

Language Support for Feature-Oriented Product Line Engineering



Wonseok Chae

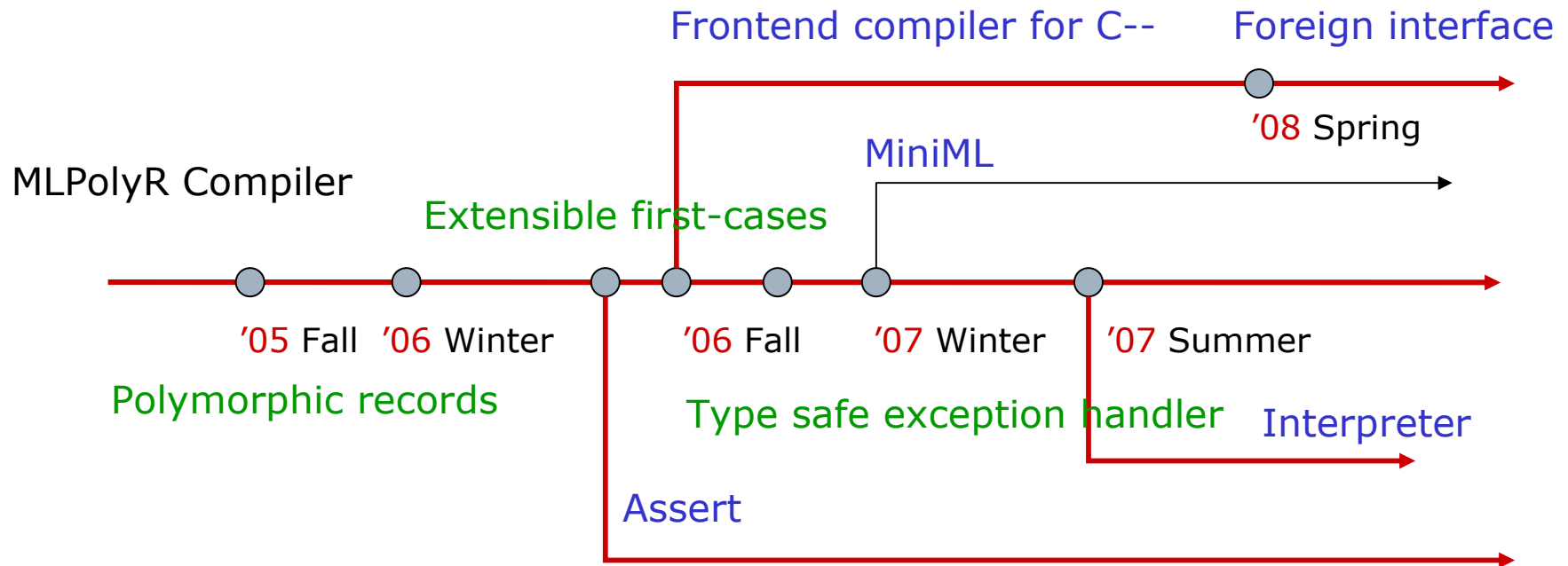
Toyota Technological Institute at Chicago

Matthias Blume

Google, Inc.

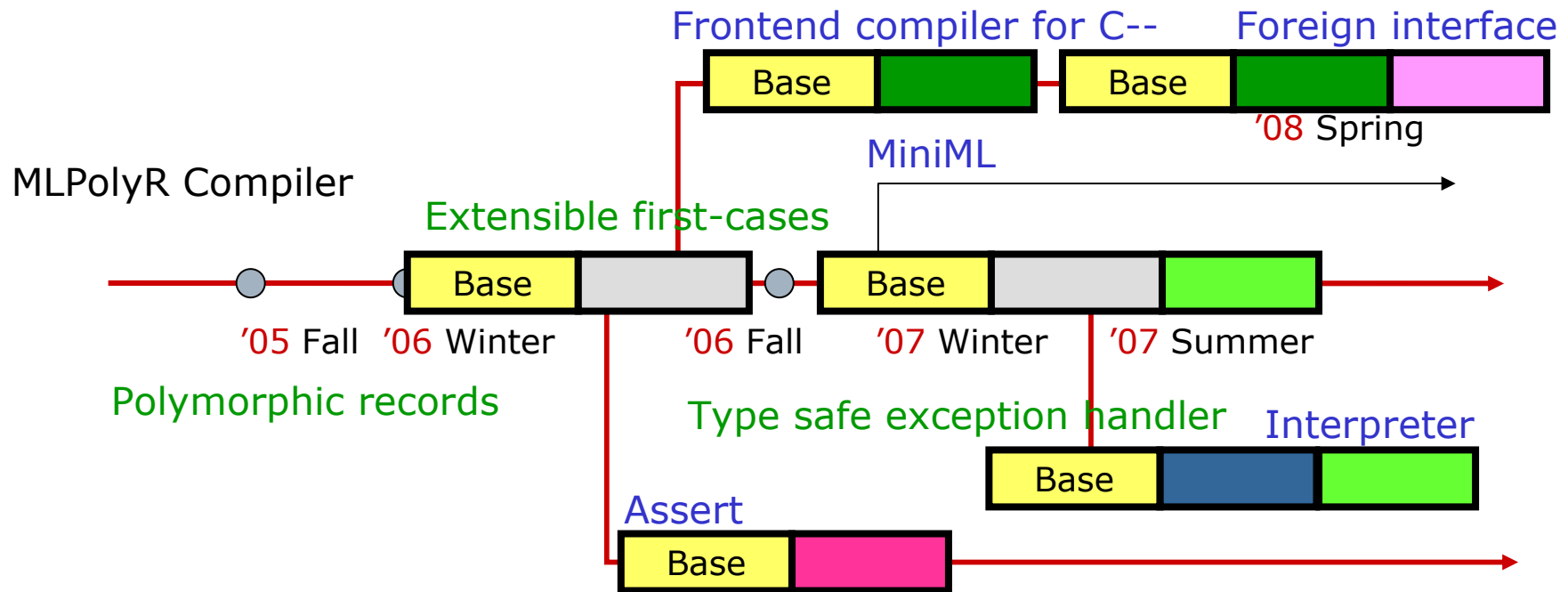
Workshop on Feature-Oriented Software
Development (FOSD'09), October 6, 2009, Denver.

Motivation (1)



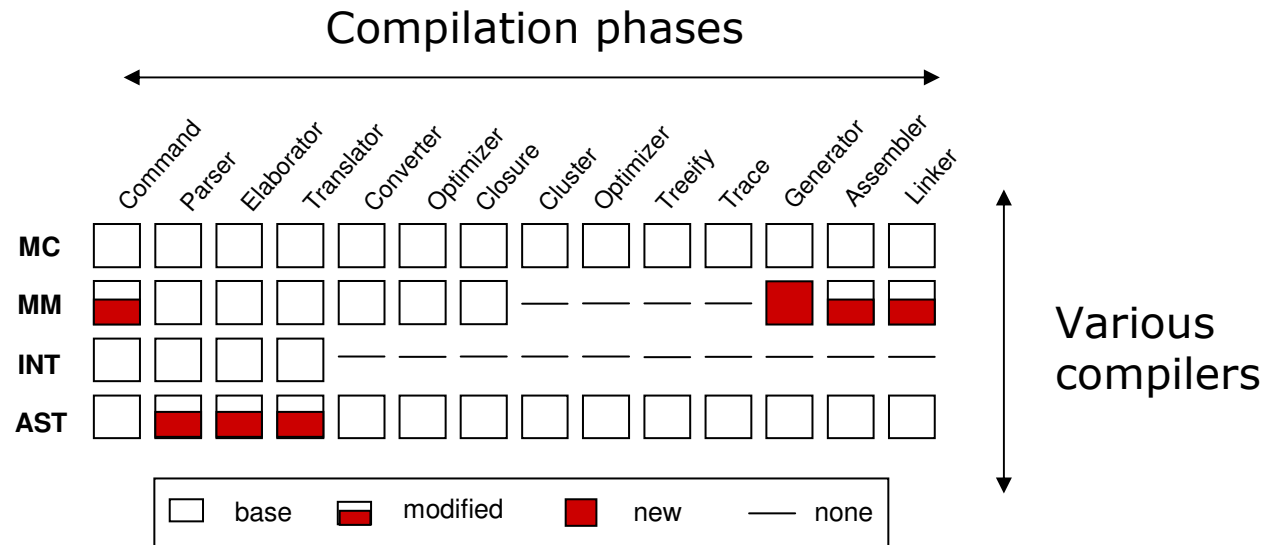
From SPLC'09

Motivation (2)

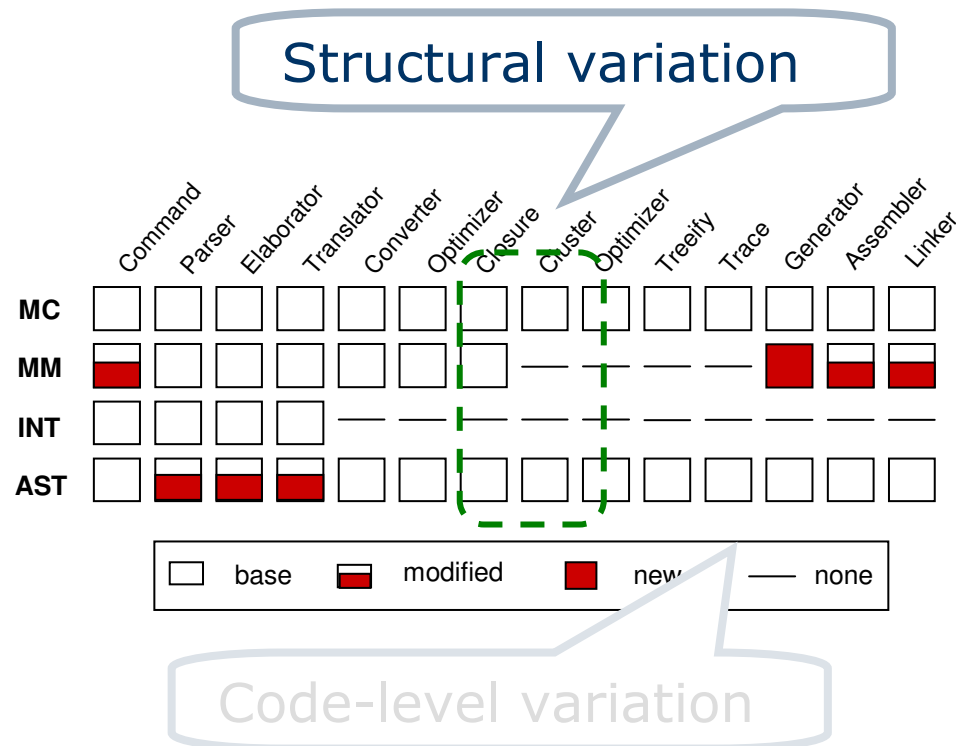


From SPLC'09

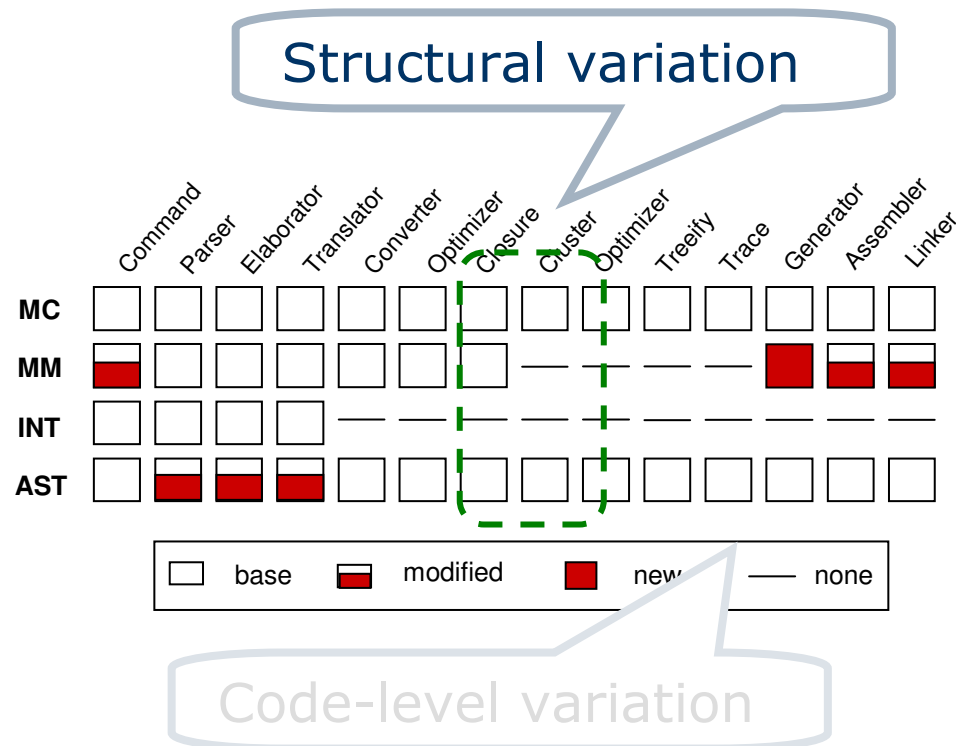
Product line Analysis



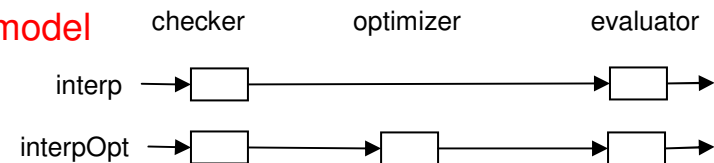
Product line analysis



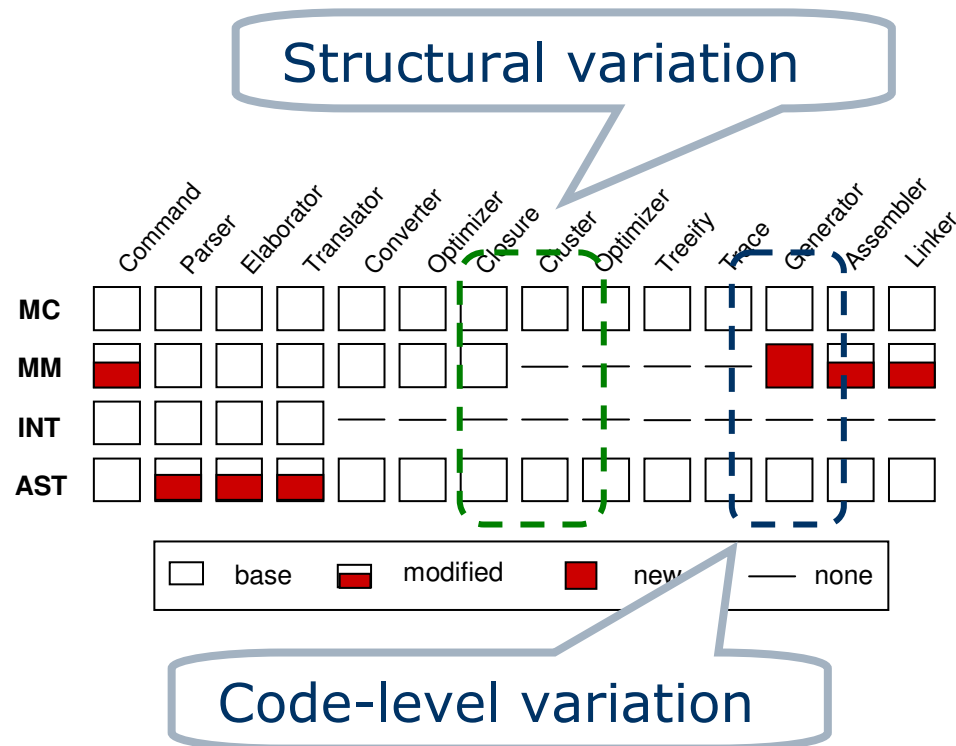
Product line architecture model



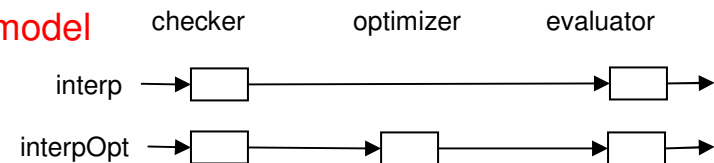
Architecture model



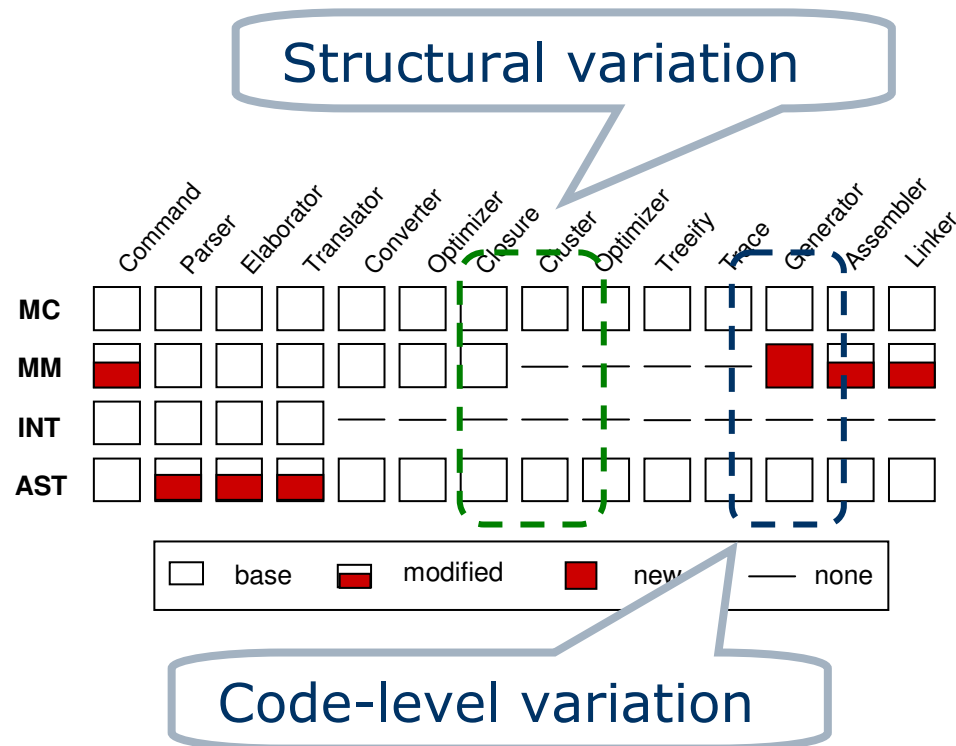
Product line analysis, cond.



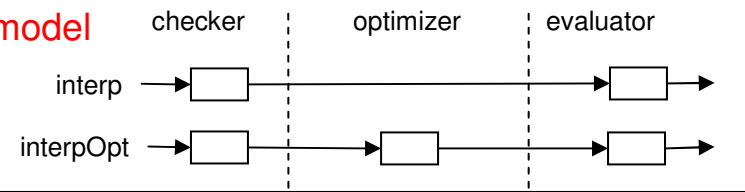
Architecture model



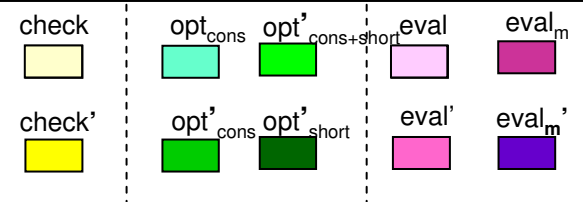
Product line component model



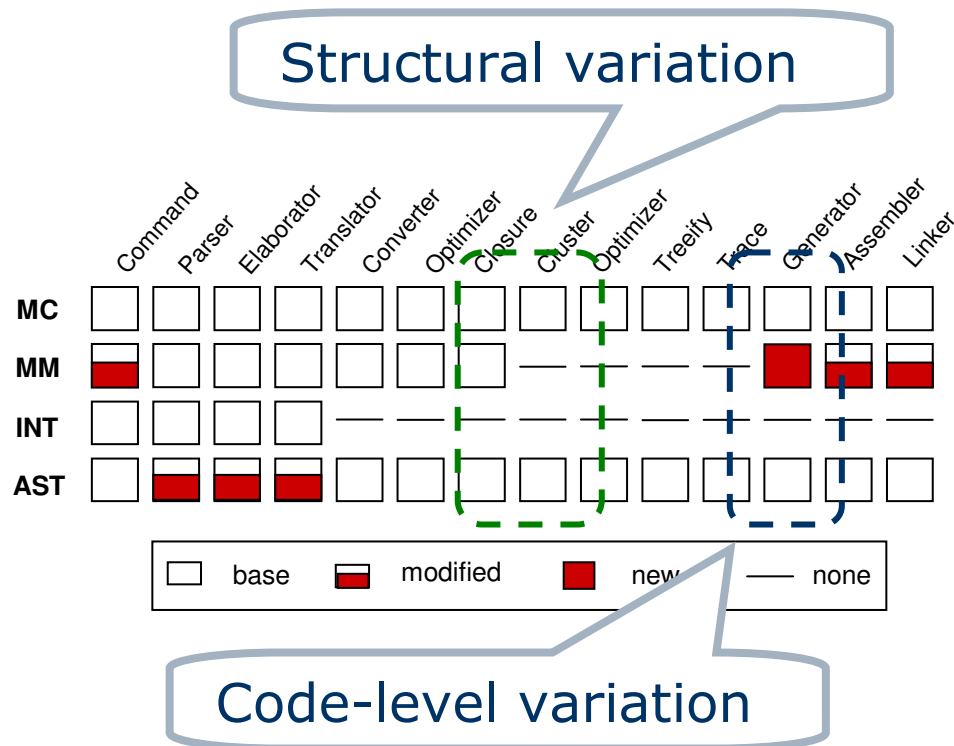
Architecture model



Component model

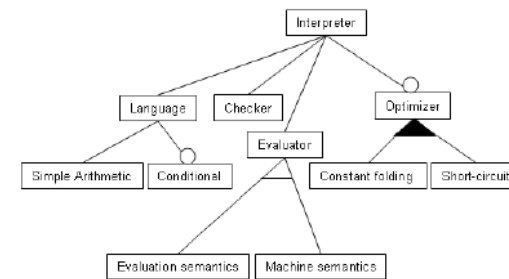


Feature model

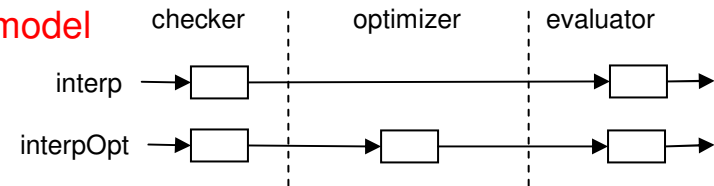


PLE core assets

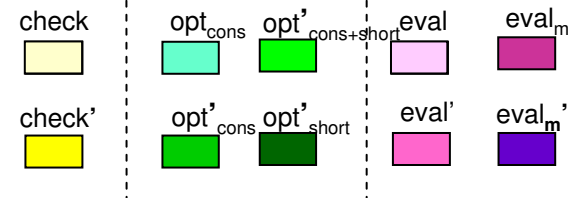
Feature model



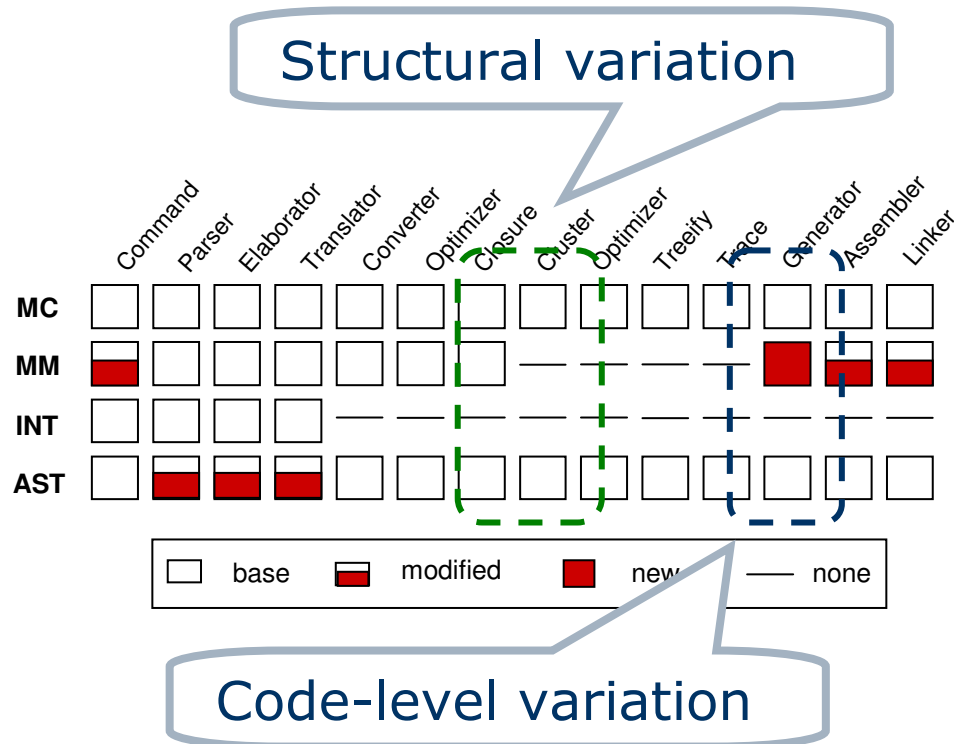
Architecture model



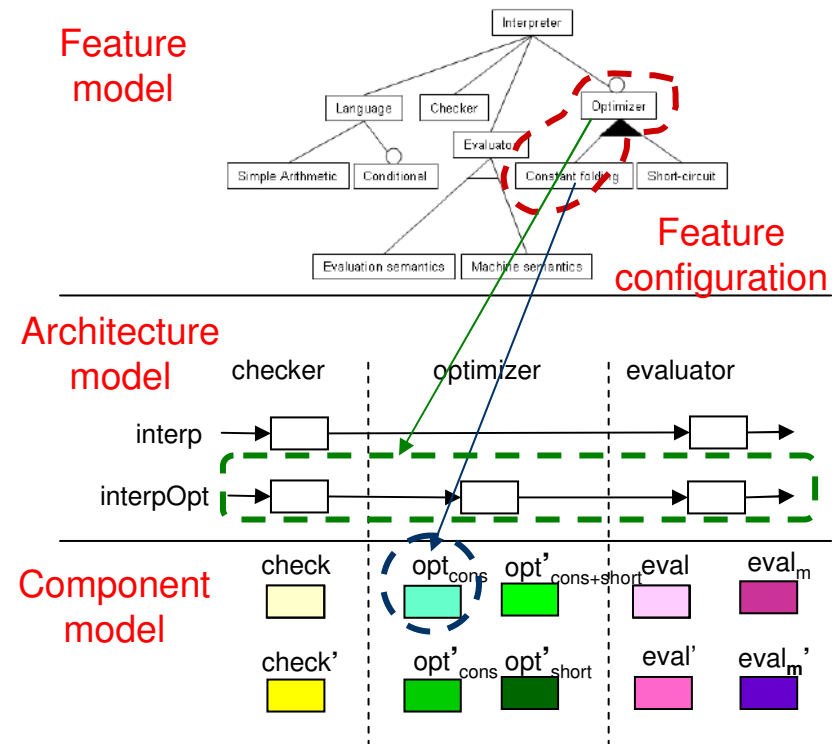
Component model



Feature configuration



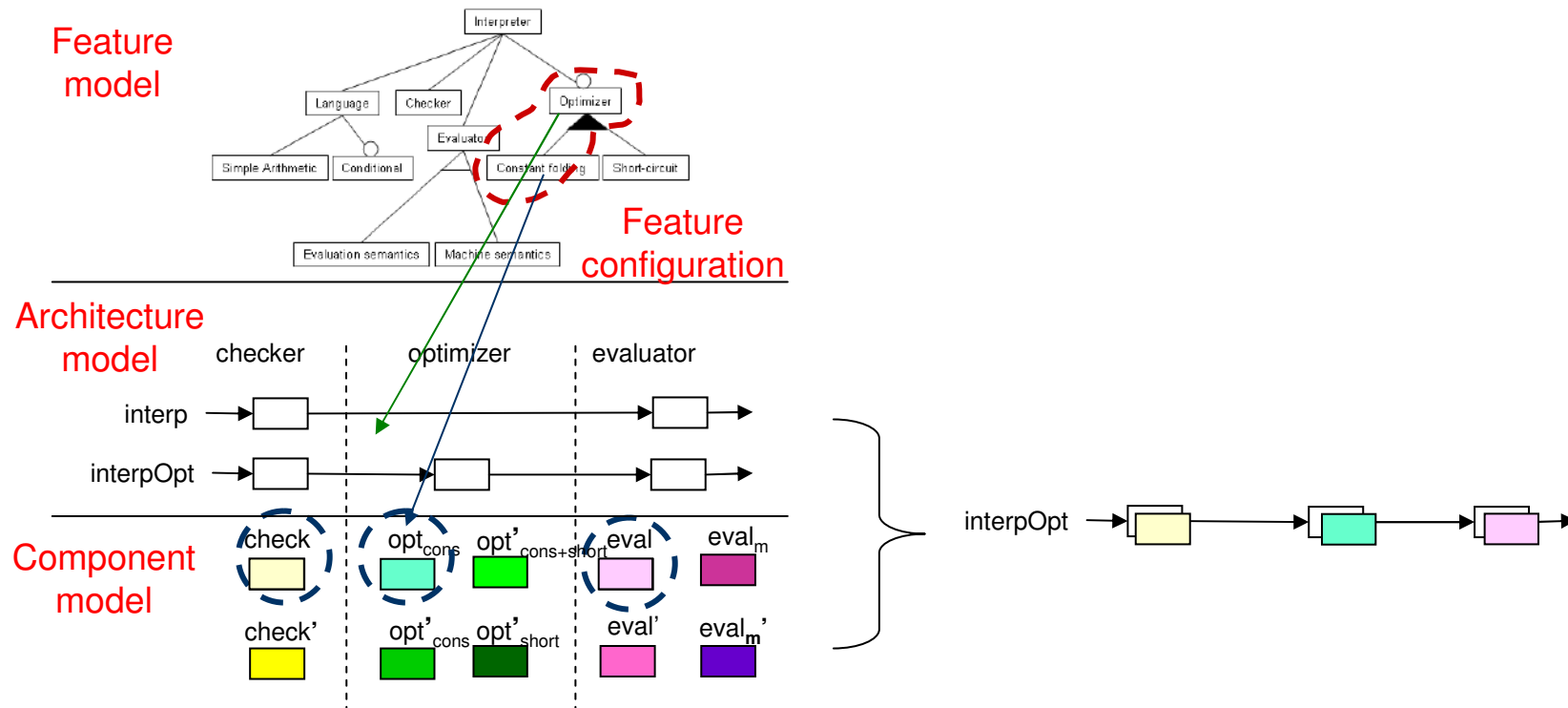
PLE core assets



Product development

PLE core assets

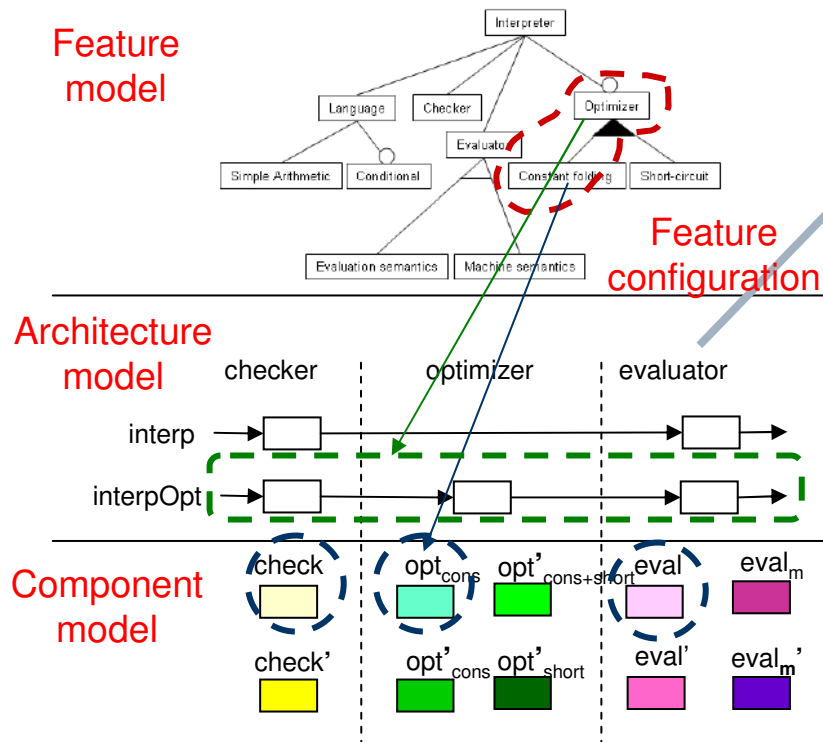
Application engineering



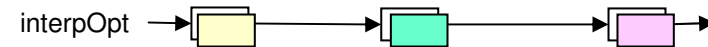
Parameterization programming

PLE core assets

Language support



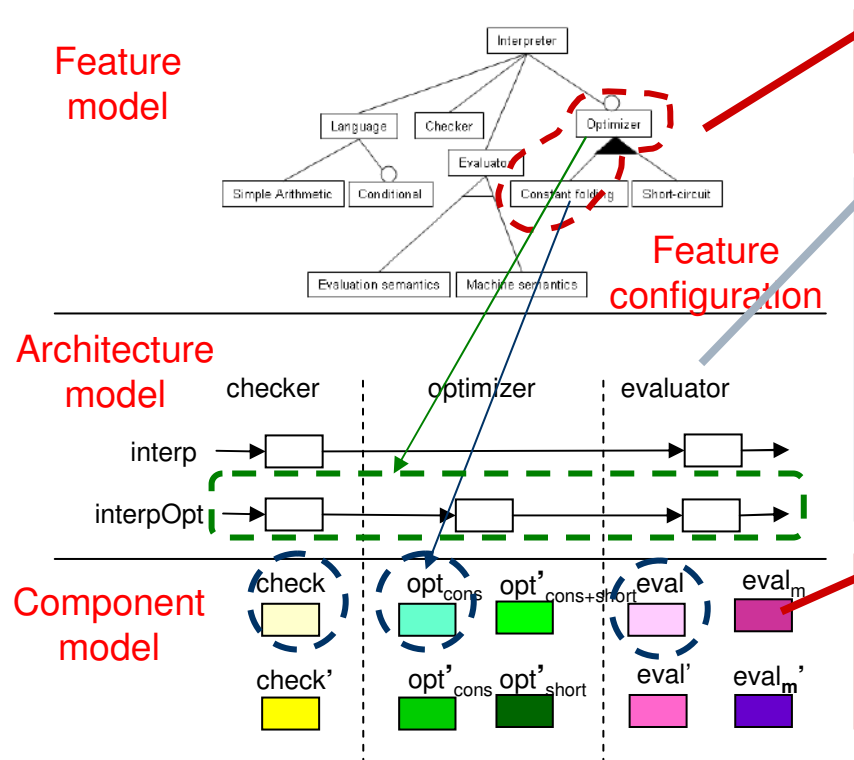
The parameterized module system of **Standard ML** is powerful enough to manage structural variations.



Limitations

PLE core assets

Language support



Implicit mapping relations make it difficult to generate a product.

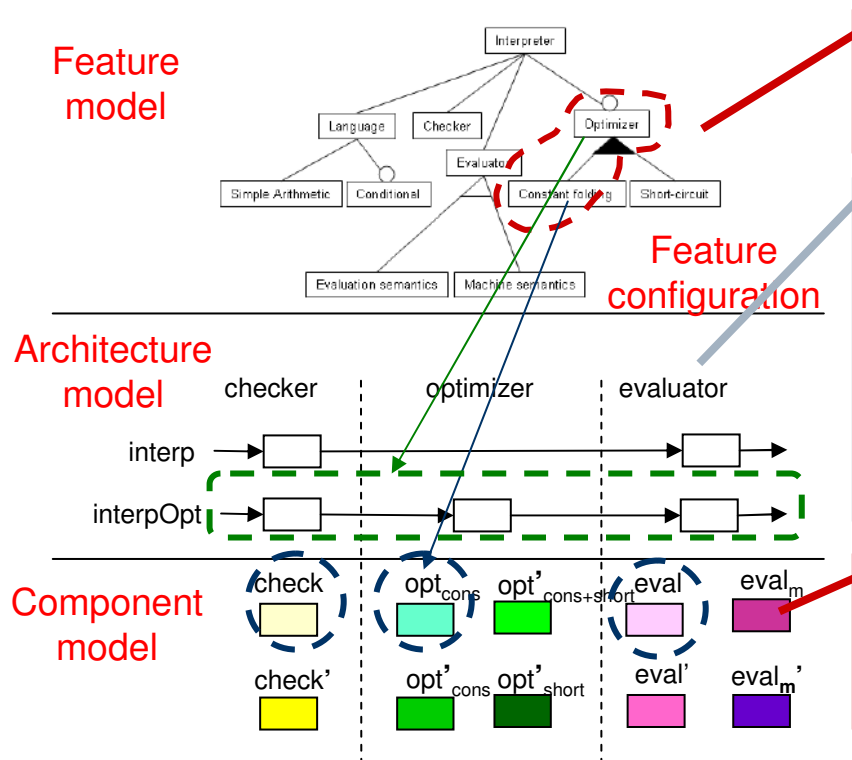
The parameterized module system of **Standard ML** is powerful enough to manage structural variations. Sometimes, however, its type system is too restrictive.

Code extensions are not properly supported.

Language support

PLE core assets

Language support



Annotative approach

Parameterization approach

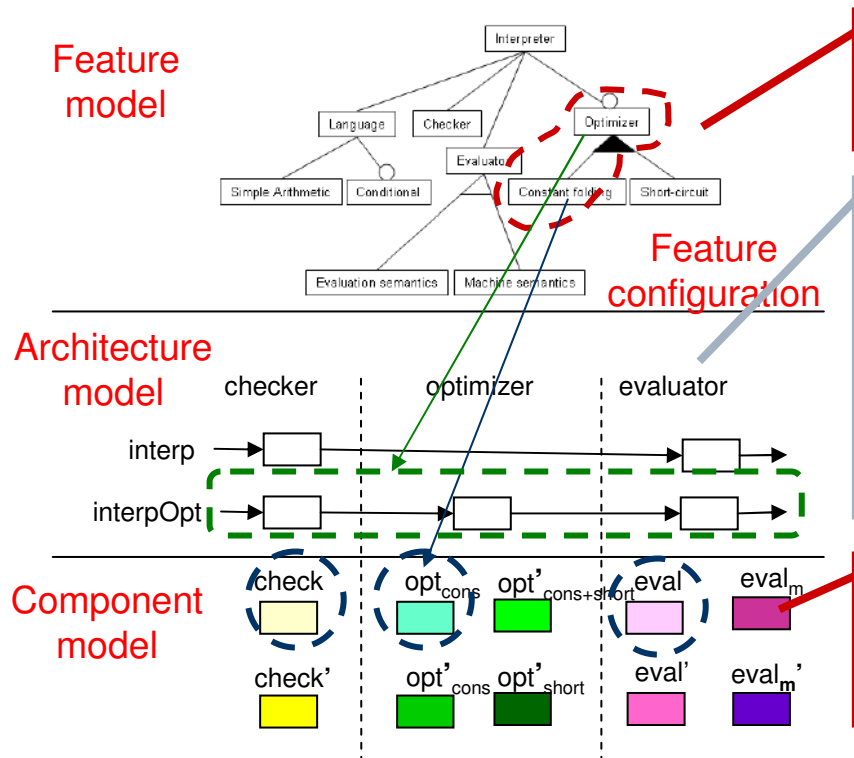
Compositional approach

Implementing a language for a product line is a difficult task. The parameterized module system of Standard ML is powerful enough to model a wide range of languages. Sometimes, however, its type system is too restrictive. Code reuse is not properly supported.

The MLPolyR language

PLE core assets

Language support



Annotative approach

Macro system

Parameterization approach

Parameterized, extensible module

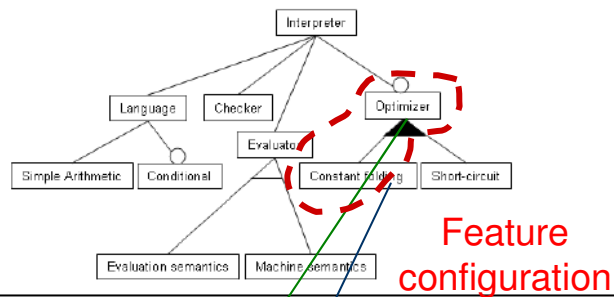
Compositional approach

Type-safe extensible programming

The MLPolyR Language Second.

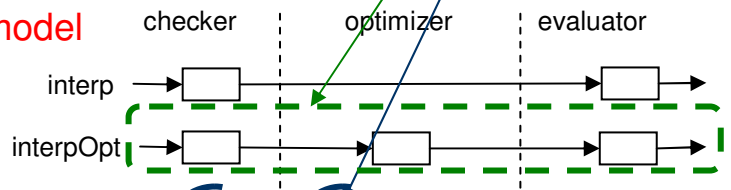
PLE core assets

Feature model

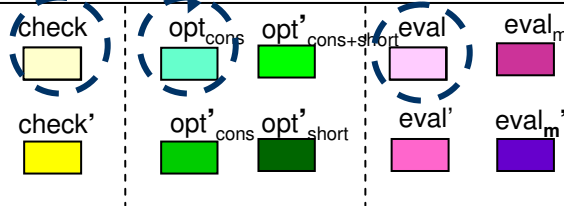


Feature configuration

Architecture model



Component model



Architecture

```
template InterFun (C, E) = {{...}}
```

```
template InterOptFun (C, O, E) = {{
  val interp = fn e =>
    E.eval (C.check e)
}}
```

Compositional approach

Component

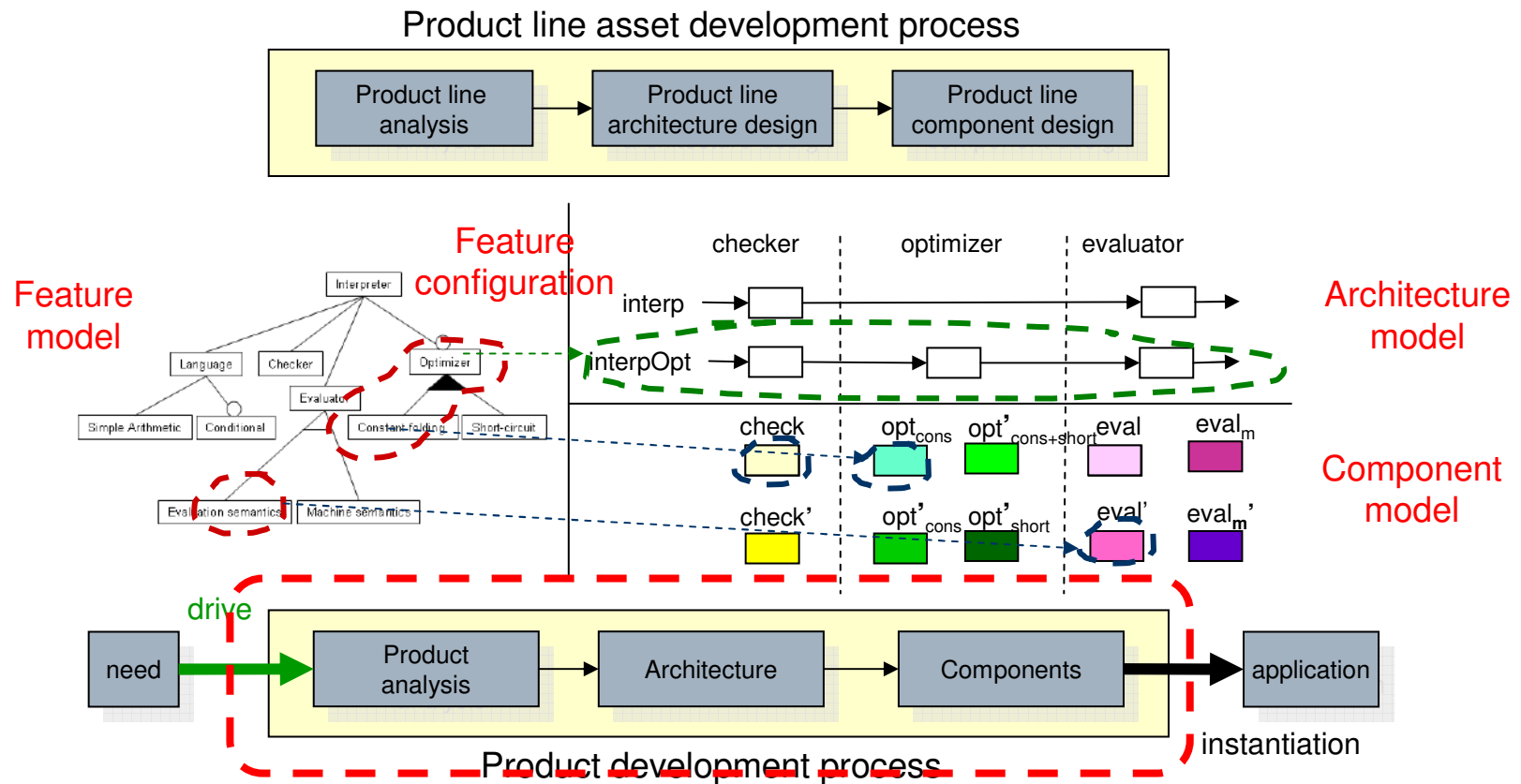
```
module BigStep ... { val eval = ... }
module EBigStep ... { val eval = ... }
```

Annotative approach

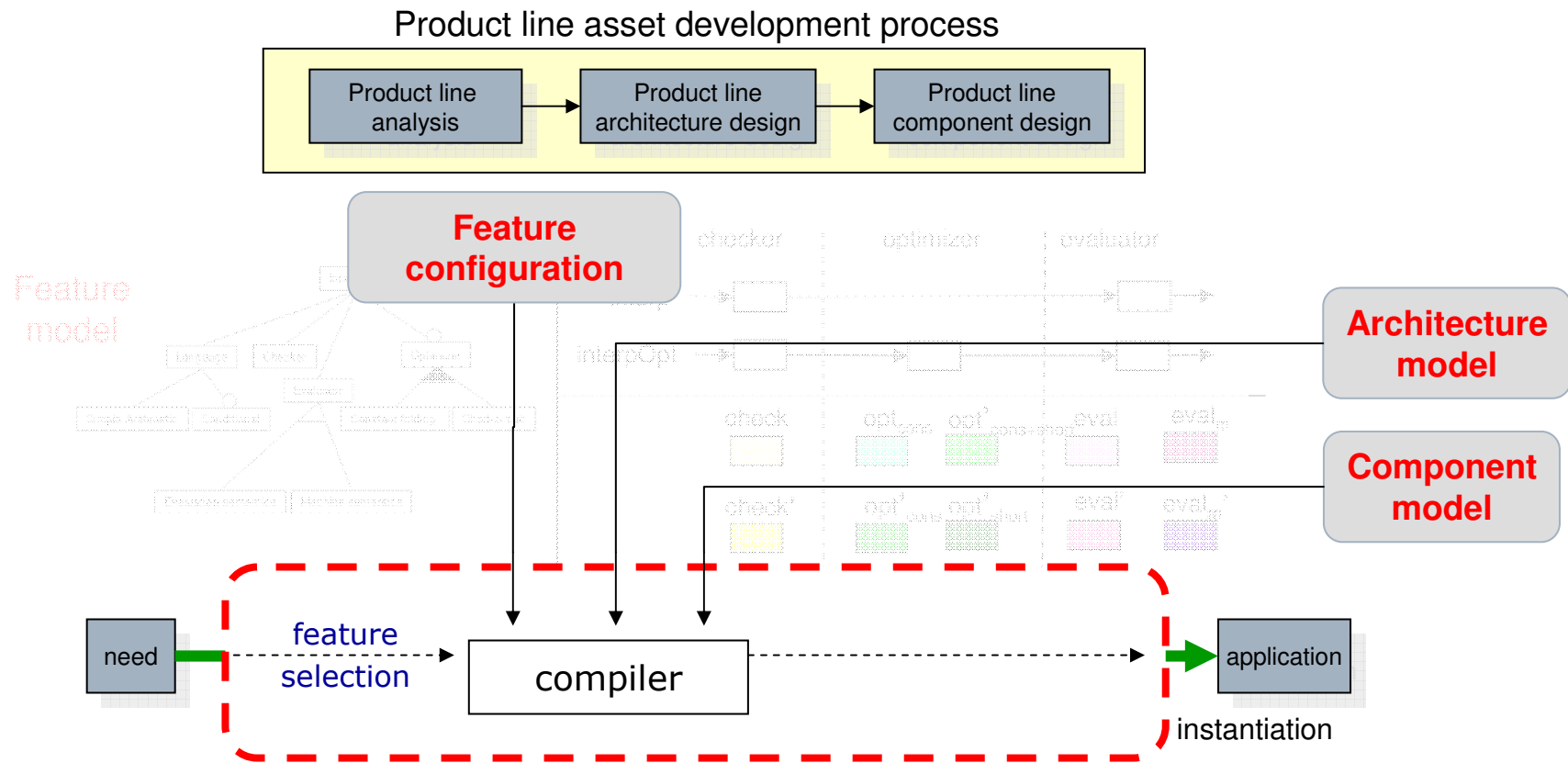
Feature configuration

```
I ::= InterpFun (C, E) {}
  | InterOptFun (C, O, E) { Optimizer }
```

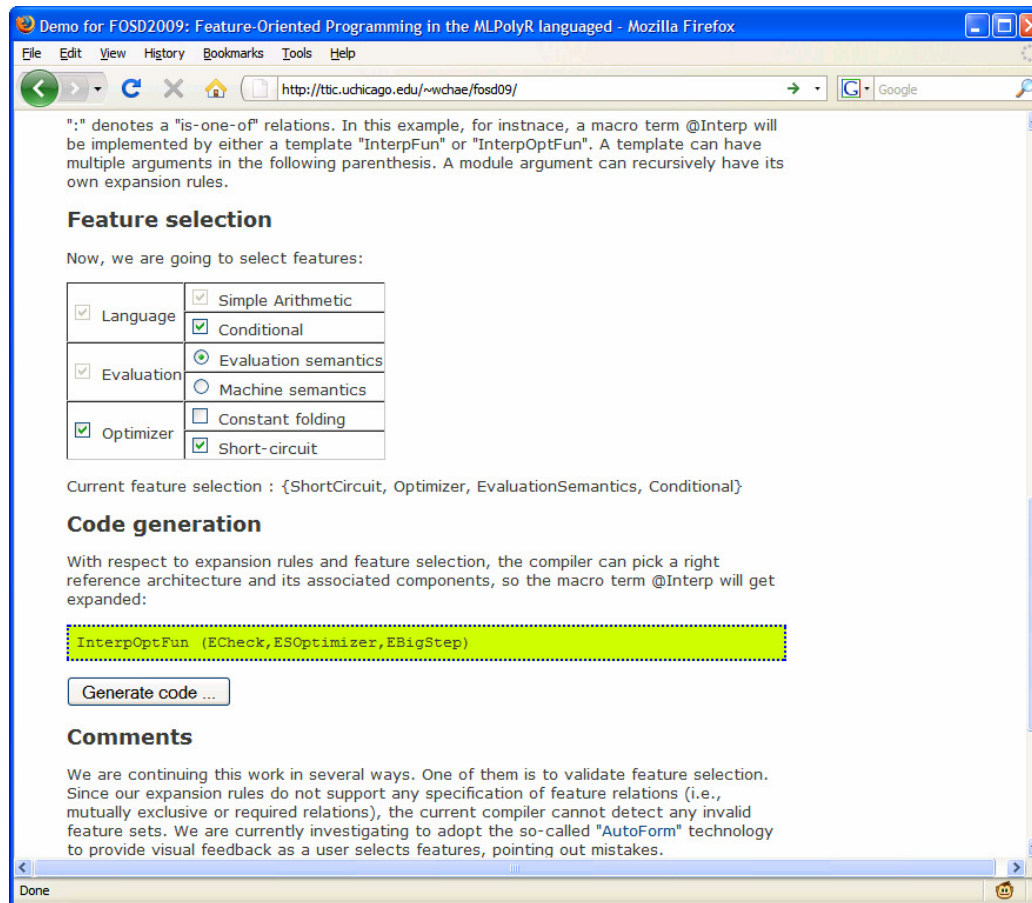
Product development process



Product development process



Demo



Related Work

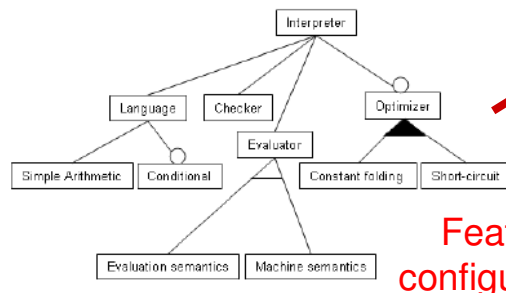
- The annotative approach
 - FORM macro language, #ifdef
- The compositional approach
 - Aspects, stepwise refinements (AHEAD)
- Parameterization approach
 - Functor (SML), Template (C++), Generics (Java)
- Hybrid approach
 - FeatureC++, CIDE, MLPolyR

Future work

PLE core assets

Language support

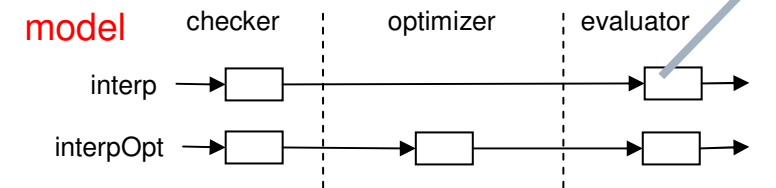
Feature model



Integrate a **feature modeling tool** as a front-end for our compiler.

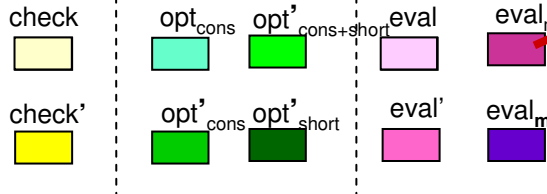
Feature configuration

Architecture model



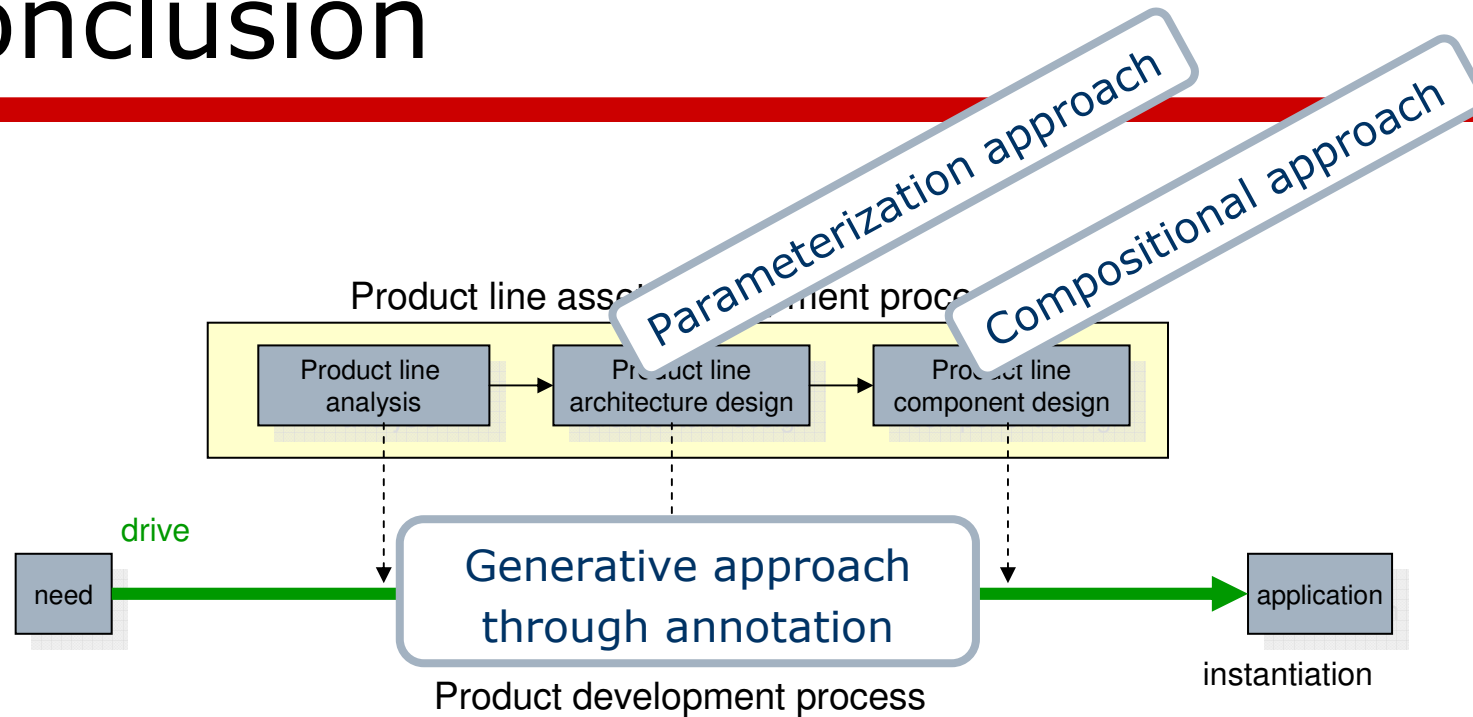
Improve **architectural expressiveness** to support various viewpoints (e.g., process, tasks)

Component model



Support AHEAD-like **stepwise refinements**.

Conclusion



Feature-oriented software development can be *efficiently* delivered by *proper* language supports.