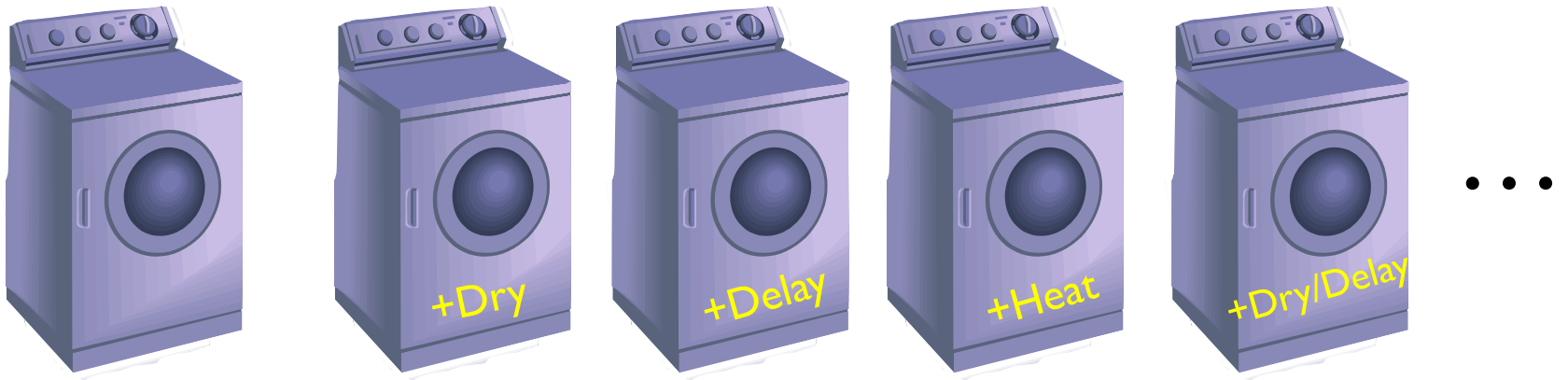
Transformations of Software Product Lines:

A Generalizing Framework based on Category Theory

Gabriele Taentzer, Rick Salay, Daniel Strüber and Marsha Chechik

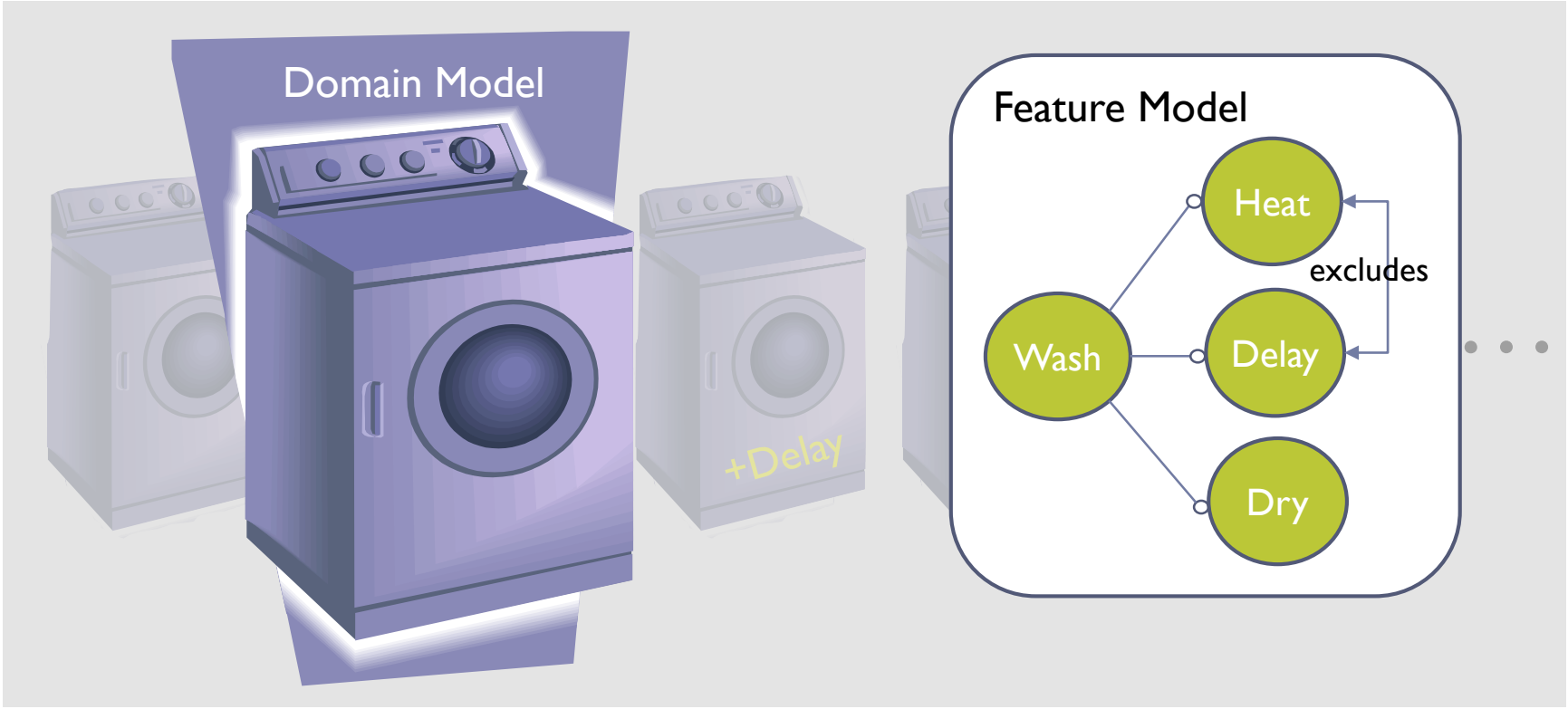
Software Product Lines (SPL)

- Manage a large number of similar but different artifact variants (products)
  - Washing Machine Co.



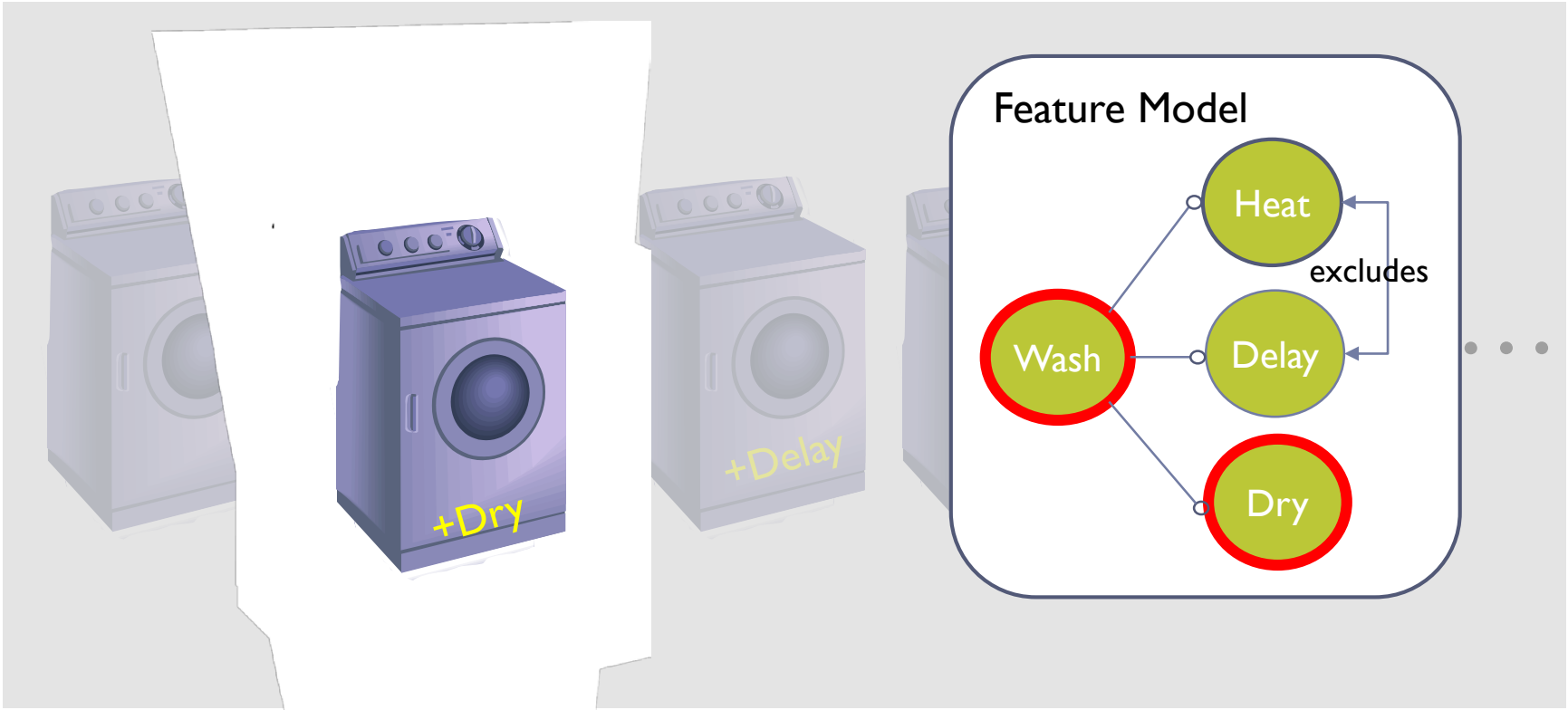
SPL Structure

- SPL (annotative) represented by
  - Domain Model – combined parts from all products
  - Feature Model – shows possible features and restrictions for products

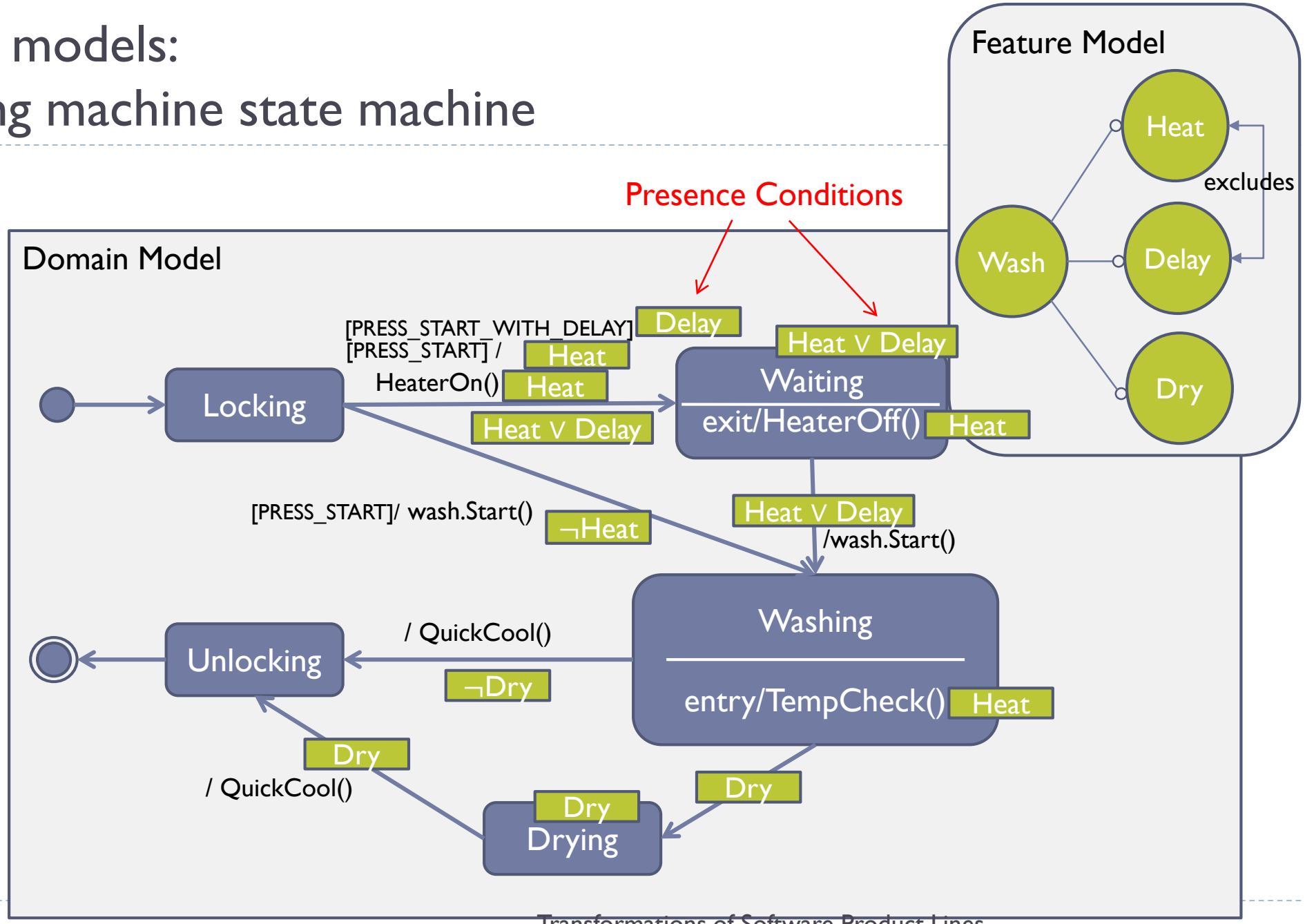


SPL Configuration – example

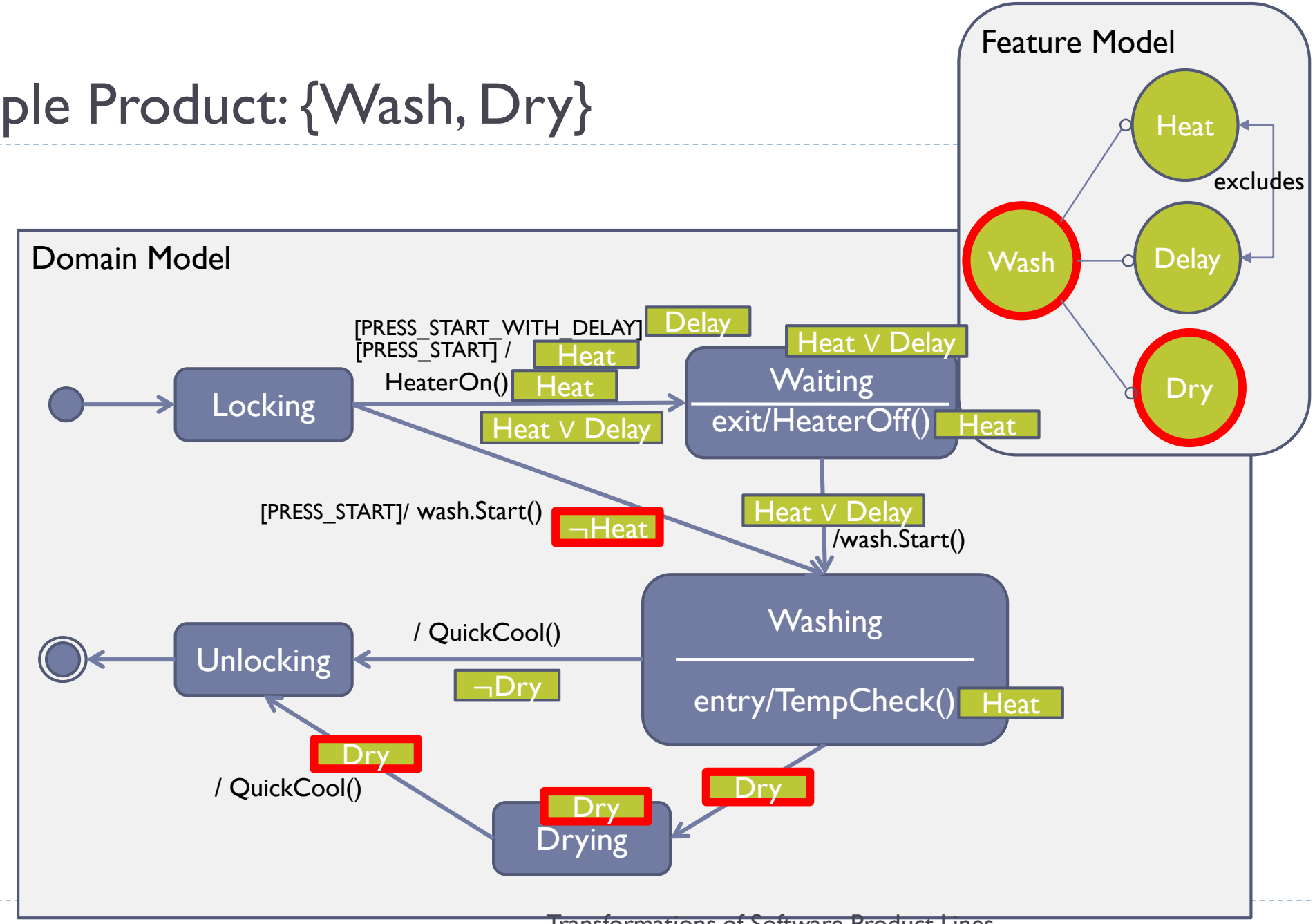
- +Dry product
  - Feature configuration: {Wash, Dry}



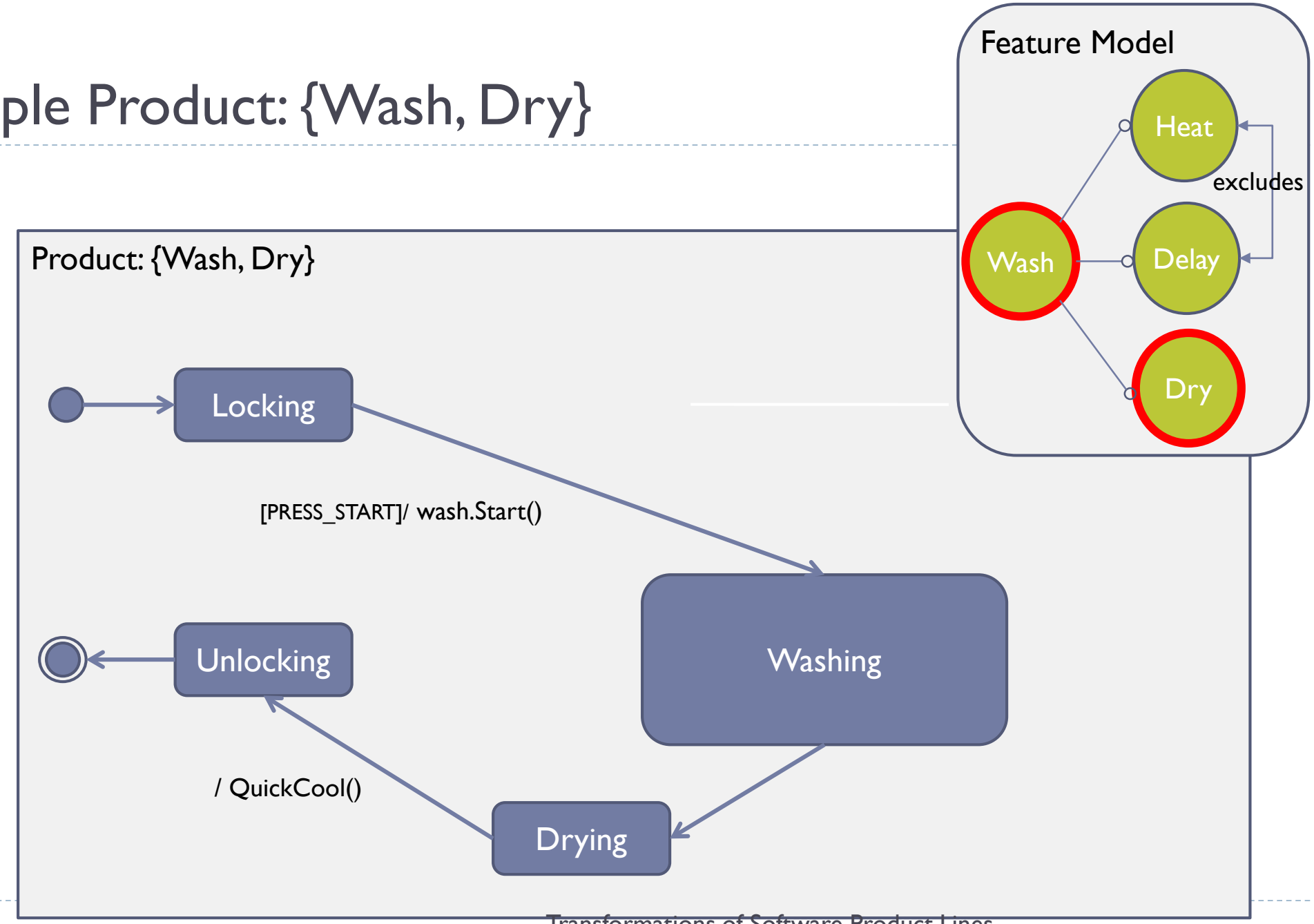
SPL of models: washing machine state machine



Example Product: {Wash, Dry}



Example Product: {Wash, Dry}



Outline

- Software Product Lines
- Transformations of SPLs
- What is the problem?
- Approach (part I): Category of SPLs
- Approach (part II): SPL Transformations using graph transformation rules
- Summary of results and next steps

Key types of SPL transformations in the Literature

1. Feature model transformations
   - Supports reasoning about additions, deletions, and modifications of features
   - e.g., Transformation rules to specify high-level feature editing operations [Bürdek et al.]
2. Lifted model transformations
   - Adapts single-product transformation rules to the entire SPL [Salay et al.]
   - Effect of lifting is same as applying the rule to each product separately.
3. SPL refinement
   - Supports safe evolution SPL by controlling impact on existing products.
   - e.g., Modifications restricted so that only a subset of products change [Sampaio et al.]

Motivation

- Types (1) & (3) apply only to feature models; type (2) applies only to domain models

  None of these types of transformation apply to entire SPL's – feature *and* domain models!

- But, this is needed in practice:
  - Addition or deletion of features usually entails the corresponding changes in the domain model
- Research Objective:

  A formal characterization of SPL transformations that addresses both feature and domain models.

Approach

- Build on existing formal theories: category theory and theory of Algebraic Graph Transformations (AGT)[1]
- Our strategy: given any suitable (to be defined) category $\boldsymbol{Mod}$ of models,
  1. Show how to define the category $PL_{\boldsymbol{Mod}}$ of SPLs having $\boldsymbol{Mod}$ models as domain models.
  2. Use AGT to define transformation rules for SPLs in $PL_{\boldsymbol{Mod}}$
- Benefits:
  - General, systematic and covers both feature and domain model parts of an SPL
  - Gets formal techniques from AGT that support SPL transformation development
    - e.g., conflict and dependency analysis, confluence analysis, etc.

[1] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer, Fundamentals of Algebraic Graph Transformation, ser. Monographs in Theoretical Computer Science. Springer, 2006

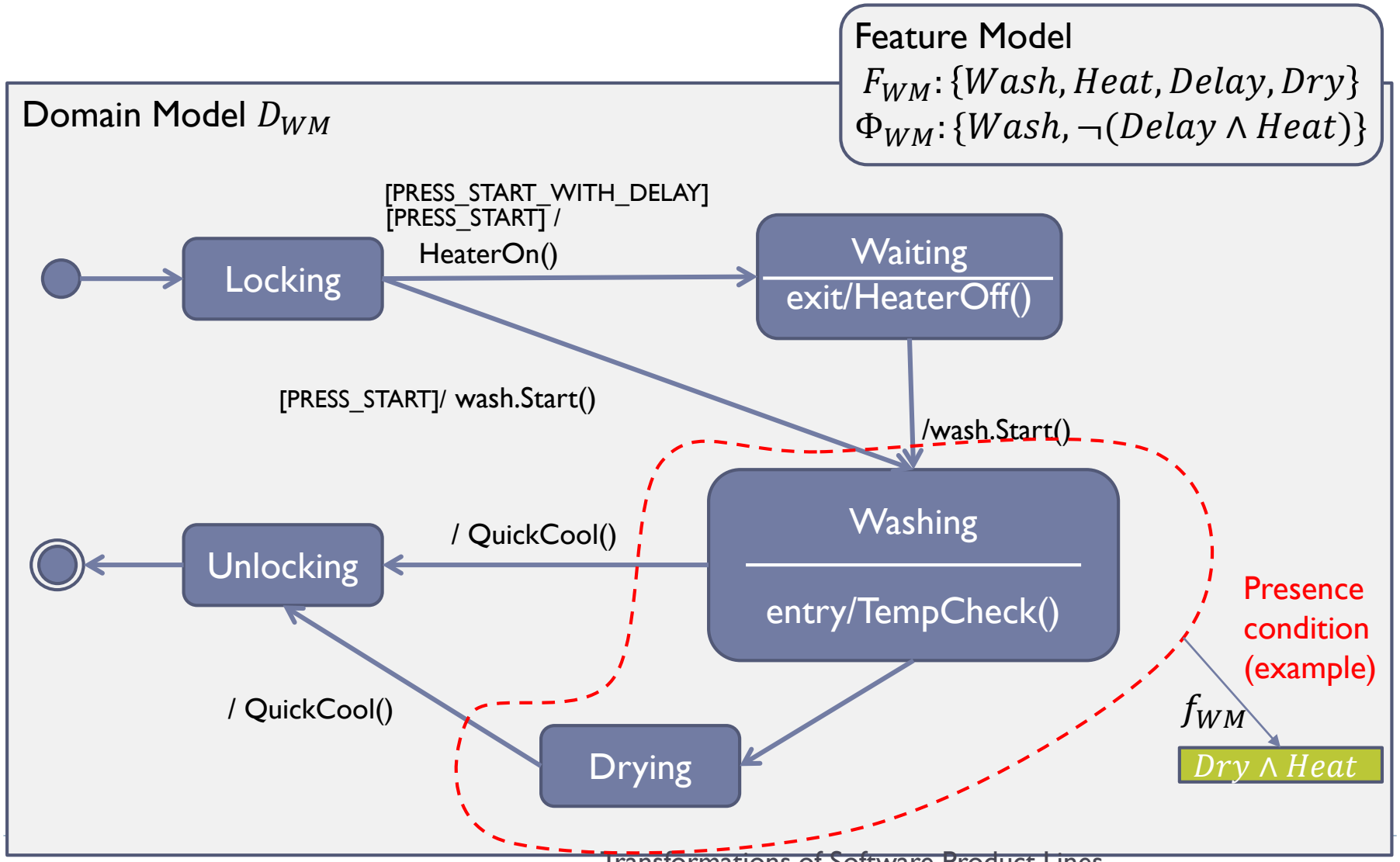
Outline

- Software Product Lines
- Transformations of SPLs
- What is the problem?
- Approach (part I): Category of SPLs
- Approach (part II): SPL Transformations using graph transformation rules
- Summary of results and next steps

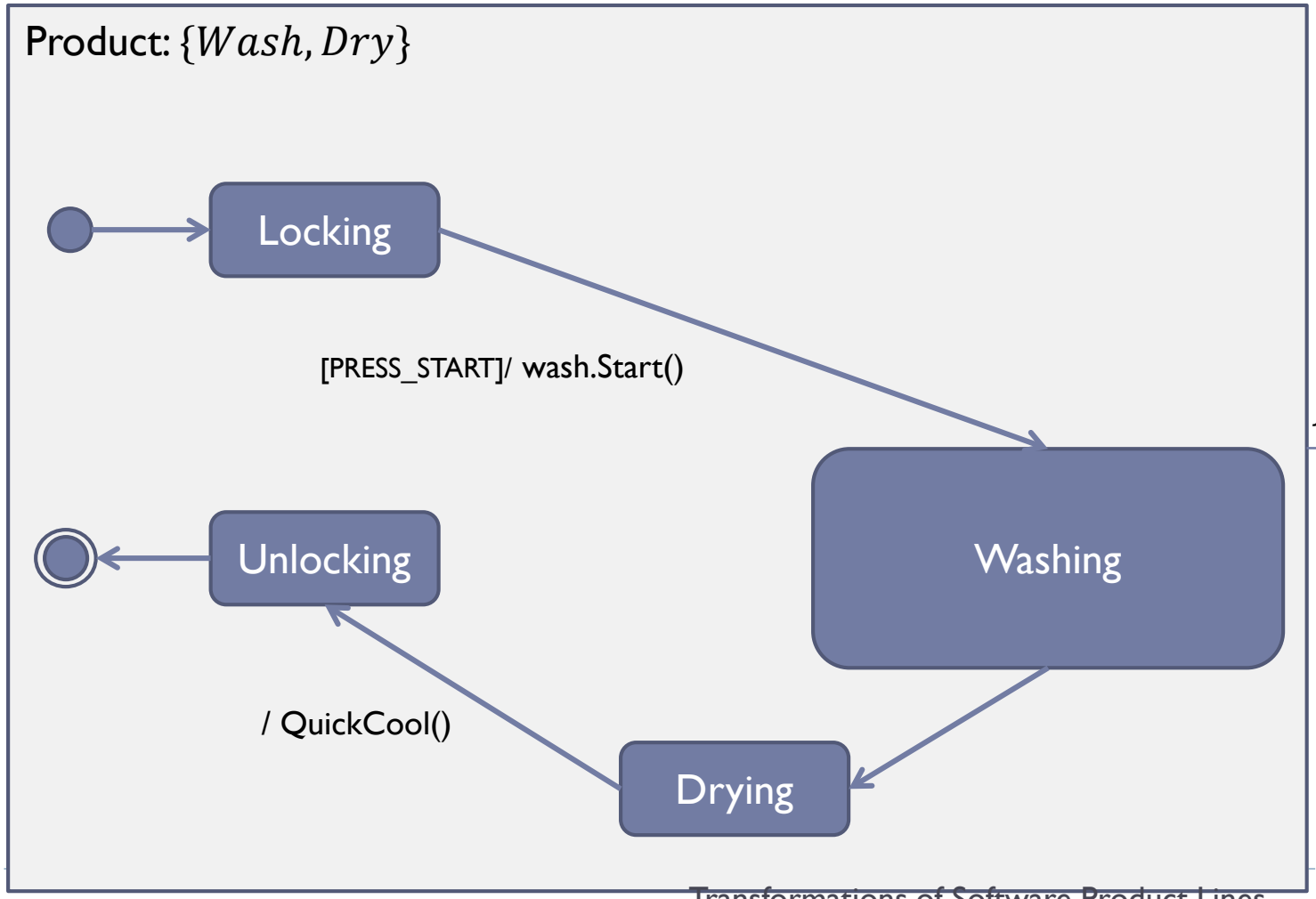
Defining category $PL_{\boldsymbol{Mod}}$ – objects and morphisms

- An SPL $P = (F_P, \Phi_P, M_P, f_P)$ of $PL_{\boldsymbol{Mod}}$ consists of:
  - **(feature model)** Set F_P of features with set Φ_P of propositional feature constraints over F_P defining allowable feature configurations
  - **(domain model)** $\boldsymbol{Mod}$-model M_P
  - **(presence conditions)** Function f_P assigns a propositional formula over F_P to each submodel of M_P defining for which feature configurations the submodel is present
- An SPL morphism $h: P \to Q$ is a mapping from SPL P to Q such that
  - **(feature mapping)** h maps F_P to F_Q
  - **(domain mapping)** h maps D_P to D_Q (using a $\boldsymbol{Mod}$-morphism)
  - above mappings constrained so that products of P map into products of Q

Washing machine SPL WM in PL_{SM}



Example Product: {Wash, Dry}

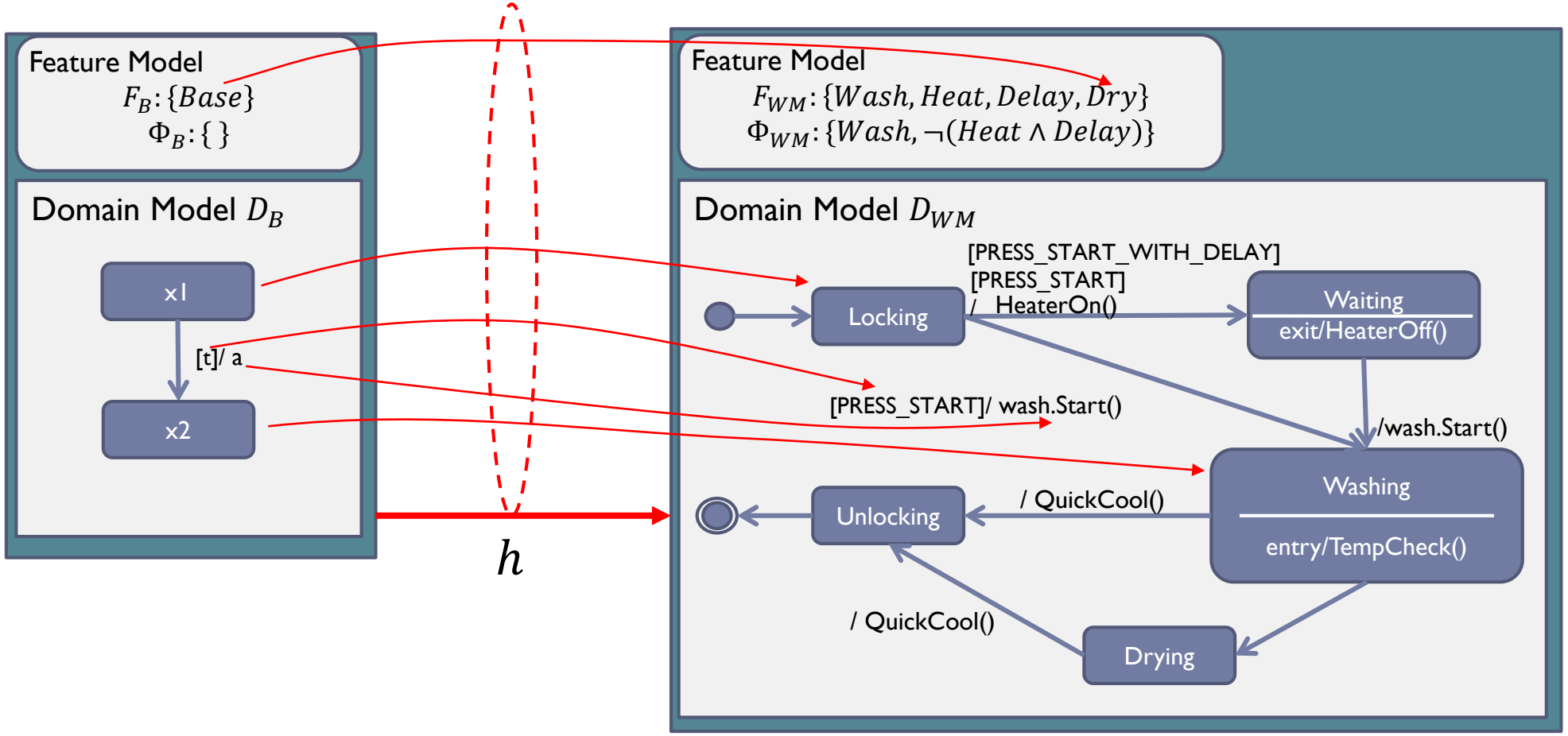


Presence condition (example)

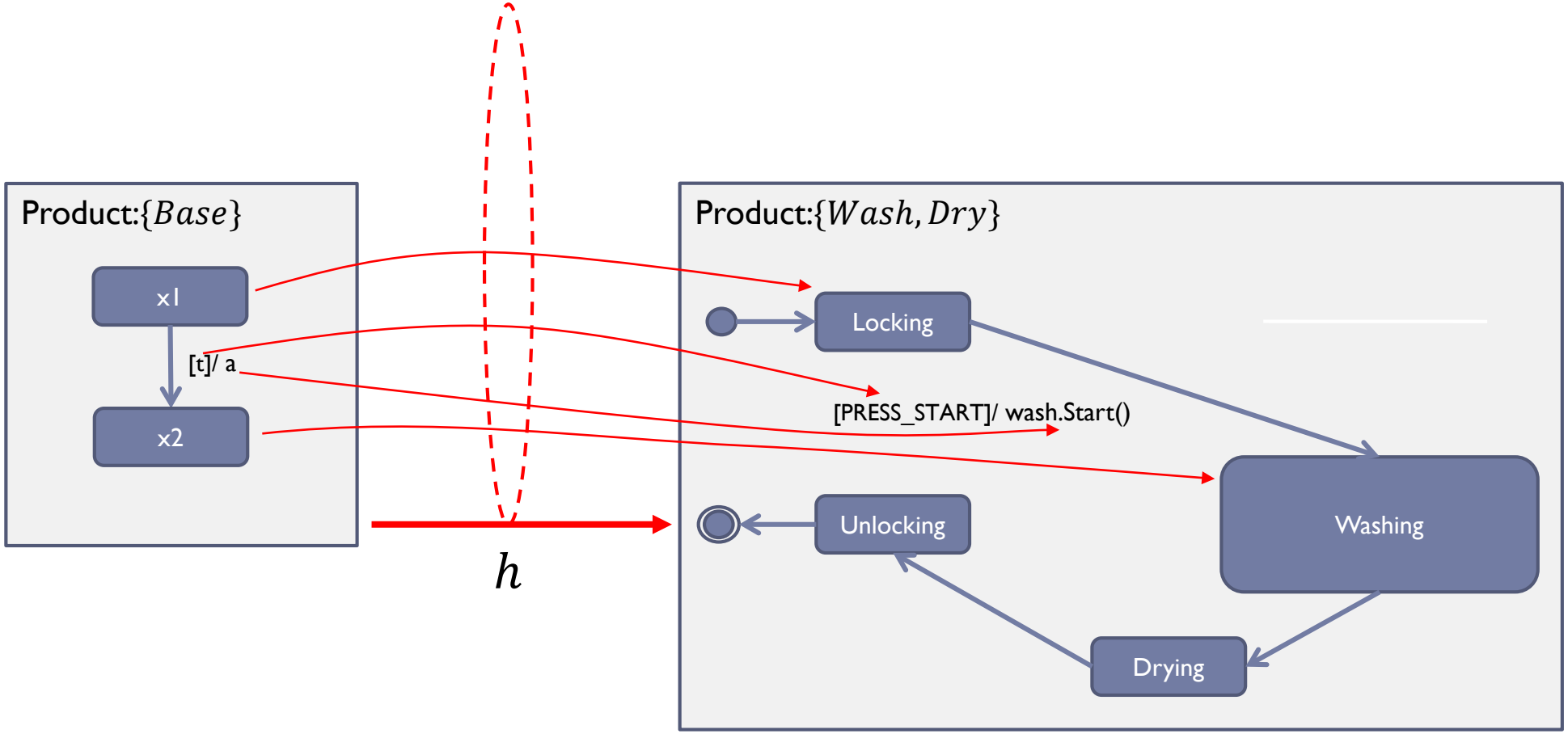
f_{WM} → $Wash \wedge Dry \wedge \neg Heat \wedge \neg Delay$

Products are the maximal submodels with presence condition for a configuration

Morphisms in PL_{SM}



Morphisms map products



Key result: Pushout construction in PL_{Mod}

- In PL_{Mod} we can use the standard category theory pushout construction to combine two SPLs that are related by a common SPL

See Paper for Details!

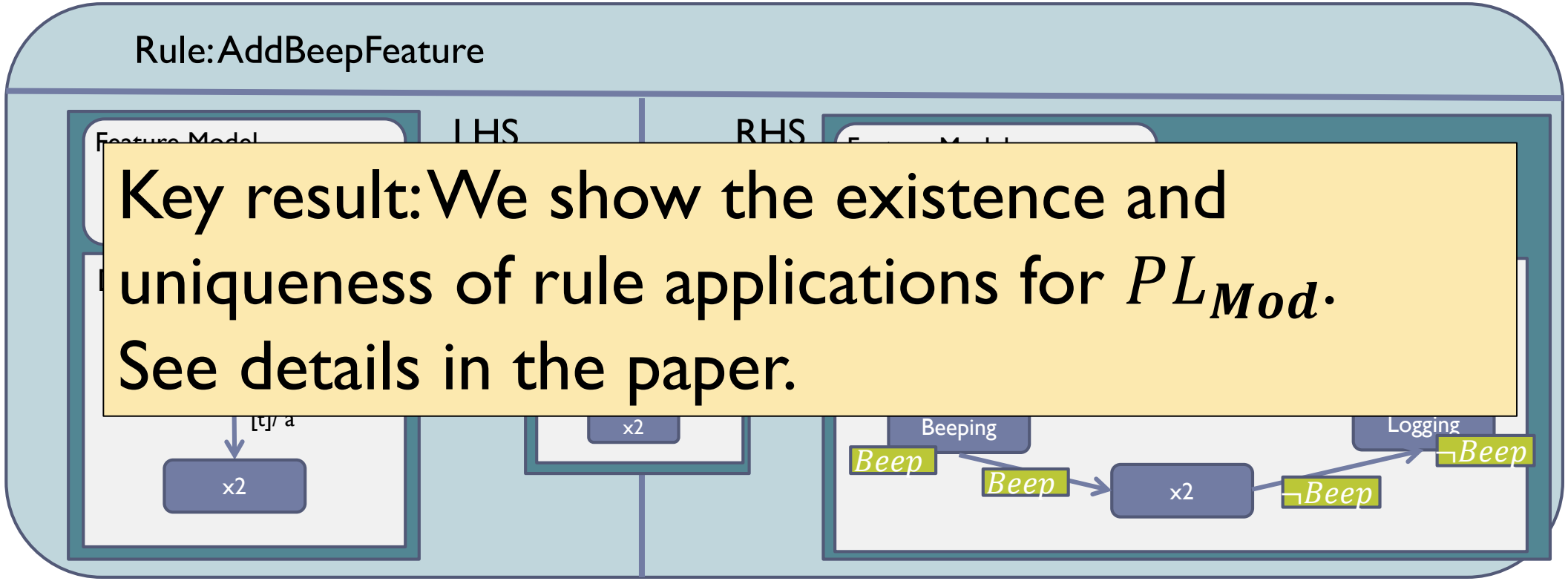
cts of SPL_S are
ucted by combining
ts of SPL_Q and SPL_R
ding to overlap in SPL_P

Outline

- Software Product Lines
- Transformations of SPLs
- What is the problem?
- Approach (part 1): Category of SPLs
- Approach (part II): SPL Transformations using graph transformation rules
- Summary of results and next steps

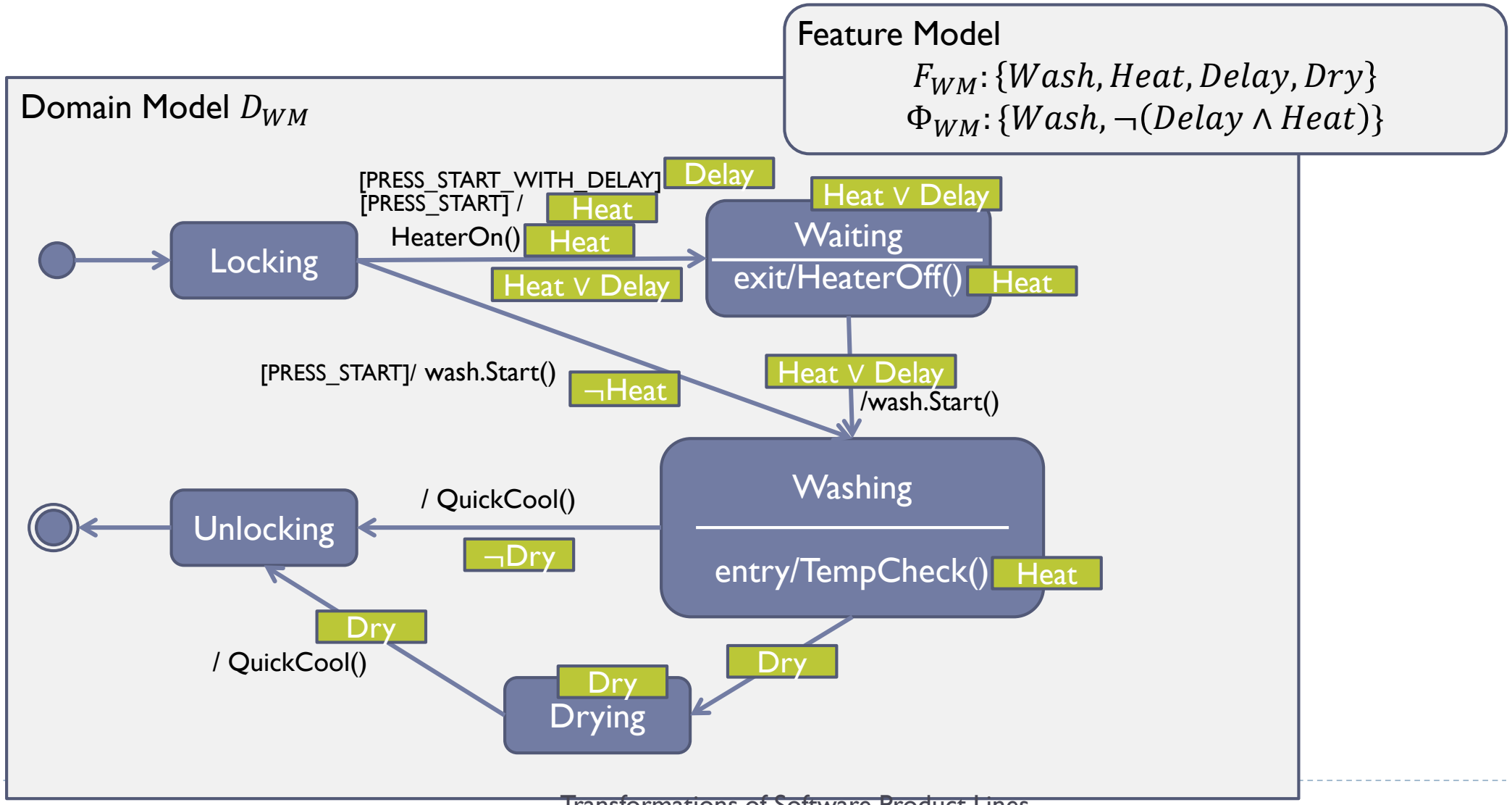
Transformations in PL_{Mod}

- Since we can construct pushouts in PL_{Mod}, we can use the double pushout approach from AGT to define transformation rules.

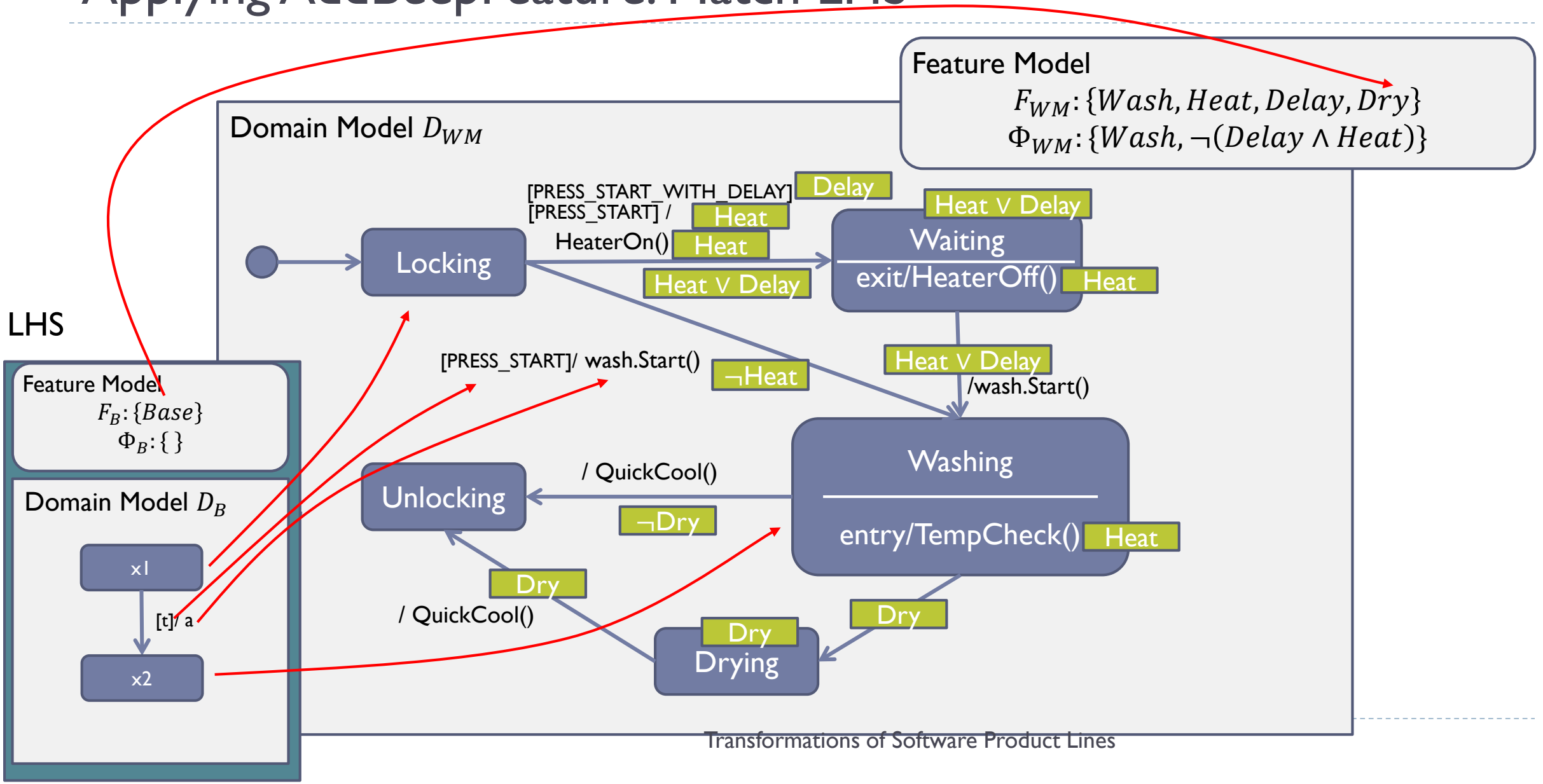


Key result: We show the existence and uniqueness of rule applications for PL_{Mod}. See details in the paper.

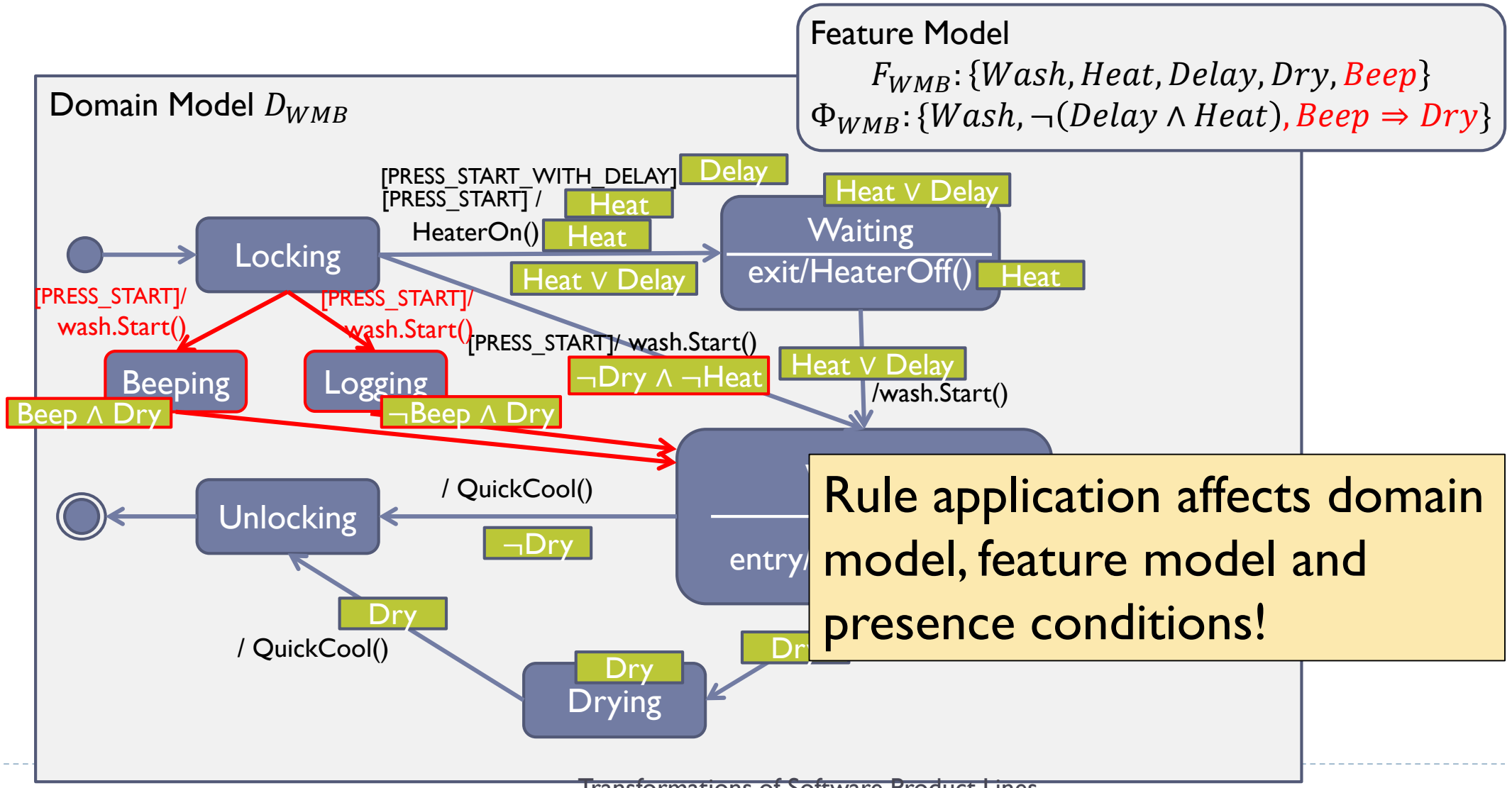
Washing machine SPL WM



Applying AddBeepFeature: Match LHS



Applying AddBeepFeature: Result SPL WMB of double pushout



Summary of Results

- Given any suitable category $\boldsymbol{Mod}$ of models,
  1. Showed how to define the category $PL_{\boldsymbol{Mod}}$ of SPLs having $\boldsymbol{Mod}$ models as domain models.
  2. Defined the pushout construction for $PL_{\boldsymbol{Mod}}$
  3. Showed how to define transformation rules for $PL_{\boldsymbol{Mod}}$ using double pushout
  4. Proved the existence and uniqueness of rule application.
- Illustrated how an SPL rule can affect both feature and domain model parts of an SPL
  - i.e., we have exceeded these limitations in the literature

Next Steps

- Have only partially proven that $PL_{\boldsymbol{Mod}}$ satisfies the formal requirements for AGT
  - We are completing this task
- Plan to implement formal analysis techniques from AGT for PL_{Mod}
  - e.g., conflict and dependency analysis, confluence analysis, etc.
  - Henshin is likely the platform
- Want to explore the scope of SPL transformations expressible using our approach.
- Want to explore the kinds of SPLs obtained by using different model categories for $\boldsymbol{Mod}$

Questions