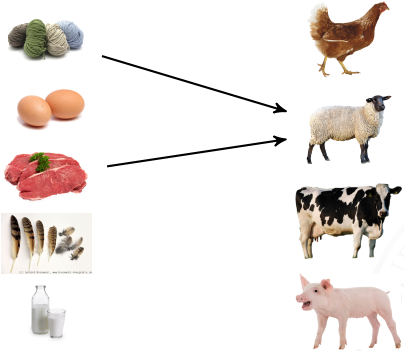
# Variability in Product-Line Engineering

Compile-time variability: components, plug-ins, feature modules, aspects, build scripts, preprocessors, virtual separation, ...



# Variability in Product-Line Engineering

Compile-time variability: components, plug-ins, feature modules, aspects, build scripts, preprocessors, virtual separation, ...



# Variability in Product-Line Engineering

Compile-time variability: components, plug-ins, feature modules, aspects, build scripts, preprocessors, virtual separation, ...



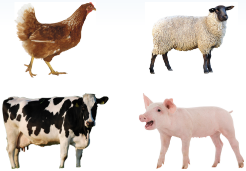




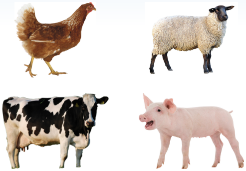


Challenges: testing, verification, specification, ...

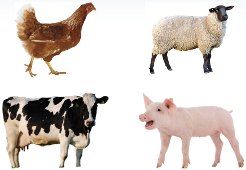
# Transition between Variability Representations



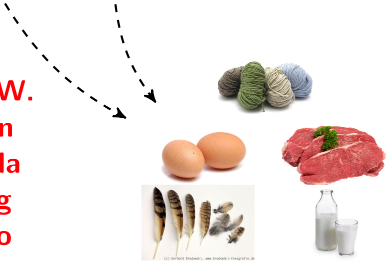
# Transition between Variability Representations



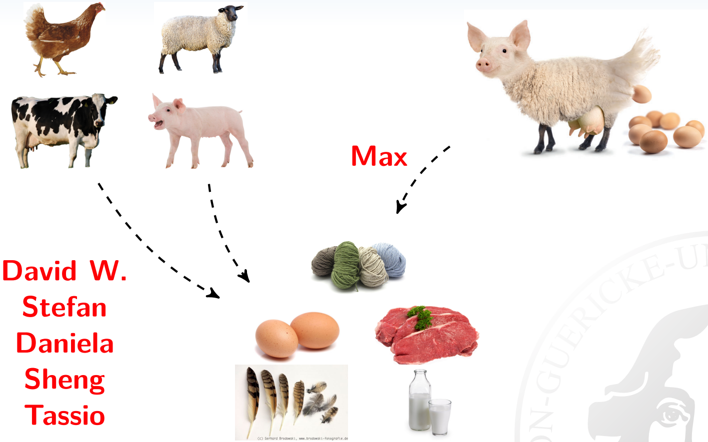
# Transition between Variability Representations



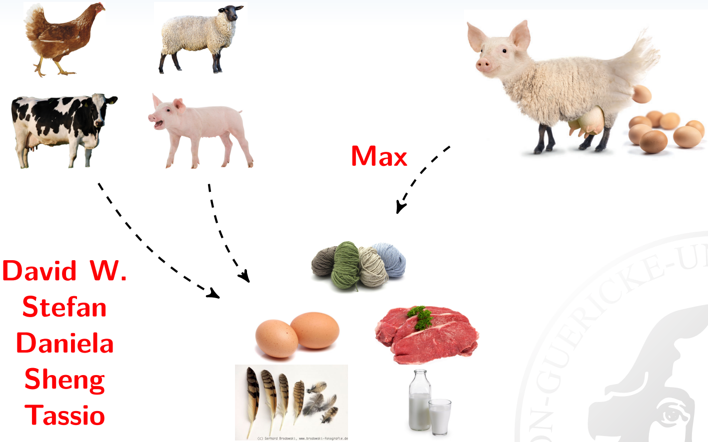
**David W.**
**Stefan**
**Daniela**
**Sheng**
**Tassio**



# Transition between Variability Representations

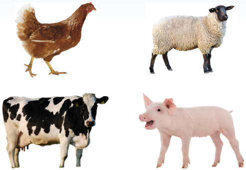


# Transition between Variability Representations




High manual effort

# Transition between Variability Representations

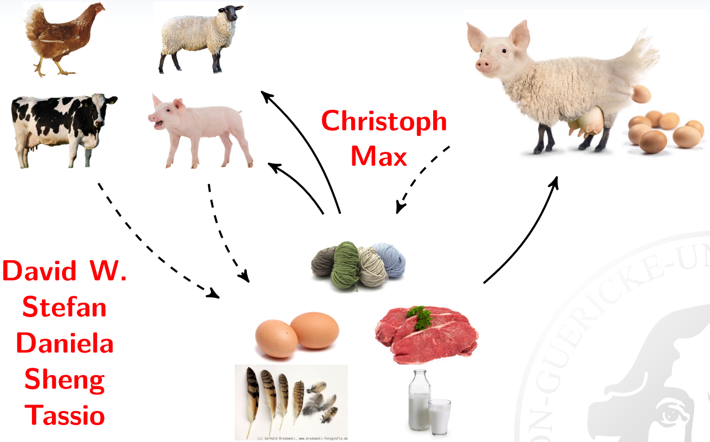


Christoph
Max

David W.
Stefan
Daniela
Sheng
Tassio

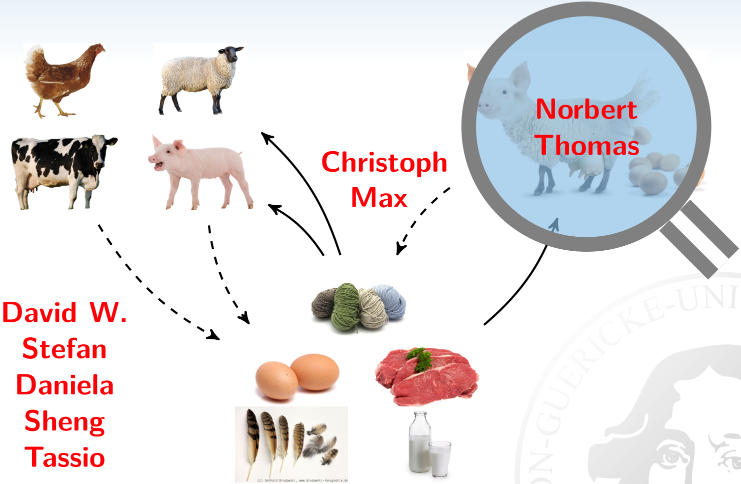
High manual effort vs. automatic generation

# Transition between Variability Representations



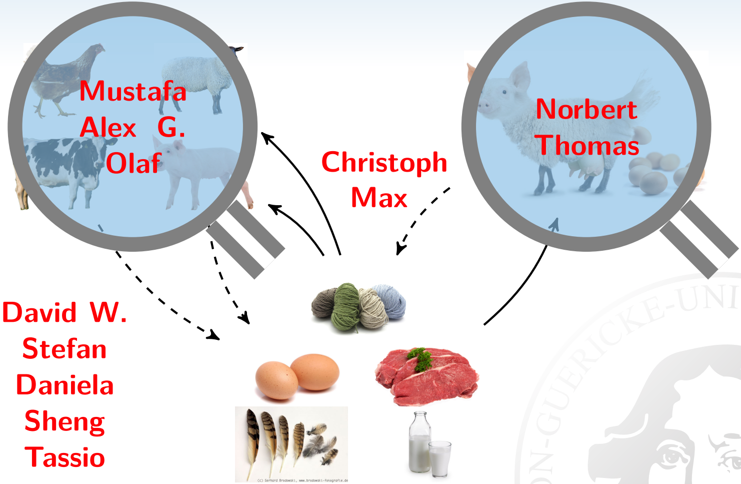

High manual effort vs. automatic generation

# Transition between Variability Representations



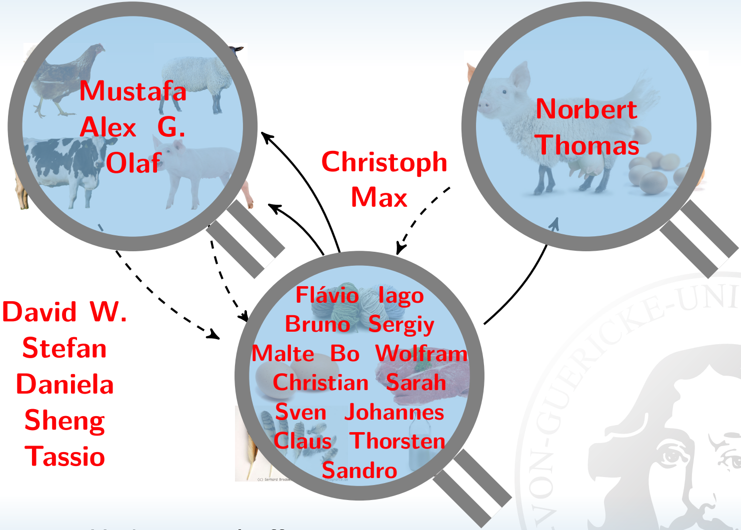

High manual effort vs. automatic generation

# Transition between Variability Representations



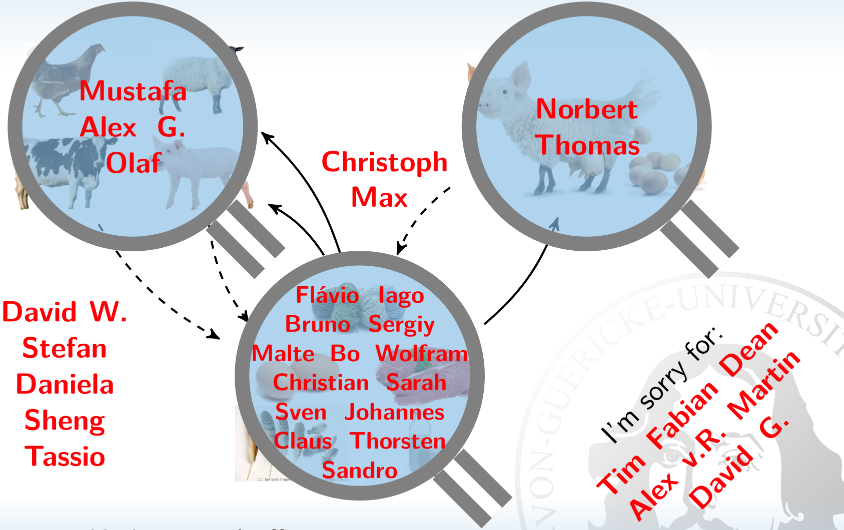

High manual effort vs. automatic generation

# Transition between Variability Representations




High manual effort vs. automatic generation

# Transition between Variability Representations




High manual effort vs. automatic generation

# Variability Encoding

Translating compile-time into run-time/load-time variability for:

- Model checking — Post and Sinz [2008], Apel et al. [2011], Classen et al. [2011], Apel et al. [2013]
- Theorem proving — Thüm et al. [2012]
- Testing — Kästner et al. [2012]
- Predicting non-functional properties — Siegmund et al. [2013] **Norbert**
- . . .

# Variability Encoding

Translating compile-time into run-time/load-time variability for:

- Model checking — Post and Sinz [2008], Apel et al. [2011], Classen et al. [2011], Apel et al. [2013]
- Theorem proving — Thüm et al. [2012]
- Testing — Kästner et al. [2012]
- Predicting non-functional properties — Siegmund et al. [2013] **Norbert**
- . . .

We can reuse tools from single-system engineering!

# Theorem Proving vs. Model Checking

- Deductive reasoning
- Code translated into first-order logic
- Transformation of logic formulas
- Methods in isolation
- Applicable to incomplete code
- Theorem provers: KeY, Coq, ...

- Exhaustive search
- Specification translated into runtime assertions
- Code (symbolically) executed
- Test scenarios
- Applicable to incomplete specifications
- Model checkers: JPF, SPIN, ...

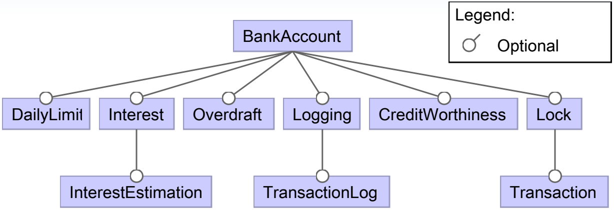
# Theorem Proving vs. Model Checking

- Deductive reasoning
- Code translated into first-order logic
- Transformation of logic formulas
- Methods in isolation
- Applicable to incomplete code
- Theorem provers: KeY, Coq, ...

- Exhaustive search
- Specification translated into runtime assertions
- Code (symbolically) executed
- Test scenarios
- Applicable to incomplete specifications
- Model checkers: JPF, SPIN, ...

What is more efficient/effective?

# Empirical Comparison




Logging ∧ Transaction ⇔ TransactionLog

- Feature modules with feature-oriented contracts
- Dependent variables: verification time, effectiveness
- Independent variables: number of features, number of defects

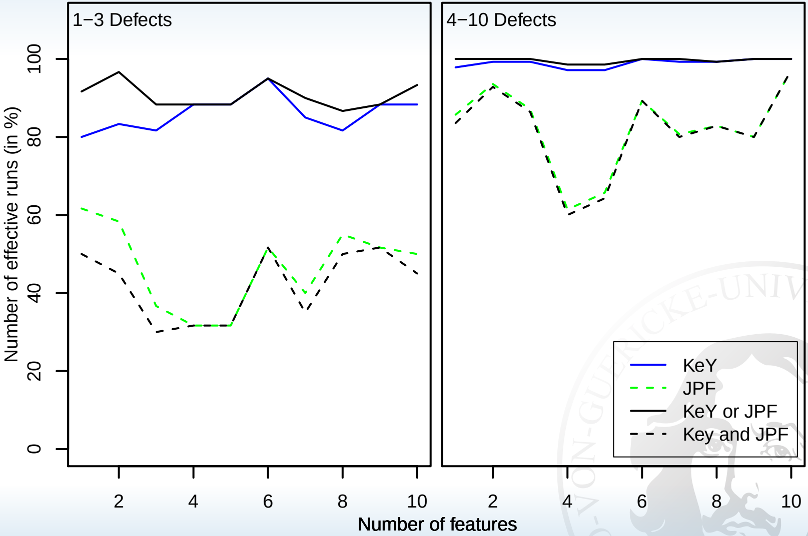
# Automatic Generation of Defects

Typical mutations from mutation testing — Jia and Harman [2011]

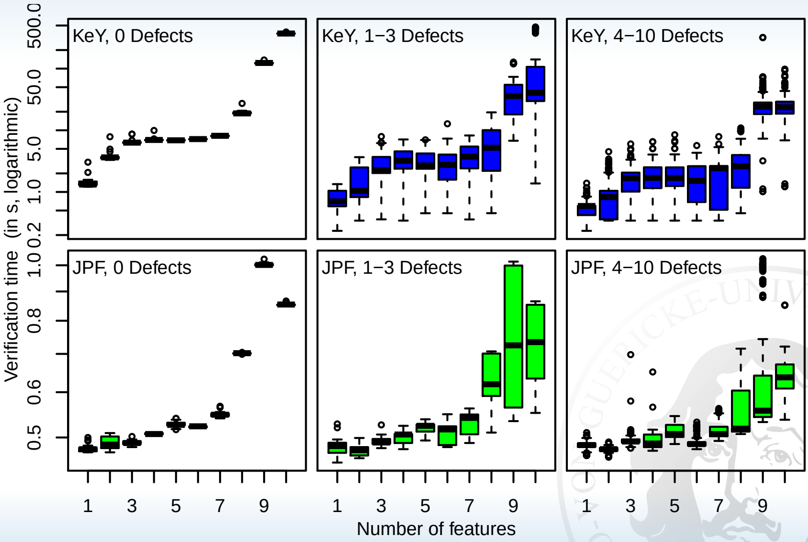
| Source/Target | Target/Source | In Java | In JML |
|---|---|---|---|
| `<` | `>` | 6 | 0 |
| `<=` | `>=` | 2 | 17 |
| `!=` | `==` | 0 | 39 |
| `&&` | `\|\|` | 0 | 11 |
| `==>` | `<==>` | 0 | 27 |
| `+` | `-` | 7 | 8 |
| `*` | `/` | 11 | 0 |
| `+=` | `-=` | 4 | 0 |
| `false` | `true` | 27 | 1 |

To simulate different stages during development

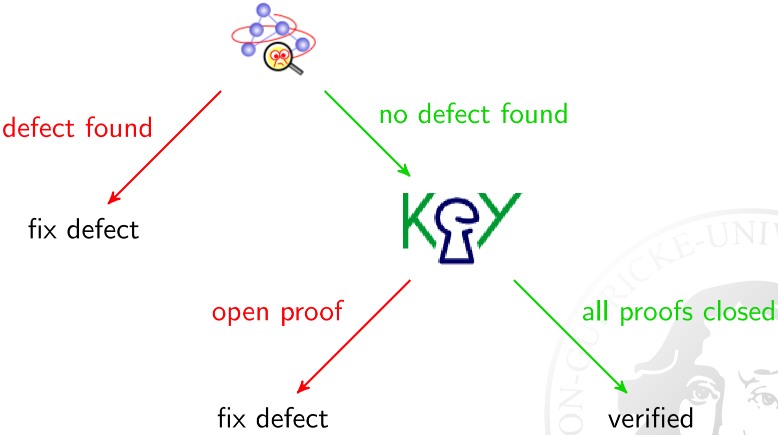
# Effectiveness of Theorem Proving and Model Checking



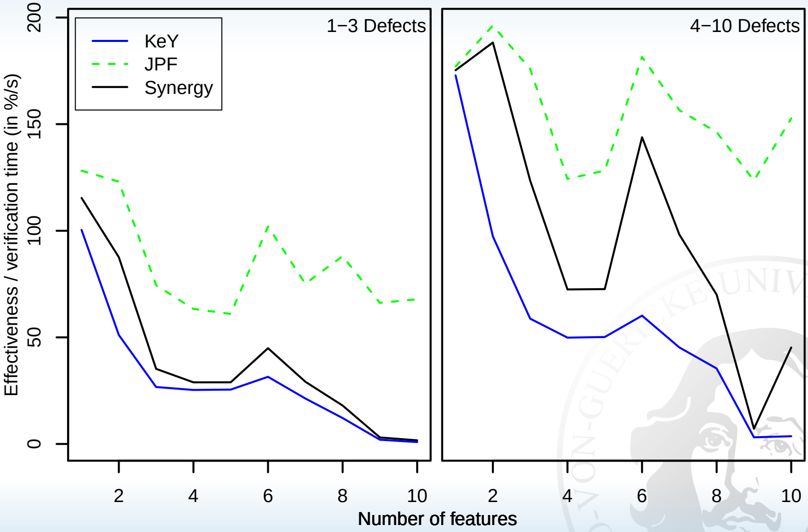
# Performance of Theorem Proving and Model Checking



# Combining Theorem Proving and Model Checking



# Efficiency of Theorem Proving and Model Checking



# Conclusion

- Theorem proving and model checking are more effective *and* efficient for many than for few defects
- Model checking is more efficient, but less effective
- Combination improves efficiency and effectiveness
- Combination especially more effective for few defects

# FOSD Meeting 2014

# References I


Sven Apel, Hendrik Speidel, Philipp Wendler, Alexander von Rhein, and Dirk Beyer. Detection of Feature Interactions using Feature-Aware Verification. In *Proc. Int'l Conf. Automated Software Engineering (ASE)*, pages 372–375, Washington, DC, USA, 2011. IEEE.

Sven Apel, Alexander von Rhein, Philipp Wendler, Armin Größlinger, and Dirk Beyer. Strategies for Product-Line Verification: Case Studies and Experiments. In *Proc. Int'l Conf. Software Engineering (ICSE)*, pages 482–491, Piscataway, NJ, USA, May 2013. IEEE.

Andreas Classen, Patrick Heymans, Pierre-Yves Schobbens, and Axel Legay. Symbolic Model Checking of Software Product Lines. In *Proc. Int'l Conf. Software Engineering (ICSE)*, pages 321–330, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0445-0. doi: http://doi.acm.org/10.1145/1985793.1985838.

Yue Jia and Mark Harman. An Analysis and Survey of the Development of Mutation Testing. *IEEE Trans. Software Engineering (TSE)*, 37(5):649–678, September 2011. ISSN 0098-5589. doi: 10.1109/TSE.2010.62.

# References II

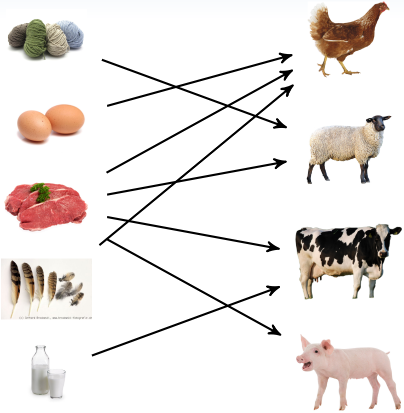
Christian Kästner, Alexander von Rhein, Sebastian Erdweg, Jonas Pusch, Sven Apel, Tillmann Rendel, and Klaus Ostermann. Toward Variability-Aware Testing. In *Proc. Int'l Workshop Feature-Oriented Software Development (FOSD)*, pages 1–8, New York, NY, USA, September 2012. ACM. ISBN 978-1-4503-1309-4. doi: 10.1145/2377816.2377817.

Hendrik Post and Carsten Sinz. Configuration Lifting: Software Verification meets Software Configuration. In *Proc. Int'l Conf. Automated Software Engineering (ASE)*, pages 347–350, Washington, DC, USA, 2008. IEEE.

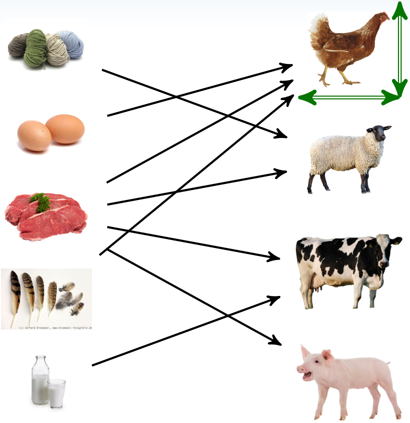
Norbert Siegmund, Alexander von Rhein, and Sven Apel. Family-based Performance Measurement. In *Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE)*, pages 95–104, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2373-4. doi: 10.1145/2517208.2517209.

Thomas Thüm, Ina Schaefer, Sven Apel, and Martin Hentschel. Family-Based Deductive Verification of Software Product Lines. In *Proc. Int'l Conf. Generative Programming and Component Engineering (GPCE)*, pages 11–20, New York, NY, USA, September 2012. ACM. ISBN 978-1-4503-1129-8. doi: 10.1145/2371401.2371404.

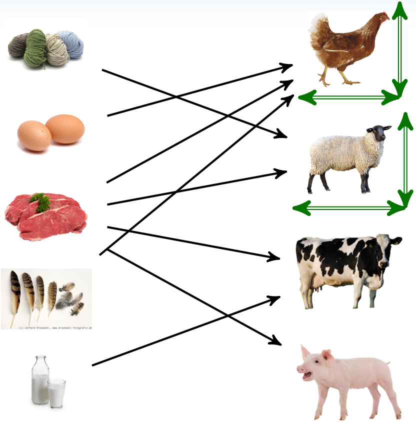
# Product-Based Specification



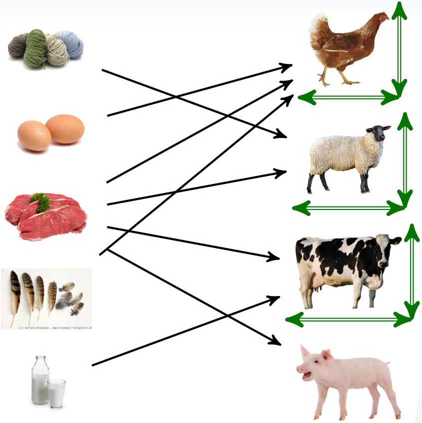
# Product-Based Specification



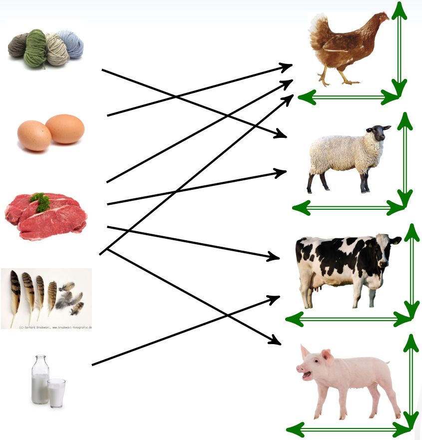
# Product-Based Specification



# Product-Based Specification

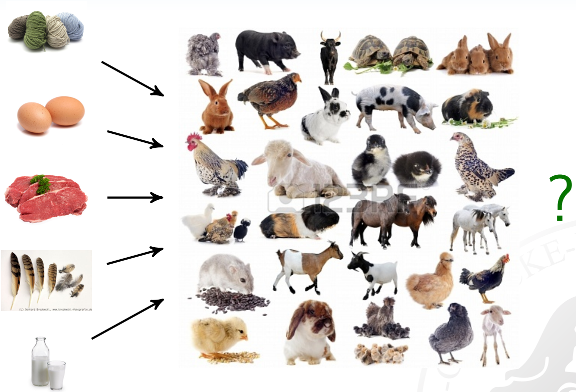


# Product-Based Specification



Problems: specification clones, scalability

# Product-Based Specification




Problems: specification clones, scalability

# Feature-Based Specification

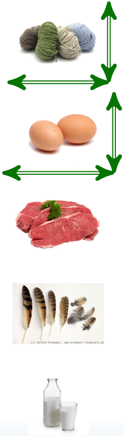
FASE'12, CSUR'14:

# Feature-Based Specification

FASE'12, CSUR'14:

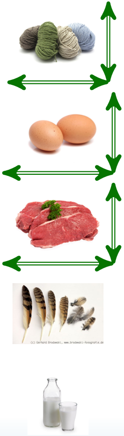
# Feature-Based Specification

FASE'12, CSUR'14:



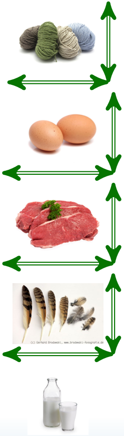
# Feature-Based Specification

FASE'12, CSUR'14:



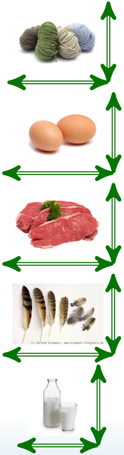
# Feature-Based Specification

FASE'12, CSUR'14:



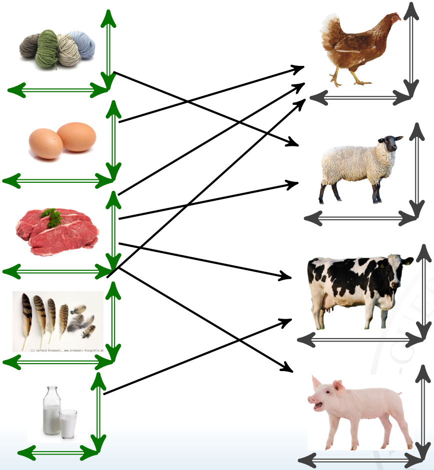
# Feature-Based Specification

FASE'12, CSUR'14:



# Feature-Based Specification

FASE'12, CSUR'14:



# Family-Based Specification

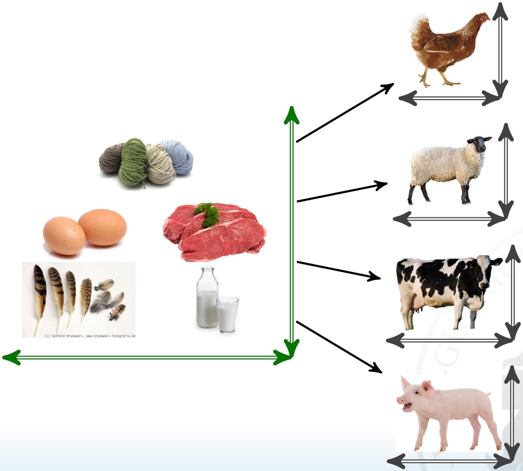
CSUR'14:

# Family-Based Specification

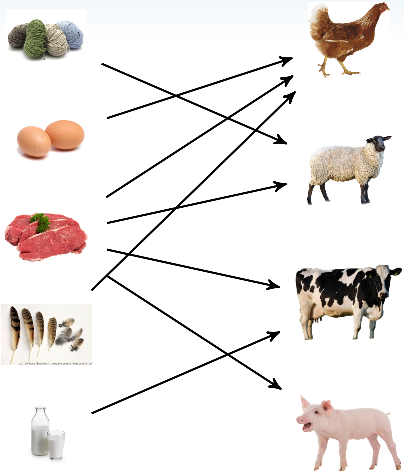
CSUR'14:

# Family-Based Specification

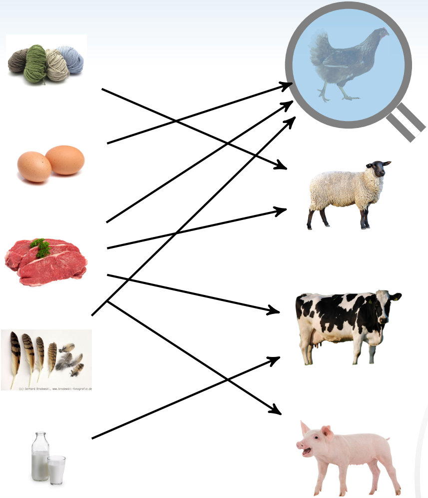
CSUR'14:



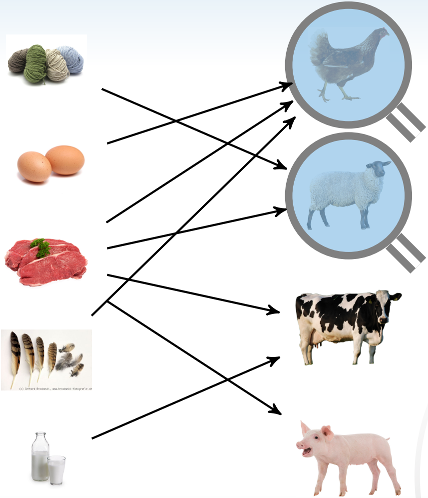
# Product-Based Analysis



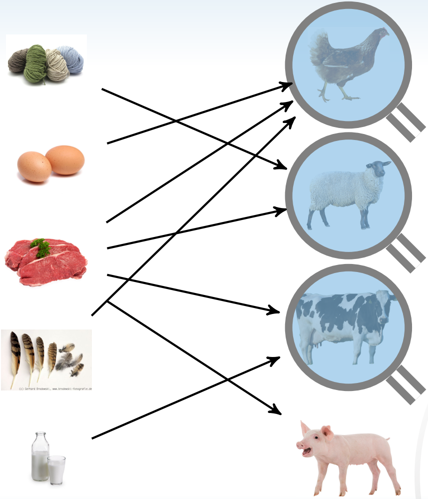
# Product-Based Analysis



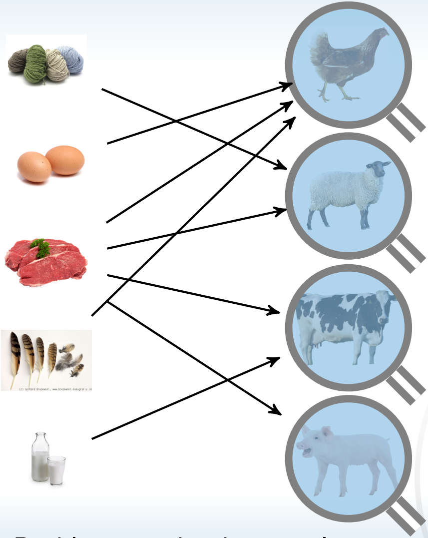
# Product-Based Analysis



# Product-Based Analysis



# Product-Based Analysis

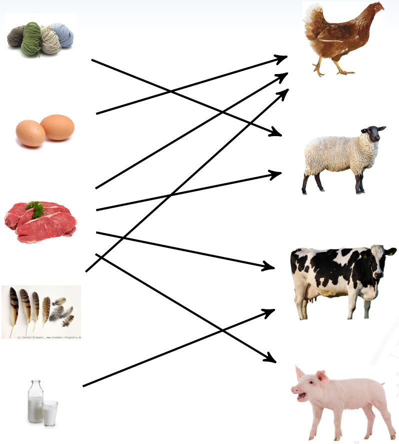


Problems: redundant analysis, scalability

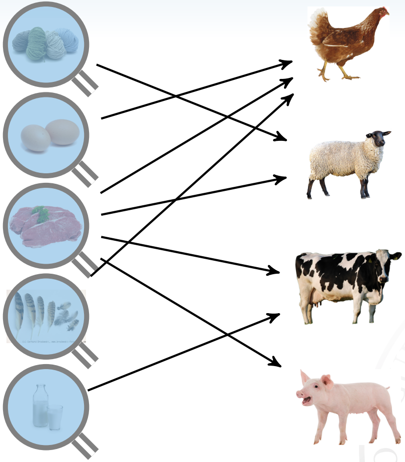
# Product-Based Analysis


Problems: redundant analysis, scalability

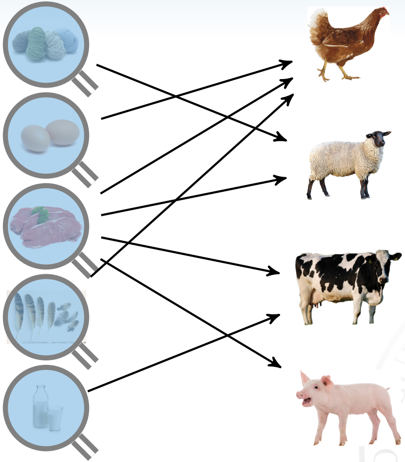
# Feature-Based Analysis



# Feature-Based Analysis



# Feature-Based Analysis

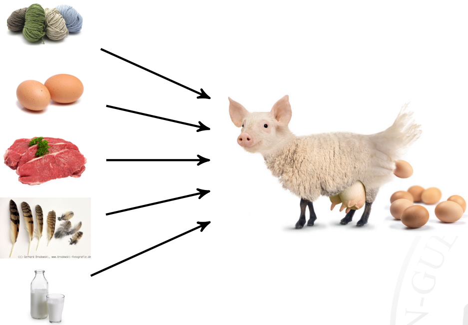


Limitation: only compositional properties

# Family-Based Analysis

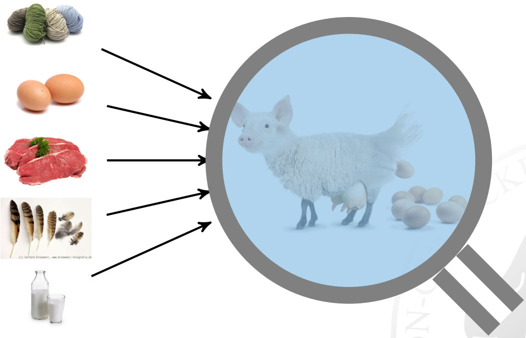
# Family-Based Analysis

Automatic (!) transition of compile-time into runtime variability only for analysis



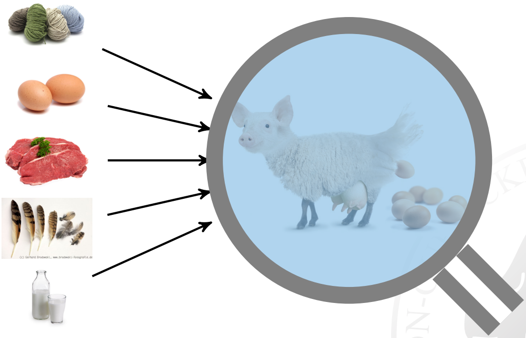
# Family-Based Analysis

Automatic (!) transition of compile-time into runtime variability
only for analysis



# Family-Based Analysis

Automatic (!) transition of compile-time into runtime variability only for analysis



Enables reuse of analysis tools from single-system engineering

# Implementation vs. Specification vs. Analysis

Possible combinations of the strategies:

| Impl. \ Spec. | Product-based | Family-based | Feature-based |
|---|---|---|---|
| Product-based | | | |
| Family-based | | | |
| Feature-based | | | |

# Implementation vs. Specification vs. Analysis

Possible combinations of the strategies:

| Impl. \ Spec. | Product-based | Family-based | Feature-based |
|---|---|---|---|
| Product-based | P | P | P |
| Family-based | P | P | P |
| Feature-based | P | P | P |

Legend: P/F/f - product/family/feature-based analysis

# Implementation vs. Specification vs. Analysis

Possible combinations of the strategies:

| Impl. \ Spec. | Product-based | Family-based | Feature-based |
|---|---|---|---|
| Product-based | P | P | P |
| Family-based | P | P F | P F |
| Feature-based | P | P F | P F |

Legend: P/F/f - product/family/feature-based analysis

# Implementation vs. Specification vs. Analysis

Possible combinations of the strategies:

| Impl. \ Spec. | Product-based | Family-based | Feature-based |
|---|---|---|---|
| Product-based | P | P | P |
| Family-based | P | P F | P F |
| Feature-based | P | P F | P F f |

Legend: P/F/f - product/family/feature-based analysis