Feature-Oriented Requirements Modeling and Analysis

Jo Atlee
Shoham Ben-David
Pourya Shaker
Sandy Beidu
David Dietrich
Xiaoni Lai

David R. Cheriton School of Computer Science
University of Waterloo

Super cruise

EN-V

Driver Assist Package

- Rear Automatic Braking
- Full-Speed Range Adaptive Cruise Control
- Intelligent Brake Assist
- Forward Collision Alert
- Safety Alert Seat
- Automatic Collision Preparation
- Lane Departure Warning
- Side Blind Zone Alert
- Rear Cross Traffic Alert
- Adaptive Forward Lighting
- …

(semi) autonomous vehicles – many features

many features

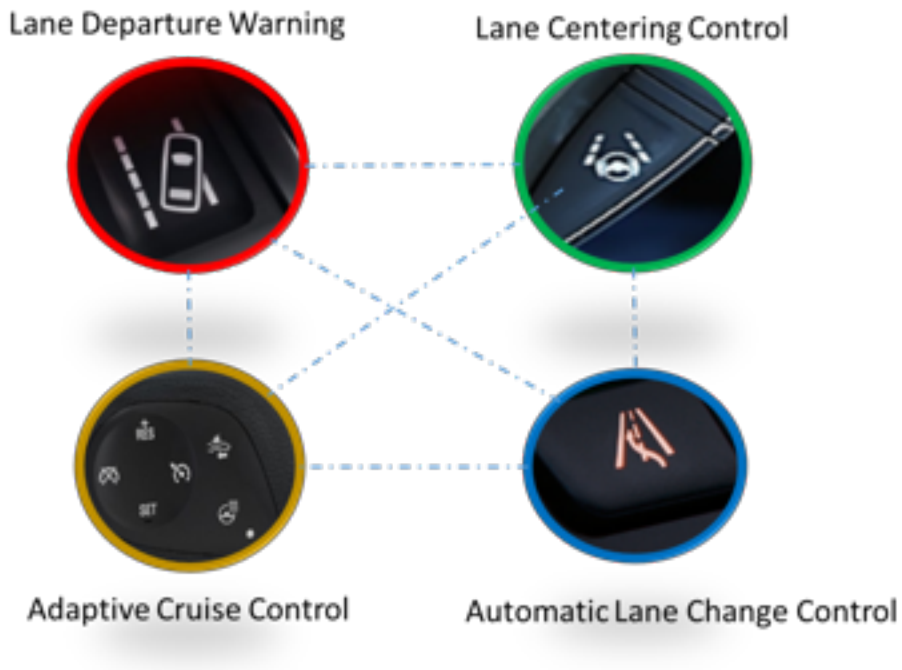
excited user

hmmm … so many
feature interactions

overwhelmed engineer

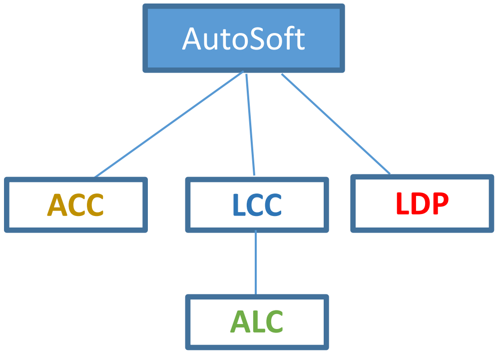
feature-oriented software development

feature: a unit of functionality or added value in the product



product lines

feature model = valid configurations



feature interactions

feature interaction: features influence each other in defining overall system behaviour

- conflicts over shared context
- violations of global correctness properties
- emergent behaviours

feature interaction problem: the number of potential interactions is exponential in the number of features

in this presentation

modelling feature requirements

- feature modularity
- modelling intended interactions

composing features

- to obtain a product-line model

analyzing product-line model

- strategies to scale analysis
- properties for detecting feature interaction

FORML: feature-oriented requirements modelling language

world model

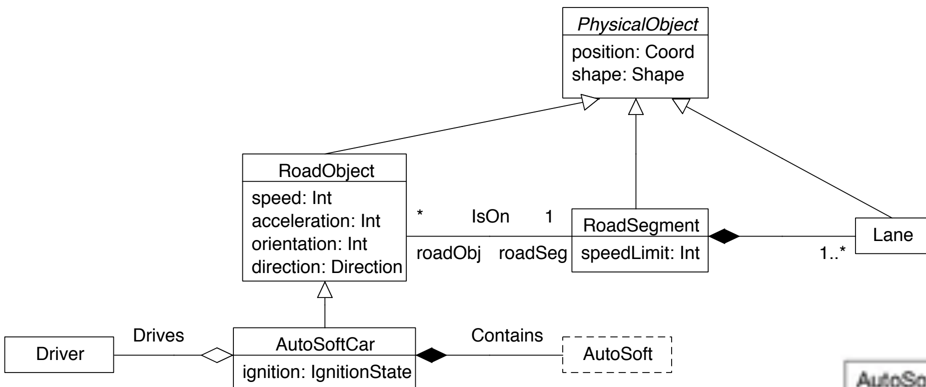
a conceptual model of the problem world

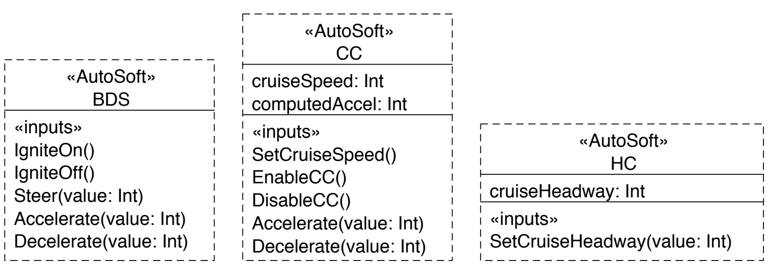
- defines possible world states
- includes feature phenomena
- and feature model

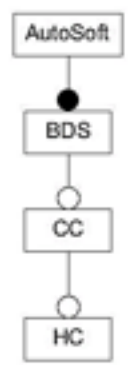
behaviour model

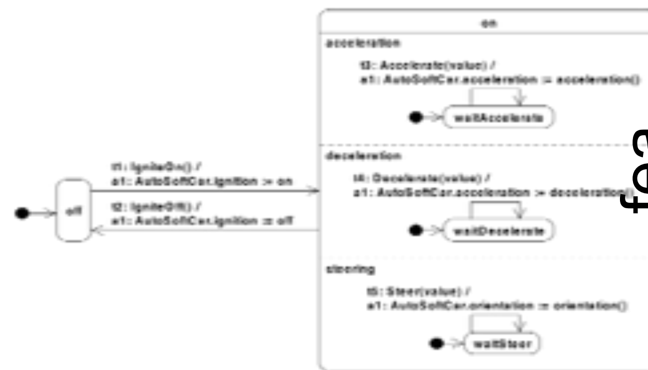
state-machine models (of features)

- whose events, conditions and actions are expressions over world phenomena
- and over feature phenomena

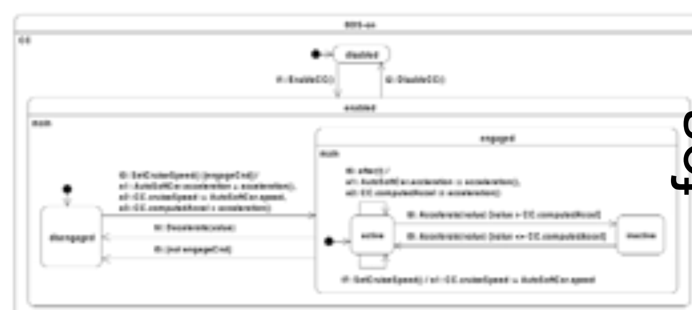








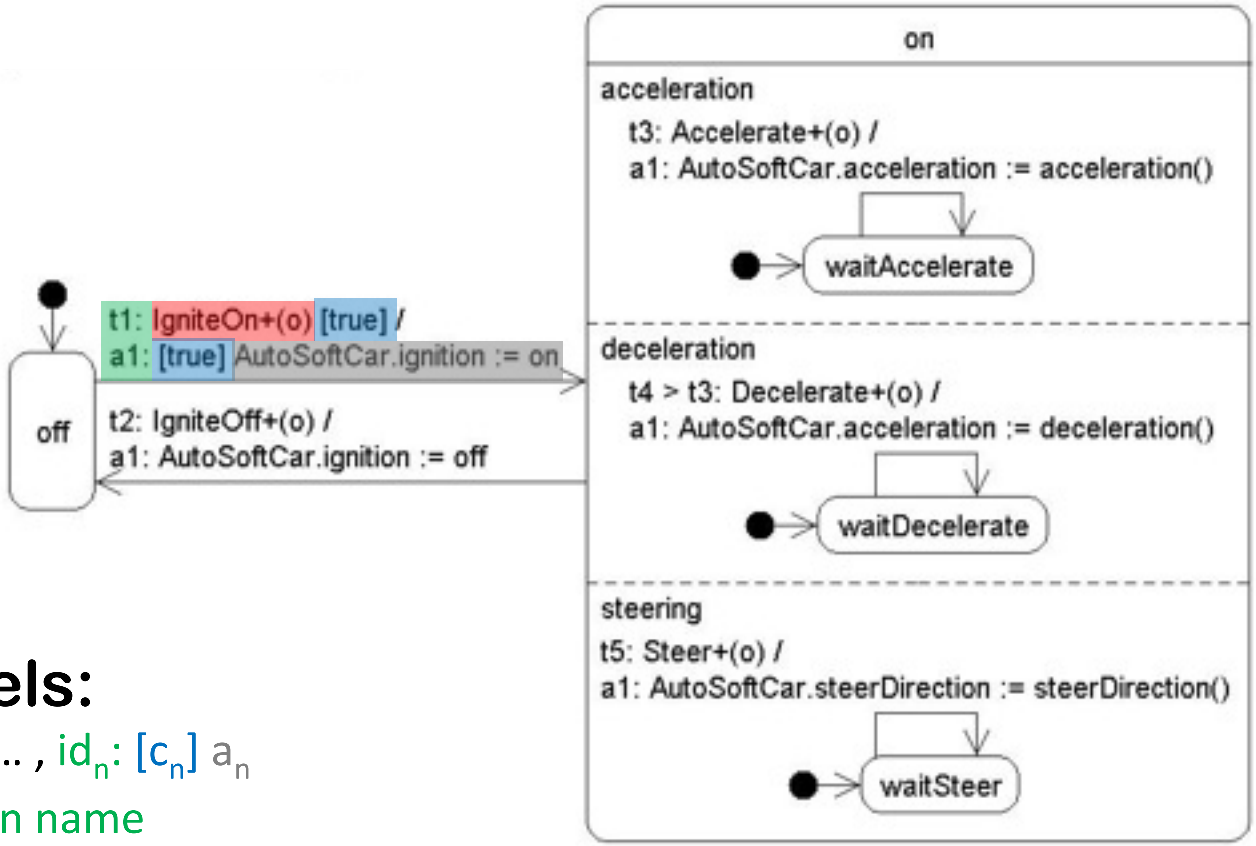
feature



feature

modelling features

features are modelled as hierarchical state machines that sense and control the world



transition labels:

id: e [c] / id_1: [c_1] a_1, … , id_n: [c_n] a_n

- transition or action name
- triggering event: a change in the world
- guard condition: predicate over the world
- action: a prescribed change to the world

a new feature may…

introduce behaviours

- **via:** new machines

eliminate behaviours

- **via:** new or stronger enabling conditions on existing actions or transitions

substitute behaviours

- **via:** new *pre-empting* actions or transitions

a new feature may...

introduce behaviours

- **via:** new machines

eliminate behaviours

- **via:** new or stronger enabling conditions on existing actions or transitions

substitute behaviours

- **via:** new *pre-empting* actions or transitions

intended interactions:

modelled as structural extensions at extension points in existing features

a new feature may...

introduce behaviours

- **via:** new machines

can also be expressed as extensions to existing features:
new regions, new states,
new transitions,
weakened enabling conditions

eliminate behaviours

- **via:** new or stronger enabling conditions on existing actions or transitions

substitute behaviours

- **via:** new *pre-empting* actions or transitions

intended interactions:
modelled as structural extensions at extension points in existing features

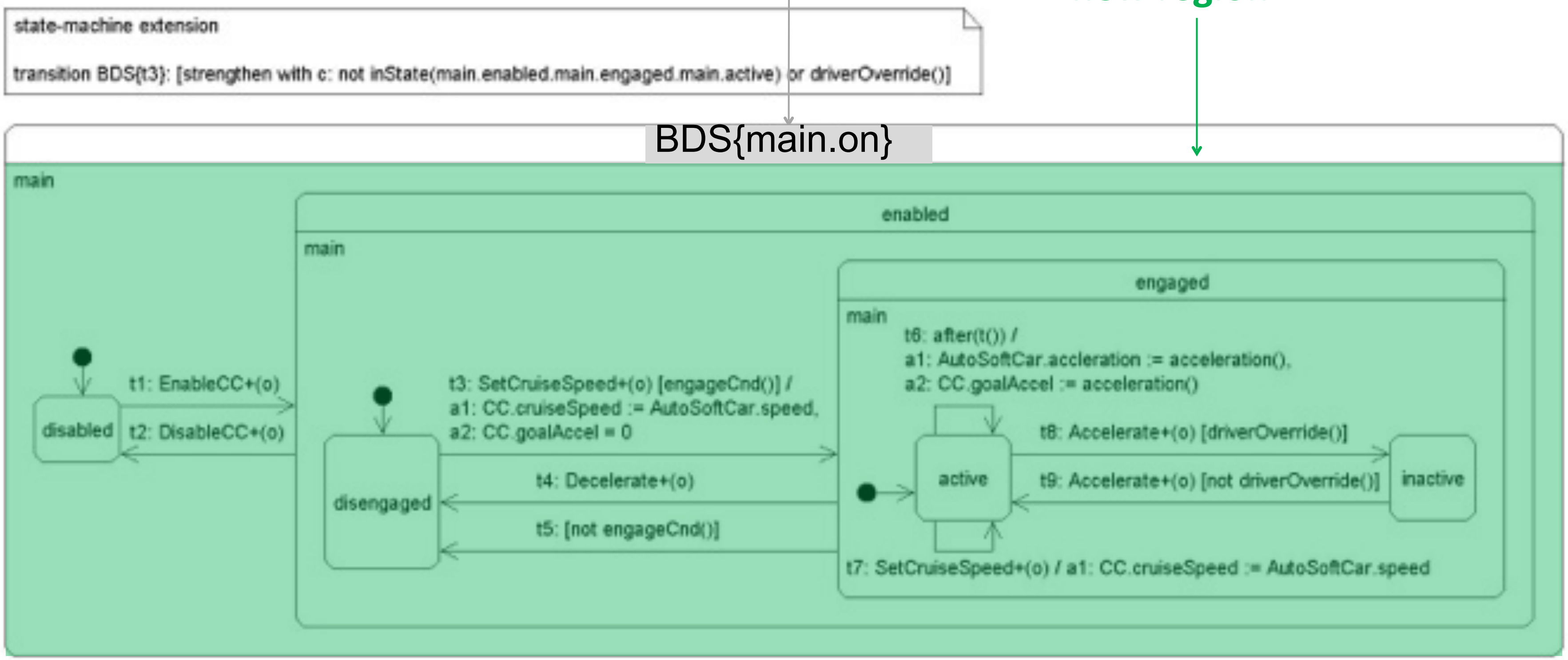
adding behaviours

BDS

Cruise Control (CC)

extends BDS state

new region



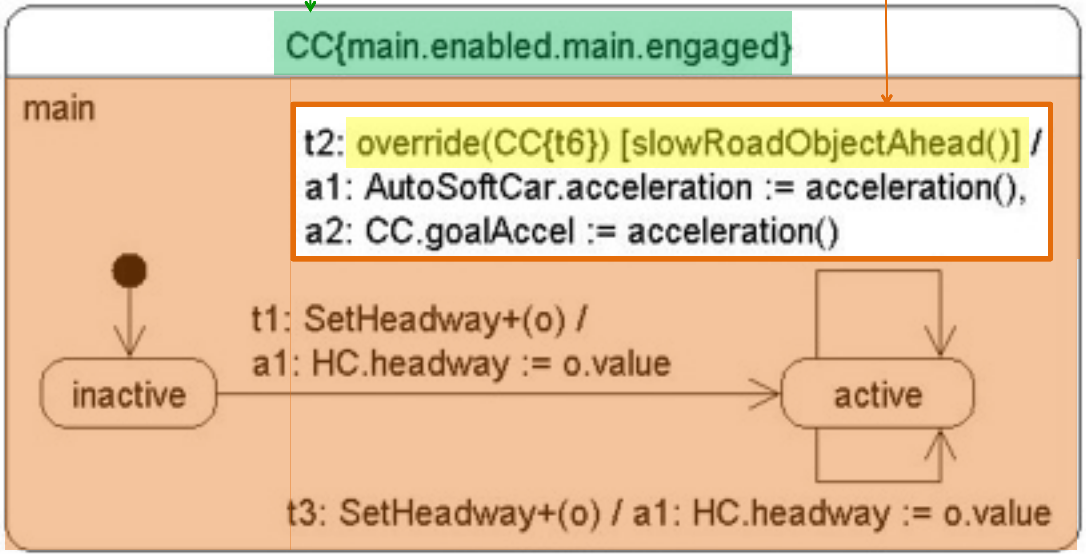
replacing behaviours

CC

Headway Control (HC)

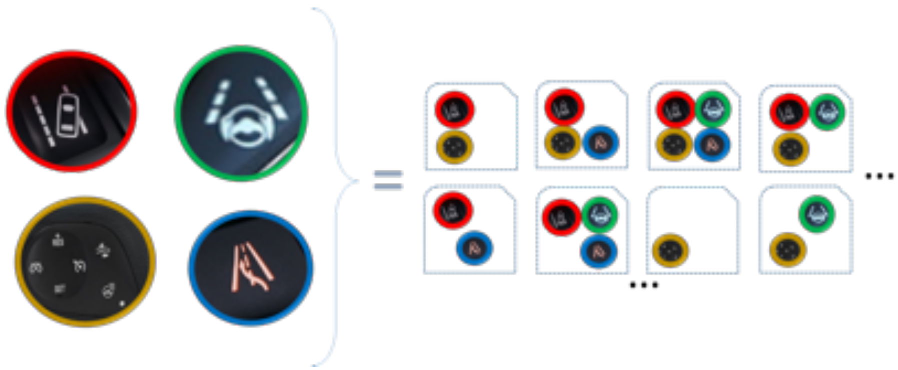
extends CC state

new region includes pre-empting transition:
models HC intentionally prohibiting CC



product-based strategy

generate and analyze each product in isolation



advantages

- parallel analysis
- can use existing analysis techniques and tools
- sound and complete

disadvantages

- exponential number of products to analyse
- sampling is not complete
- analysis is inefficient as it doesn't exploit commonalities among features

family-based analysis

compose features into one product-line model

advantages

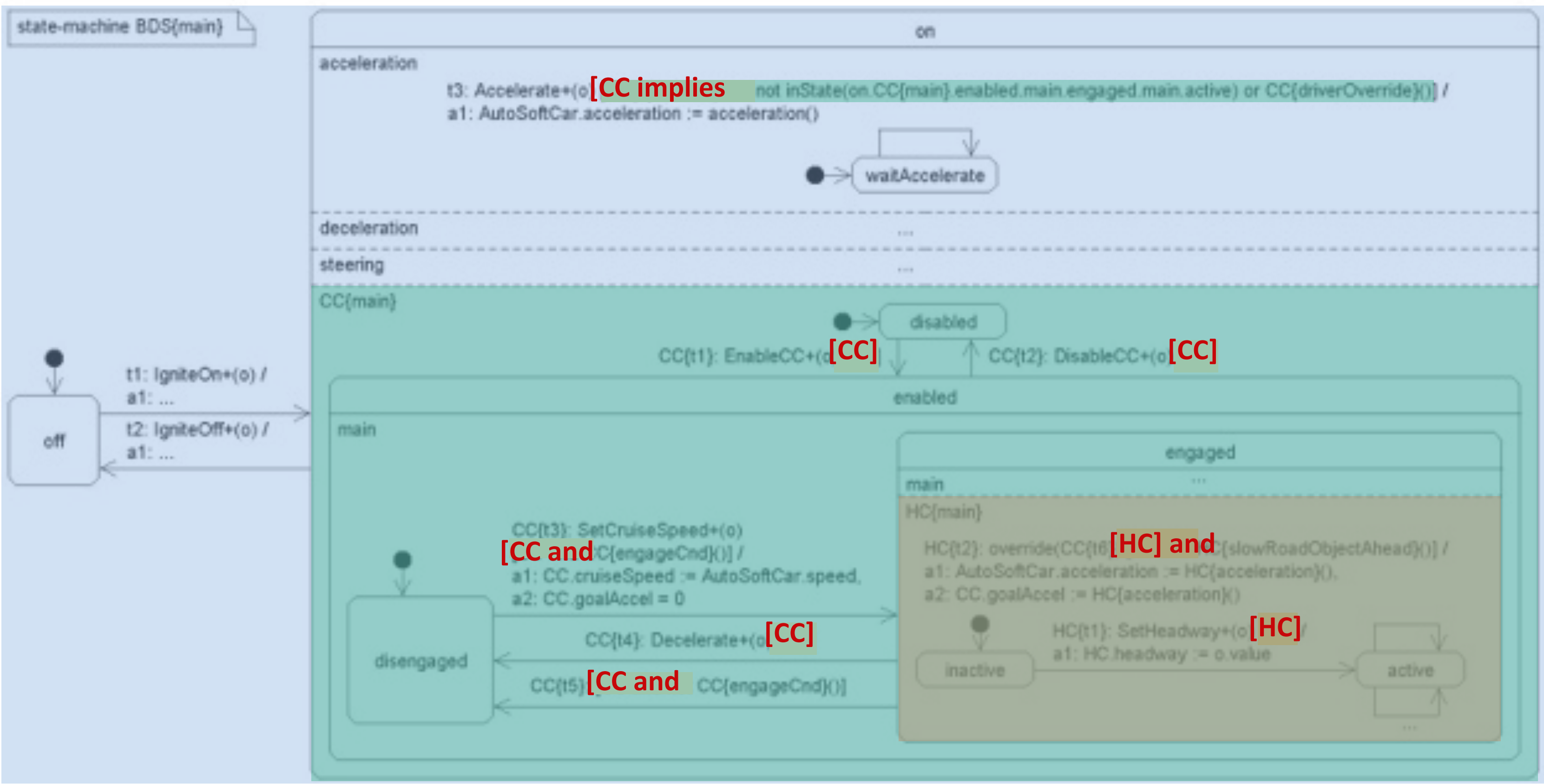
- efficient analysis by exploiting commonalities
- sound and complete

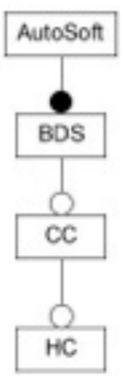
disadvantages

- huge model susceptible to state-space explosion
- cannot perform parallel analysis
- requires new (modified) tools

composition is a product line

transitions, actions, clauses are guarded by presence conditions (of their declaring feature)

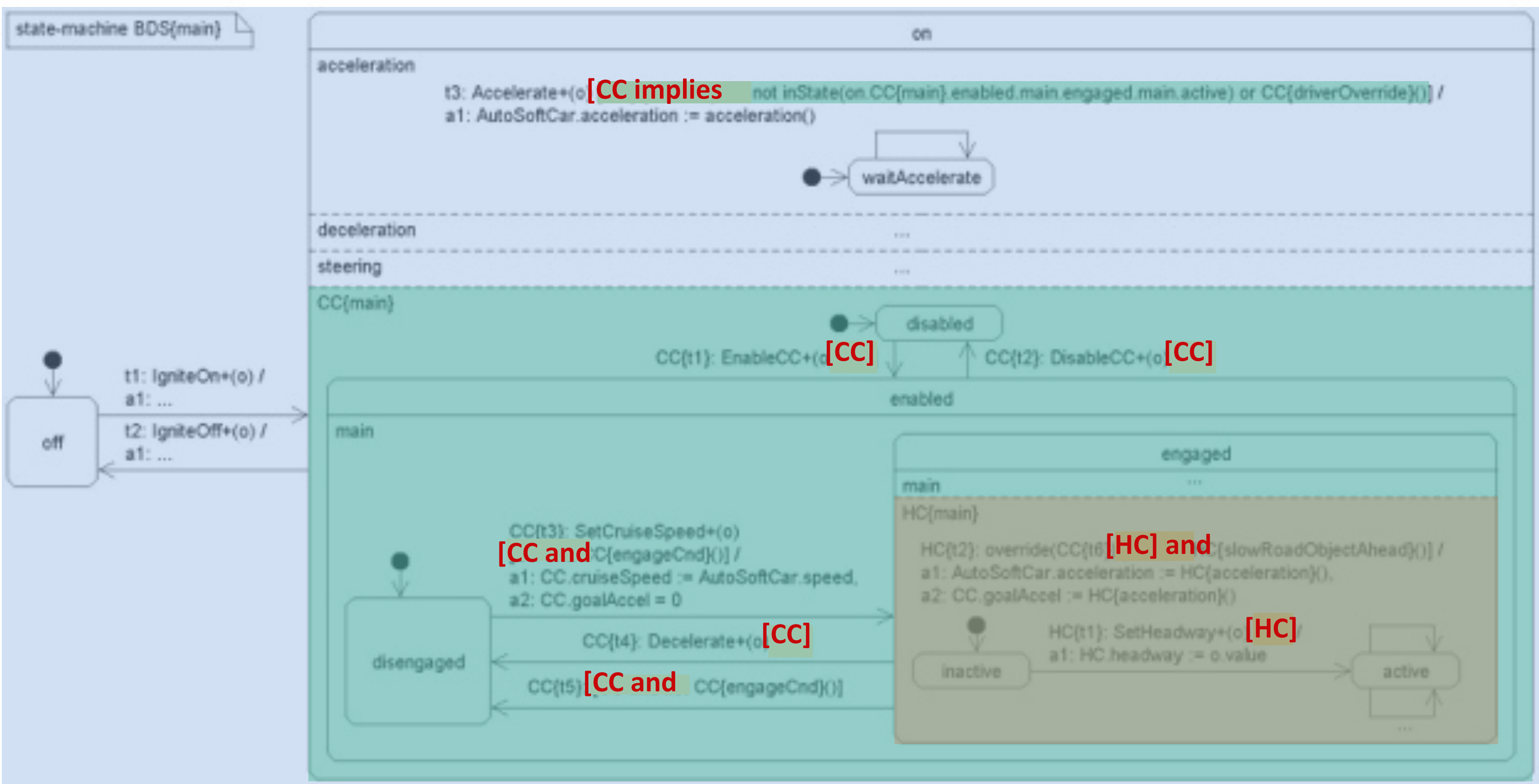




product line = {BDS, BDS + CC, BDS + CC + HC}

composition is commutative

- **order of composition does not affect behaviour**
- **enables incremental composition**

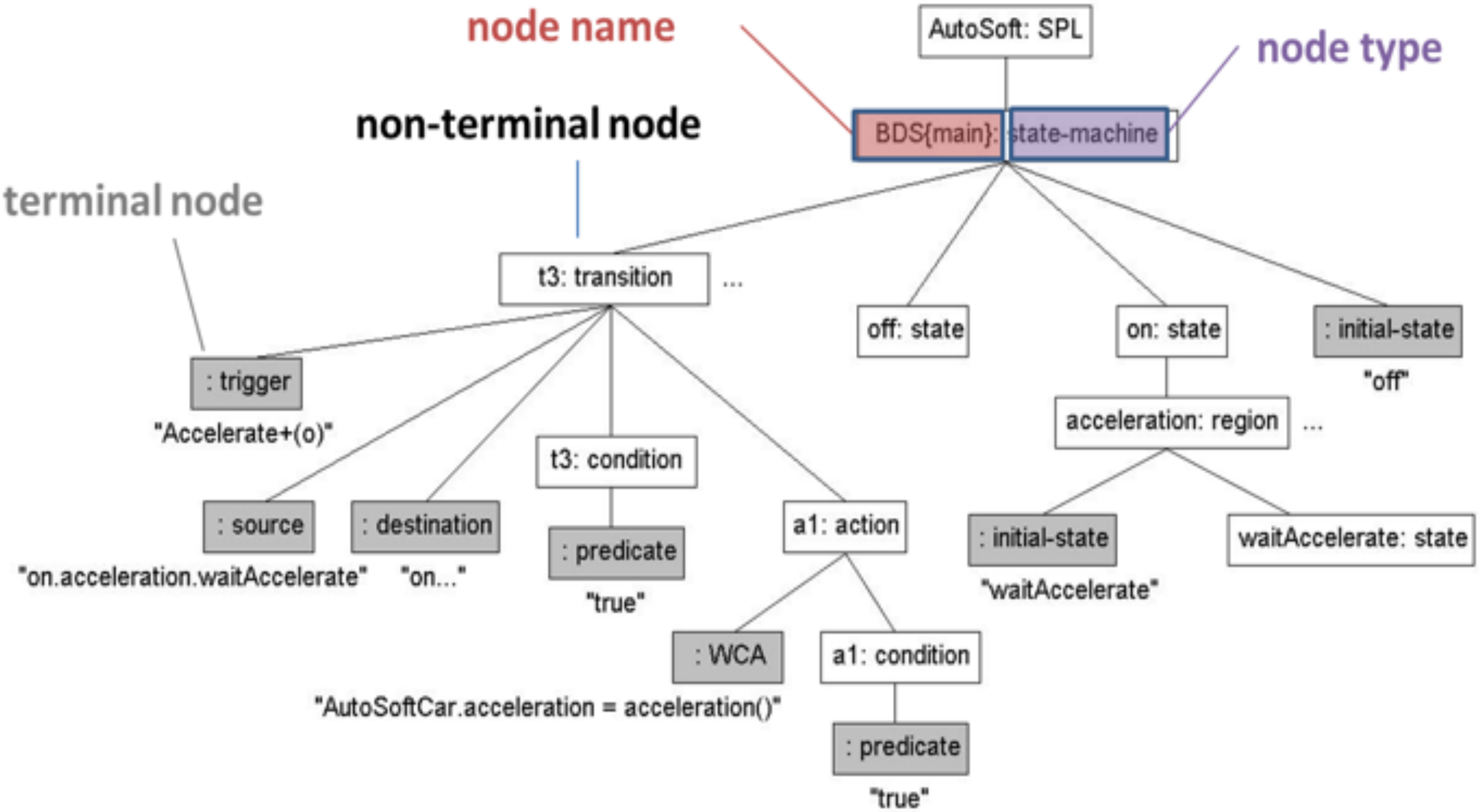


semantics of feature-module Composition

the composition of feature modules is the superimposition of their feature structure trees (FSTs)

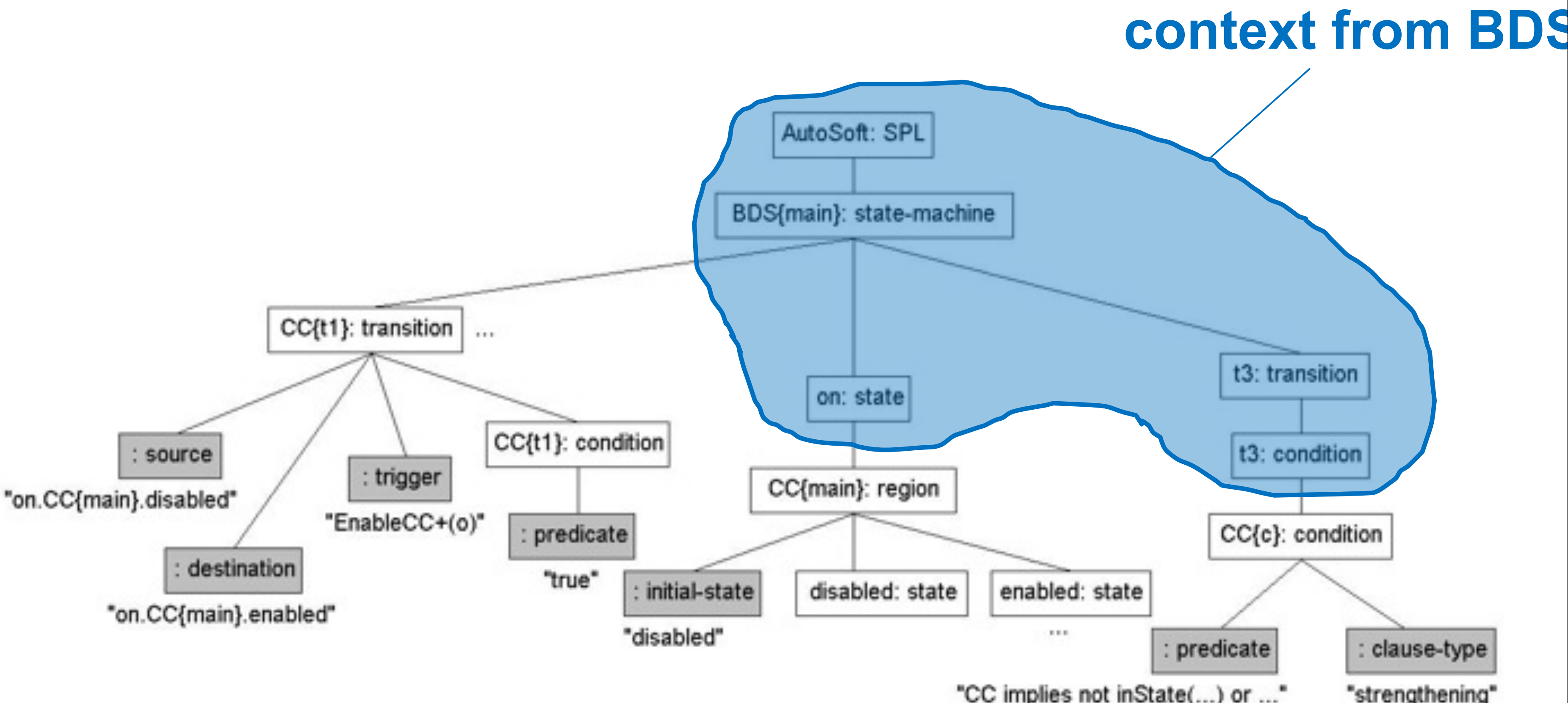
- **FST – abstract-syntax tree of feature module**
- **superimposition – overlay of FSTs with merging of nodes with same name and type**

Feature Structure Tree (FST)



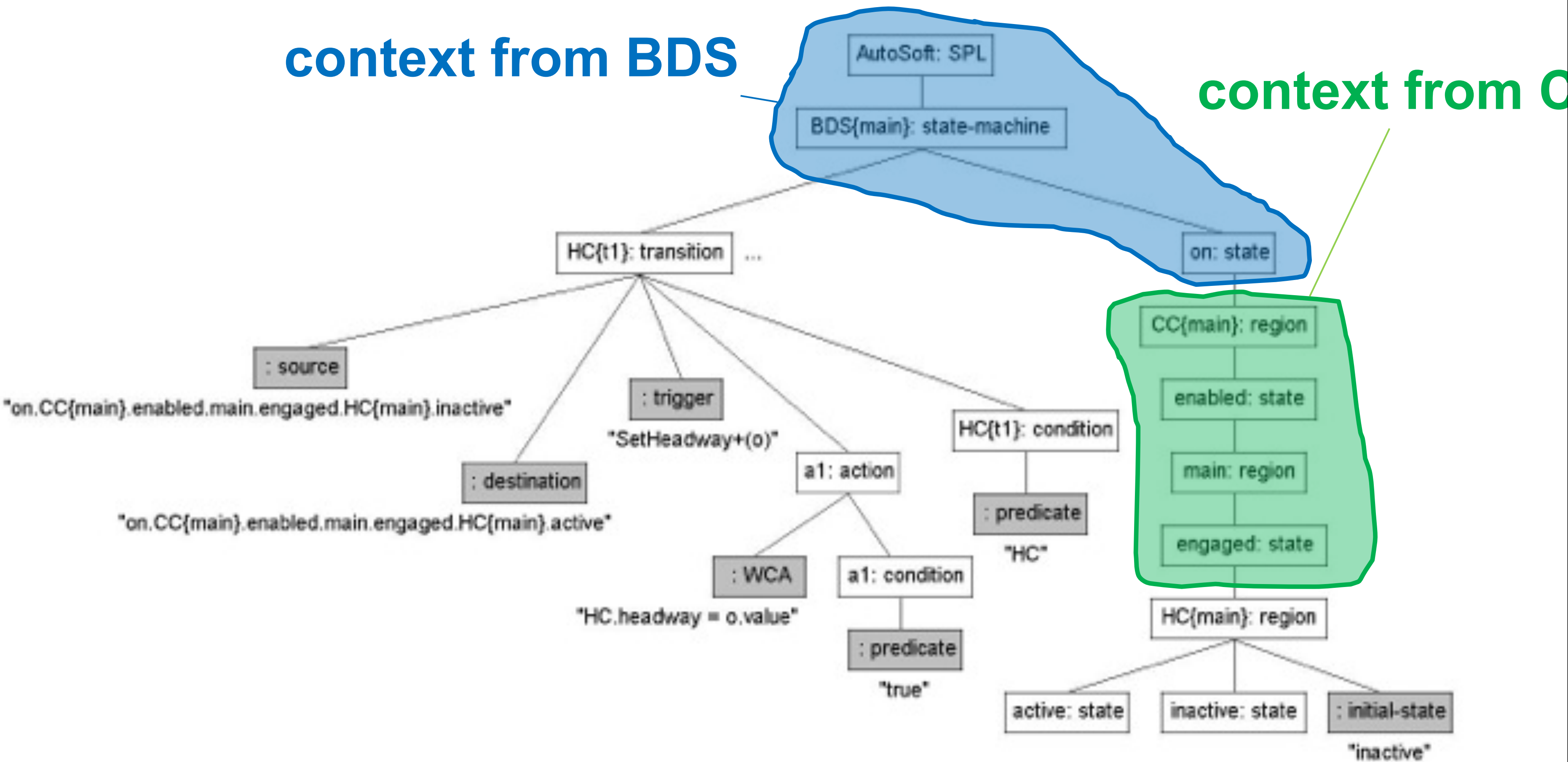
BDS FST

Feature Structure Trees (FST)



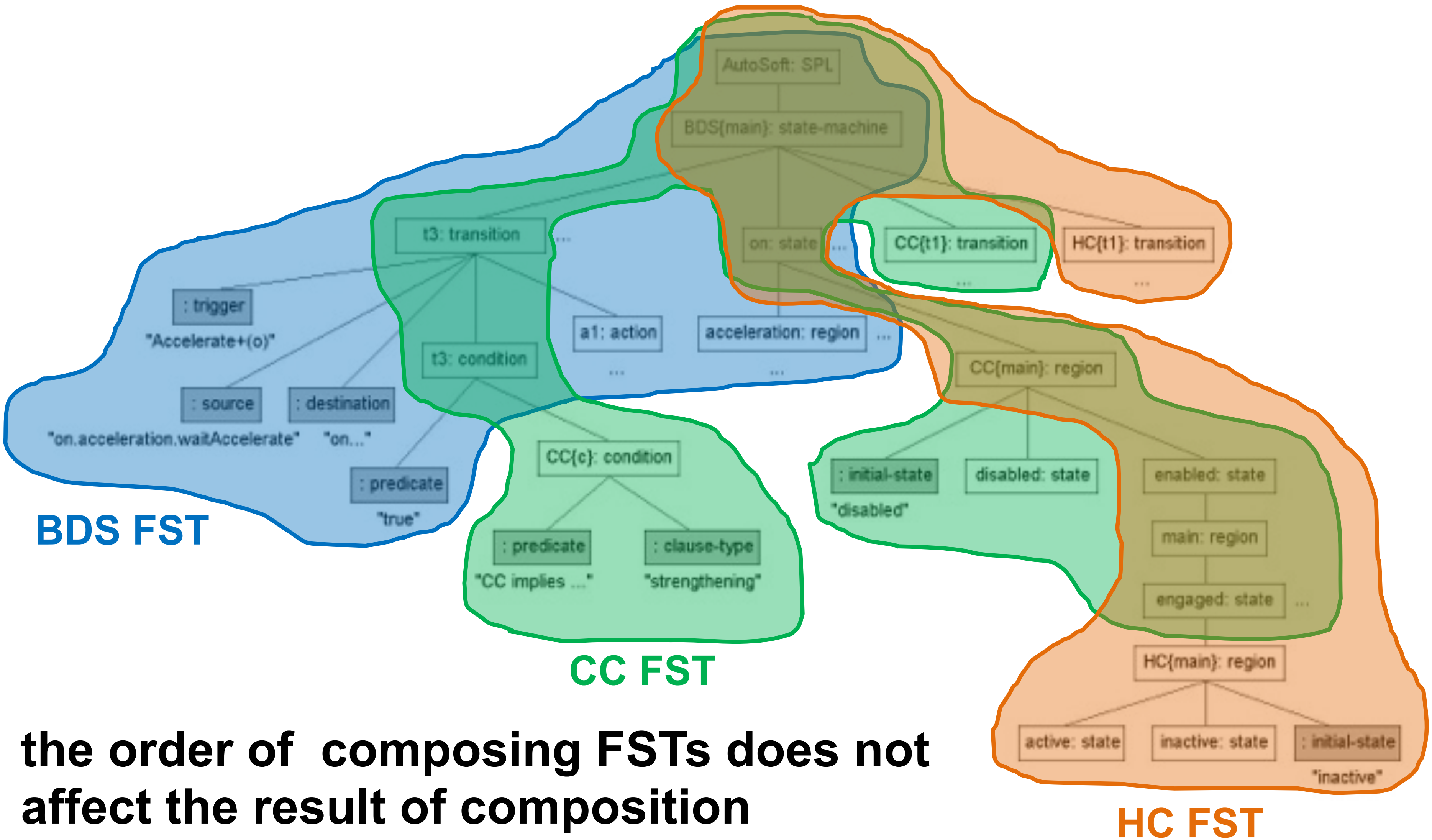
CC FST

Feature Structure Trees (FST)



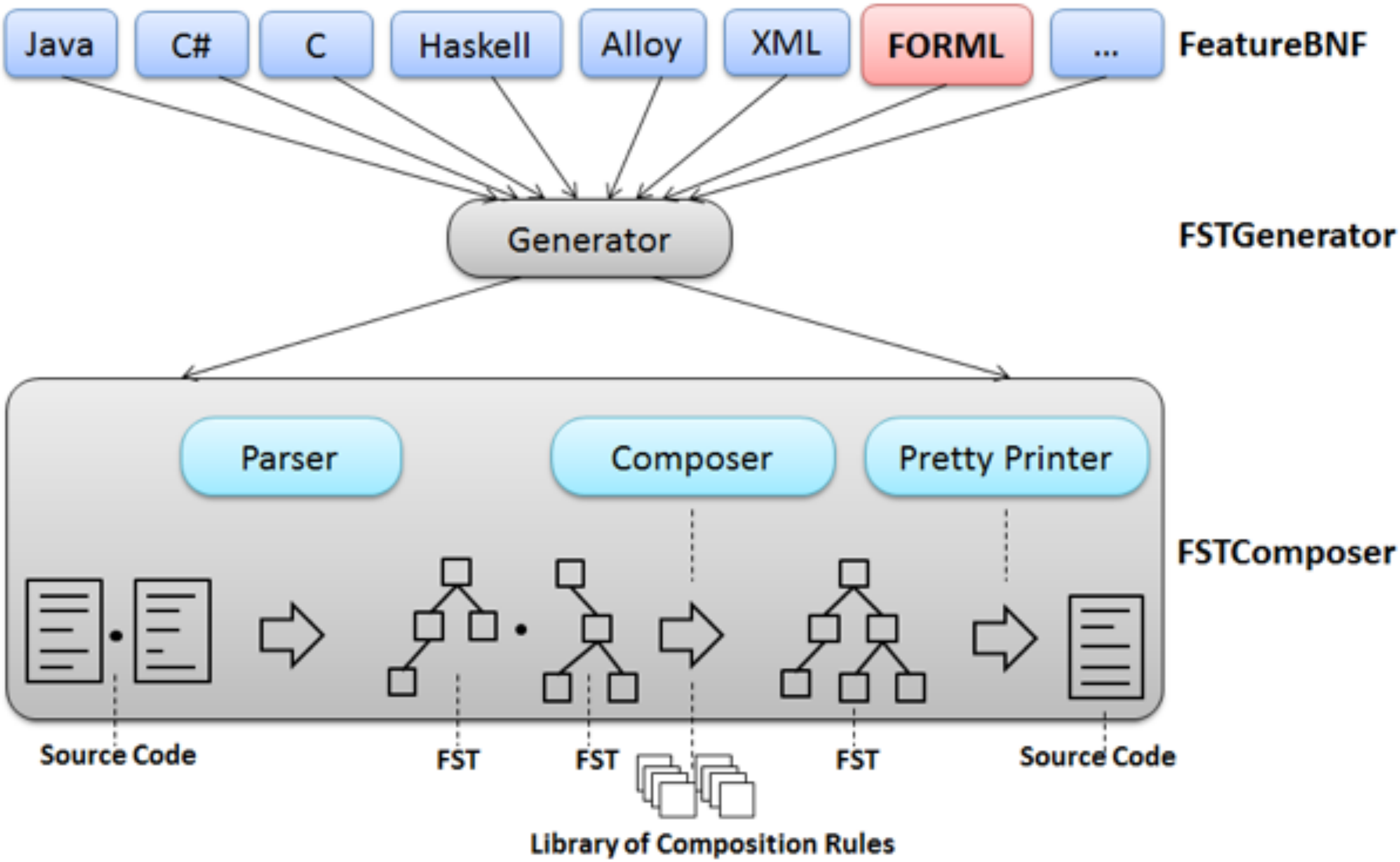
HC FST

superimposition of FSTs



the order of composing FSTs does not affect the result of composition

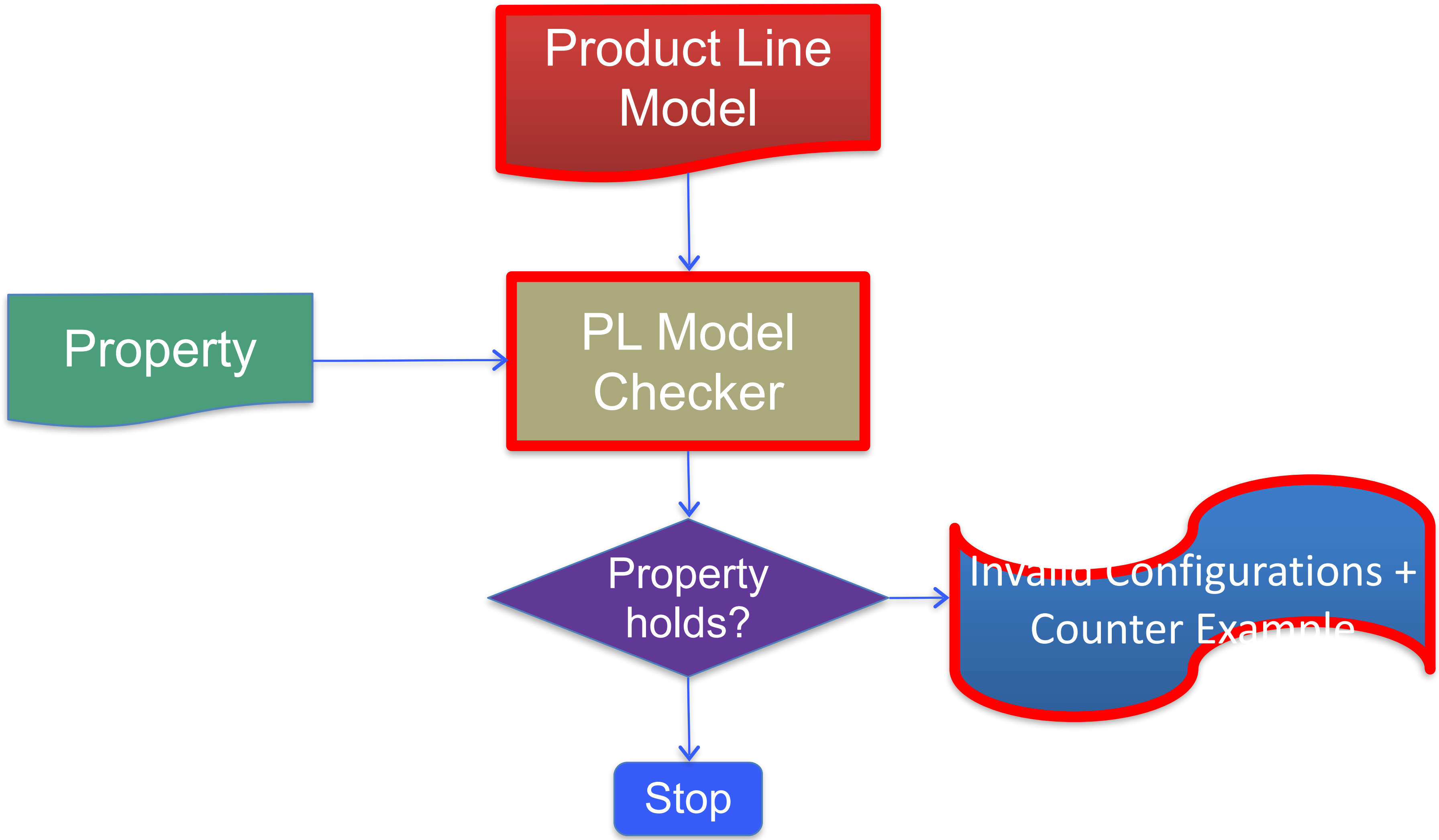
feature composition in featureHouse



FeatureHouse Architecture

product-line model checking

Classen, Heymans, Schobbens, Legay, Raskin, ICSE'10



SAT-based product-line model checking

S. Ben-David, B. Sterin, J. M. Atlee, and S. Beidu, ICSE'15

previous PL model checking implemented in BDDs

› BDDs have been outperformed by SAT-based methods

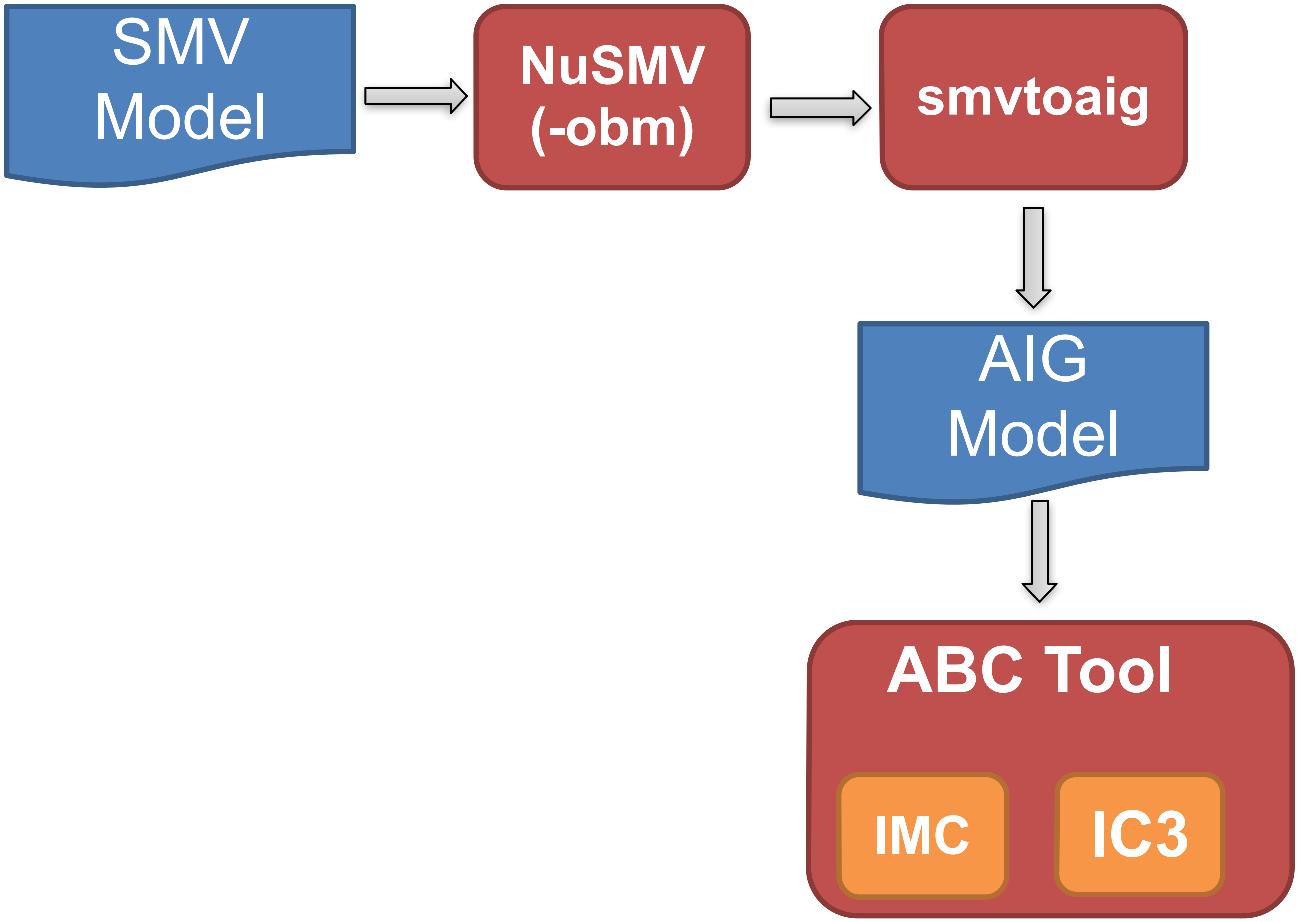
we have implemented PL model checking on top of two existing SAT-based algorithms in the ABC Verification tool

› **IMC**: Bounded Model Checking
  - complemented with Interpolation

› **IC3**: Incremental Construction of Inductive Clauses for Indubitable Correctness

SAT-based product-line model checking

S. Ben-David, B. Sterin, J. M. Atlee, and S. Beidu, ICSE'15



SAT-based product-line model checking

S. Ben-David, B. Sterin, J. M. Atlee, and S. Beidu, ICSE'15

- we experimented with 3 models: two from the literature (small) and an in-house one (large)
- results suggest improvement of 1 to 3 orders of magnitude.

Sample results:

Model Size	BDDs	IC3	IMC	Speed up
40	68	2.1	1.36	×50
40	999	0.33	0.55	×3027
76	44	0.02	0.03	×1110
76	80	0.21	0.39	×387
526	—	8.2	25	—
526	—	50	224	—

family-based analysis

compose features into one product-line model

advantages

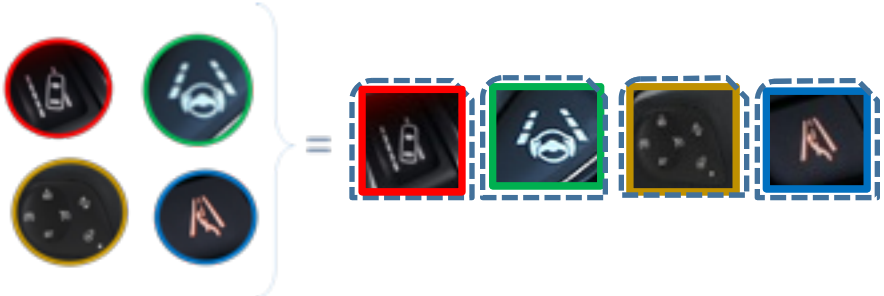
- efficient analysis by exploiting commonalities
- sound and complete

disadvantages

- huge model susceptible to state-space explosion
- cannot perform parallel analysis
- requires new (modified) tools

feature-based analysis

each feature is analyzed wrt open environment



advantages

- small model size
- linear analysis tasks
- can use existing analysis techniques and tools
- parallel analysis

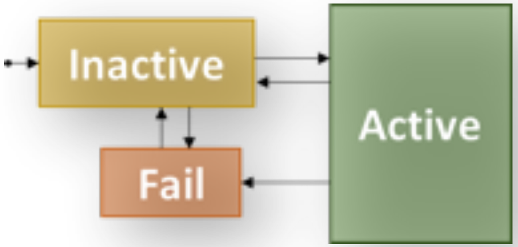
disadvantages

- **cannot detect feature interactions**

observations

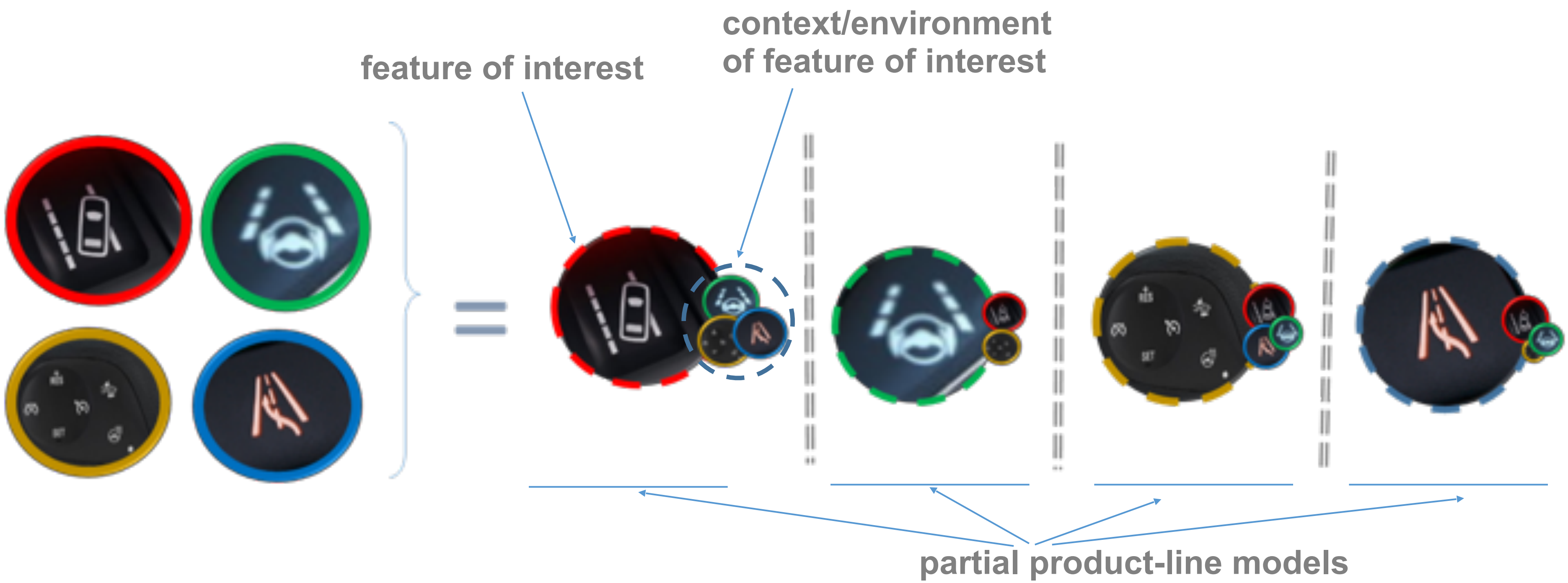
- typically a feature is designed to satisfy its requirements in specific contexts
- features can be designed to have interfaces
- some class of feature interactions can be automatically detected

mode-based pattern for features



90% of references are to high level modes

a new approach –
interface-based analysis



analyze each feature with respect to its <u>minimal environment</u>

a new approach – interface-based analysis

benefits

- **efficient analysis**
- **relatively small model size**
- **can detect feature interactions**
- **linear analysis tasks**
- **parallel analysis**

challenge: **deriving the minimal environment**

- should be as small as possible
- must preserve behaviour of feature of interest

a new approach – interface-based analysis

benefits

- **efficient analysis**
- **relatively small model size**
- **can detect feature interactions**
- **linear analysis tasks**
- **parallel analysis**

challenge: **deriving the minimal environment**

- should be as small as possible
- must preserve behaviour of feature of interest

we use model slicing to derive the context of a feature of interest

model slicing

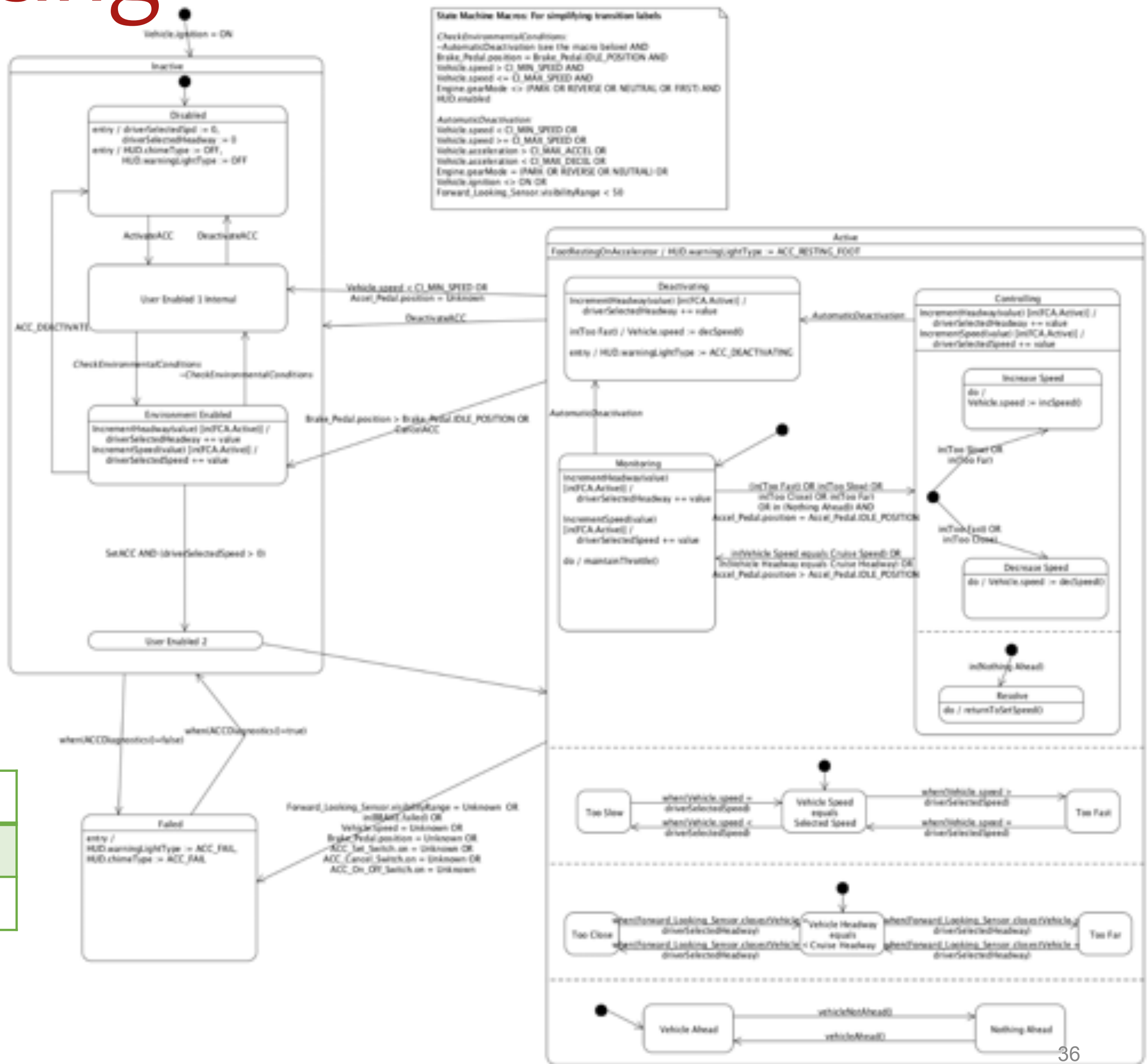
Slicing Step 1:

1. Identify the variables that are read or written by the slicing feature module.

Slicing Step 2:

1. Remove Irrelevant Variables
2. Mark contributing transitions
   a) Transitions that contain relevant variables
   b) Irrelevant transitions that make the model executable
3. Remove non-contributing transitions, states and concurrent regions
4. Merge states if possible

model slicing



	Original	Sliced
States	***21***	***10***
Transitions	53	25

interface rule

automatically generate property, Prop, from feature of interest

sat **Prop**

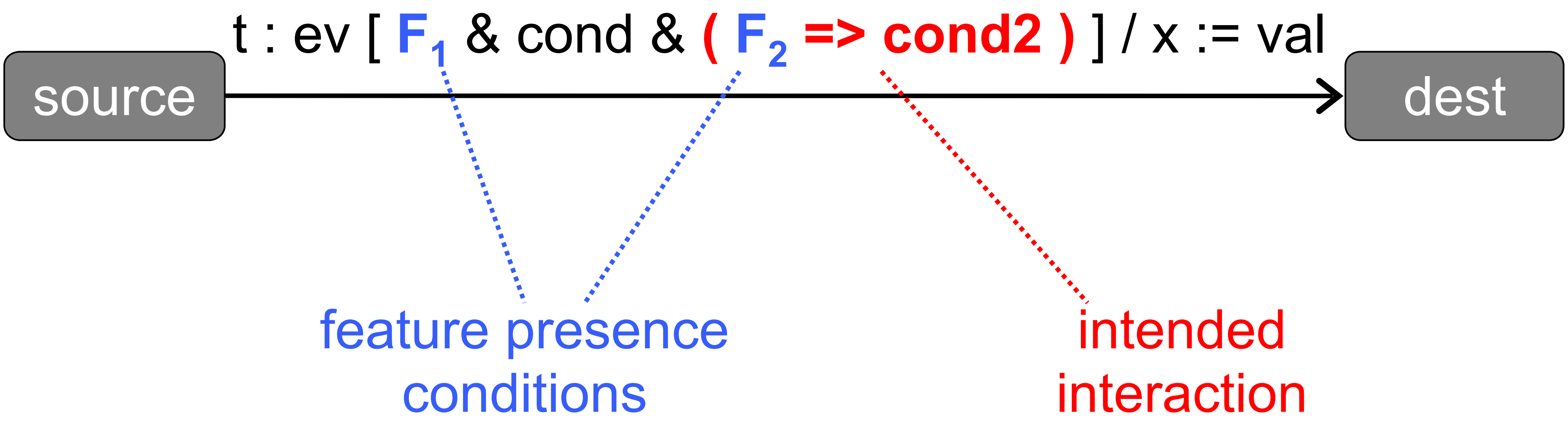
implies

sat **Prop**

properties

- **desired behaviour of features**
- **conditional on whether feature is present**
- **accommodate intended interactions**

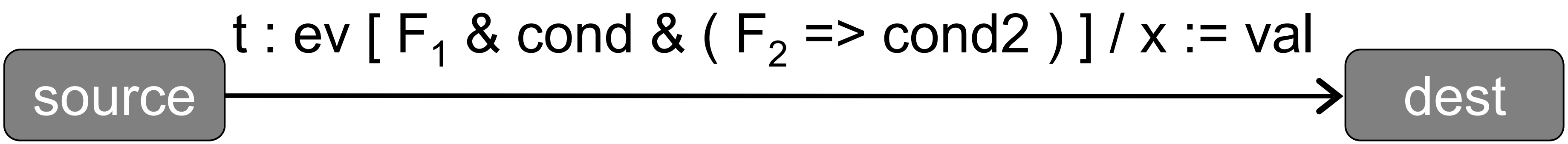
› which affect whether a transition executes



properties

property for each executing transition:

- if transition executes, the effects of its actions are realized
- can be generated automatically from feature



AG (t_execute -> AX(x = val))

All paths

Globally

ne**X**t state

summary of progress