Comparing CNF Transformations for Feature-Model Analysis

Elias Kuiter | FOSD Meeting 2022

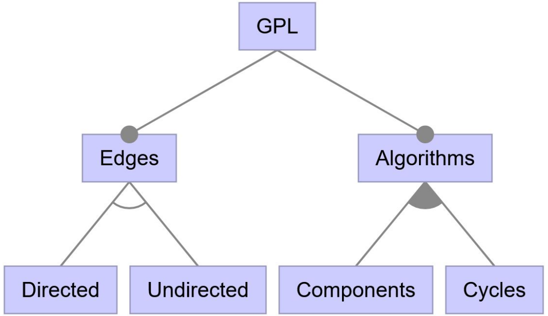
w/ Sebastian Krieter, Chico Sundermann, Thomas Thüm, Gunter Saake

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

Databases and Software Engineering

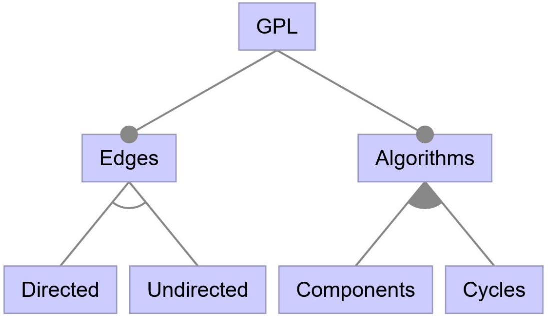
universität ulm

SAT-Based Analysis of Feature Models



Components ∨ Cycles ⇒ Directed

SAT-Based Analysis of Feature Models



Components ∨ Cycles ⇒ Directed

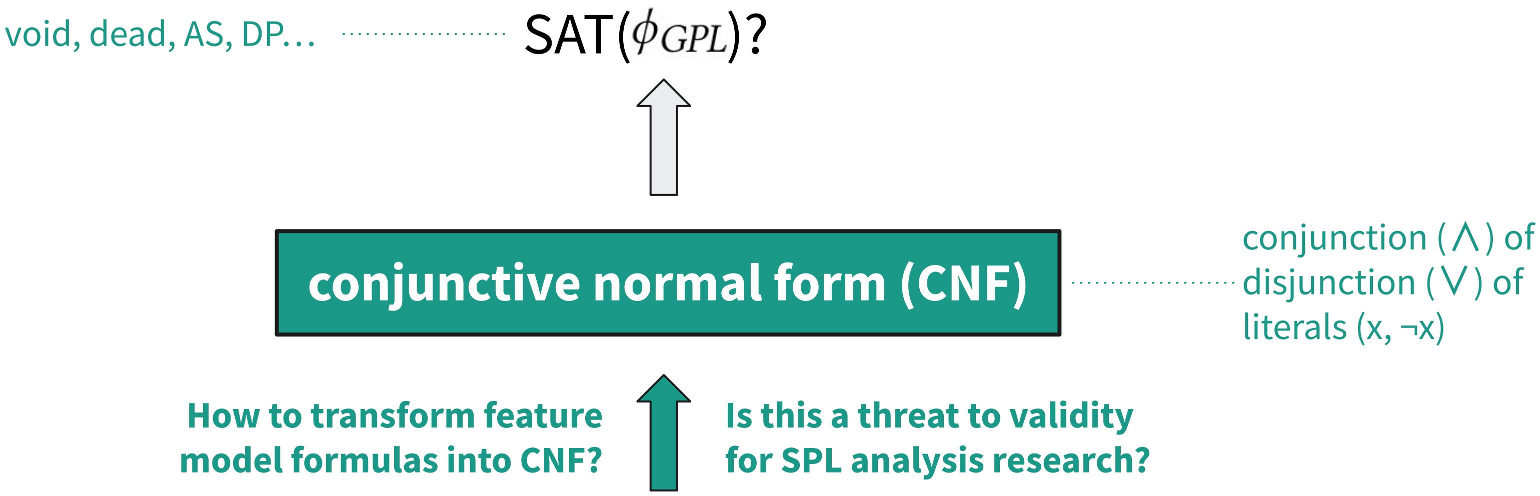
$$\begin{aligned}\phi_{GPL} = {} & GPL \land Edges \land Algorithms \land (Directed \lor Undirected) \\ & \land (\neg Directed \lor \neg Undirected) \land (Components \lor Cycles) \\ & \land (\neg(Components \lor Cycles) \lor Directed)\end{aligned}$$

SAT-Based Analysis of Feature Models

void, dead, AS, DP… ······················ SAT(ϕ_{GPL})?

$$\begin{aligned}\phi_{GPL} = {} & GPL \wedge Edges \wedge Algorithms \wedge (Directed \vee Undirected) \\ & \wedge (\neg Directed \vee \neg Undirected) \wedge (Components \vee Cycles) \\ & \wedge (\neg(Components \vee Cycles) \vee Directed)\end{aligned}$$

SAT-Based Analysis of Feature Models



void, dead, AS, DP…

$\mathrm{SAT}(\phi_{GPL})$?

conjunctive normal form (CNF)

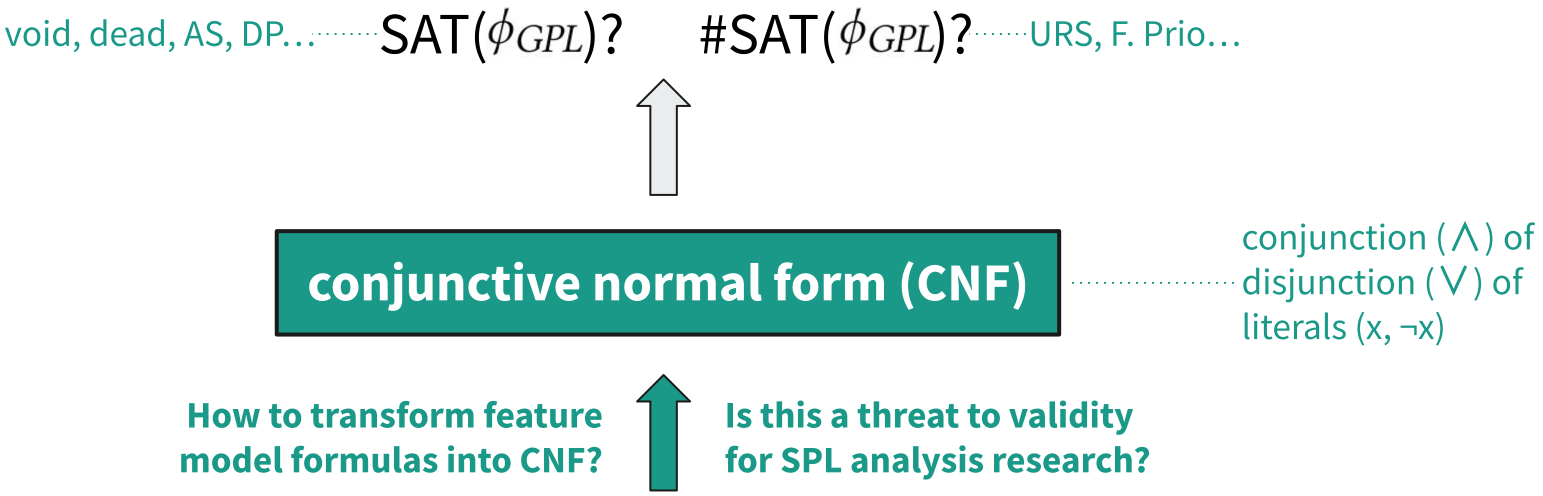
conjunction (∧) of disjunction (∨) of literals (x, ¬x)

How to transform feature model formulas into CNF?

Is this a threat to validity for SPL analysis research?

$$\begin{aligned}\phi_{GPL} = {} & GPL \land Edges \land Algorithms \land (Directed \lor Undirected)\\ & \land (\neg Directed \lor \neg Undirected) \land (Components \lor Cycles)\\ & \land (\neg(Components \lor Cycles) \lor Directed)\end{aligned}$$

# (#)SAT-Based Analysis of Feature Models



void, dead, AS, DP… $\mathrm{SAT}(\phi_{GPL})$? $\#\mathrm{SAT}(\phi_{GPL})$? URS, F. Prio…

conjunctive normal form (CNF)

conjunction (∧) of disjunction (∨) of literals (x, ¬x)

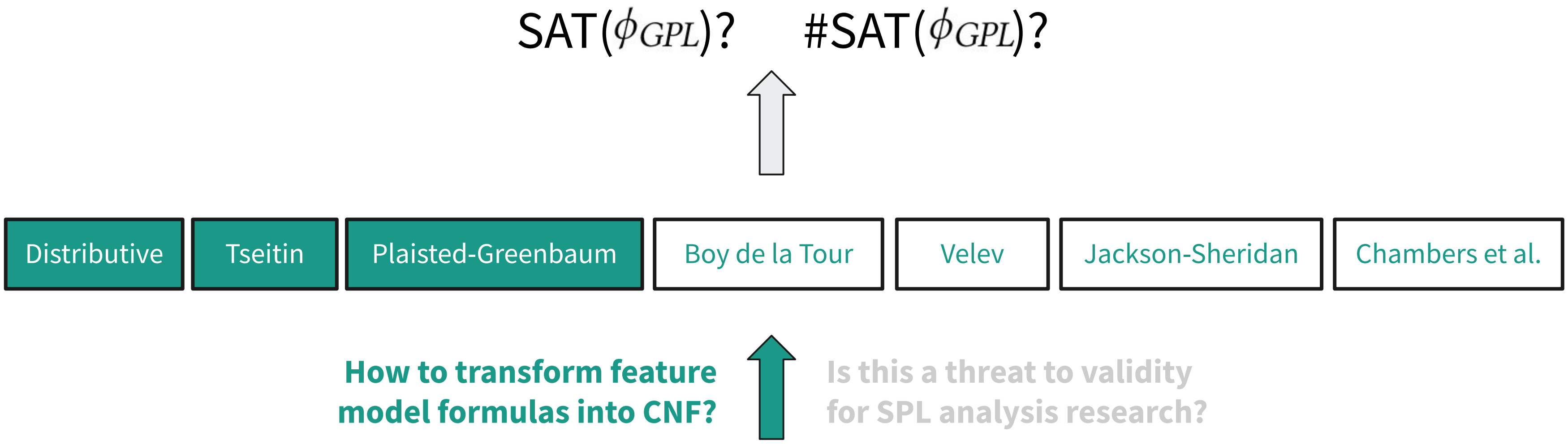
How to transform feature model formulas into CNF?

Is this a threat to validity for SPL analysis research?

$$\begin{aligned}\phi_{GPL} = {} & GPL \land Edges \land Algorithms \land (Directed \lor Undirected)\\ & \land (\neg Directed \lor \neg Undirected) \land (Components \lor Cycles)\\ & \land (\neg(Components \lor Cycles) \lor Directed)\end{aligned}$$

Chico Sundermann, Michael Nieke, Paul M. Bittner, Tobias Heß, Thomas Thüm, and Ina Schaefer. 2021. Applications of #SAT Solvers on Feature Models. In *VaMoS'21*. ACM, NY, USA, Article 12, 1–10.

# (#)SAT-Based Analysis of Feature Models



$\mathrm{SAT}(\phi_{GPL})$? $\#\mathrm{SAT}(\phi_{GPL})$?

Distributive

Tseitin

Plaisted-Greenbaum

Boy de la Tour

Velev

Jackson-Sheridan

Chambers et al.

How to transform feature model formulas into CNF?

Is this a threat to validity for SPL analysis research?

$$\begin{aligned}\phi_{GPL} = {} & GPL \land Edges \land Algorithms \land (Directed \lor Undirected)\\ & \land (\neg Directed \lor \neg Undirected) \land (Components \lor Cycles)\\ & \land (\neg(Components \lor Cycles) \lor Directed)\end{aligned}$$

Alouneh, S., Abed, S., Al Shayeji, M.H. et al. A comprehensive study and analysis on SAT-solvers: advances, usages and achievements. Artif Intell Rev 52, 2575–2601 (2019).

How to transform into CNF?

How to transform into CNF?

Distributive Transformation

Idea: Apply *distributive* and *De Morgan's laws*

Example:

$... \wedge (Co \vee Cy) \wedge (\neg(Co \vee Cy) \vee D)$

$... \wedge (Co \vee Cy) \wedge ((\neg Co \wedge \neg Cy) \vee D)$

$... \wedge (Co \vee Cy) \wedge (\neg Co \vee D) \wedge (\neg Cy \vee D)$

Properties:

- preserves equivalence
- easy to implement
- exponential space complexity

How to transform into CNF?

Distributive Transformation

Idea: Apply *distributive* and *De Morgan's laws*

Example:

$$\ldots \wedge (Co \vee Cy) \wedge (\neg(Co \vee Cy) \vee D)$$

$$\ldots \wedge (Co \vee Cy) \wedge ((\neg Co \wedge \neg Cy) \vee D)$$

$$\ldots \wedge (Co \vee Cy) \wedge (\neg Co \vee D) \wedge (\neg Cy \vee D)$$

Properties:

- preserves equivalence
- easy to implement
- <u>exponential</u> space complexity

Tseitin Transformation

Idea: Use ↔ to define variables ("shortcuts") for subformulas

Example: $\square_{\phi} \wedge \ldots$

$$\wedge\ (\square_{\phi} \leftrightarrow \ \ldots \wedge \square_{Co \vee Cy} \wedge \square_{(\neg Co \wedge \neg Cy) \vee D}))$$

$$\wedge\ (\square_{Co \vee Cy} \leftrightarrow (Co \vee Cy))$$

$$\wedge\ (\square_{(\neg Co \wedge \neg Cy) \vee D} \leftrightarrow (\square_{(\neg Co \wedge \neg Cy)} \vee D))$$

$$\wedge\ (\square_{\neg Co \wedge \neg Cy} \leftrightarrow (\neg Co \wedge \neg Cy))$$

Properties:

- preserves assignments + count
- introduces <u>artificial variables</u>
- linear space complexity

Tseitin, G. S. (1983). On the complexity of derivation in propositional calculus. In Automation of reasoning (pp. 466-483). Springer, Berlin, Heidelberg.

How to transform into CNF?

Distributive Transformation

Idea: Apply *distributive* and *De Morgan's laws*

Example:

$\ldots \wedge (Co \vee Cy) \wedge (\neg(Co \vee Cy) \vee D)$

$\ldots \wedge (Co \vee Cy) \wedge ((\neg Co \wedge \neg Cy) \vee D)$

$\ldots \wedge (Co \vee Cy) \wedge (\neg Co \vee D) \wedge (\neg Cy \vee D)$

Properties:

- preserves equivalence
- easy to implement
- exponential space complexity

Tseitin Transformation

often mixed up

Idea: Use ↔ to define variables ("shortcuts") for subformulas

Example: $\Box_\phi \wedge \ldots$

$\wedge\ (\Box_\phi \leftrightarrow \ldots \wedge \Box_{Co \vee Cy} \wedge \Box_{(\neg Co \wedge \neg Cy) \vee D}))$

$\wedge\ (\Box_{Co \vee Cy} \leftrightarrow (Co \vee Cy))$

$\wedge\ (\Box_{(\neg Co \wedge \neg Cy) \vee D} \leftrightarrow (\Box_{(\neg Co \wedge \neg Cy)} \vee D))$

$\wedge\ (\Box_{\neg Co \wedge \neg Cy} \leftrightarrow (\neg Co \wedge \neg Cy))$

Properties:

- preserves assignments + count
- introduces artificial variables
- linear space complexity

Plaisted-Greenbaum Trans.

Idea: Like Tseitin, but using → ("half-definitions")

Example: $\Box_\phi \wedge \ldots$

$\wedge\ (\Box_\phi \rightarrow \ldots \wedge \Box_{Co \vee Cy} \wedge \Box_{(\neg Co \wedge \neg Cy) \vee D}))$

$\wedge\ (\Box_{Co \vee Cy} \rightarrow (Co \vee Cy))$

$\wedge\ (\Box_{(\neg Co \wedge \neg Cy) \vee D} \rightarrow (\Box_{(\neg Co \wedge \neg Cy)} \vee D))$

$\wedge\ (\Box_{\neg Co \wedge \neg Cy} \rightarrow (\neg Co \wedge \neg Cy))$

Properties:

- preserves assignments
- does not preserve count
- requires less space than Tseitin

Plaisted, D. A., & Greenbaum, S. (1986). A structure-preserving clause form translation. Journal of Symbolic Computation, 2(3), 293-304.

Evaluation

Is this a threat to validity?

Research Questions

Research Questions

1. Does the CNF transfomation influence

   the **performance of SAT and #SAT-based analyses**?

Research Questions

1. Does the CNF transfomation influence

   the **performance of SAT and #SAT-based analyses**?

2. Does this influence depend on the …

   a. **size of the feature model** (do larger models imply larger performance differences)?

   b. **size increase of the formula** (is it costly to introduce variables/literals)?

Research Questions

1. Does the CNF transfomation influence

   the **performance of SAT and #SAT-based analyses**?

2. Does this influence depend on the …

   a. **size of the feature model** (do larger models imply larger performance differences)?

   b. **size increase of the formula** (is it costly to introduce variables/literals)?

3. Does the CNF transformation affect the **correctness of #SAT-based analyses**?

Subject Systems

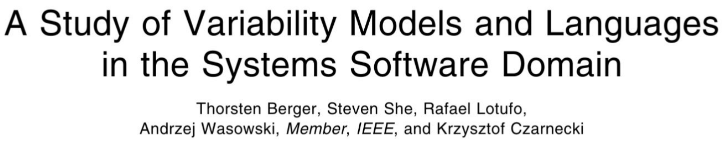
Subject Systems

Berger et al. 2013

Kconfig-based feature models

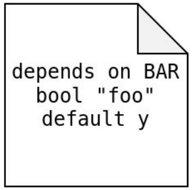
Current versions of 9 FOSS systems:

axTLS, Buildroot, BusyBox, EmbToolkit, Fiasco, Freetz-NG, Linux, toybox, uClibc-ng



A Study of Variability Models and Languages in the Systems Software Domain

Thorsten Berger, Steven She, Rafael Lotufo,
Andrzej Wasowski, *Member*, *IEEE*, and Krzysztof Czarnecki



```
depends on BAR
 bool "foo"
 default y
```

T. Berger, S. She, R. Lotufo, A. Wasowski and K. Czarnecki, "A Study of Variability Models and Languages in the Systems Software Domain," in IEEE Transactions on Software Engineering, vol. 39, no. 12, pp. 1611-1640, Dec. 2013, doi: 10.1109/TSE.2013.34.

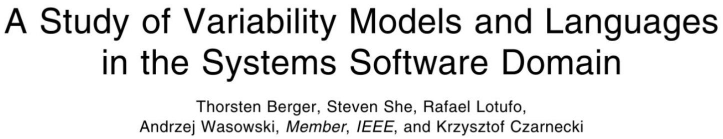
Subject Systems

Berger et al. 2013

Kconfig-based feature models

Current versions of 9 FOSS systems:

axTLS, Buildroot, BusyBox, EmbToolkit, Fiasco, Freetz-NG, Linux, toybox, uClibc-ng



A Study of Variability Models and Languages in the Systems Software Domain

Thorsten Berger, Steven She, Rafael Lotufo,
Andrzej Wasowski, *Member*, *IEEE*, and Krzysztof Czarnecki

```
depends on BAR
 bool "foo"
 default y
```

Knüppel et al. 2017

Large feature models (w/ tree)

4 automotive and 8 FOSS systems:

axTLS, BusyBox, eCos/CDL (3), EmbToolkit, Linux, uClibc, uClinux-base/distribution

Is There a Mismatch between Real-World Feature Models and Product-Line Research?

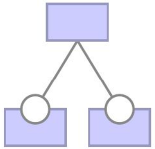
Alexander Knüppel
TU Braunschweig, Germany
a.knueppel@tu-bs.de

Thomas Thüm
TU Braunschweig, Germany
t.thuem@tu-bs.de

Stephan Mennicke
TU Braunschweig, Germany
mennicke@ips.cs.tu-bs.de

Jens Meinicke
University of Magdeburg, Germany
Carnegie Mellon University, USA
meinicke@ovgu.de

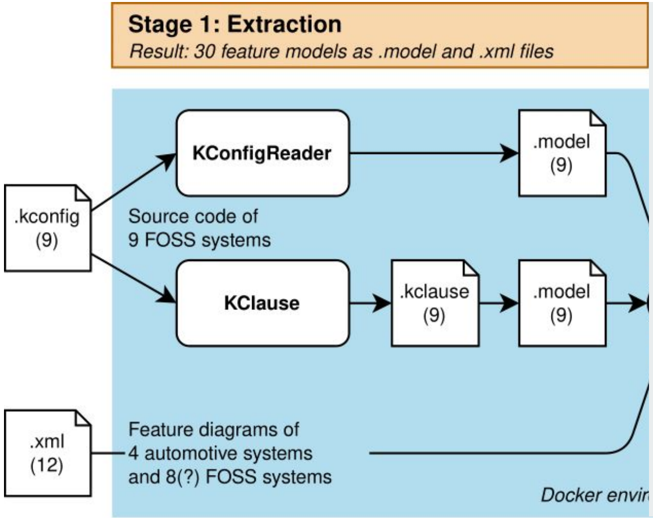
Ina Schaefer
TU Braunschweig, Germany
i.schaefer@tu-bs.de



Knüppel, A., Thüm, T., Mennicke, S., Meinicke, J., & Schaefer, I. (2017, August). Is there a mismatch between real-world feature models and product-line research?. In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (pp. 291-302).

Experimental Setup

Experimental Setup



Stage 1: Extraction

Result: 30 feature models as .model and .xml files

.kconfig (9)

Source code of 9 FOSS systems

KConfigReader

.model (9)

KClause

.kclause (9)

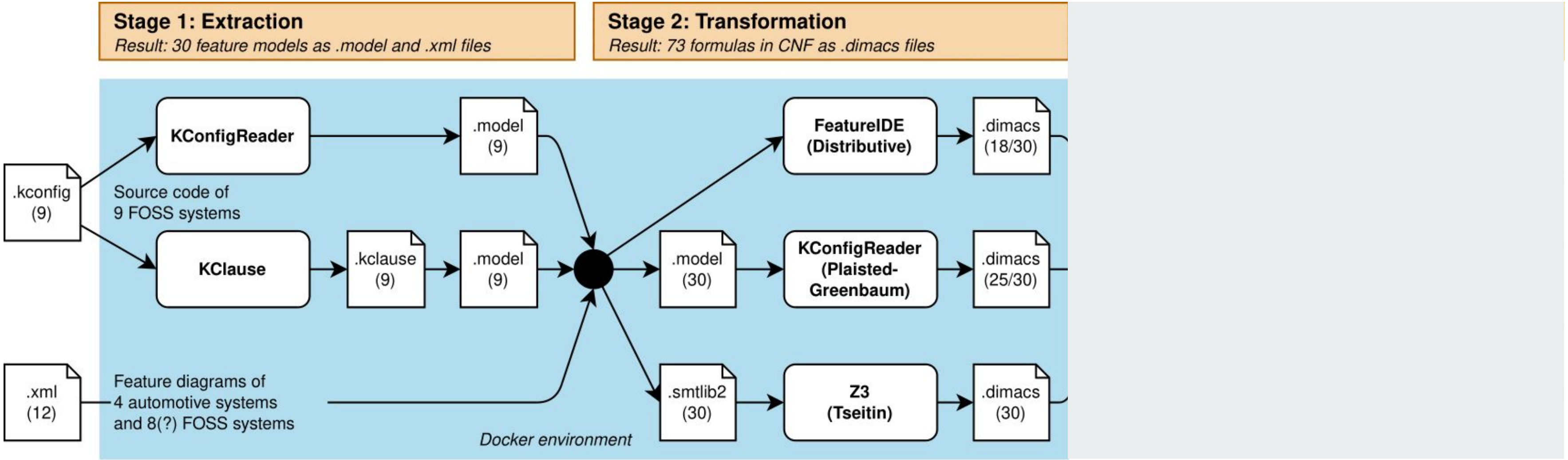
.model (9)

.xml (12)

Feature diagrams of 4 automotive systems and 8(?) FOSS systems

Docker envir

Experimental Setup



Stage 1: Extraction

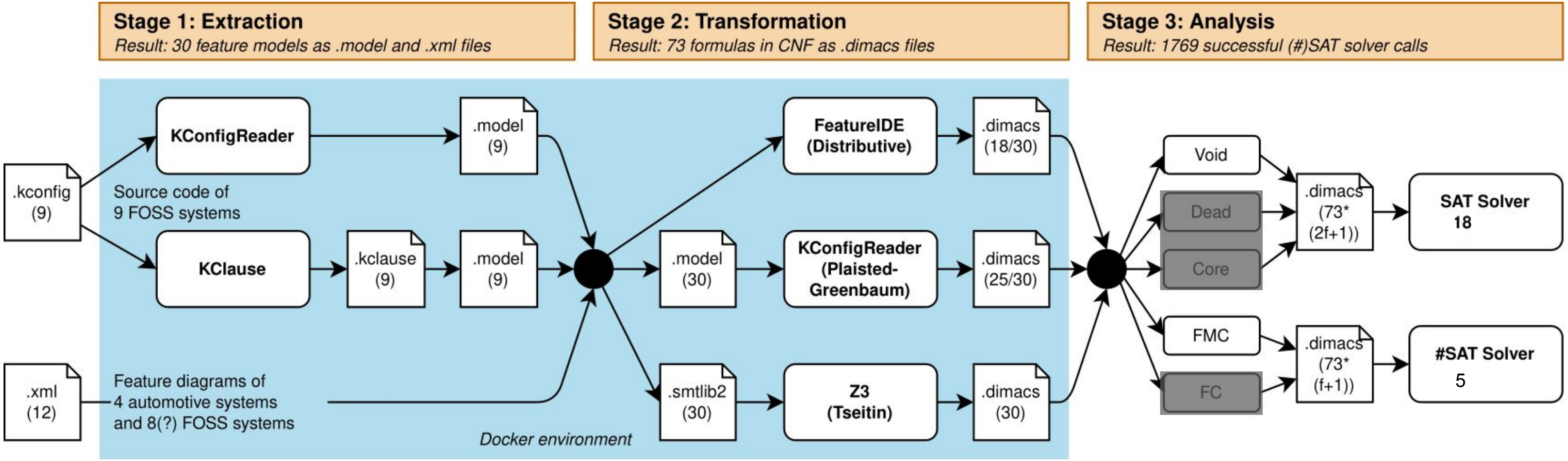
Result: 30 feature models as .model and .xml files

Stage 2: Transformation

Result: 73 formulas in CNF as .dimacs files

Experimental Setup

3 iterations
180s timeout



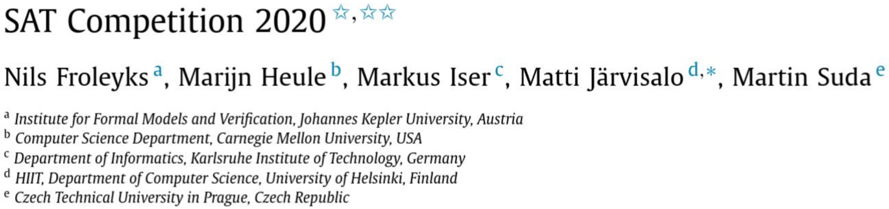
SAT and #SAT Solvers

SAT and #SAT Solvers

www.satcompetition.org, SAT Heritage

International SAT Competition

18 SAT solvers (winners in main track 2002-2021):
*zchaff (2), Forklift, SatELiteGTI, MiniSat, RSat,
precosat, CryptoMiniSat, glucose (2),
lingeling (2), Maple (4), Kissat (2)*



SAT Competition 2020[☆,☆☆]

Nils Froleyks[a], Marijn Heule[b], Markus Iser[c], Matti Järvisalo[d,*], Martin Suda[e]

[a] *Institute for Formal Models and Verification, Johannes Kepler University, Austria*
[b] *Computer Science Department, Carnegie Mellon University, USA*
[c] *Department of Informatics, Karlsruhe Institute of Technology, Germany*
[d] *HIIT, Department of Computer Science, University of Helsinki, Finland*
[e] *Czech Technical University in Prague, Czech Republic*

http://www.satcompetition.org/
https://github.com/sat-heritage/docker-images/

SAT and #SAT Solvers

www.satcompetition.org, SAT Heritage

International SAT Competition

18 SAT solvers (winners in main track 2002-2021):
zchaff (2), Forklift, SatELiteGTI, MiniSat, RSat, precosat, CryptoMiniSat, glucose (2), lingeling (2), Maple (4), Kissat (2)

SAT Competition 2020 ☆,☆☆

Nils Froleyks[a], Marijn Heule[b], Markus Iser[c], Matti Järvisalo[d,*], Martin Suda[e]

[a] *Institute for Formal Models and Verification, Johannes Kepler University, Austria*
[b] *Computer Science Department, Carnegie Mellon University, USA*
[c] *Department of Informatics, Karlsruhe Institute of Technology, Germany*
[d] *HIIT, Department of Computer Science, University of Helsinki, Finland*
[e] *Czech Technical University in Prague, Czech Republic*

Sundermann et al. 2020

Evaluating #SAT solvers

5 fastest exact #SAT solvers:
countAntom, d4, dSharp, GANAK, sharpSAT

Evaluating #SAT Solvers on Industrial Feature Models

Chico Sundermann
Technische Universität Braunschweig

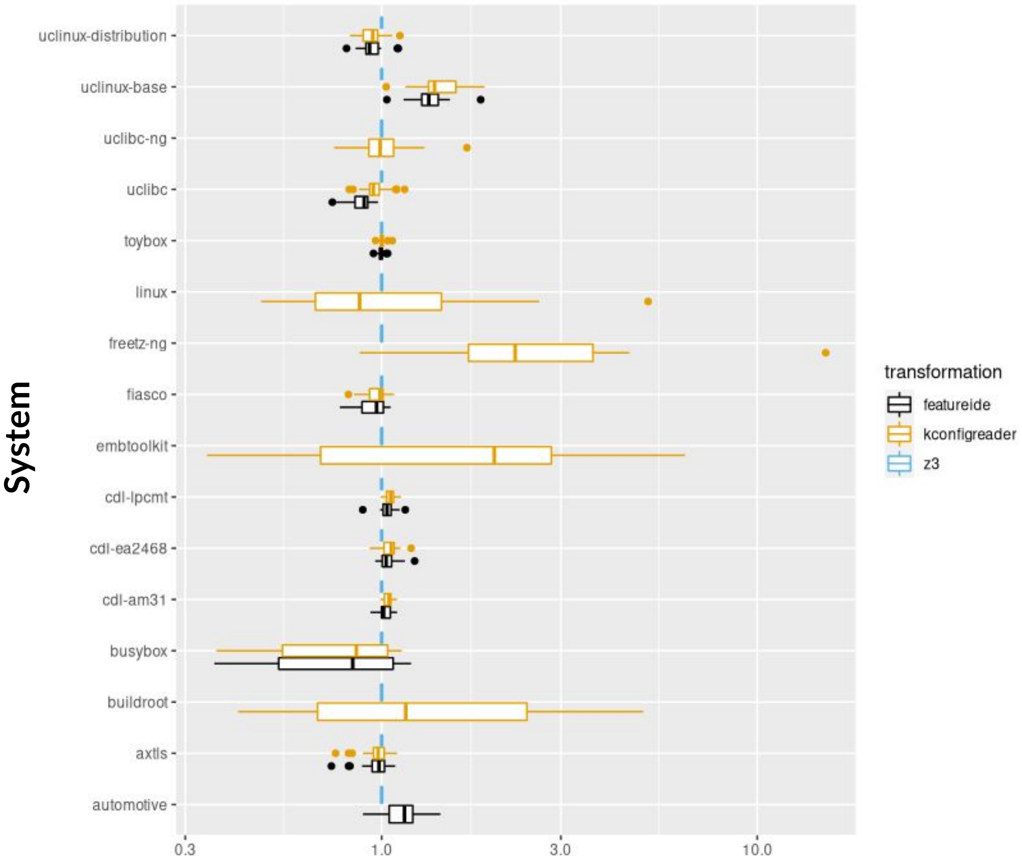
Thomas Thüm
University of Ulm

Ina Schaefer
Technische Universität Braunschweig

Chico Sundermann, Thomas Thüm, and Ina Schaefer. 2020. Evaluating #SAT solvers on industrial feature models. In VAMOS '20. ACM, NY, USA, Article 3, 1–9.

Preliminary Results

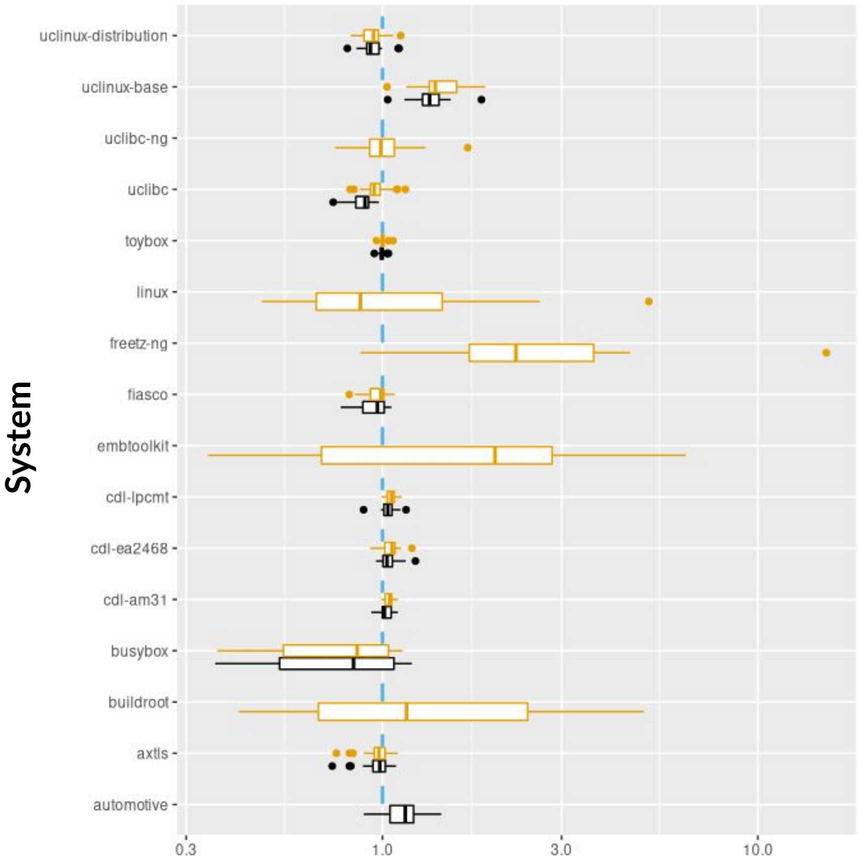
RQ1: Performance of SAT and #SAT



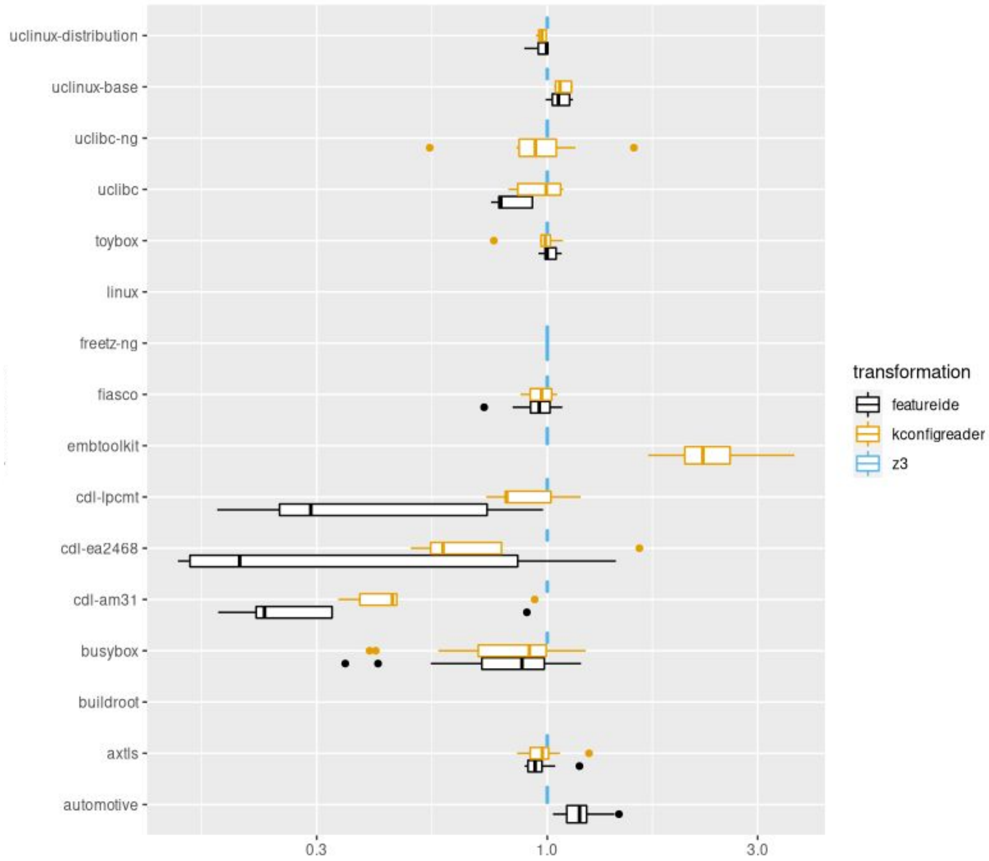
System

Log. **SAT time** relative to Z3

RQ1: Performance of SAT and #SAT

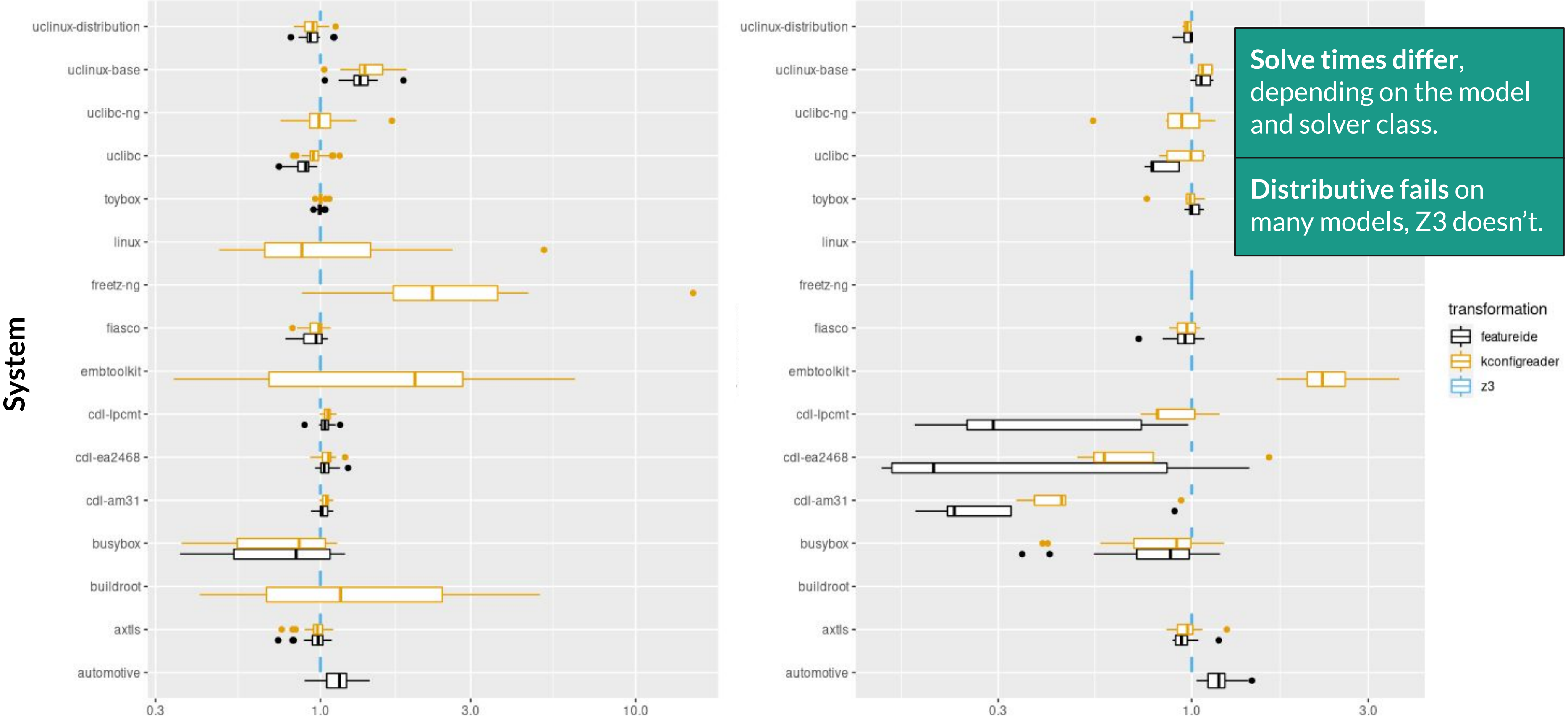


Log. **SAT time** relative to Z3



Log. **#SAT time** relative to Z3

RQ1: Performance of SAT and #SAT



Log. **SAT time** relative to Z3

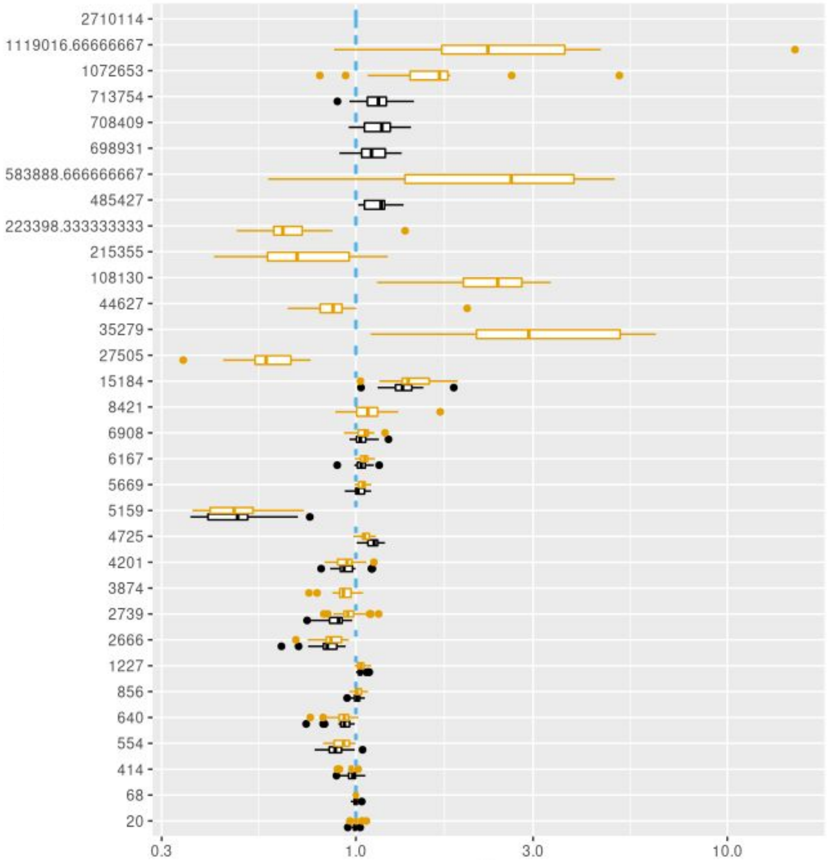
Log. **#SAT time** relative to Z3

Solve times differ, depending on the model and solver class.

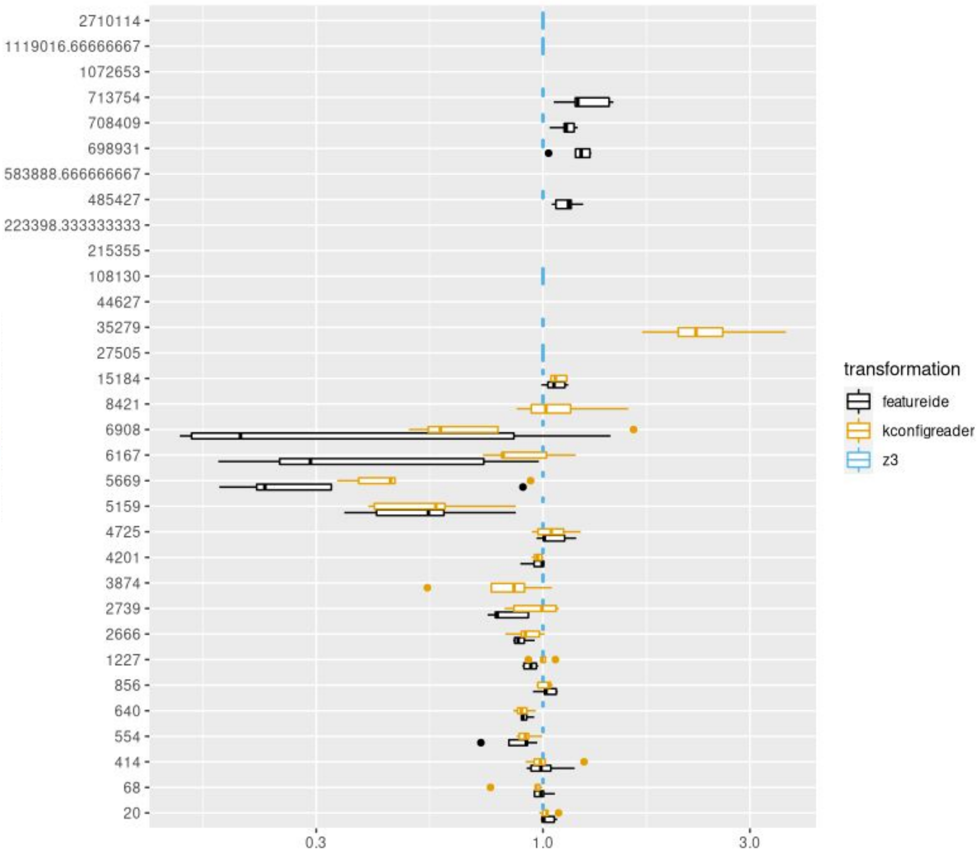
Distributive fails on many models, Z3 doesn't.

RQ2a: Feature Model Size

Log. **#Literals**

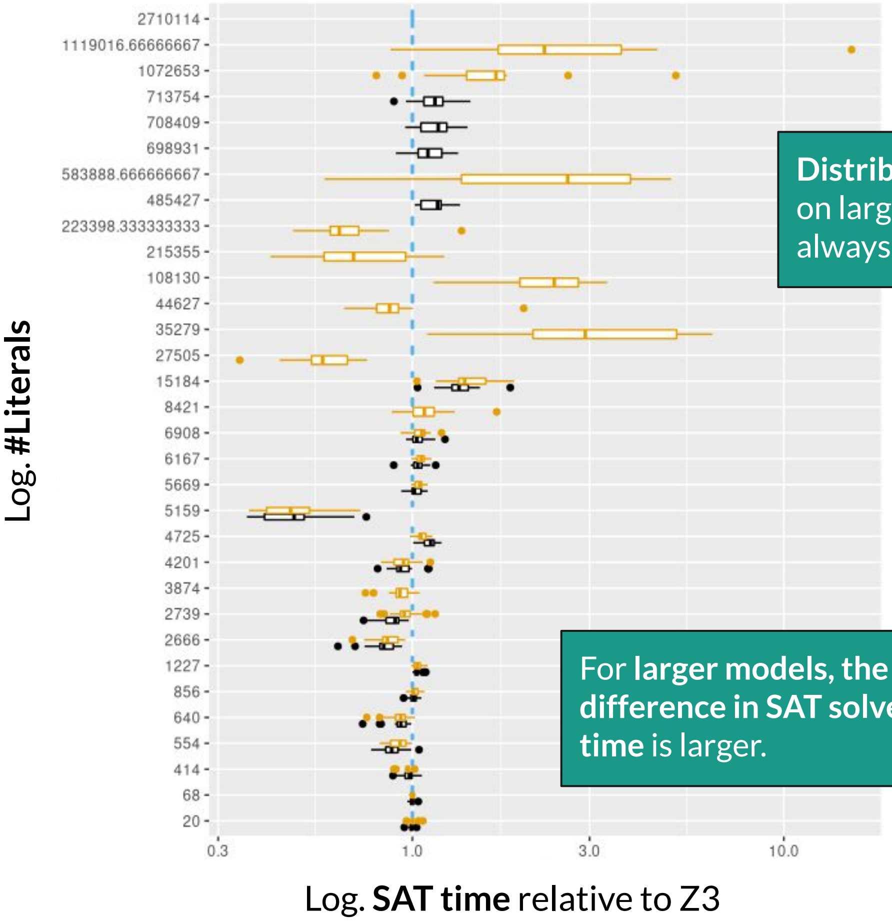


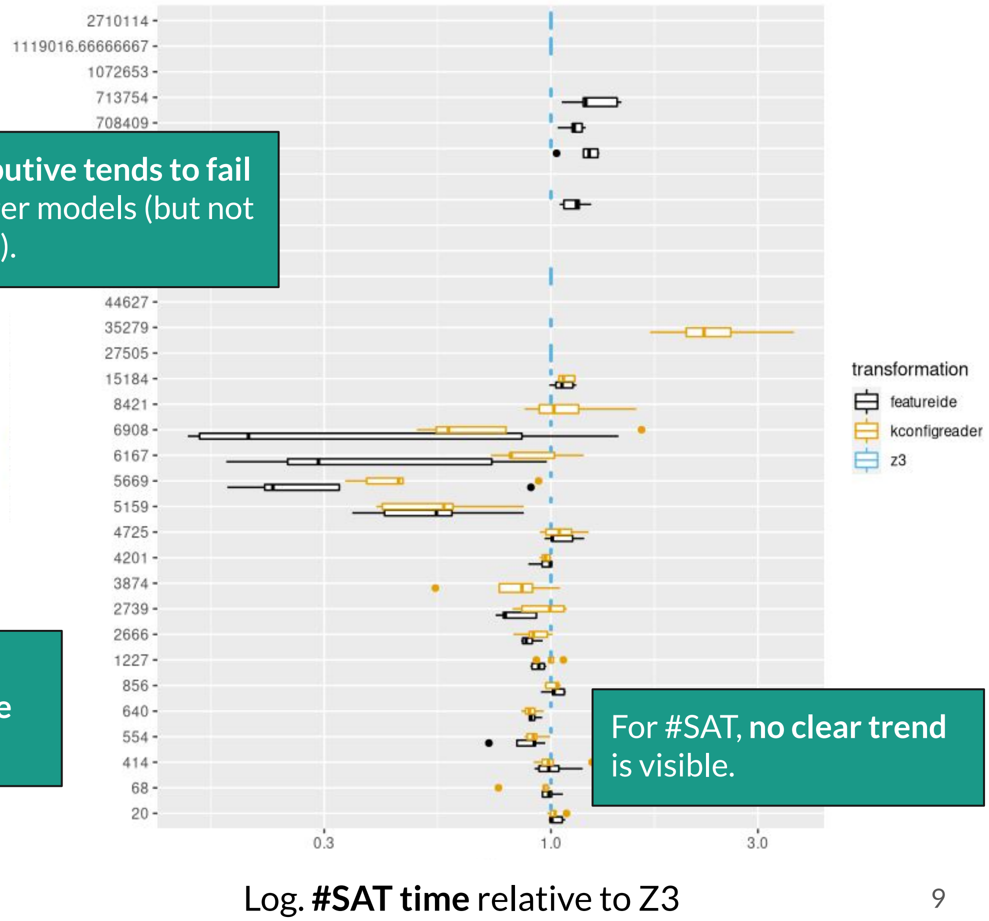
Log. **SAT time** relative to Z3



Log. **#SAT time** relative to Z3

RQ2a: Feature Model Size





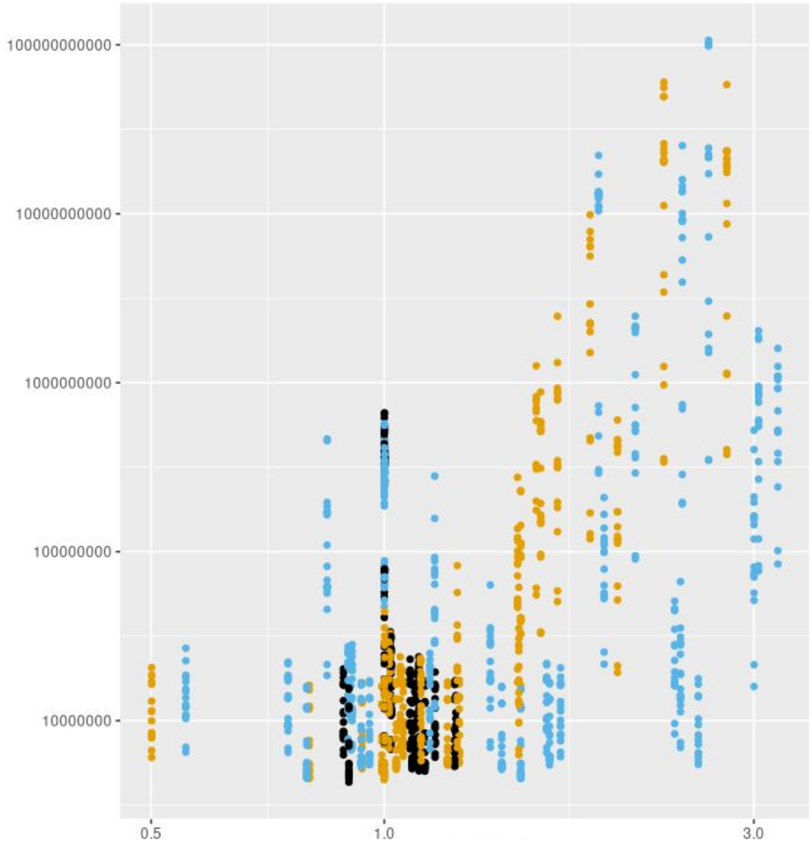
Distributive tends to fail on larger models (but not always).

For **larger models, the difference in SAT solve time** is larger.

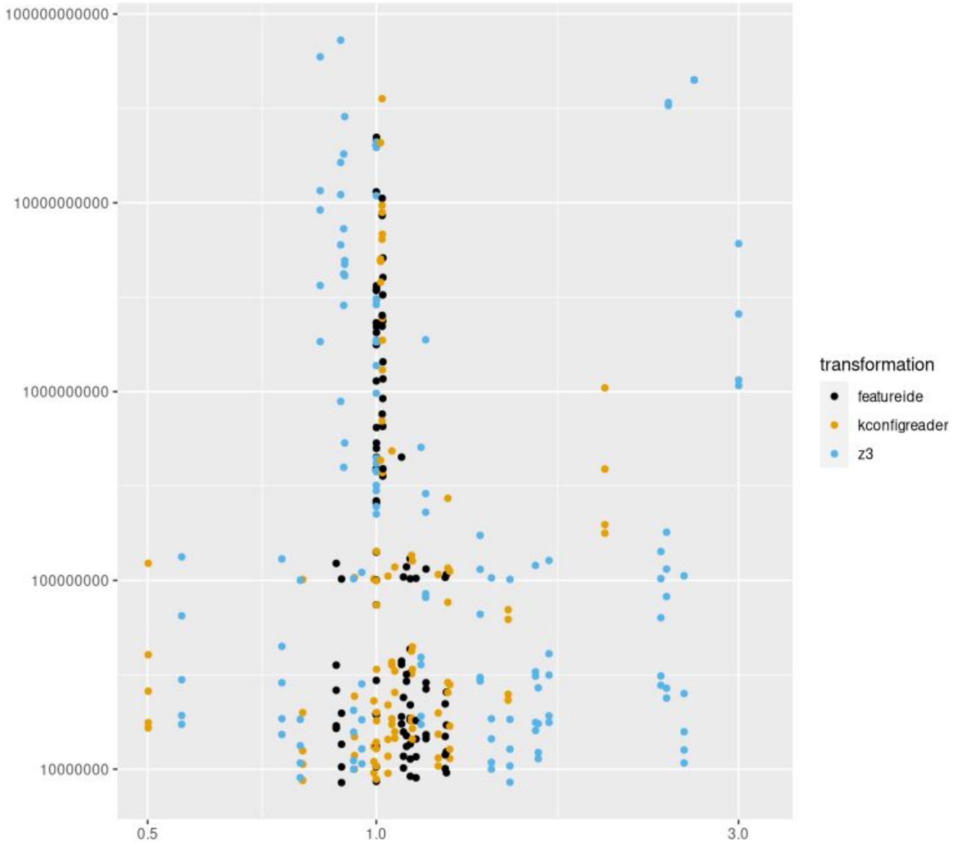
For #SAT, **no clear trend** is visible.

RQ2b: Formula Size Increase

Log. **solve time** (ns)



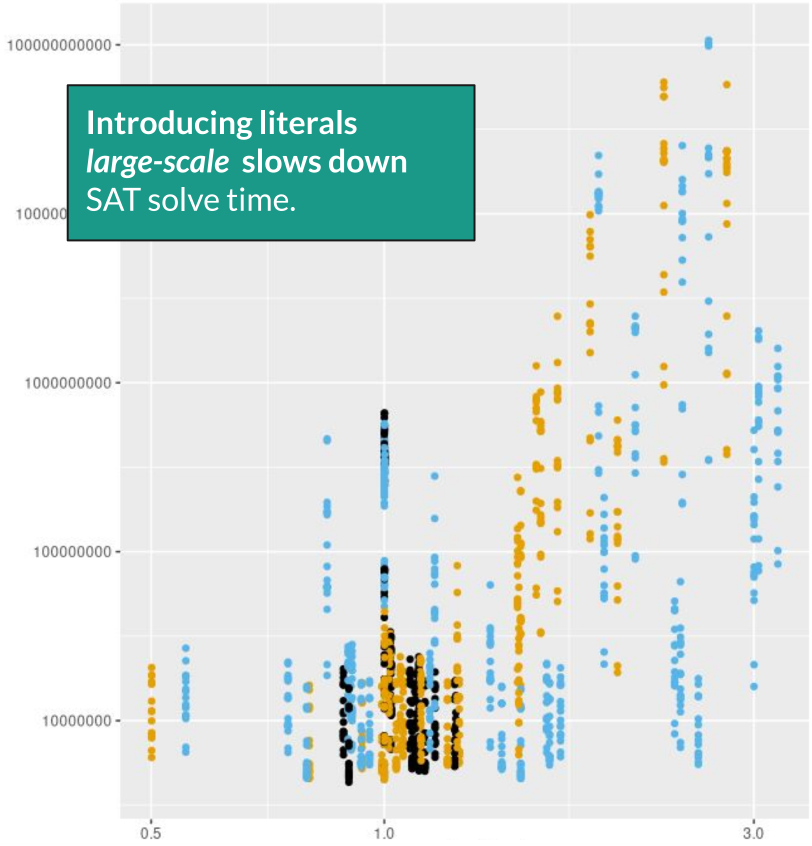
Log. **introduced literals** (%) for SAT



Log. **introduced literals** (%) for #SAT

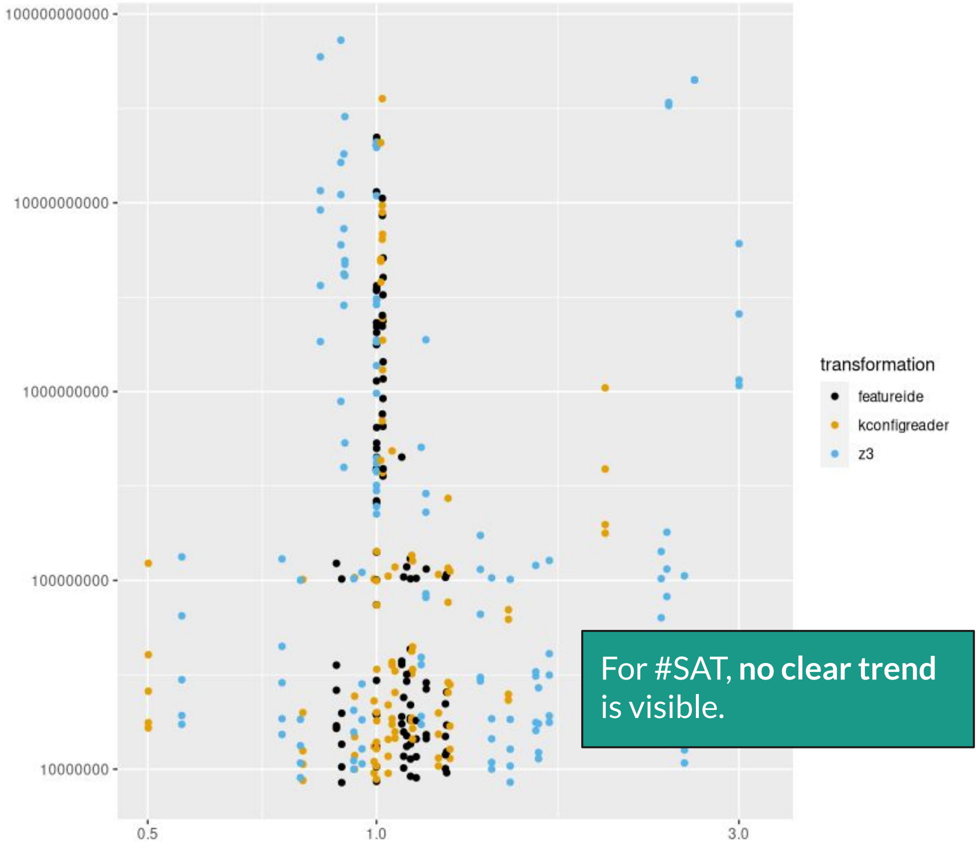
RQ2b: Formula Size Increase

Log. **solve time** (ns)



Introducing literals *large-scale* slows down SAT solve time.

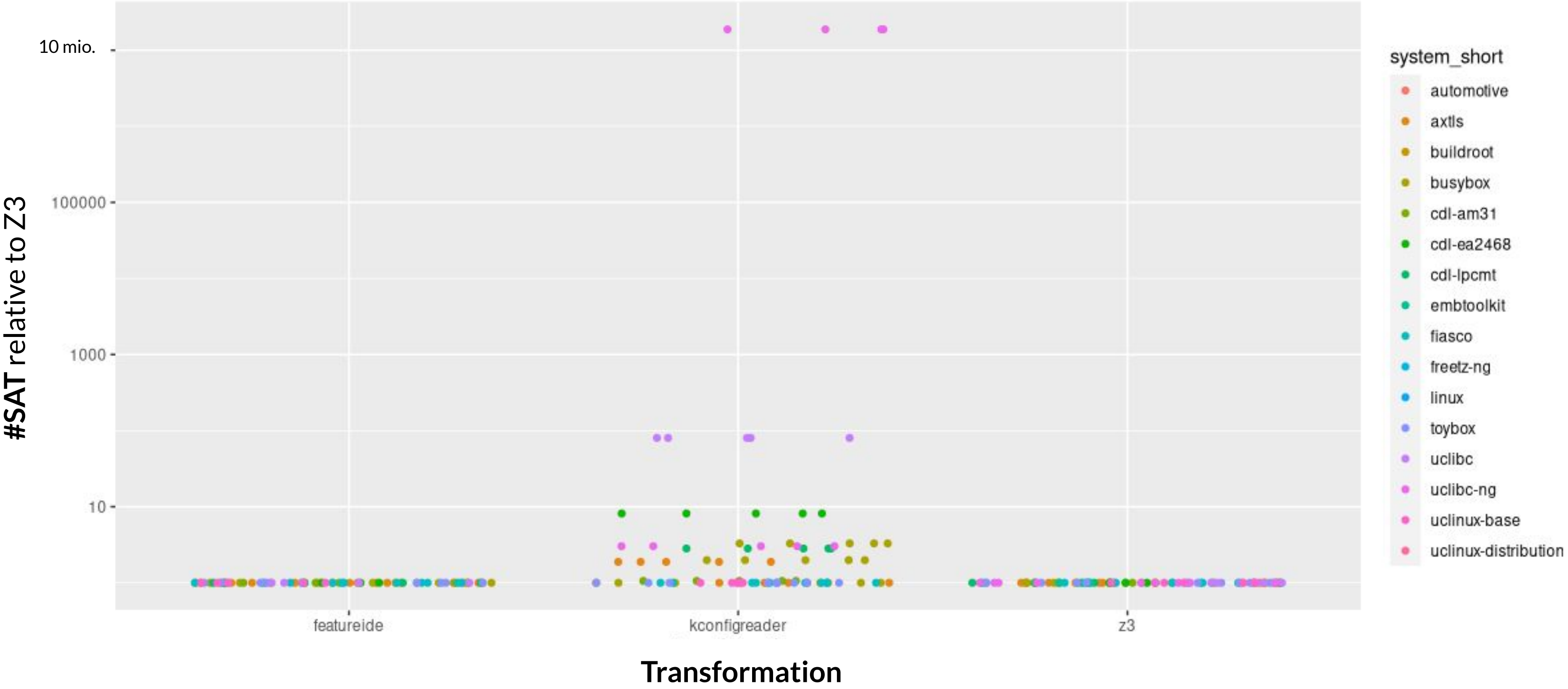
Log. **introduced literals** (%) for SAT



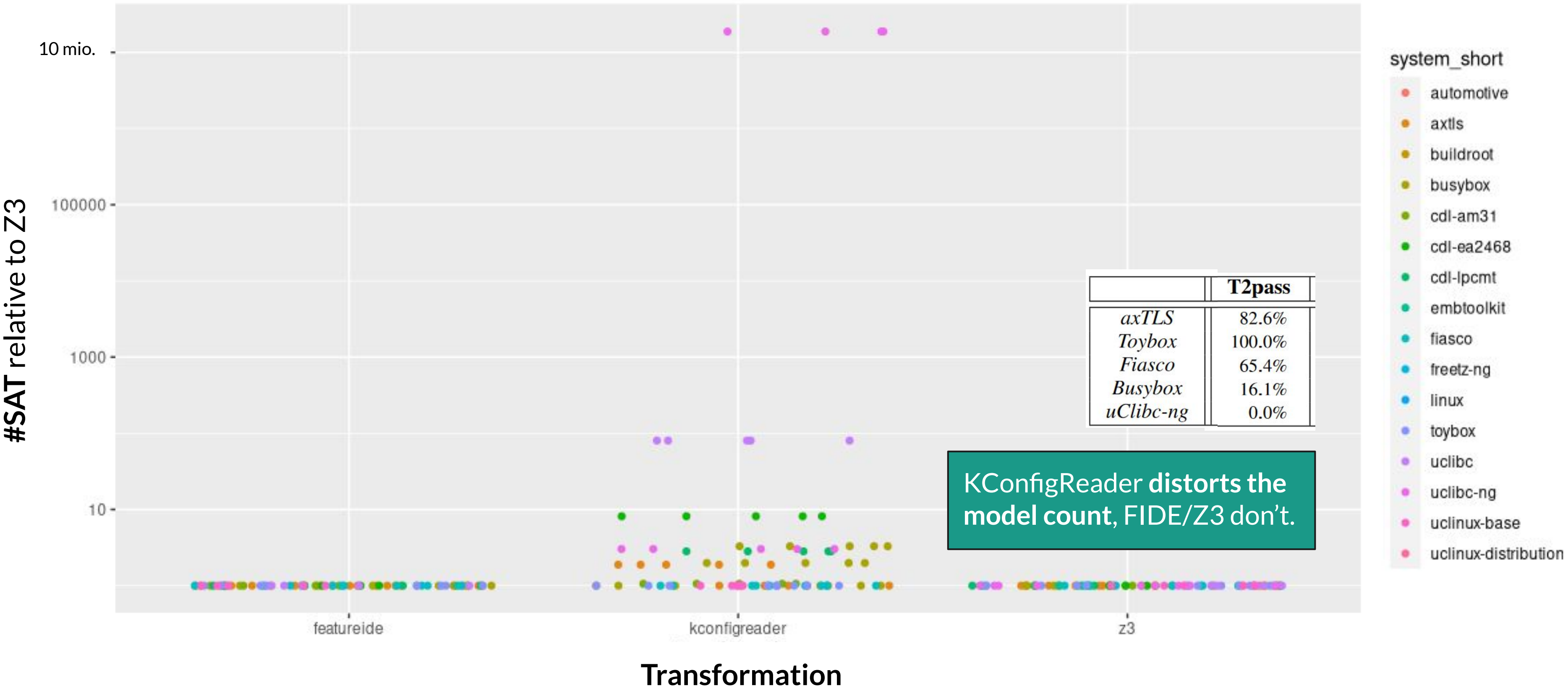
For #SAT, **no clear trend** is visible.

Log. **introduced literals** (%) for #SAT

RQ3: Correctness of #SAT



RQ3: Correctness of #SAT



	T2pass
axTLS	82.6%
Toybox	100.0%
Fiasco	65.4%
Busybox	16.1%
uClibc-ng	0.0%

KConfigReader **distorts the model count**, FIDE/Z3 don't.

Oh, J., Batory, D.S., Heule, M.J., Myers, M., & Gazzillo, P. (2019). Uniform Sampling from Kconfig Feature Models. Technical Report.

Perspective

- **External threats to validity**
  - We only evaluated **specific implementations**
    → cannot draw conclusions about transformations themselves
  - We chose **specific systems**/extractors/transformations/solvers
  - We do not account for **non-Boolean** (e.g., numeric) variability

Perspective

- **External threats to validity**
  - We only evaluated **specific implementations**
    → cannot draw conclusions about transformations themselves
  - We chose **specific systems**/extractors/transformations/solvers
  - We do not account for **non-Boolean** (e.g., numeric) variability
- **Future work**
  - **Controlled, parametrized** evaluation of all three transformations by implementing them all in Z3 and KConfigReader
    → make **recommendations** for which transformation to choose

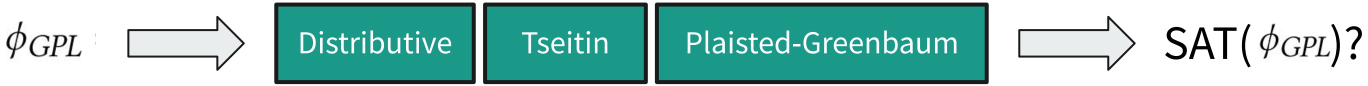
Perspective

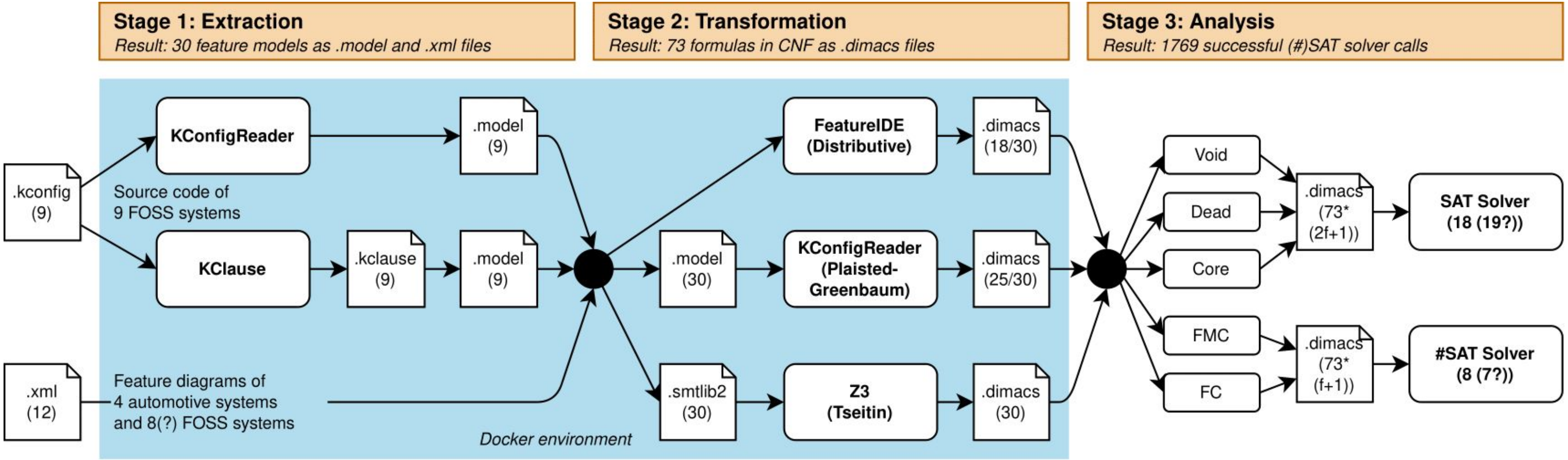
- **External threats to validity**
  - We only evaluated **specific implementations**
    → cannot draw conclusions about transformations themselves
  - We chose **specific systems**/extractors/transformations/solvers
  - We do not account for **non-Boolean** (e.g., numeric) variability

- **Future work**
  - **Controlled, parametrized** evaluation of all three transformations
    by implementing them all in Z3 and KConfigReader
    → make **recommendations** for which transformation to choose

- **Artifact:** Reproducible model extraction pipeline (VM-based, Docker-based WIP)
  https://github.com/ekuiter/feature-model-repository-pipeline

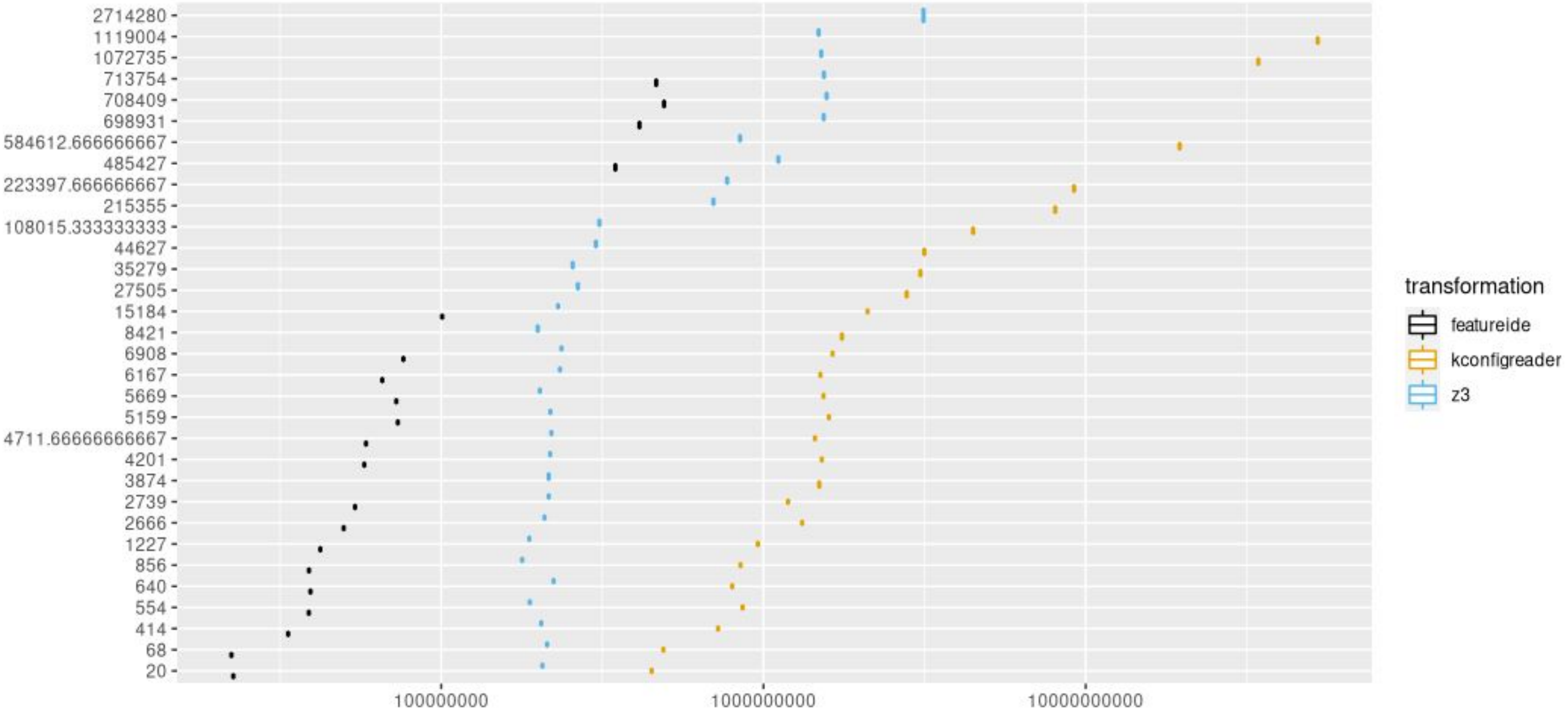
Conclusion





https://github.com/ekuiter/feature-model-repository-pipeline

Results: Performance of Transformation

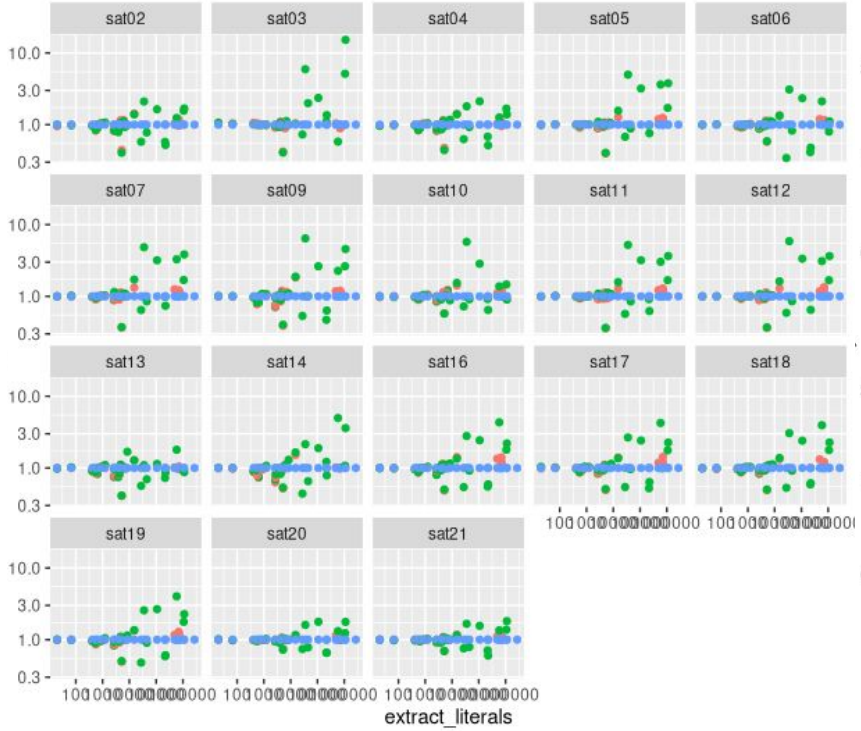


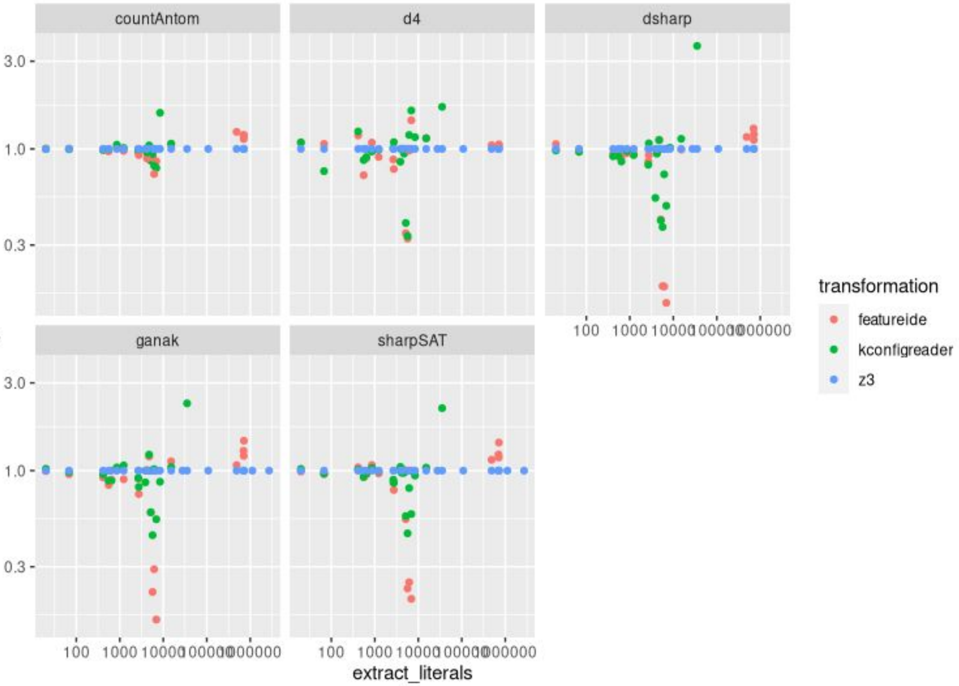
Log. **#Literals**

Log. **transformation time**

Results: Differences between Solvers

Log. **solve time** relative to Z3





Log. **#Literals**

Log. **#Literals**