Assuring Product Lines of Complex Systems

Marsha Chechik

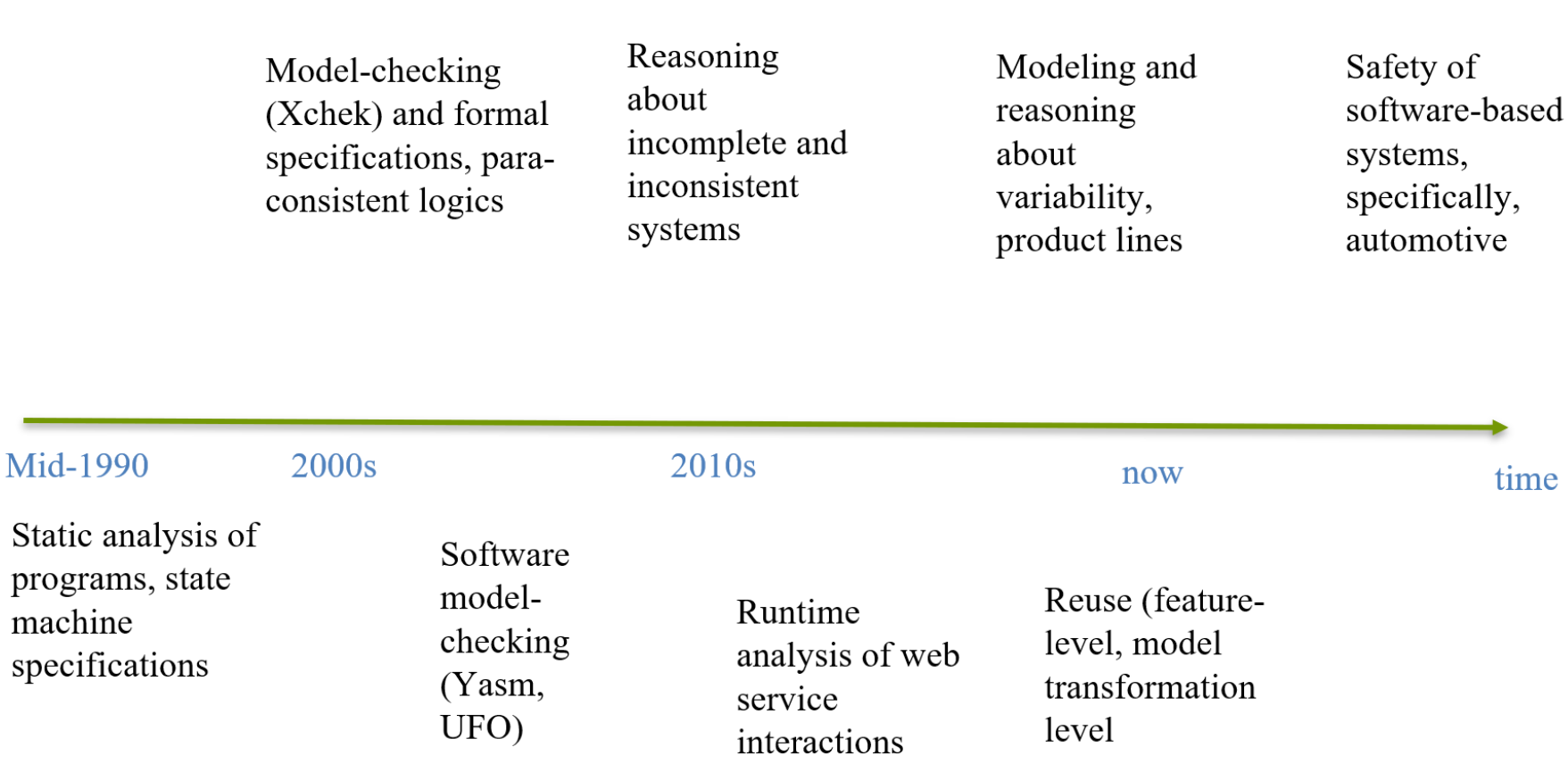
SPLC’ 25

September 4, 2025

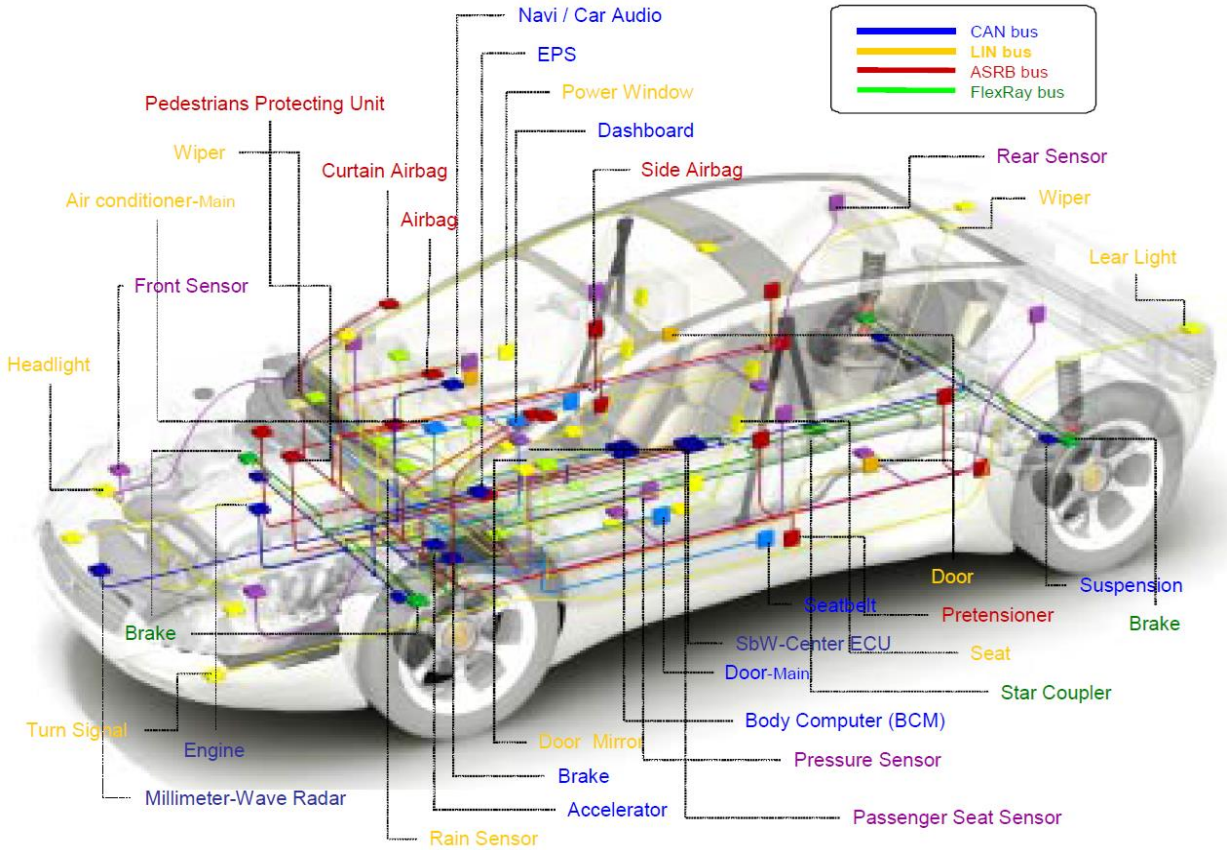
UNIVERSITY OF TORONTO

A Brief and Partial Research History

Interests: software engineering, formal specification and verification of software, modeling, assurance, product line analysis



Why Cars?

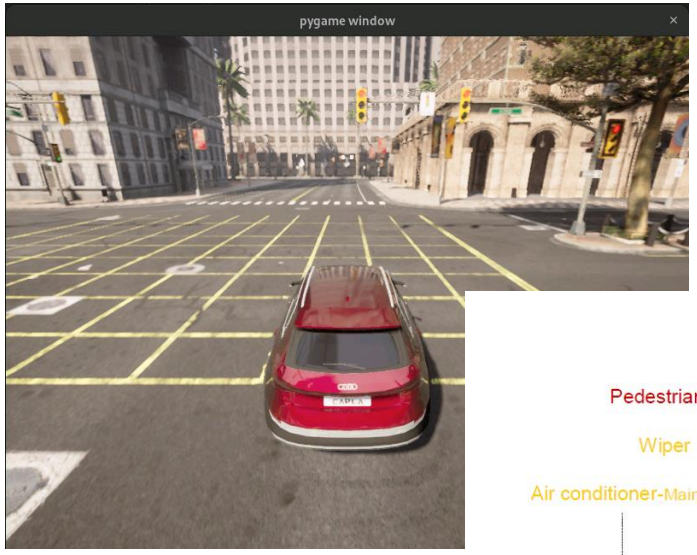


http://www.flexautomotive.net/EMCFLEXBLOG/post/2015/09/08/can-bus-for-controller-area-network)

- Extensively rely on software for their operations
- No system-wide model
- Heterogeneous components
  - legacy, generated, third-party
  - distributed ECUs
  - bus-based communications
- >100 million lines of code
- High variability (product lines)

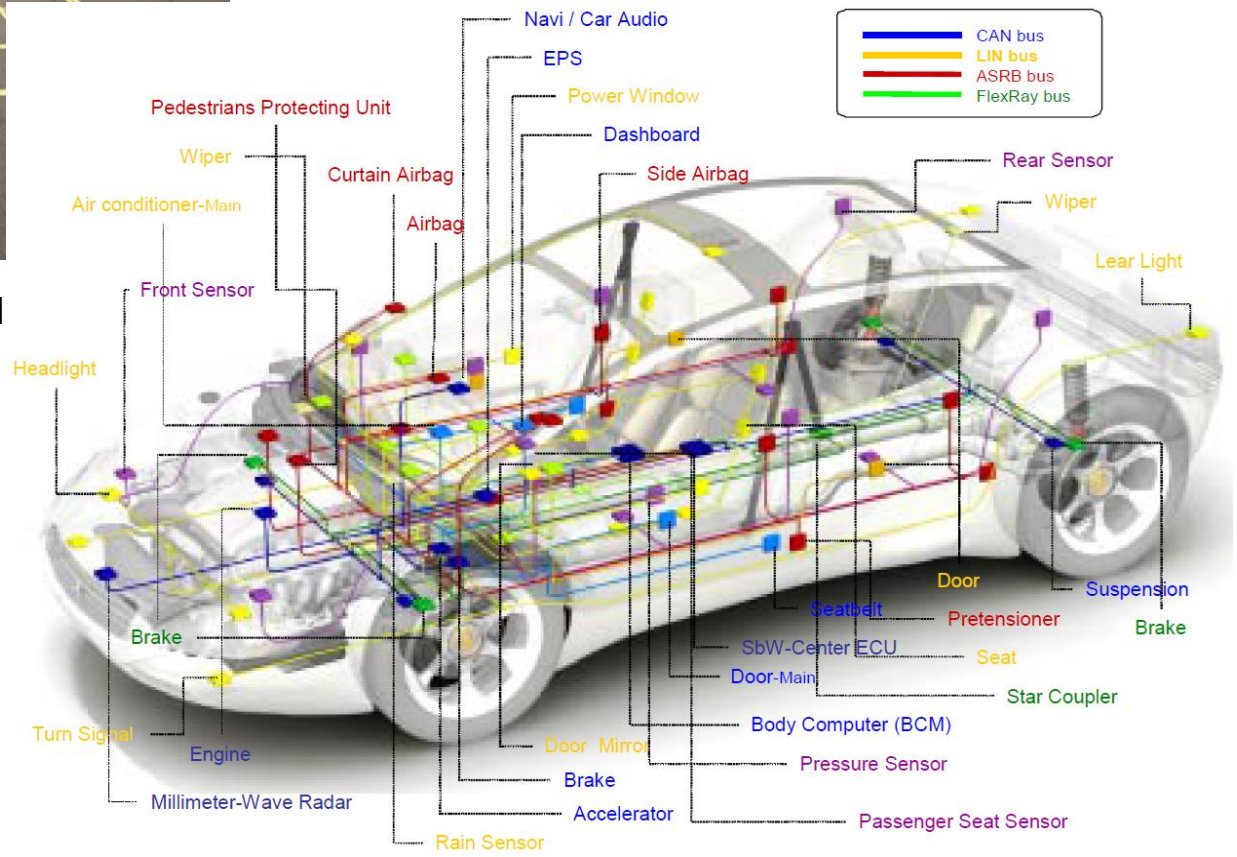
Complexity, ubiquity, significant end-user configurability, current self-driving race / accidents, over-the-air updates

SOFTWARE IN THE CAR



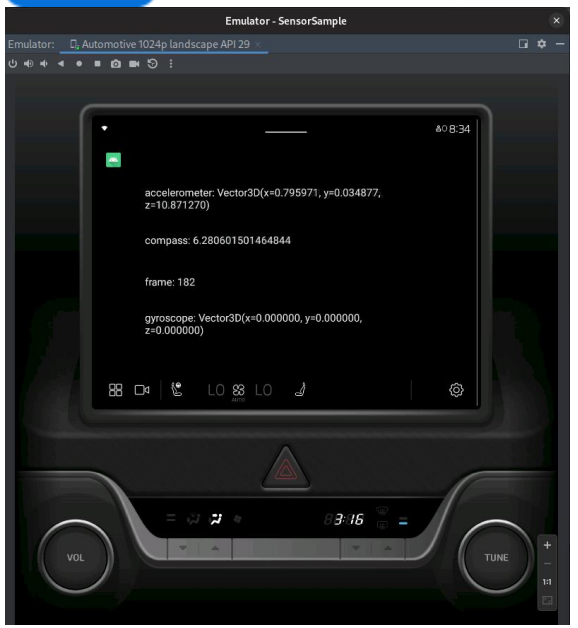
Environment sensing and perception, other cars, fleet driving

Controller



Privileged App

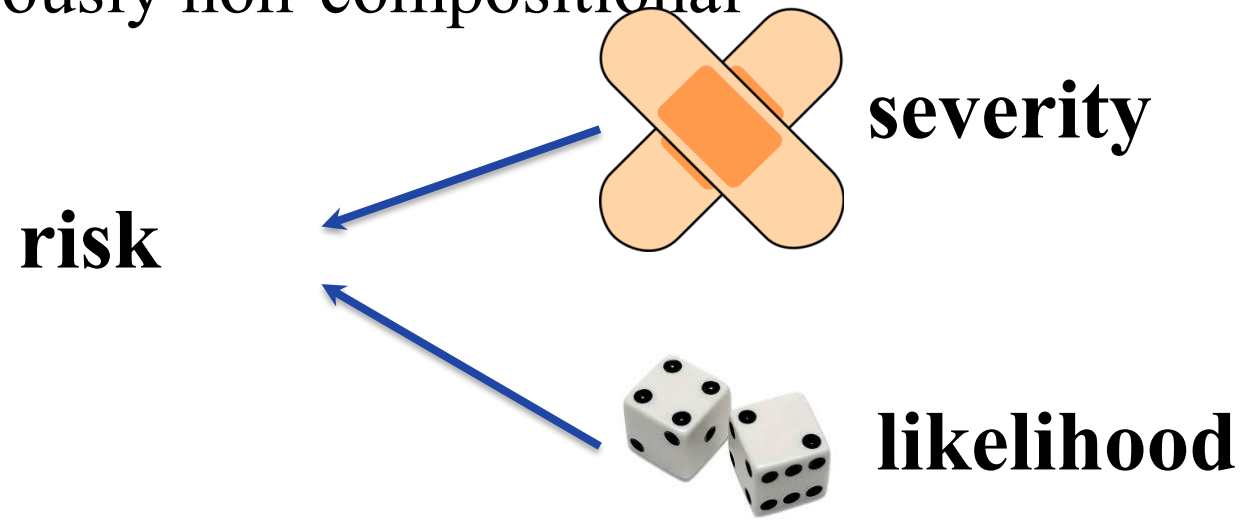
Third Party App



Android Emulator: Platform

Methods to Validate Complex Safety-Critical Systems

- Verification and Validation (V&V)
  - Mostly various forms of testing
- Safety analysis
  - Safety – absence of unreasonable risk of mishap
  - Notoriously non-compositional



"Standards are documented agreements containing technical **specifications** or other precise criteria to be used consistently as **rules**, **guidelines**, or **definitions** of characteristics, to ensure that materials, products, processes and services are fit for their purpose."

[ISO 1997]

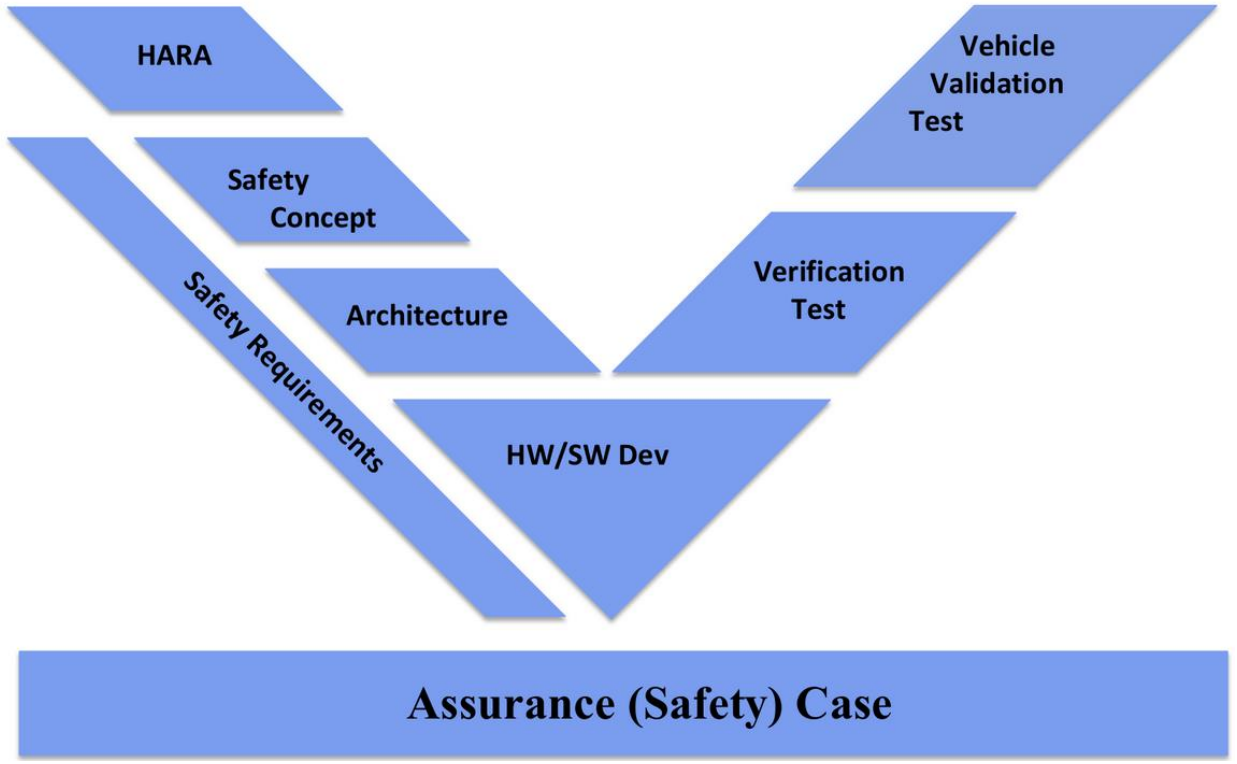
Assurance Cases (ACs)

(aka safety case, security case …)

"Reasoned, auditable **artifact** created for contention that its top level claim (or a set of claims) is satisfied, including **systematic argumentation** and its **underlyi evidence** and explicit assumption(s) that support the claim".

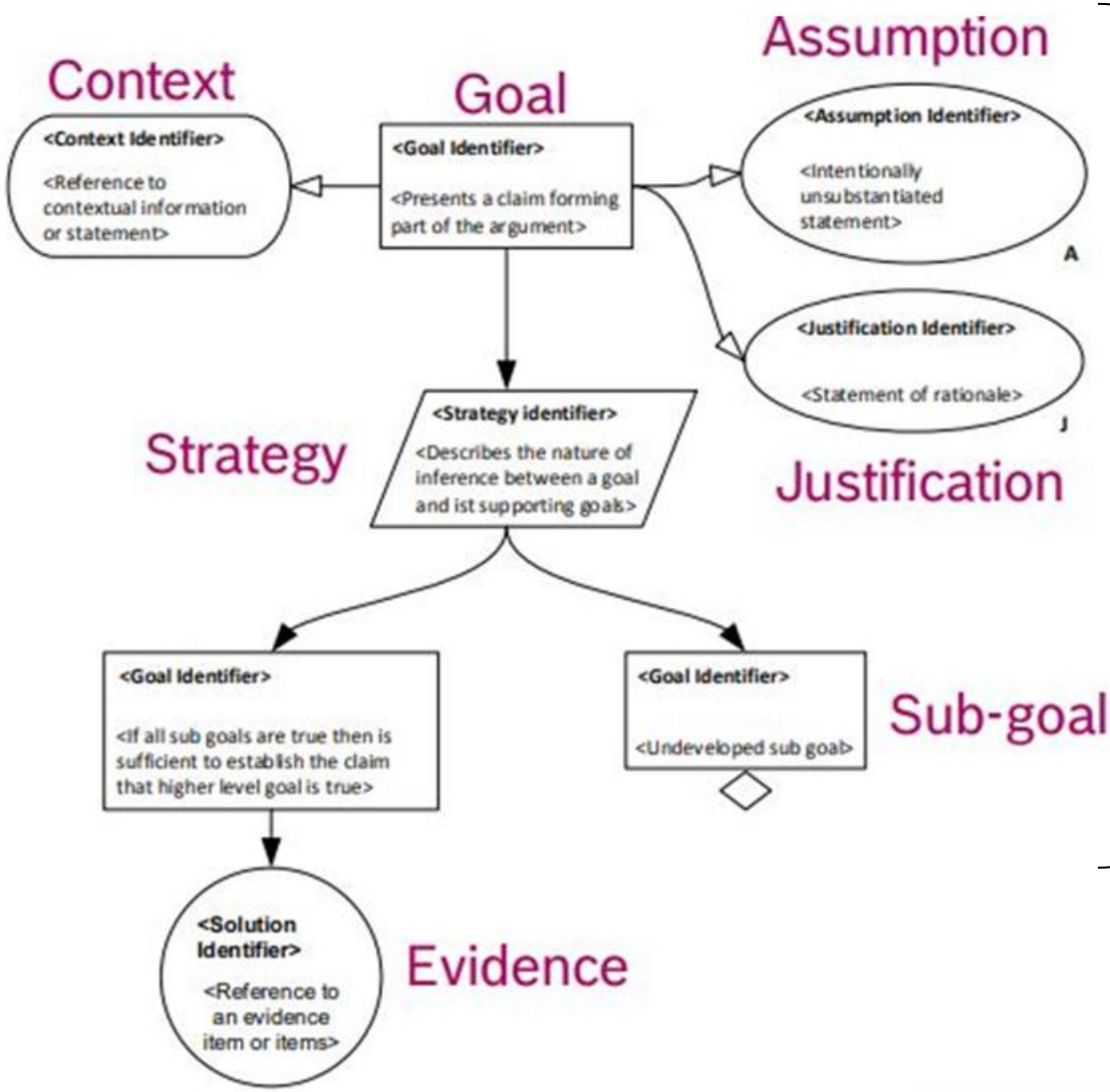
[ISO/IEC 1502

ISO 26262 – Functional Safety of Road Vehicles



Goal Structuring Notation (GSN)

Graphical notation for structuring an assurance case, linking argumentation, rationale, context, assumptions and evidence

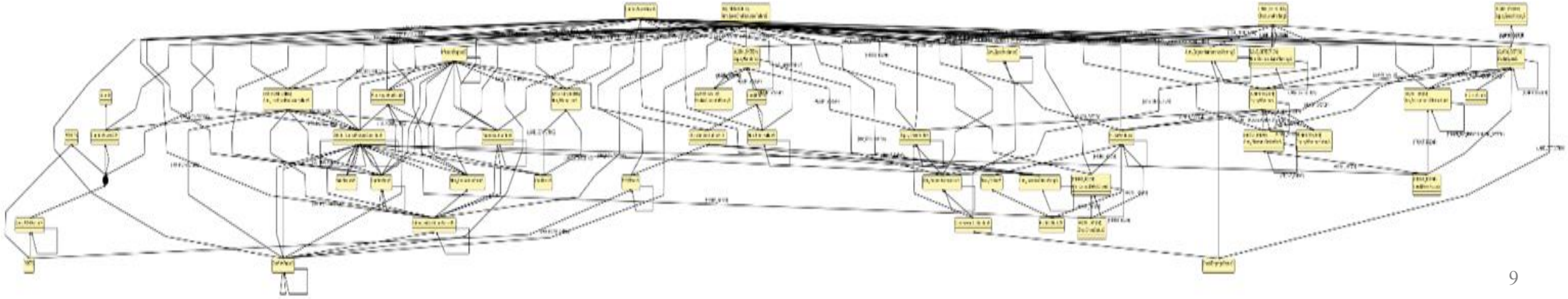


Running Example: Advanced Driving Assistance System (ADAS)

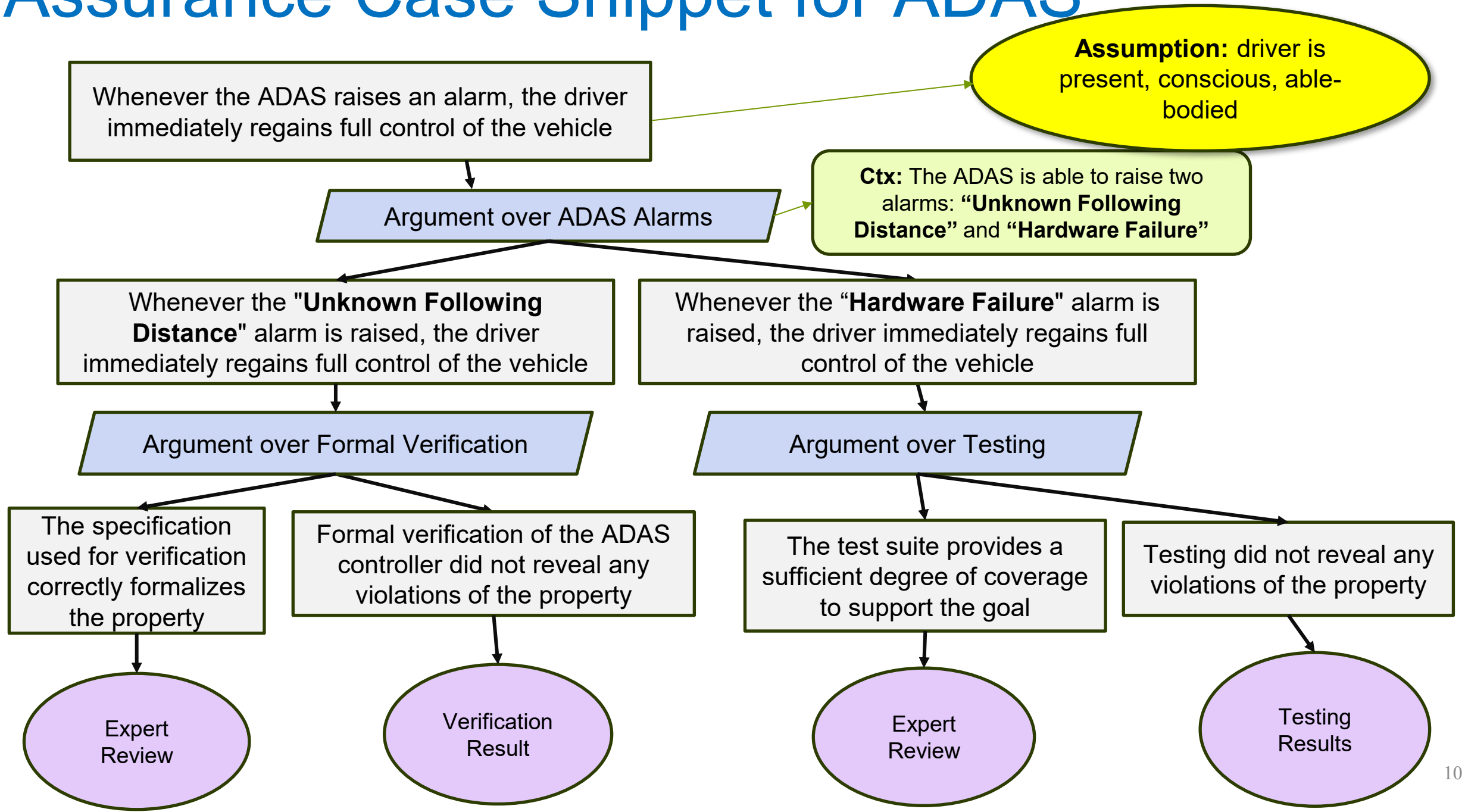
Vehicle with automated features to assist drivers in safe vehicle operations

- Lane Departure Warnings
- Lane Centering
- Driver Monitoring
- Cruise Control

Safety property: Whenever the ADAS raises an alarm (e.g., due to malfunction, unsafe conditions etc.), the driver immediately regains full control of the vehicle.



Assurance Case Snippet for ADAS



Main points

1. Assurance cases combine argument and evidence, allow to contextualize analysis and verification. Need to be reviewable

Assuring Product Lines of Complex Systems -- Talk Plan

- Motivation and goals
- Background
  - Assurance
  - Product lines – variability in space and time
- Representation: Product Line Assurance Cases (PLAC)
- Development of PLACs
- Assessing Change of PLACs
- Tooling
- Summary and Next Steps

Software Product Line Engineering

Need to develop a set of similar (but distinct) software products

- Multiple markets with different requirements/preferences
- Need to accommodate systems with different hardware components

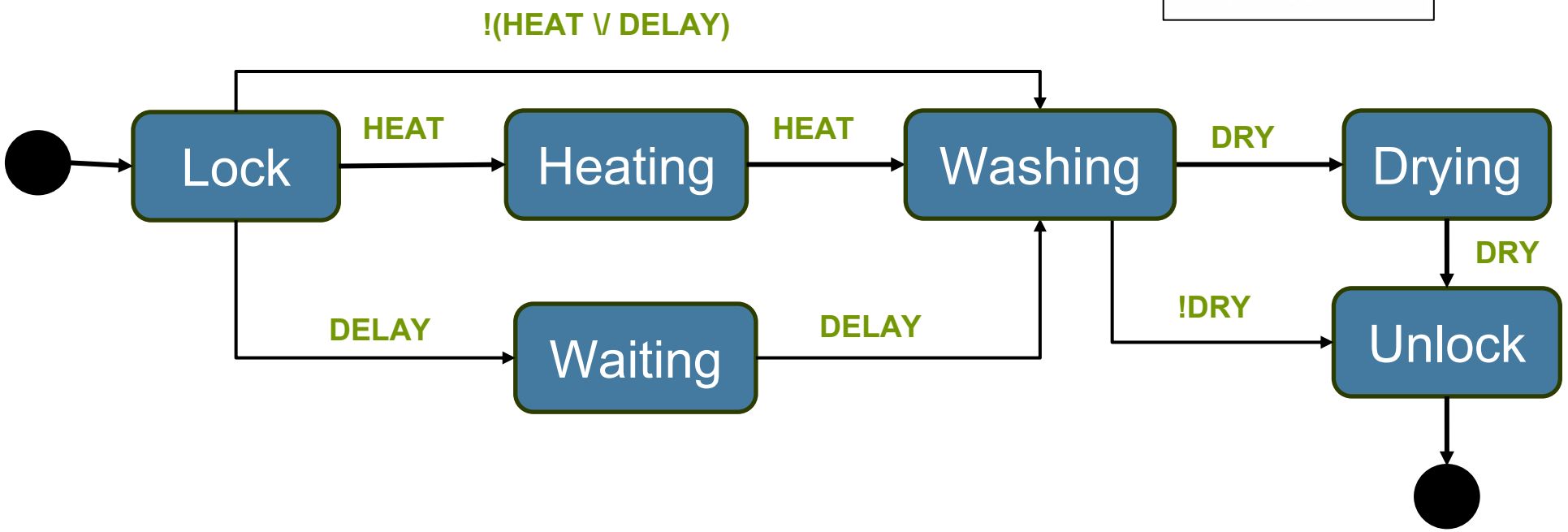
} Sources of **variability**

Idea: Systematically reuse engineering effort across the entire product line by defining and managing variability explicitly

Annotative Product Lines

Idea: Annotate variability via **presence conditions** over **product features**

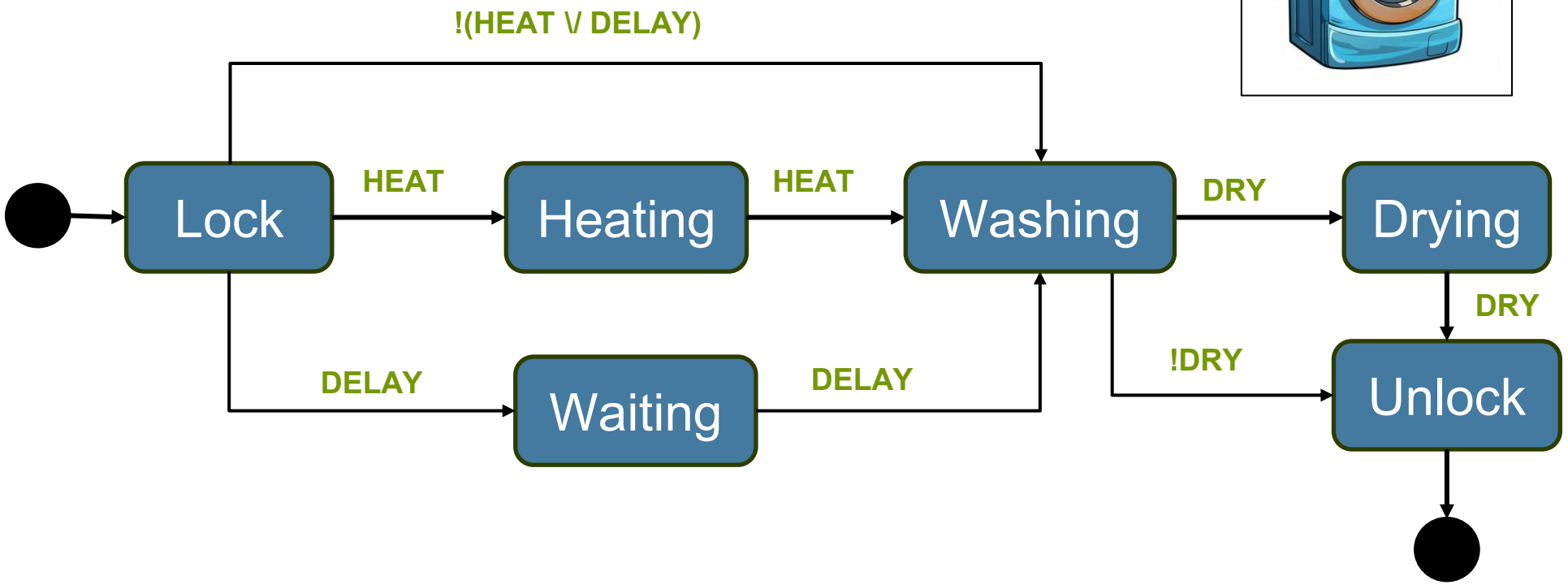
Example: A PL of washing machines with 3 optional features: heating (**HEAT**), drying (**DRY**), and time-delay (**DELAY**)



Annotative Product Lines - Derivation

Derive a product from a product line given a **configuration** of features

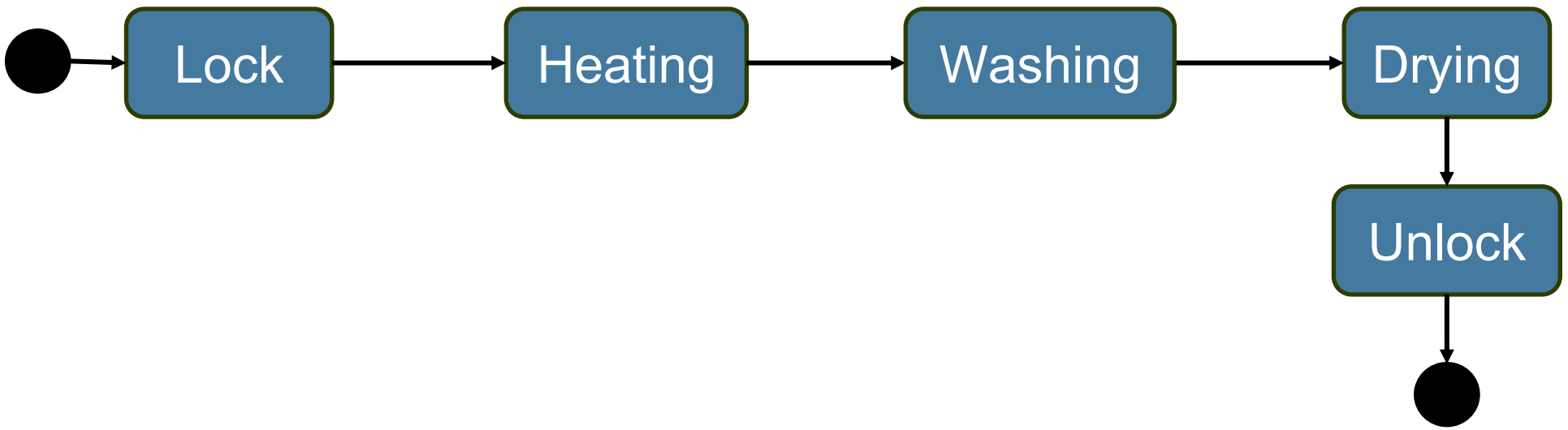
c = {HEAT, DRY}



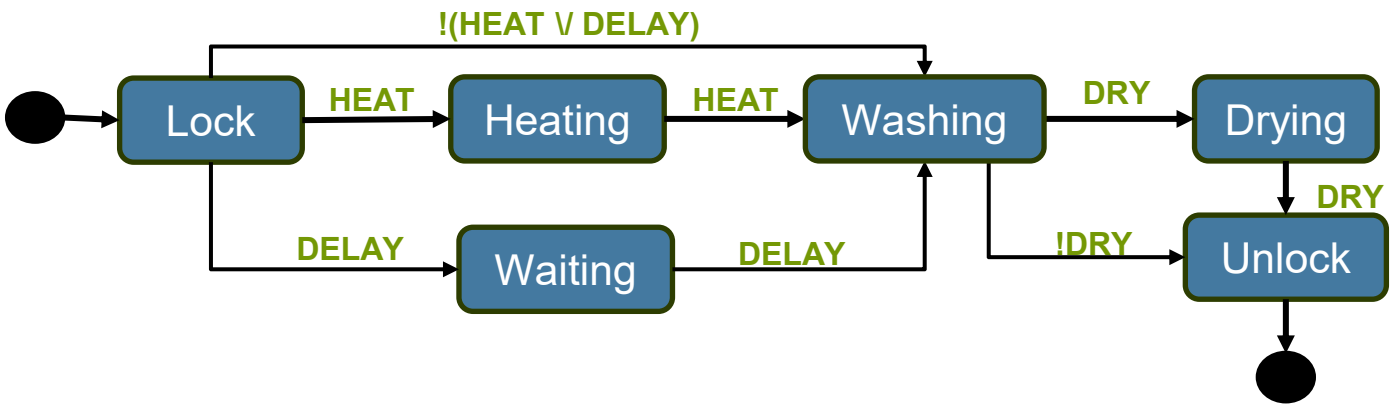
Annotative Product Lines - Derivation

Derive a product from a product line given a **configuration** of features

c = {**HEAT**, **DRY**}



Product Lines of Models



Product Lines of Code

```
int foo(int a, int b) {
  a = a * a;
#ifdef FA && FB
  b = b * b;
#elif FA && !FB
  b = b * a;
#endif
  return a + b;
}
```

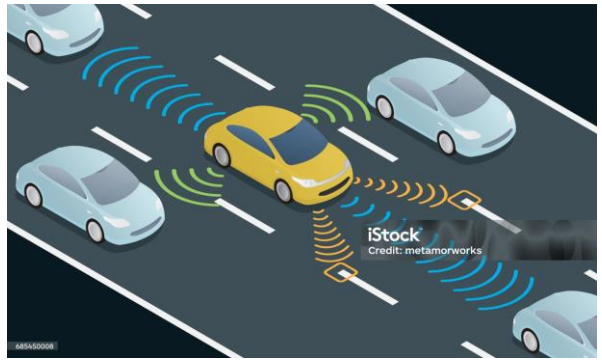
n features ➔ $O(2^n)$ distinct products!

ADAS -- Product Line Version

Representation of a *family* of vehicles with different configurations of ADAS features

Features

- HW_MONITORING
- LANE_DETECTION
- LANE_CENTERING
- FRONT_RADAR
- ALARM_SYSTEM



Feature model:

HW_MONITORING & (LANE_CENTERING => (LANE_DETECTION & ALARM_SYSTEM))

State machine mode becomes annotated with presence conditions

Software Updates

Performing software updates in cars is costly

Over the years, most automotive companies recalled many vehicles to perform updates

- in 2019, GM recalled 4.3 million vehicles
- in 2018, Volvo and Honda recalled 16.582 and 232.000 vehicles.
- in 2017, Stellantis and Tesla recalled 53.000 and 1.1 million vehicles, respectively
- ….

Over-the-air (OTA) Updates

Automotive companies are increasingly interested in over-the-air (OTA) updates. They can

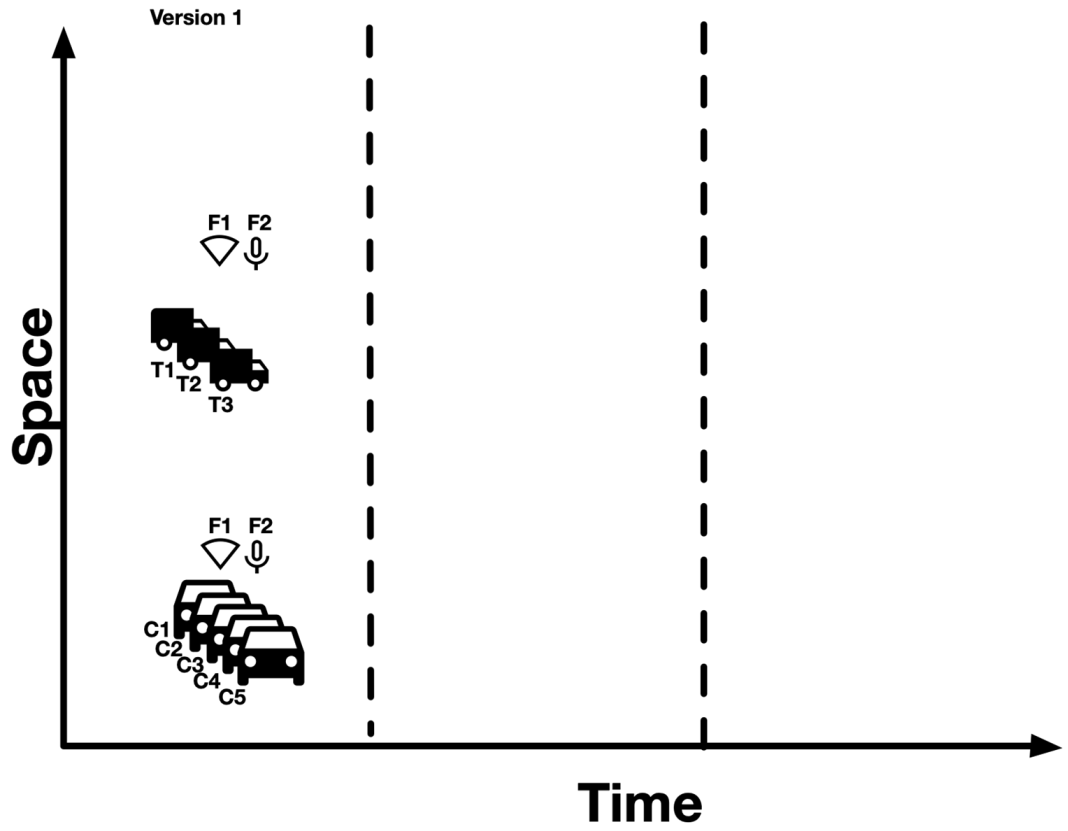
- modify the software components installed in a car wirelessly
- quickly and conveniently change the software installed in the cars.

Over-the-air (OTA) Updates: Software Configurations

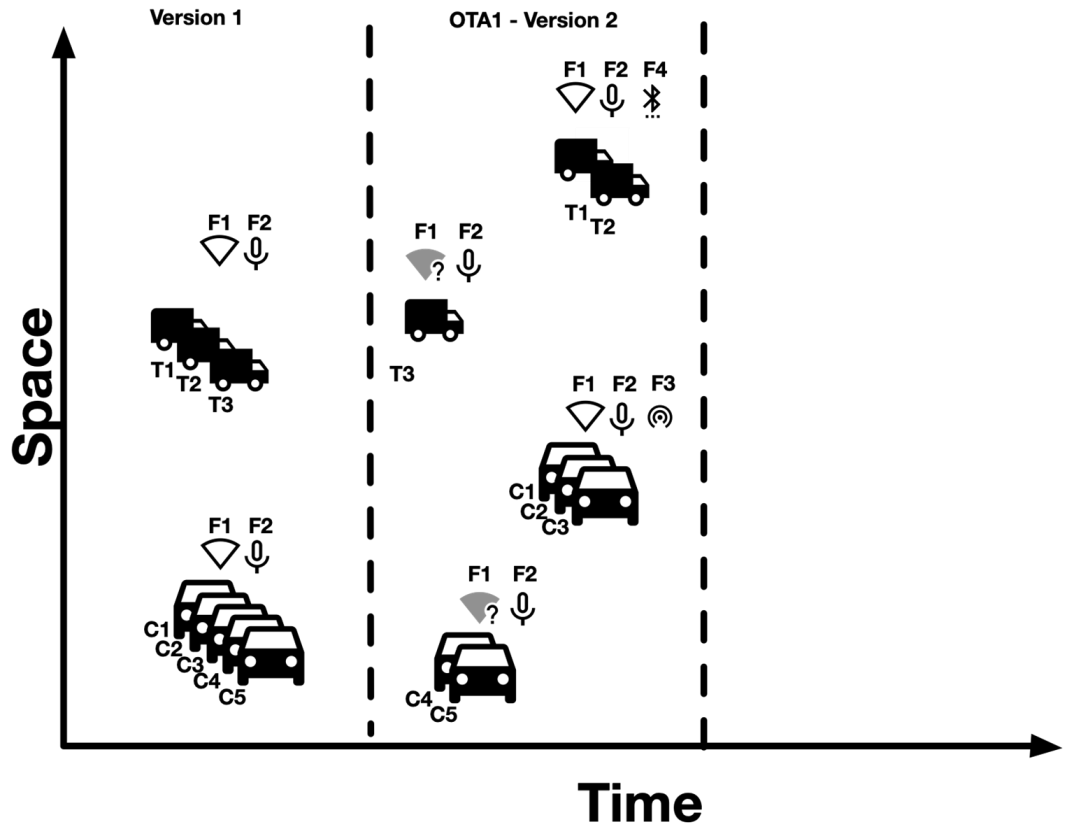
With OTA, the number of software configurations will dramatically increase

1. Some users will not install all the updates
2. Some users will have limited wifi connectivity
3. OTA updates are enabling more frequent updates of (smaller) software components
4. End-user customization and the use of third-party apps add to the complexity

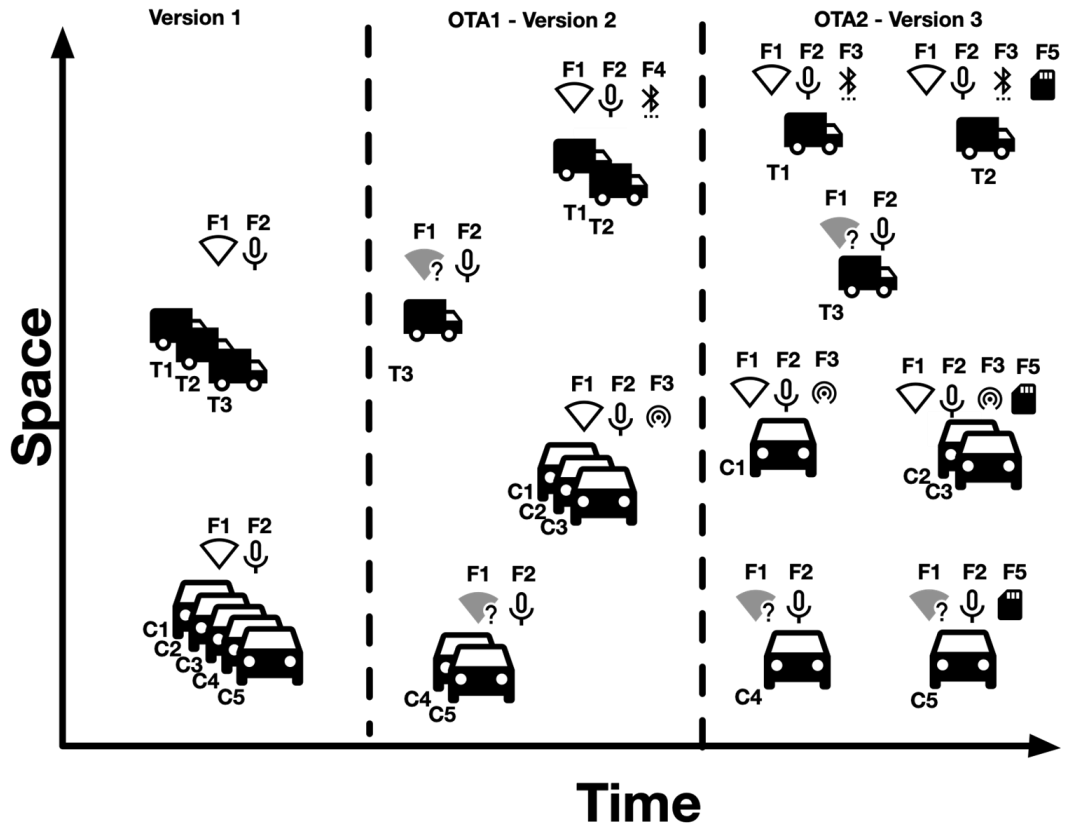
Over-the-air (OTA) Updates: Example



Over-the-air (OTA) Updates: Example



Over-the-air (OTA) Updates: Example



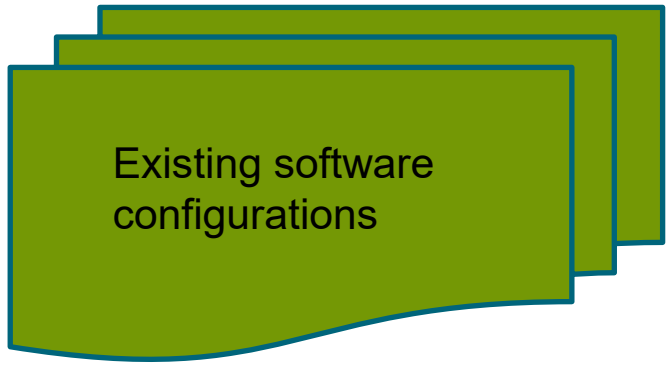
So we have safety-critical product lines in time and space that need assuring!

Main points

1. Assurance cases combine argument and evidence, allow to contextualize analysis and verification. Need to be reviewable
2. OTA updates yield product lines in time and space which need assuring

Challenges of OTA Assurance for Safety-Critical Product Lines

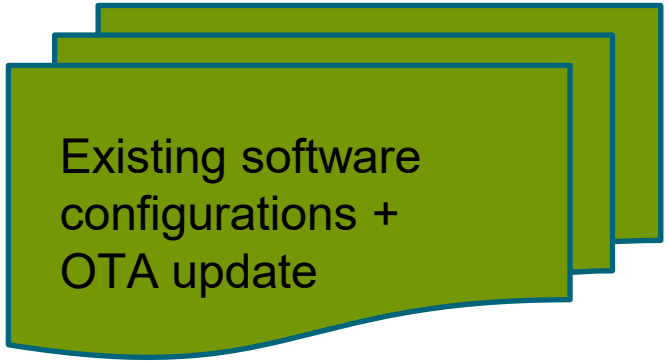
Aim to use the notion of product lines to represent variability in space (different configurations) and time (different updates) and assure them together



How to verify the update?
How to assure the update?
How to test the update?

What information to keep about each feature?
What information to keep about the entire configuration?

so that



… is safe for each configuration
… does not fail in each configuration

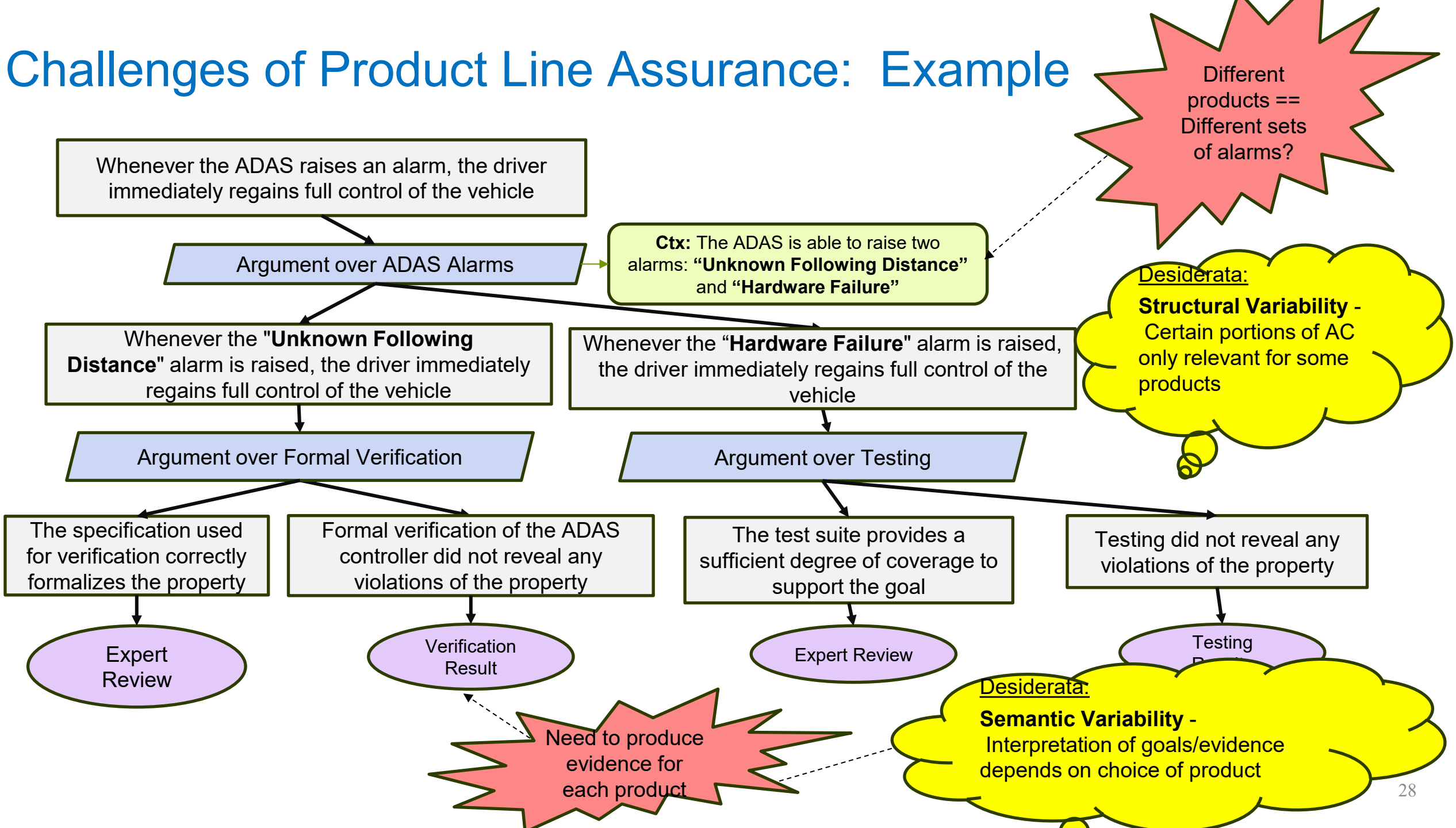
What can be proven? How can potential failures be avoided at runtime?

What does “assuring them together” mean?

1. Represent the assurance case for the product space compactly
2. Reuse verification and other evidence across similar products
3. Identify assurance-relevant variation points
4. Enable analysis of completeness of assurance across the entire product space
5. Analyze impact of a change across the entire product space

Note: some of these questions have been answered for individual products and can be “lifted” for a product-line level.

Challenges of Product Line Assurance: Example

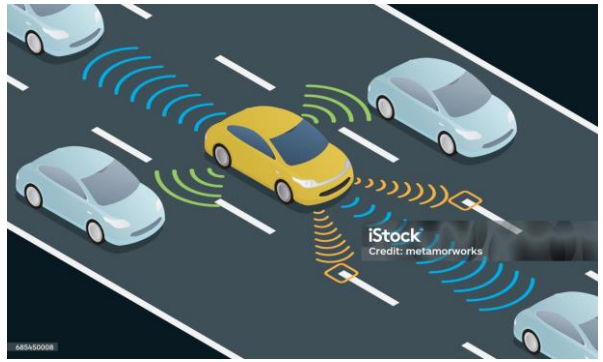


ADAS -- Product Line Version

Representation of a *family* of vehicles with different configurations of ADAS features

Features

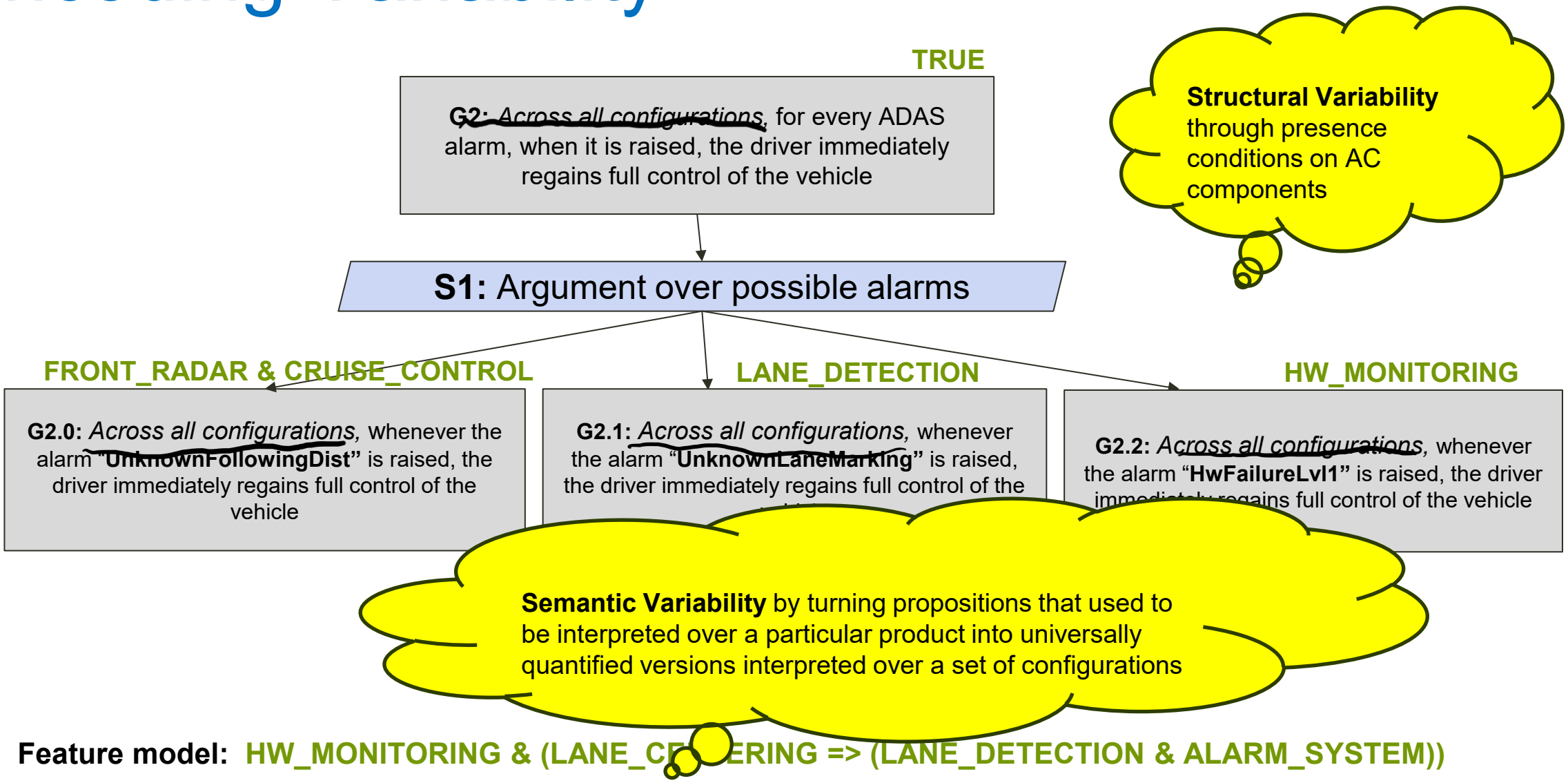
- HW_MONITORING
- LANE_DETECTION
- LANE_CENTERING
- FRONT_RADAR
- ALARM_SYSTEM



Feature model: HW_MONITORING & (LANE_CENTERING => (LANE_DETECTION & ALARM_SYSTEM))

State machine mode becomes annotated with presence conditions

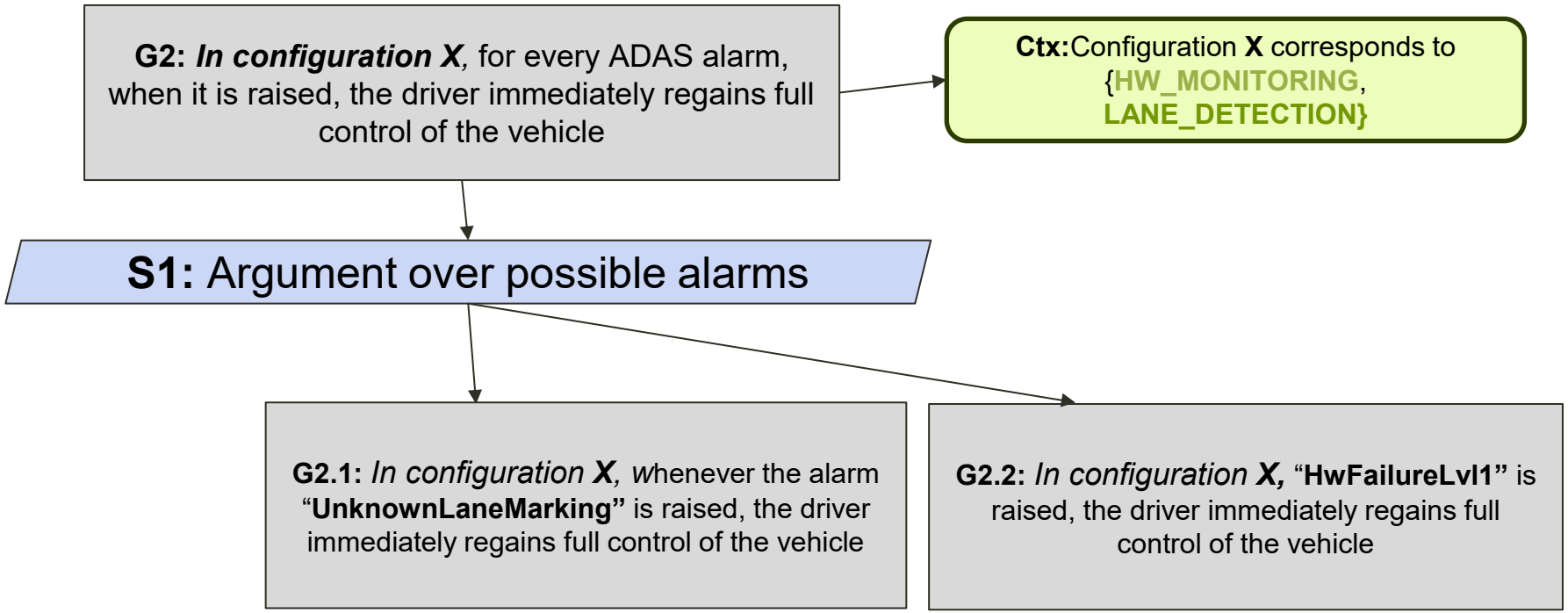
Encoding Variability



Feature model: HW_MONITORING & (LANE_CE ERING => (LANE_DETECTION & ALARM_SYSTEM))

Logan Murphy, Torin Viger, Alessio Di Sandro, Marsha Chechik: PLACIDUS: Engineering Product Lines of Rigorous Assurance Cases. IFM 2024: 87-108

Instantiation for LANE_DETECTION & HW_MONITORING



Feature model: HW_MONITORING & (LANE_CENTERING => (LANE_DETECTION & ALARM_SYSTEM))

Logan Murphy, Torin Viger, Alessio Di Sandro, Marsha Chechik: PLACIDUS: Engineering Product Lines of Rigorous Assurance Cases. IFM 2024: 87-108

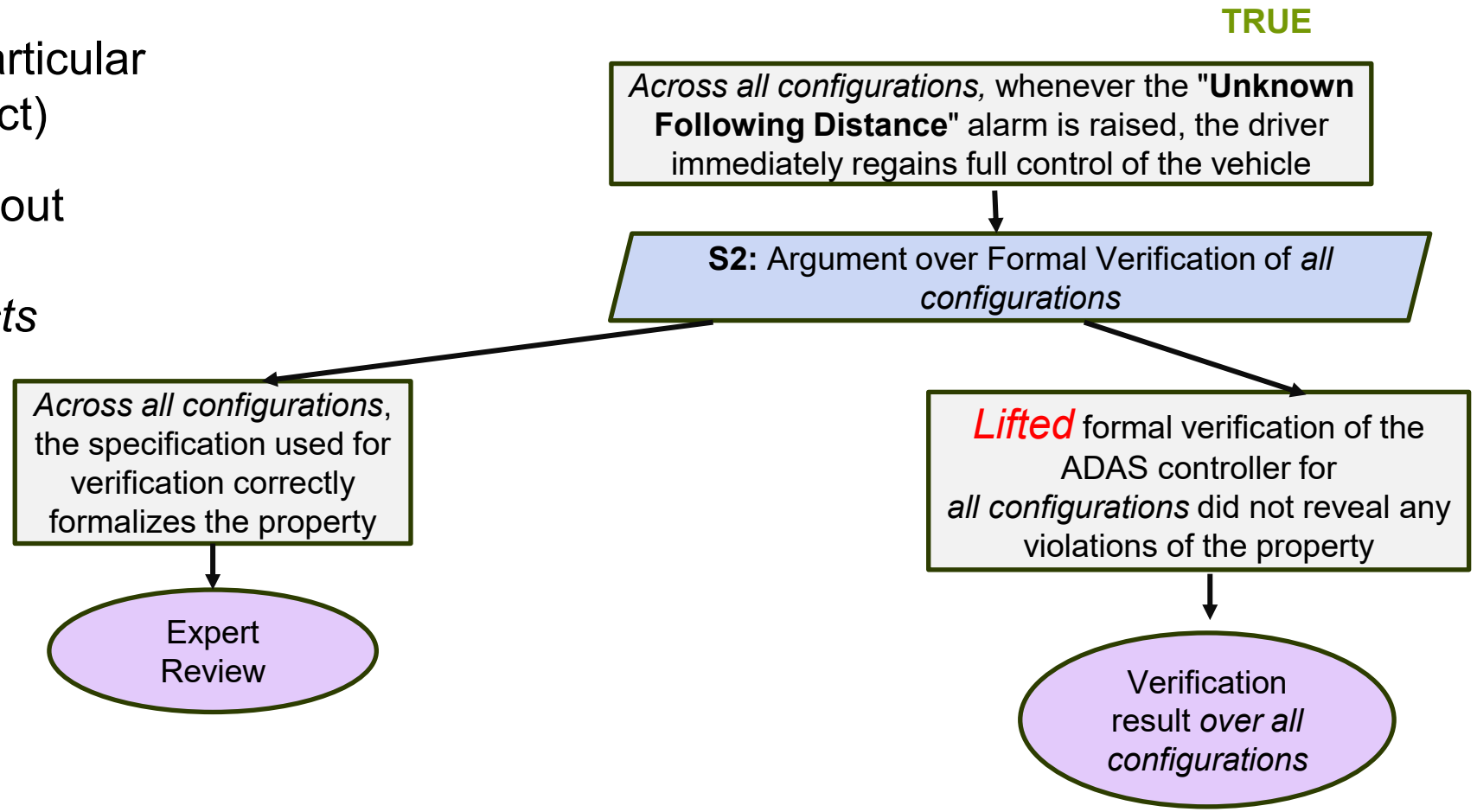
This was about “product lining” goals. What about evidence?

Encoding Variability

Regular evidence is for a particular goal (and a particular product)

Aim: evidence that talks about the *whole set of goals* representing *a set of products*

Thus, we interpret goals and evidence over sets of products



Feature model: **HW_MONITORING & (LANE_CENTERING => (LANE_DETECTION & ALARM_SYSTEM))**

Logan Murphy, Torin Viger, Alessio Di Sandro, Marsha Chechik: PLACIDUS: Engineering Product Lines of Rigorous Assurance Cases. IFM 2024: 87-108

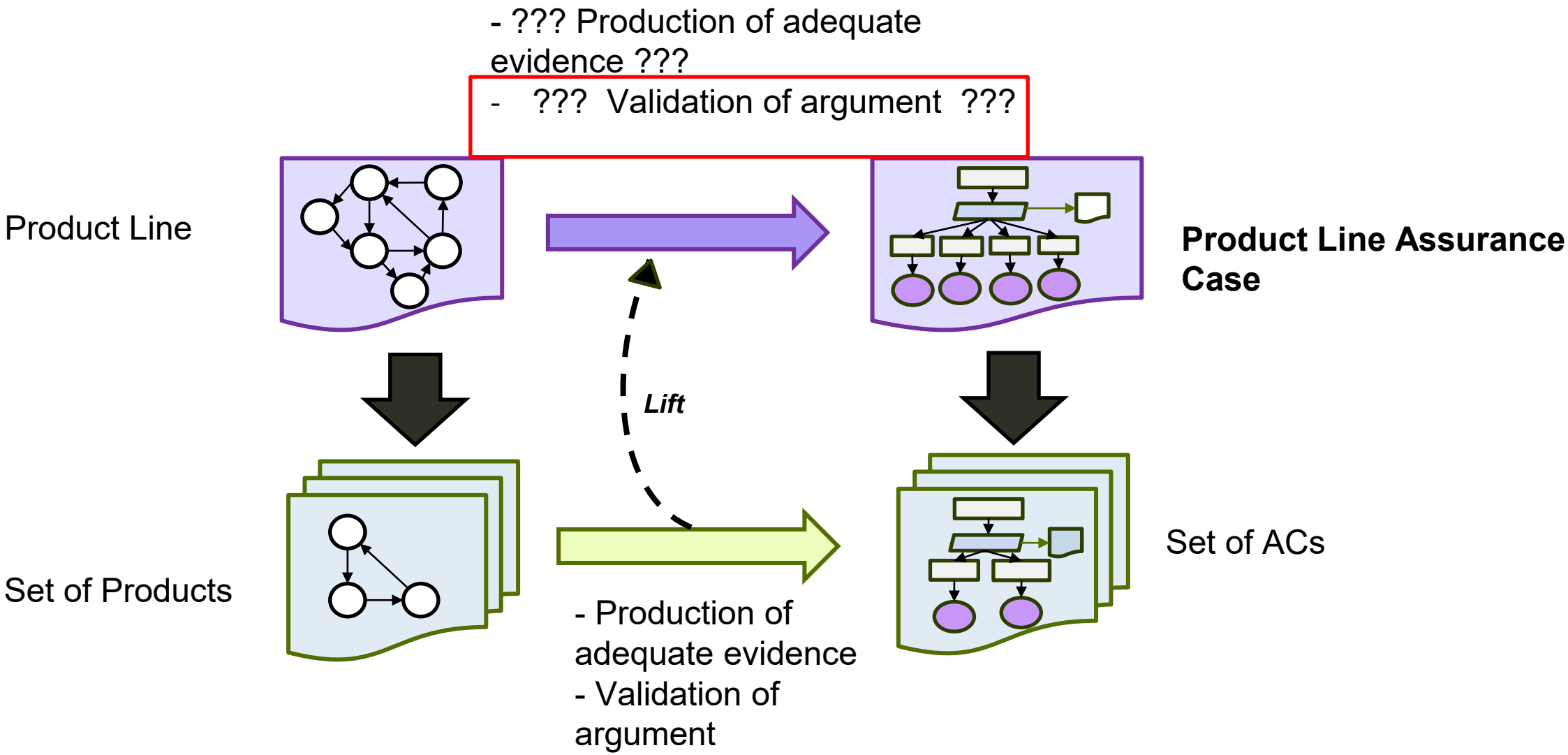
Main points

1. Assurance cases combine argument and evidence, allow to contextualize analysis and verification. Need to be reviewable
2. OTA updates yield product lines in time and space which need assuring
3. To assure product lines, reinterpret arguments and evidence to apply to sets of products, developing PLAS (product line assurance case)

Assuring Product Lines of Complex Systems -- Talk Plan

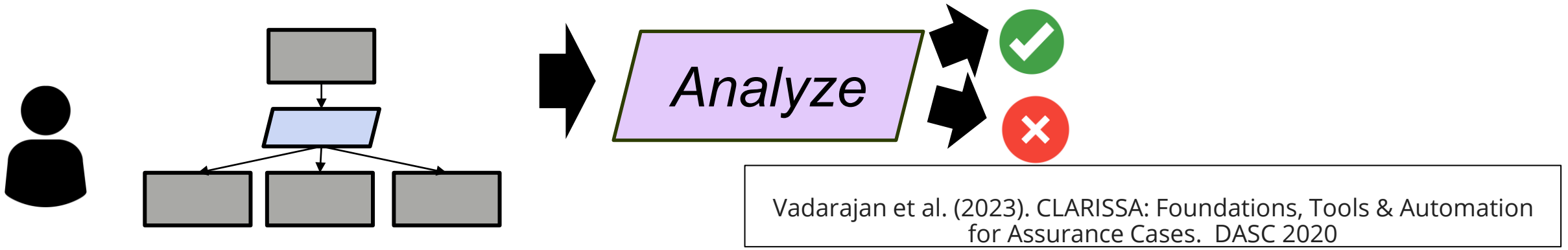
- Motivation and goals
- Background
  - Assurance
  - Product lines – variability in space and time
- Representation: Product Line Assurance Cases (PLAC)
- Development of PLACs
- Assessing Change of PLACs
- Tooling
- Summary and Next Steps

Development of Product Line Assurance



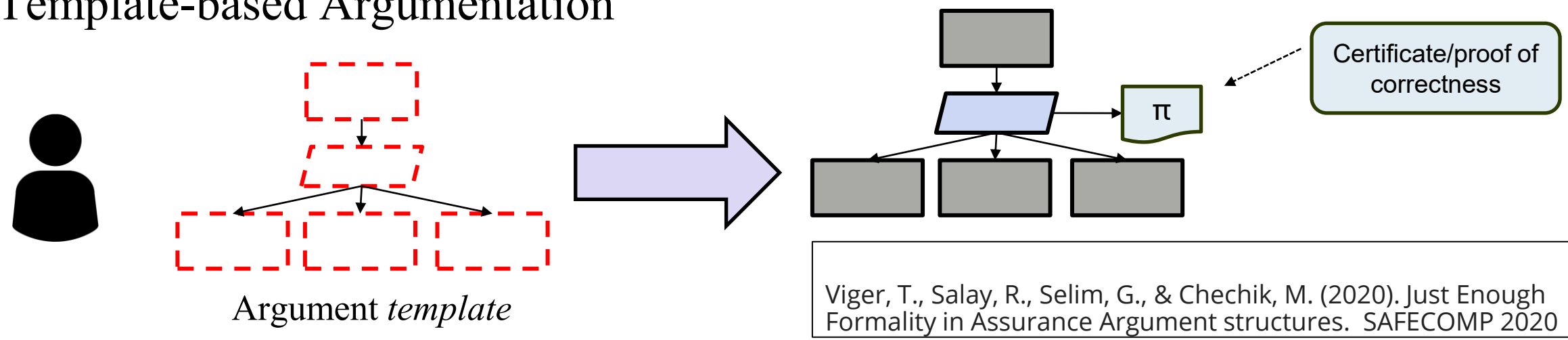
Validating Assurance Arguments

Post hoc validation



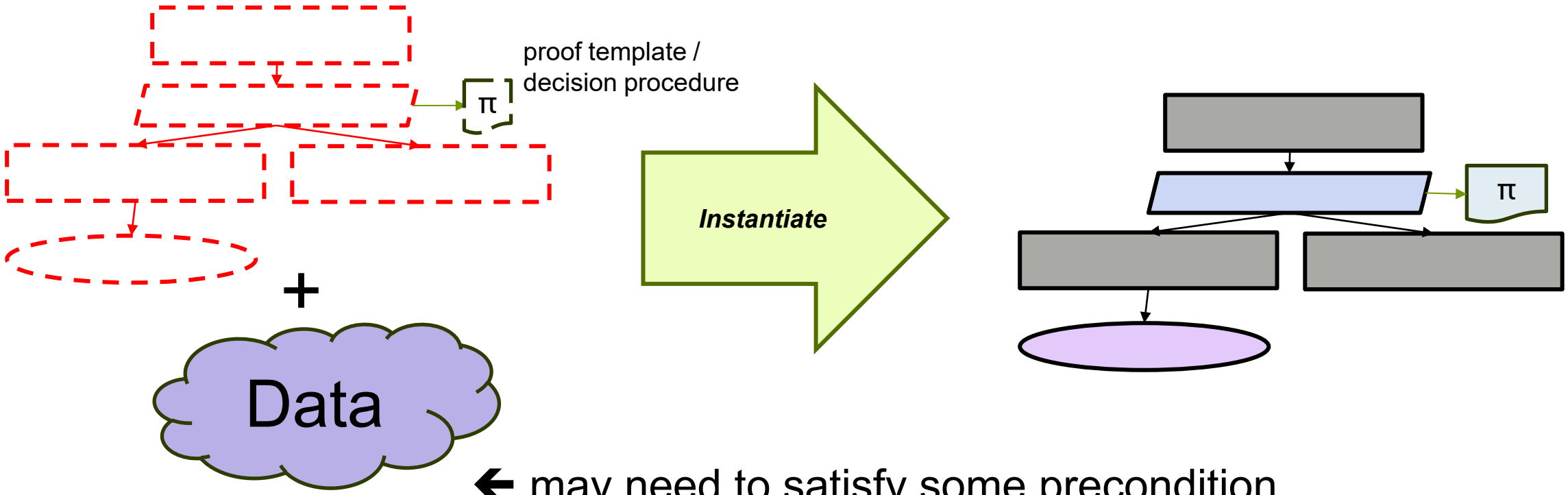
Vadarajan et al. (2023). CLARISSA: Foundations, Tools & Automation for Assurance Cases. DASC 2020

Template-based Argumentation



Viger, T., Salay, R., Selim, G., & Chechik, M. (2020). Just Enough Formality in Assurance Argument structures. SAFECOMP 2020

Formal Assurance Case Template

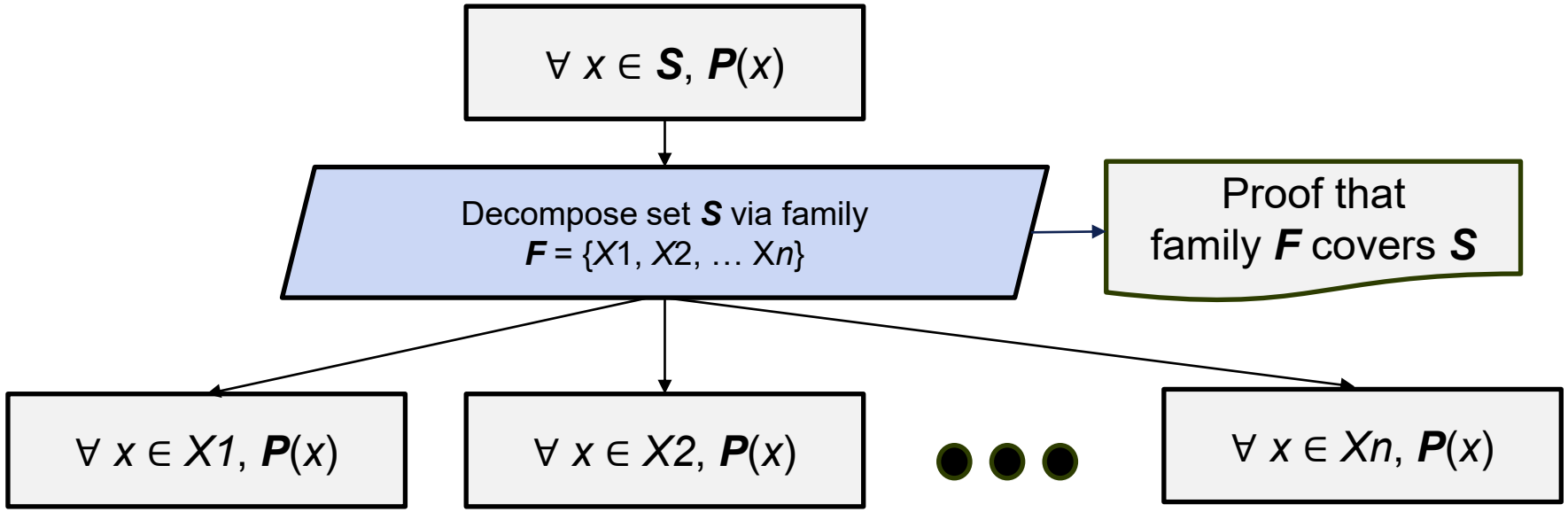


🡐 may need to satisfy some precondition

Template is **valid** iff every instantiation satisfying precondition results in a sound argument

Formal Assurance Case Template – Example

Domain decomposition template

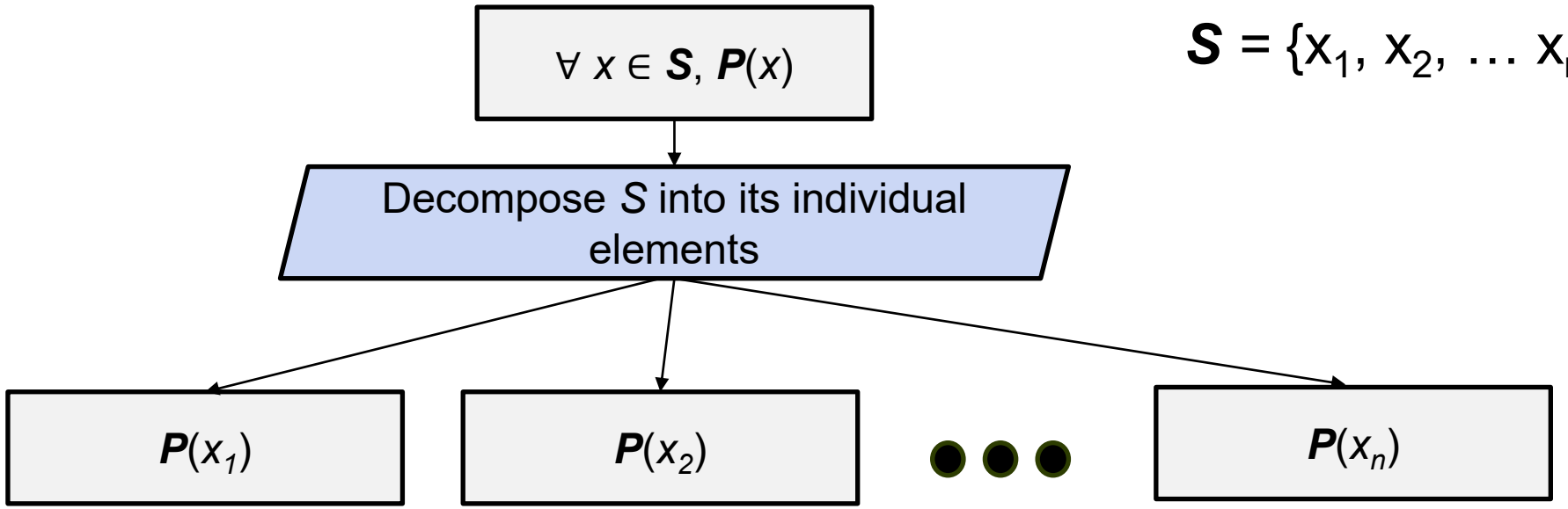


Instantiation requires specifying set ***S,*** family ***F,*** property ***P***

Viger, T., Salay, R., Selim, G., & Chechik, M. (2020). Just Enough Formality in Assurance Argument structures. SAFECOMP 2020

Special Case of Domain Decomposition

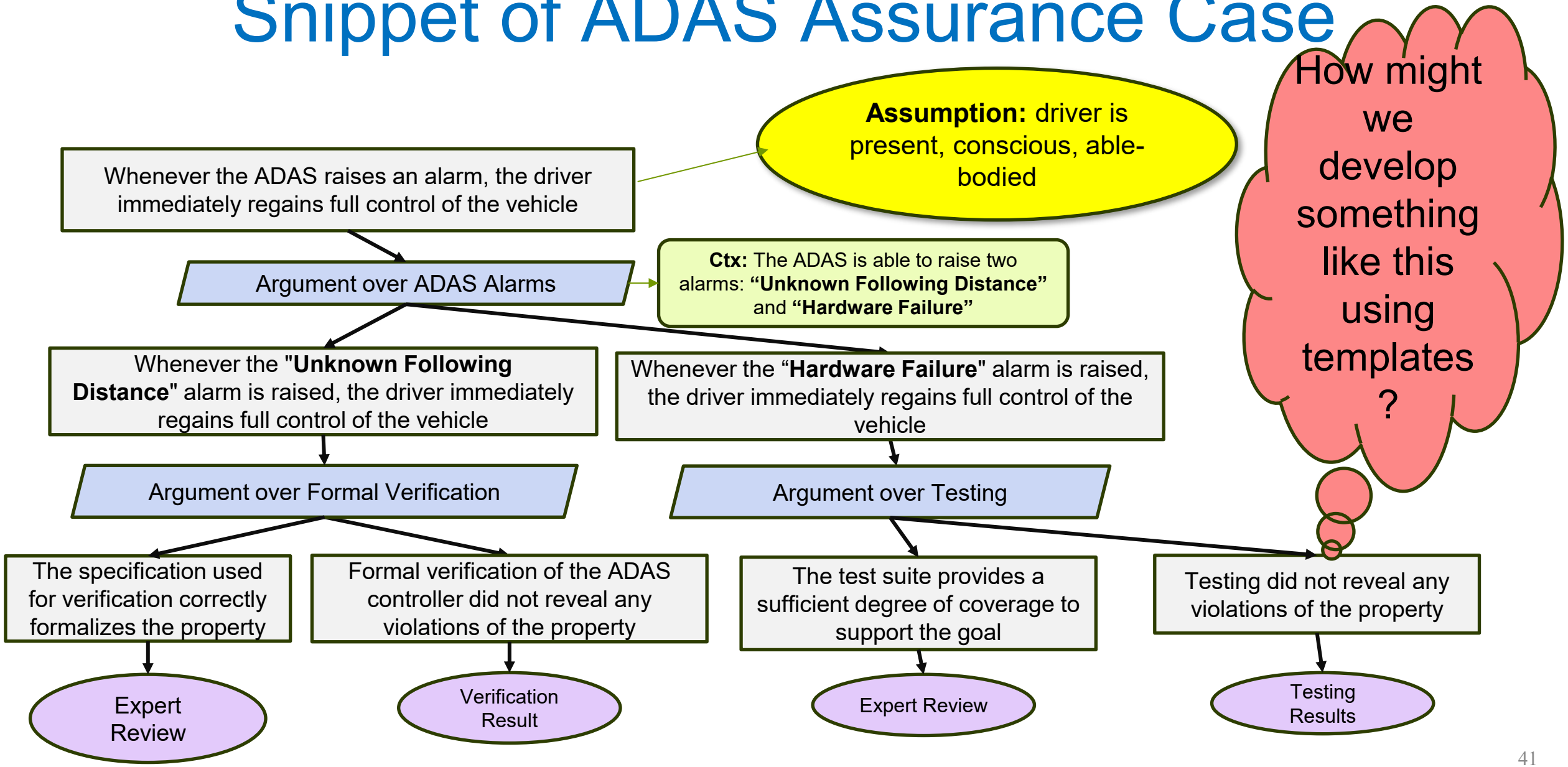
Enumeration template



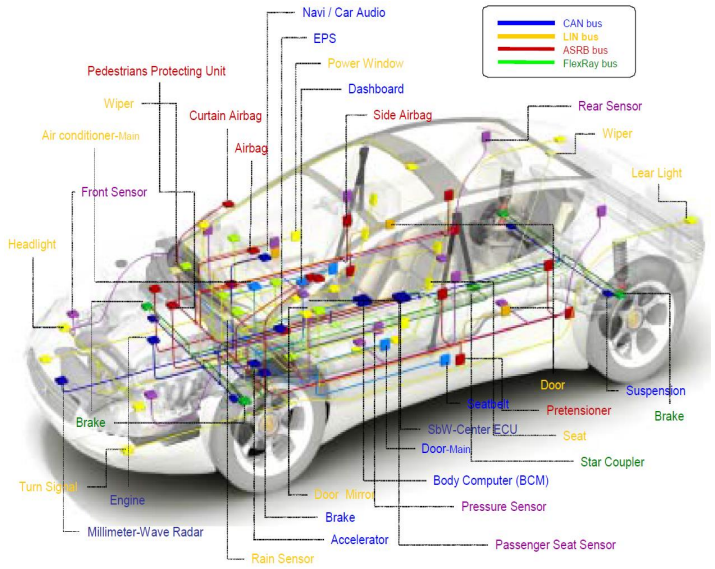
$S = \{x_1, x_2, \ldots x_n\}$

Instantiation requires specifying set ***S,*** property ***P***

Snippet of ADAS Assurance Case



Generalized Assurance Strategy



System (model)

Set of system alarms

Alarm 1 → Test/ Verify

Alarm 2 → Test/ Verify

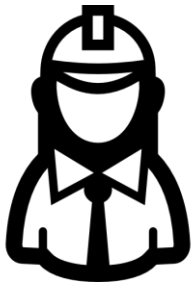
Alarm 3 → Test/ Verify

Alarm *n* → Test/ Verify

Identification of alarms (maybe by querying)

Decomposition into individual alarm scenarios

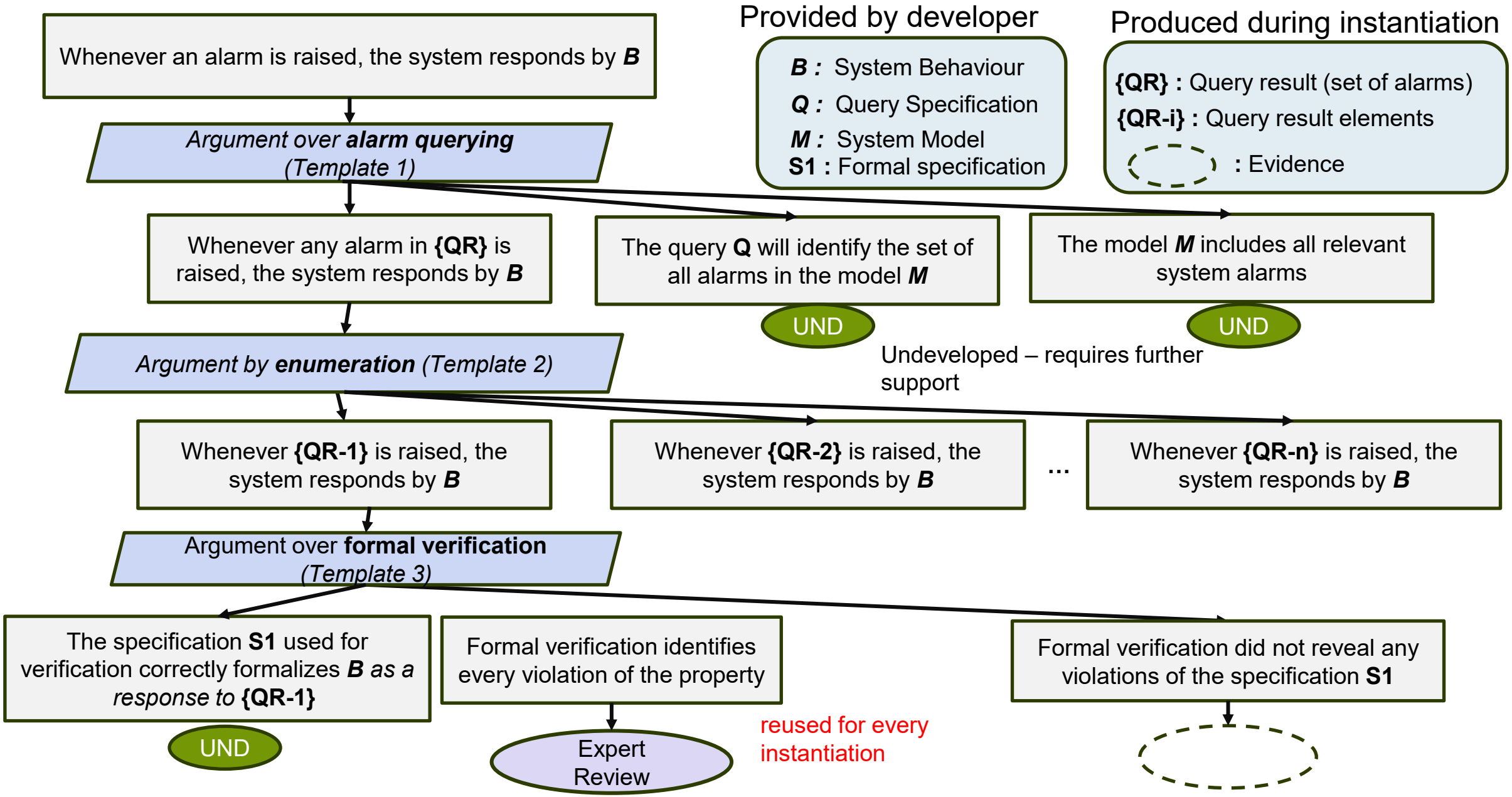
Verification of individual alarm scenarios



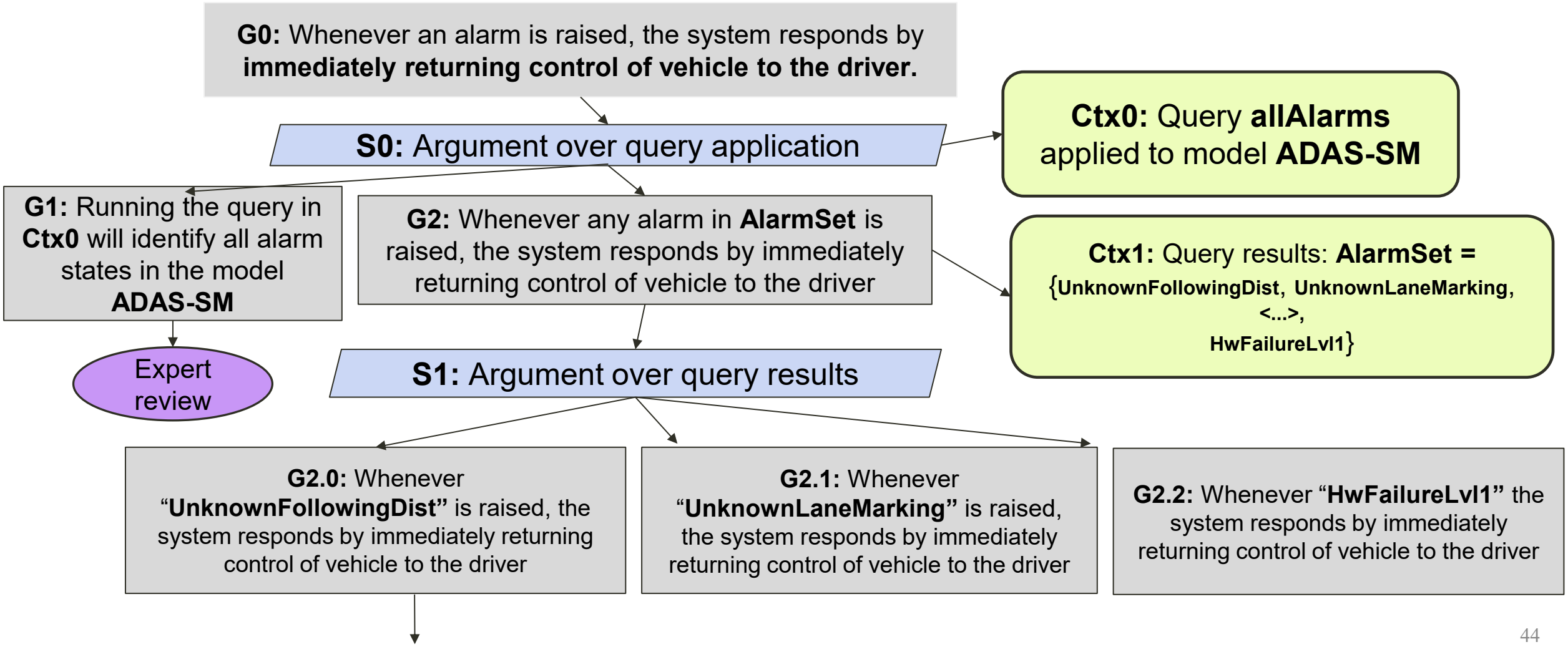
Assurance Engineer

Template-able

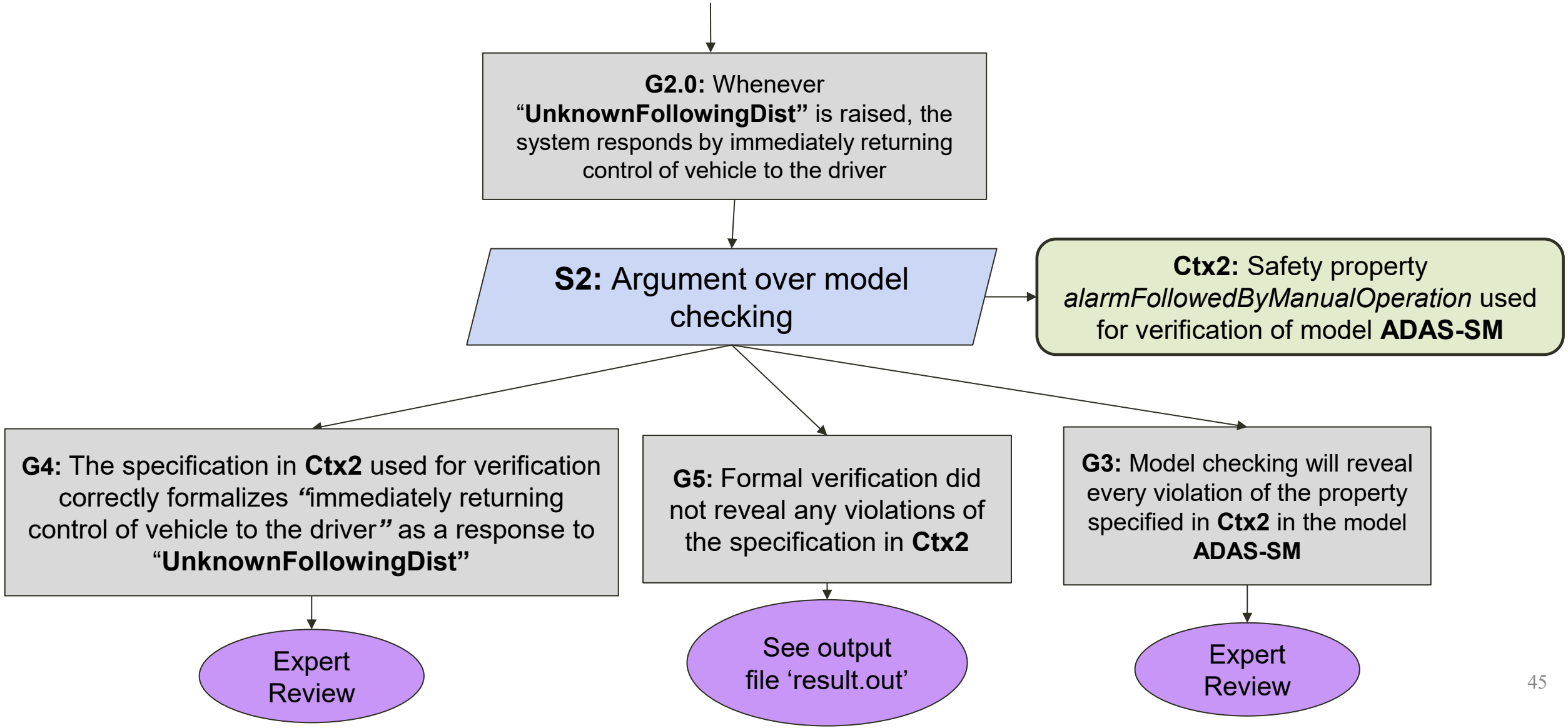
Templates for the ADAS AC



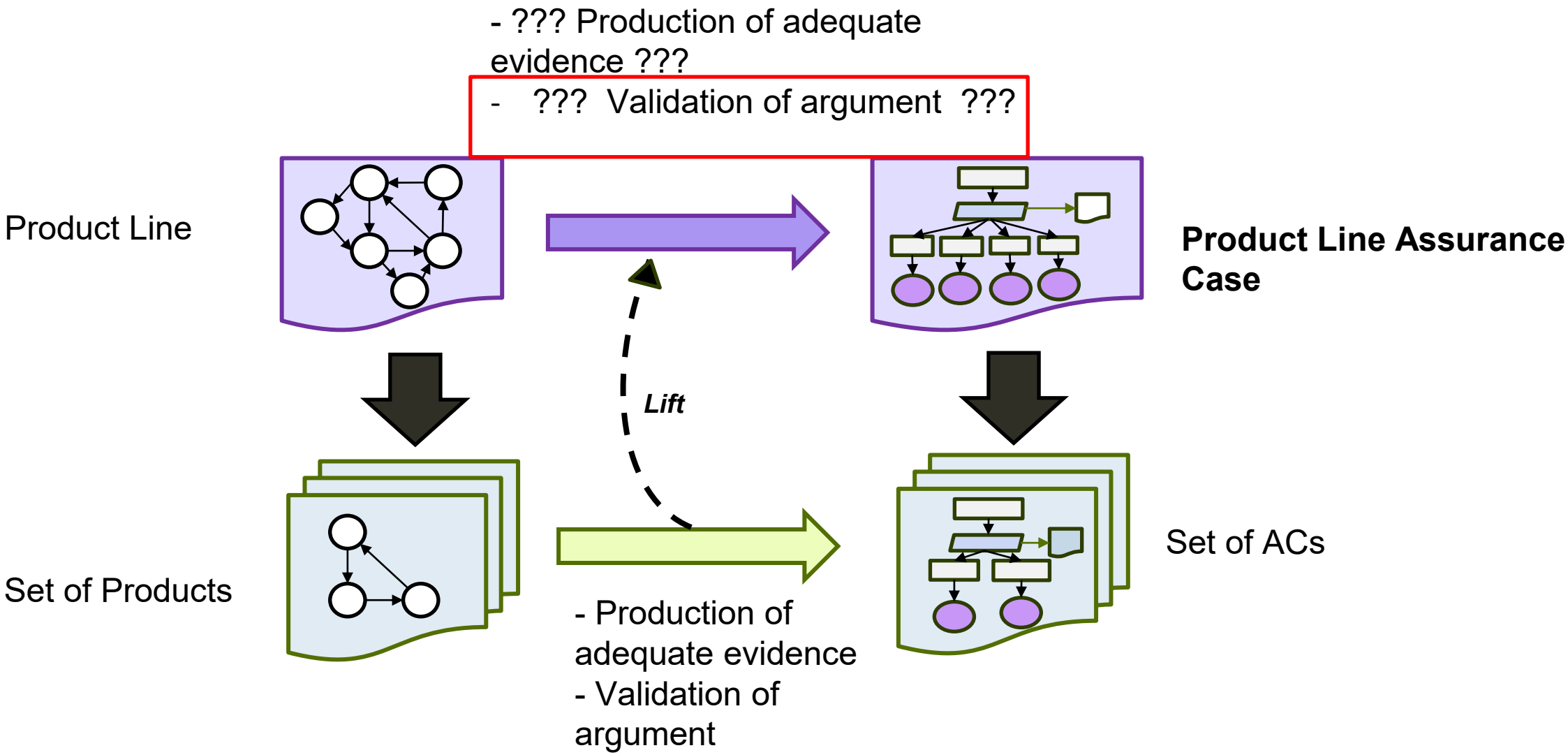
Template 1 and 2 Instantiation for ADAS AC



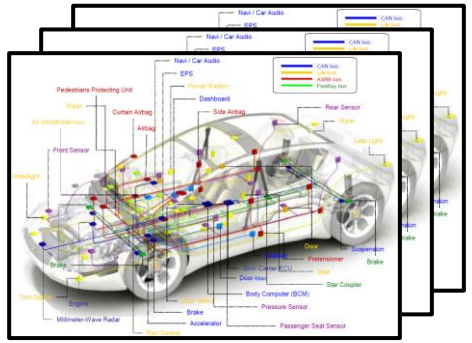
Template 3 Instantiation for ADAS AC



Development of Product Line Assurance



Functional View of AC development

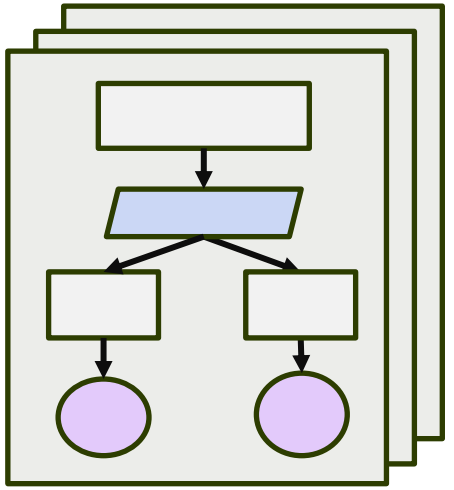


Product Line System (model)

Lifted Querying Template

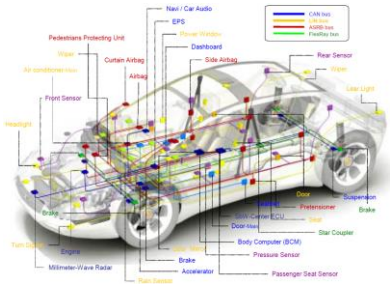
Lifted Decomposition Template

Lifted Verification Template



Product Line Assurance Case

Each template is just a composable function. These can be lifted and validated individually, resulting in desired composition



Product System (model)

Querying Template

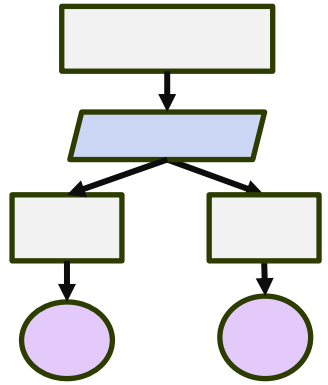
depends on running analysis

Decomposition Template

"syntactic"

Verification Template

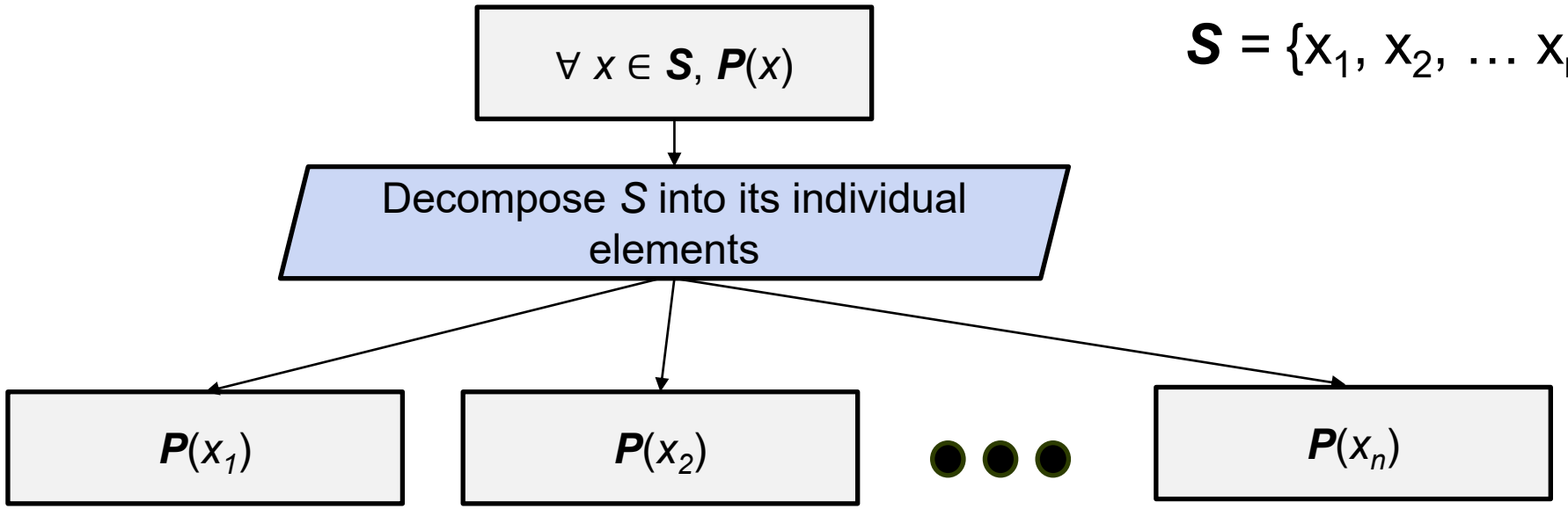
depends on running analysis



Product Assurance Case

Special Case of Domain Decomposition

Enumeration template



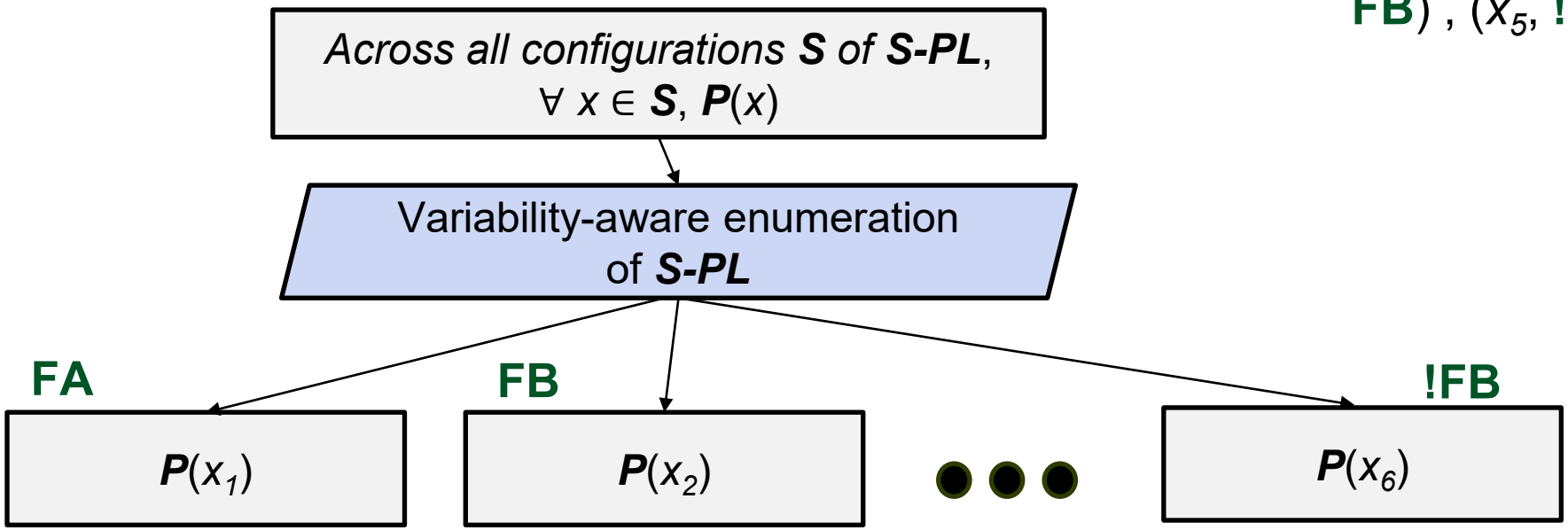
$\boldsymbol{S} = \{x_1, x_2, \ldots x_n\}$

Special Case of Domain Decomposition

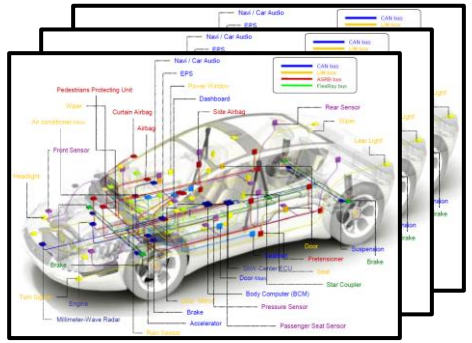
Lifted Enumeration template

Turns *semantic* variability into *syntactic* variability

S-PL = { (x_1, **FA**) , (x_2, **FB**) , (x_3, **FA & FB**), (x_4, **FA & FB**) , (x_5, **!FA**) , (x_6, **!FB**) }



Functional View of AC development

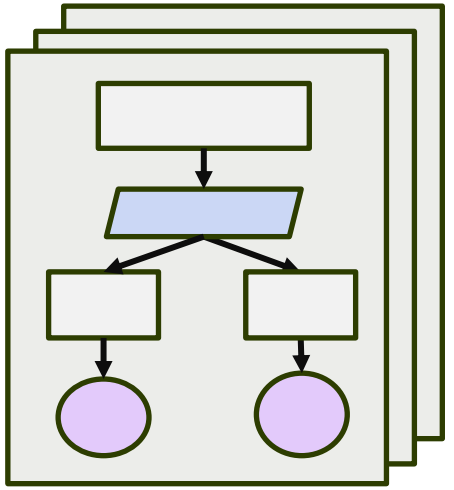


Product Line System (model)

Lifted Querying Template

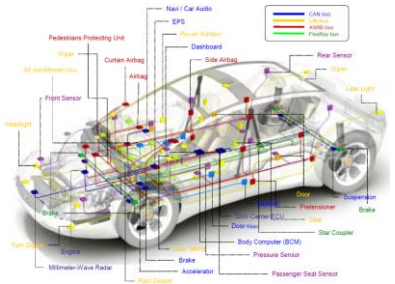
Lifted Decomposition Template

Lifted Verification Template



Product Line Assurance Case

Each of templates is just a function. These can be lifted and validated individually



Product System (model)

Querying Template

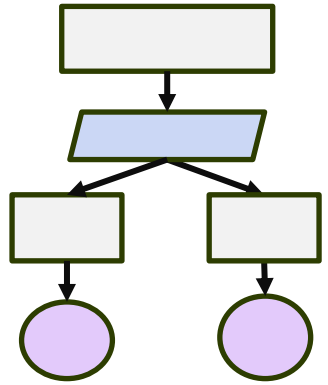
depends on running analysis ("analytic")

Decomposition Template

"syntactic"

Verification Template

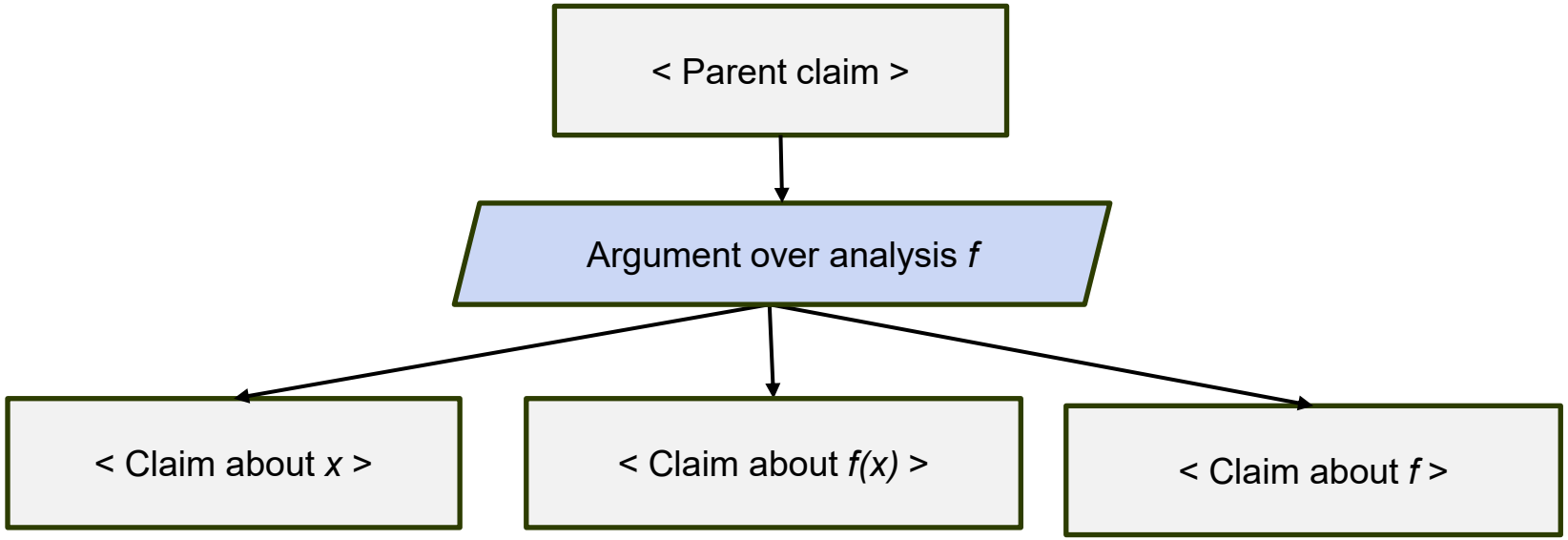
depends on running analysis ("analytic")



Product Assurance Case

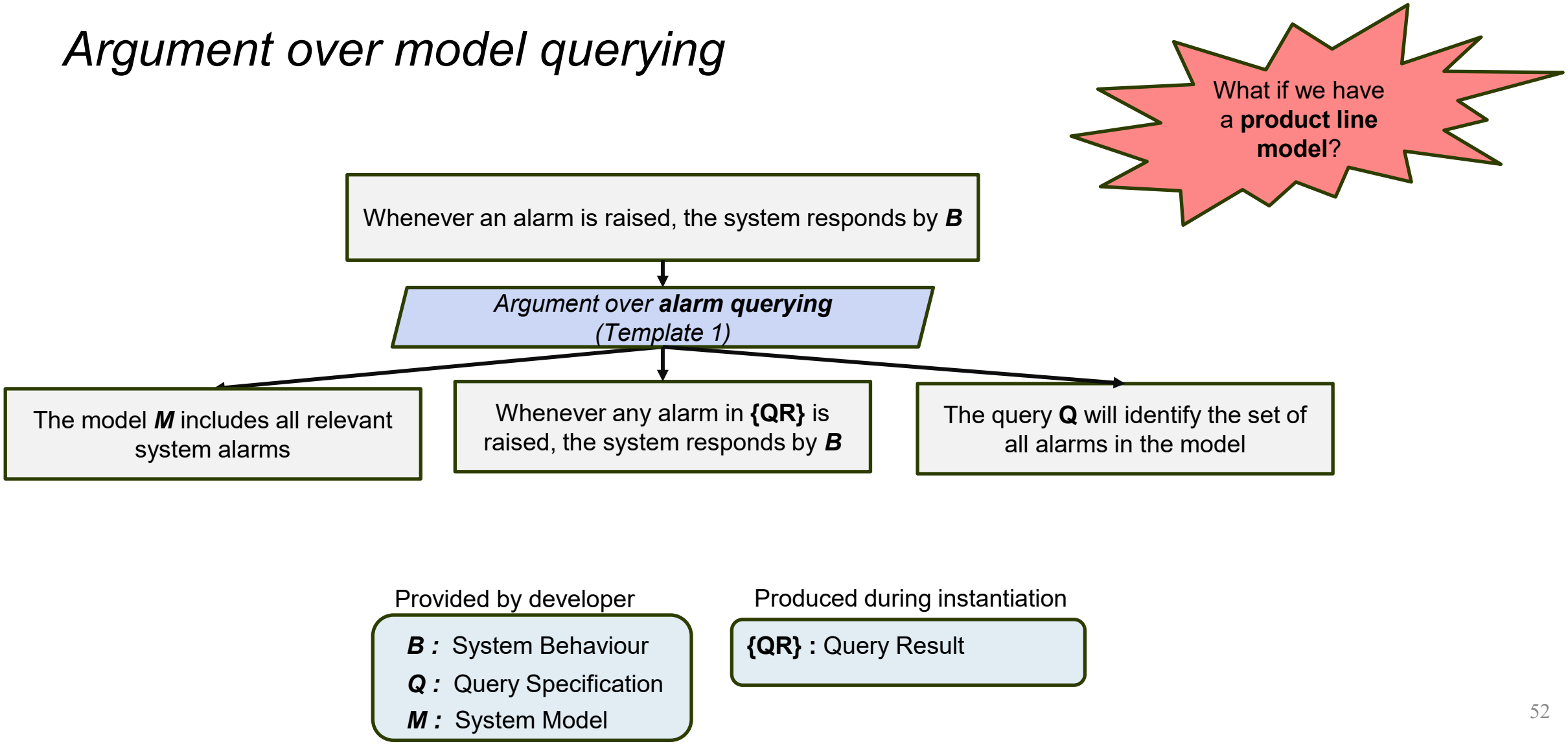
Analytic Argumentation

Idea: Support a goal by applying some analysis *f* to some object *x* and make some assertion about the result.



Analytic Template - Example

Argument over model querying

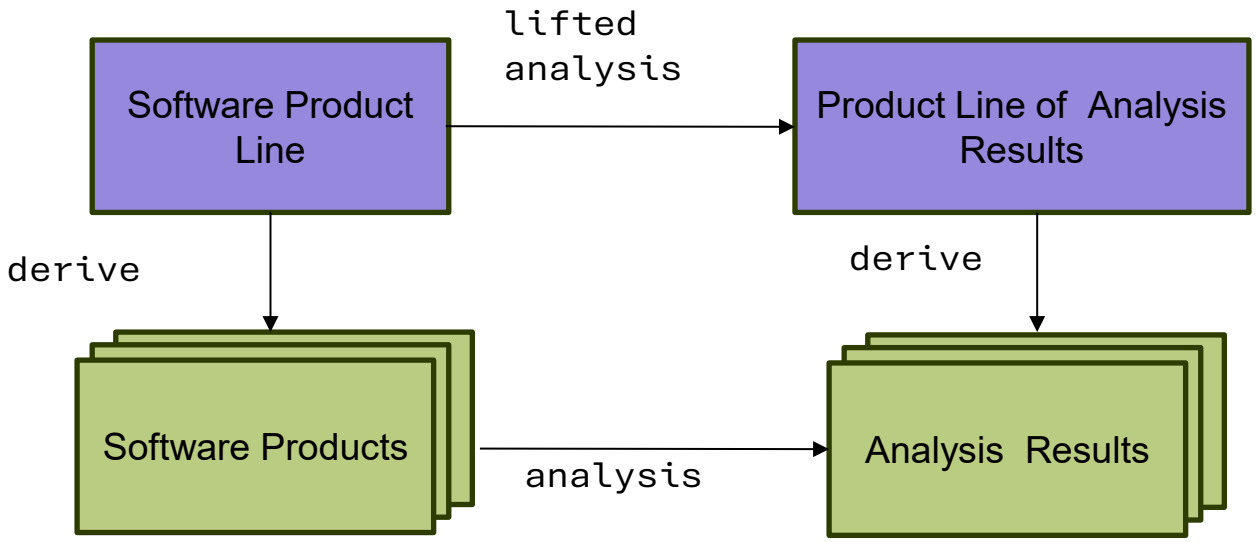


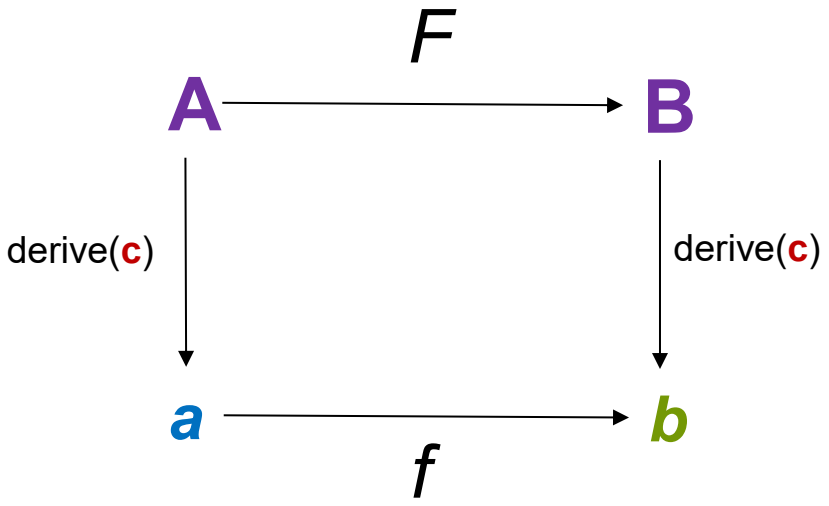
Aside: Lifted Software Analysis

Recall: Given n features, $O(2^n)$ distinct products!

Cannot analyze a PL in a product-by-product fashion

Idea: Redefine (lift) the analysis to give a "product-line" of analysis results



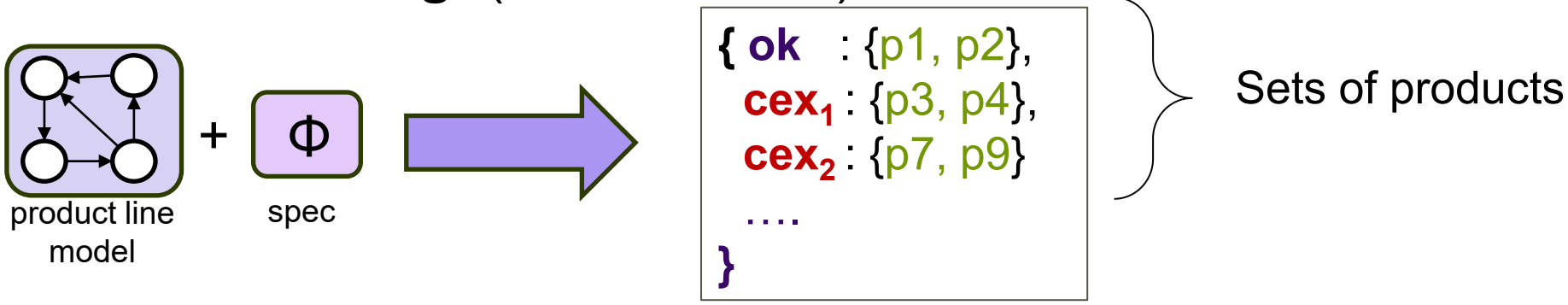


Aside: Lifted Software Analysis (Example)

Standard Model Checking:

Extensive literature: Lifted parsing, type-checking, abstract interpretation, dataflow analysis, synthesis…

Lifted Model Checking: (Classen et al.)



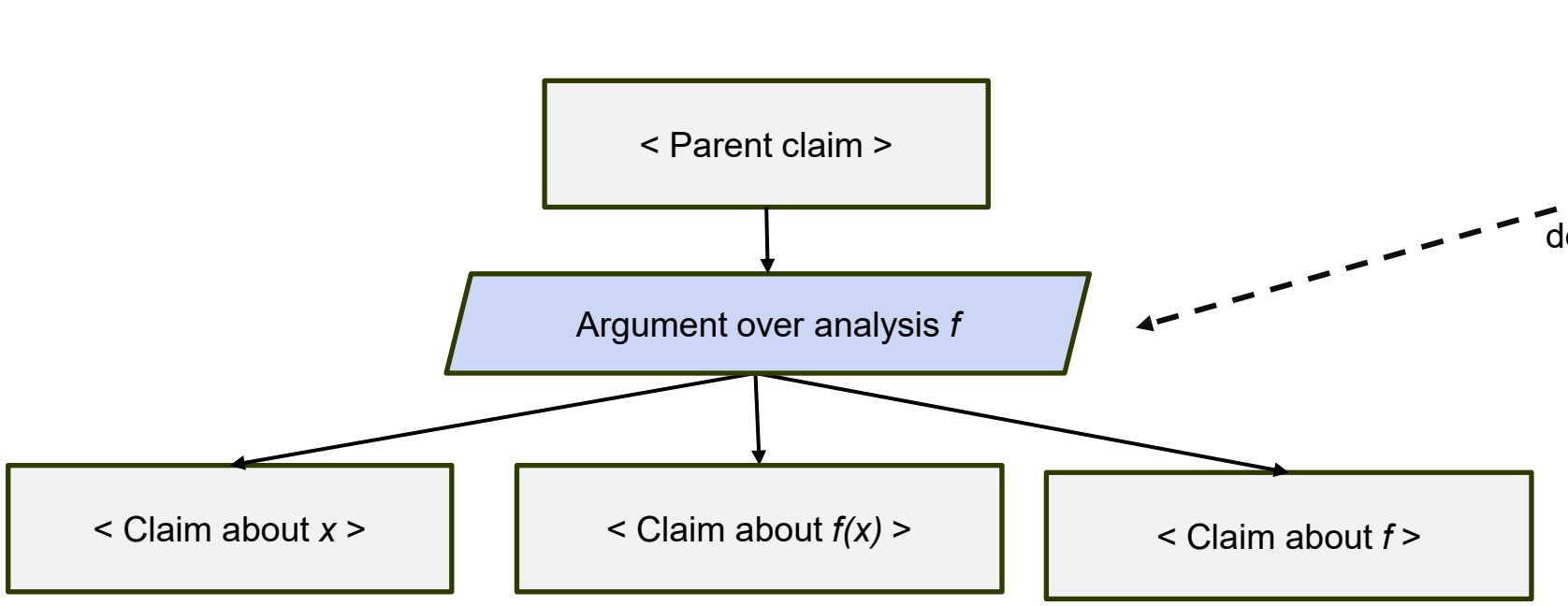
Classen, A. et al. (2012). Model checking software product lines with SNIP. *International Journal on Software Tools for Technology Transfer*

Aside Continues:
A long history of lifting analyses in my group

- Lifting transformations [ICSE'14, ICMT'15] and queries [VaMoS'23]
- Lifting analyses written in functional languages
  - Lifting Datalog [FSE'19, PADL'19, TSE'23]
  - Applications to GM controllers [MODELS'20, EMSE'23]
  - Lifting PCF+ [OOPSLA'20]
- Lifting change impact analysis and formally verifying correctness of lifting [SAFECOMP'21]
- Lifting software equivalence checking [SPLC'23a]

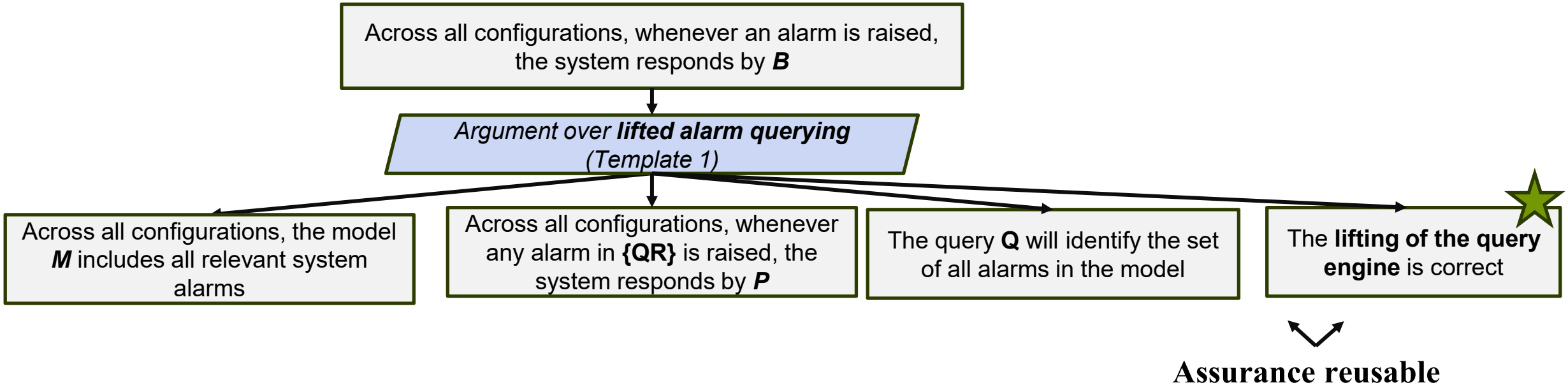
Lifting Analytic Templates

Idea: Given an argument over *f*, can we use the same argument using a lift of *f* ?



Lifted Analytic Argumentation

Theorem: For any analytic template over analysis f, if f is lifted as F, then replacing f with F preserves validity of the template.



Provided by developer

B : System Behaviour
Q : Query Specification
M : Product Line Model

Produced during instantiation

{QR} : Lifted Query Result (variational set)

If template is shown to be valid at product level, it is automatically valid at the product line level!!!!

Main points

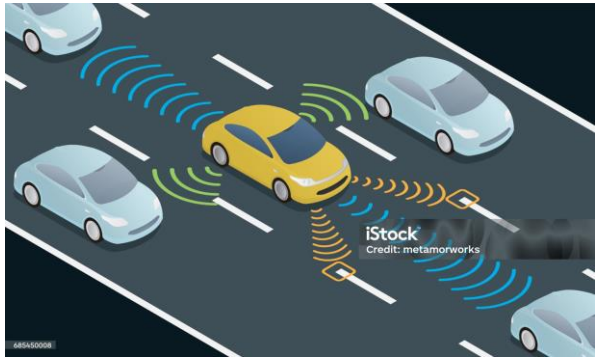
1. Assurance cases combine argument and evidence, allow to contextualize analysis and verification. Need to be reviewable
2. OTA updates yield product lines in time and space which need assuring
3. To assure product lines, reinterpret arguments and evidence to apply to sets of products
4. Assurance cases can be defined using analytic templates which can be lifted, preserving correctness, and composed

ADAS -- Product Line Version

Representation of a *family* of vehicles with different configurations of ADAS features

Features

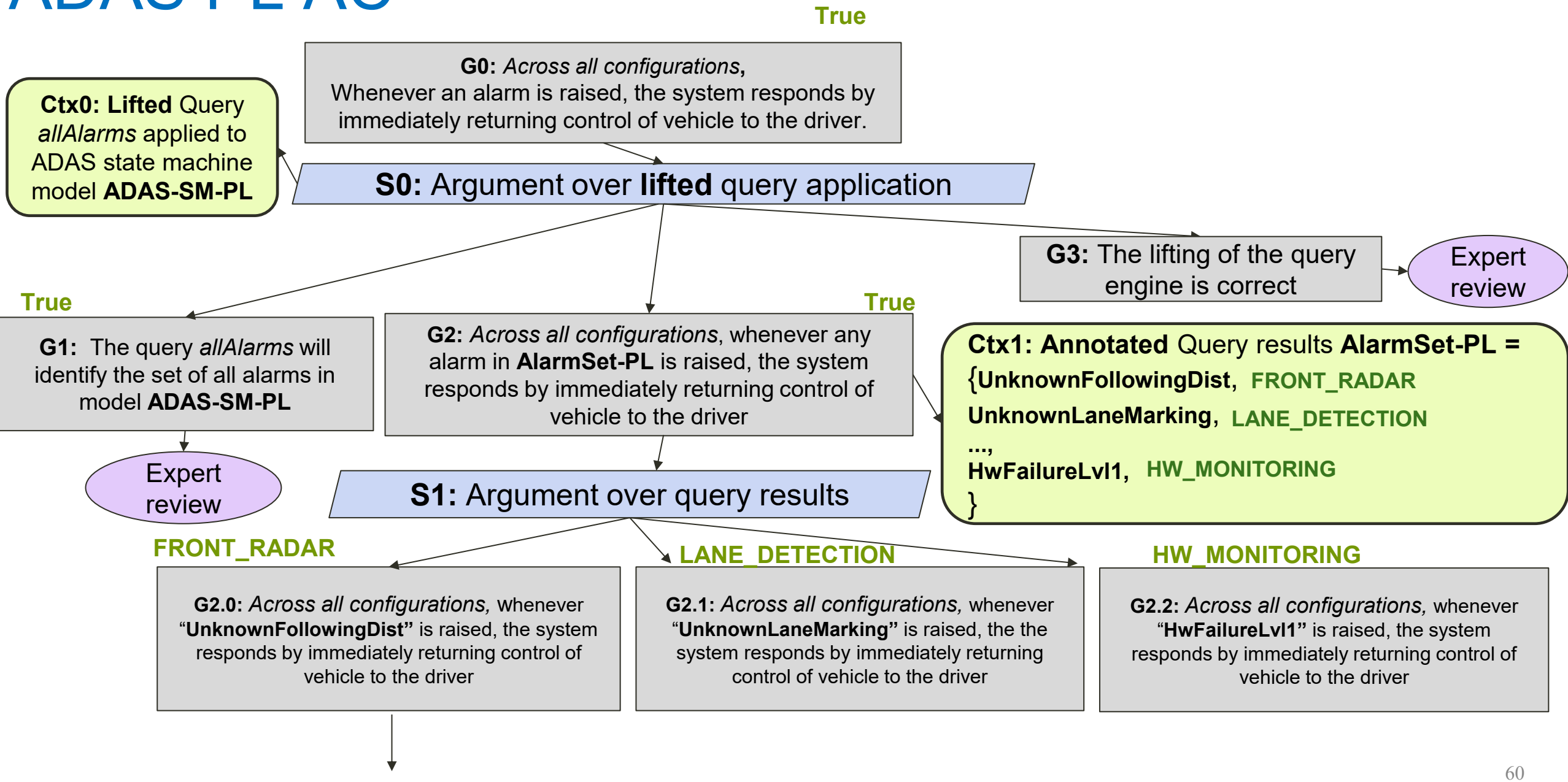
- HW_MONITORING
- LANE_DETECTION
- LANE_CENTERING
- FRONT_RADAR
- ALARM_SYSTEM



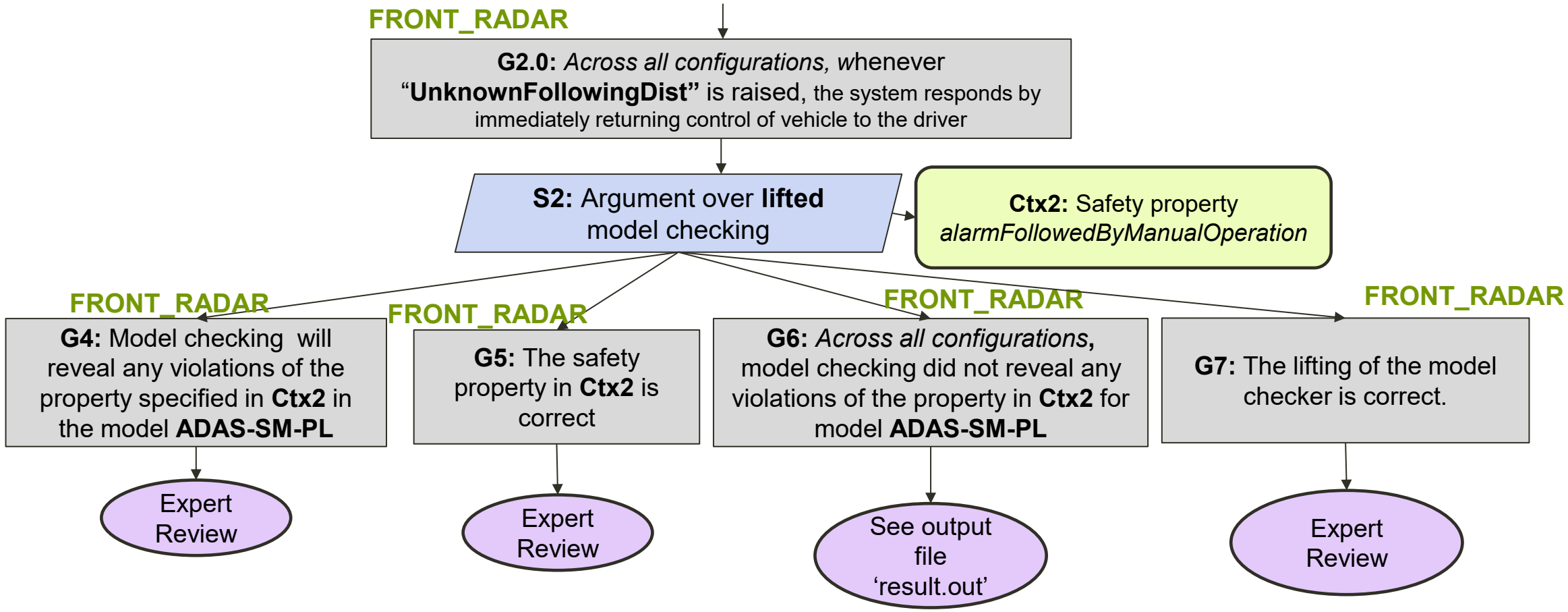
Feature model: HW_MONITORING & (LANE_CENTERING => (LANE_DETECTION & ALARM_SYSTEM))

State machine mode becomes annotated with presence conditions

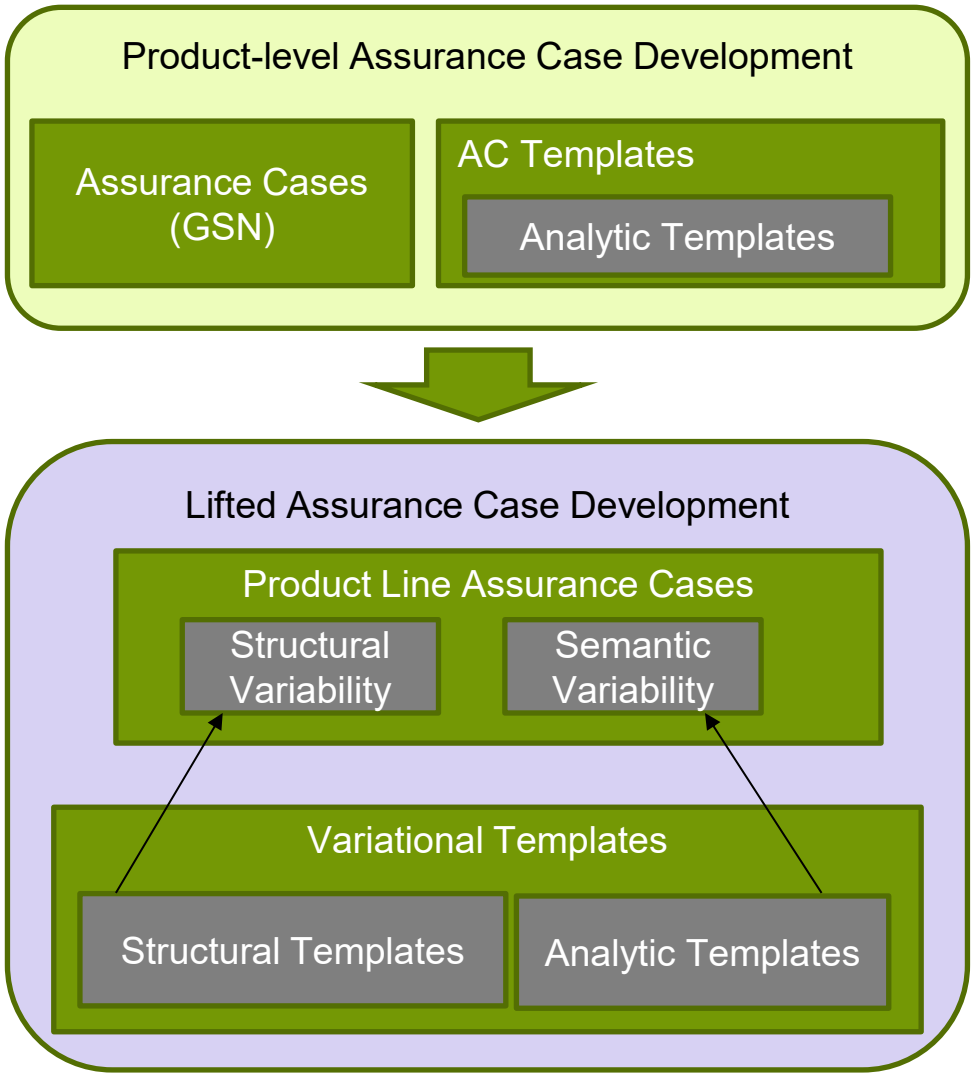
ADAS PL AC


True
G0: Across all configurations, Whenever an alarm is raised, the system responds by immediately returning control of vehicle to the driver.
Ctx0: Lifted Query allAlarms applied to ADAS state machine model ADAS-SM-PL
S0: Argument over lifted query application
G3: The lifting of the query engine is correct
Expert review
True
G1: The query allAlarms will identify the set of all alarms in model ADAS-SM-PL
Expert review
True
G2: Across all configurations, whenever any alarm in AlarmSet-PL is raised, the system responds by immediately returning control of vehicle to the driver
Ctx1: Annotated Query results AlarmSet-PL = {UnknownFollowingDist, FRONT_RADAR
UnknownLaneMarking, LANE_DETECTION
...,
HwFailureLvl1, HW_MONITORING
}
S1: Argument over query results
FRONT_RADAR
G2.0: Across all configurations, whenever "UnknownFollowingDist" is raised, the system responds by immediately returning control of vehicle to the driver
LANE_DETECTION
G2.1: Across all configurations, whenever "UnknownLaneMarking" is raised, the the system responds by immediately returning control of vehicle to the driver
HW_MONITORING
G2.2: Across all configurations, whenever "HwFailureLvl1" is raised, the system responds by immediately returning control of vehicle to the driver

ADAS PL AC


FRONT_RADAR
G2.0: Across all configurations, whenever "UnknownFollowingDist" is raised, the system responds by immediately returning control of vehicle to the driver
S2: Argument over lifted model checking
Ctx2: Safety property alarmFollowedByManualOperation
FRONT_RADAR
G4: Model checking will reveal any violations of the property specified in Ctx2 in the model ADAS-SM-PL
Expert Review
FRONT_RADAR
G5: The safety property in Ctx2 is correct
Expert Review
FRONT_RADAR
G6: Across all configurations, model checking did not reveal any violations of the property in Ctx2 for model ADAS-SM-PL
See output file 'result.out'
FRONT_RADAR
G7: The lifting of the model checker is correct.
Expert Review

Formal Foundations for Lifted AC Development



Formalized in

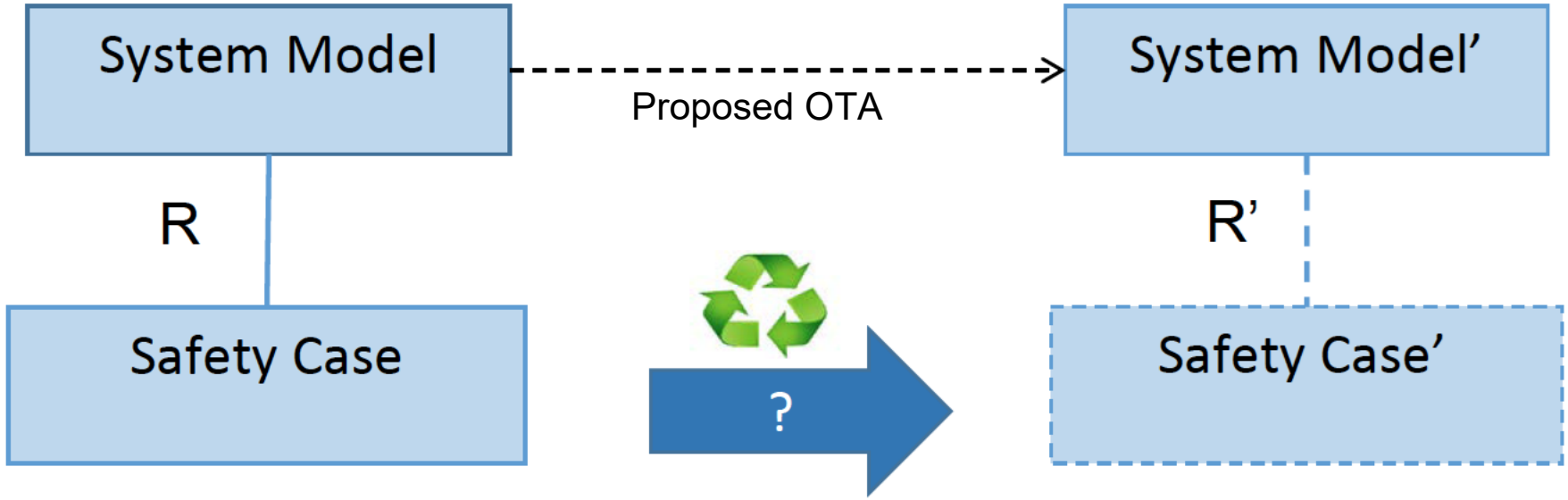
LEMMA

- Verified soundness proofs
- Foundation for integrating theorem proving + lifted AC development

Assuring Product Lines of Complex Systems -- Talk Plan

- Motivation and goals
- Background
  - Assurance
  - Product lines – variability in space and time
- Representation: Product Line Assurance Cases (PLAC)
- Development of PLACs
- Assessing Change of PLACs
- Tooling
- Summary and Next Steps

Change Impact Assessment



Problem: Can we aid the safety engineer in constructing a safety case for an evolved system by reusing the components of the original safety case as *much* and as *soundly* as possible, thus reducing the overall revision cost incurred?

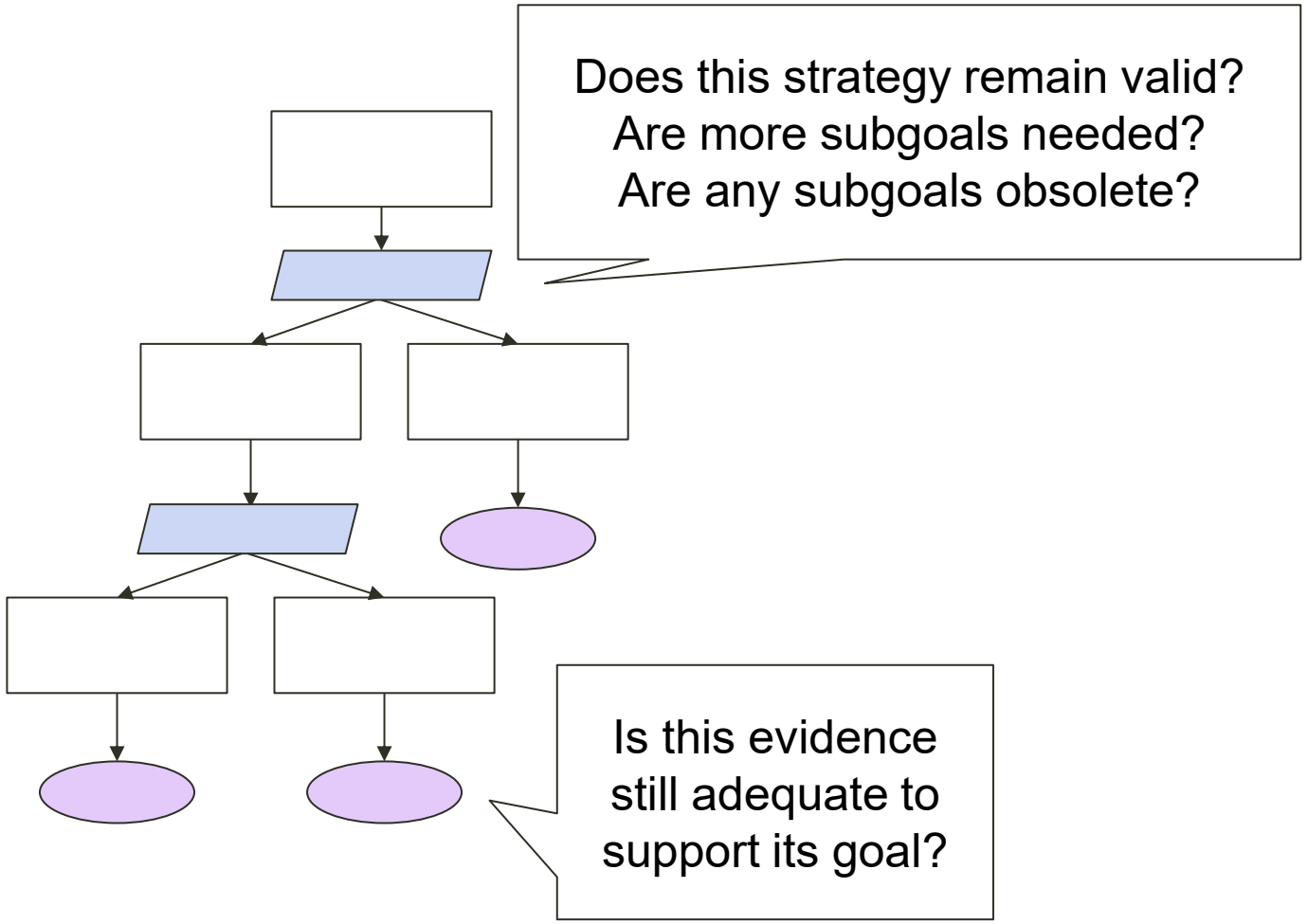
Necessary step: Impact assessment to identify how changes in the system affect the safety case.

Impact of Changes on Arguments and Evidence

Primary objective: determine which *goals* have/have not lost their assurance.

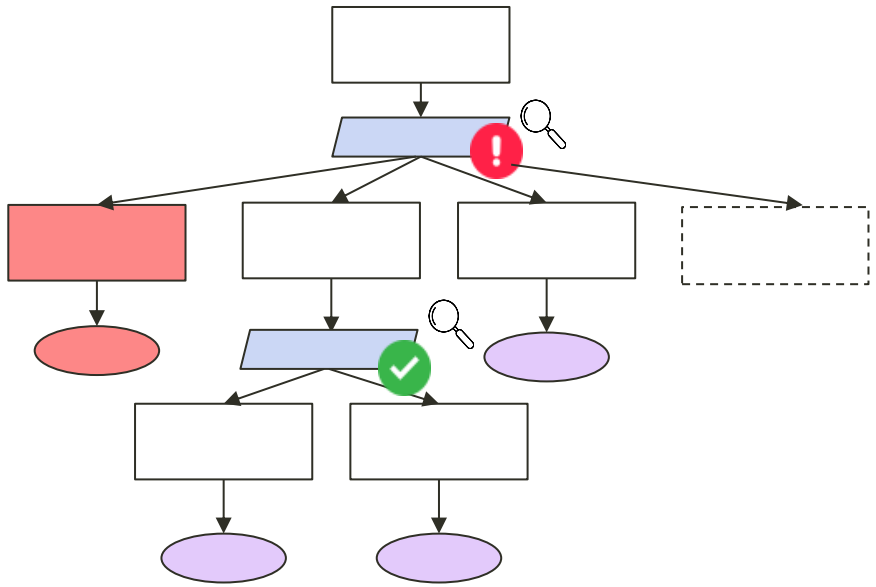
To do this, we need to determine impact on

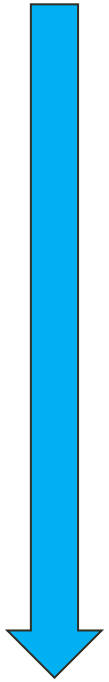
- evidence
- strategies



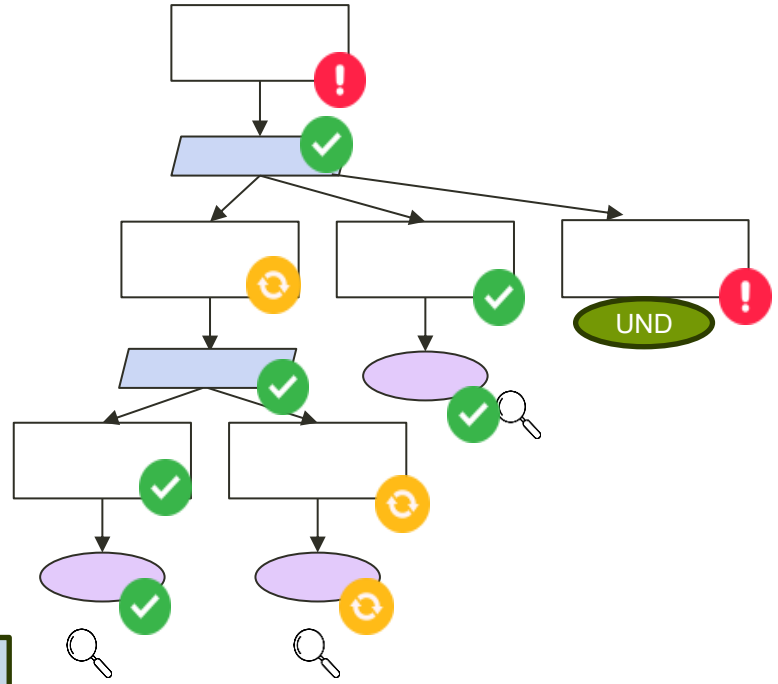
Product-Based Impact Analysis

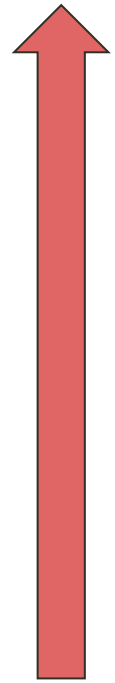
Stage 1 (Top-down): Check each strategy, identify goals which become obsolete, or new goals which are missing

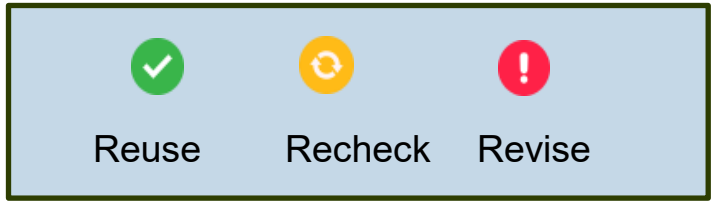




Stage 2 (Bottom-up): For each (non-obsolete) branch, determine the adequacy for each piece of evidence, and propagate the results back up through the AC

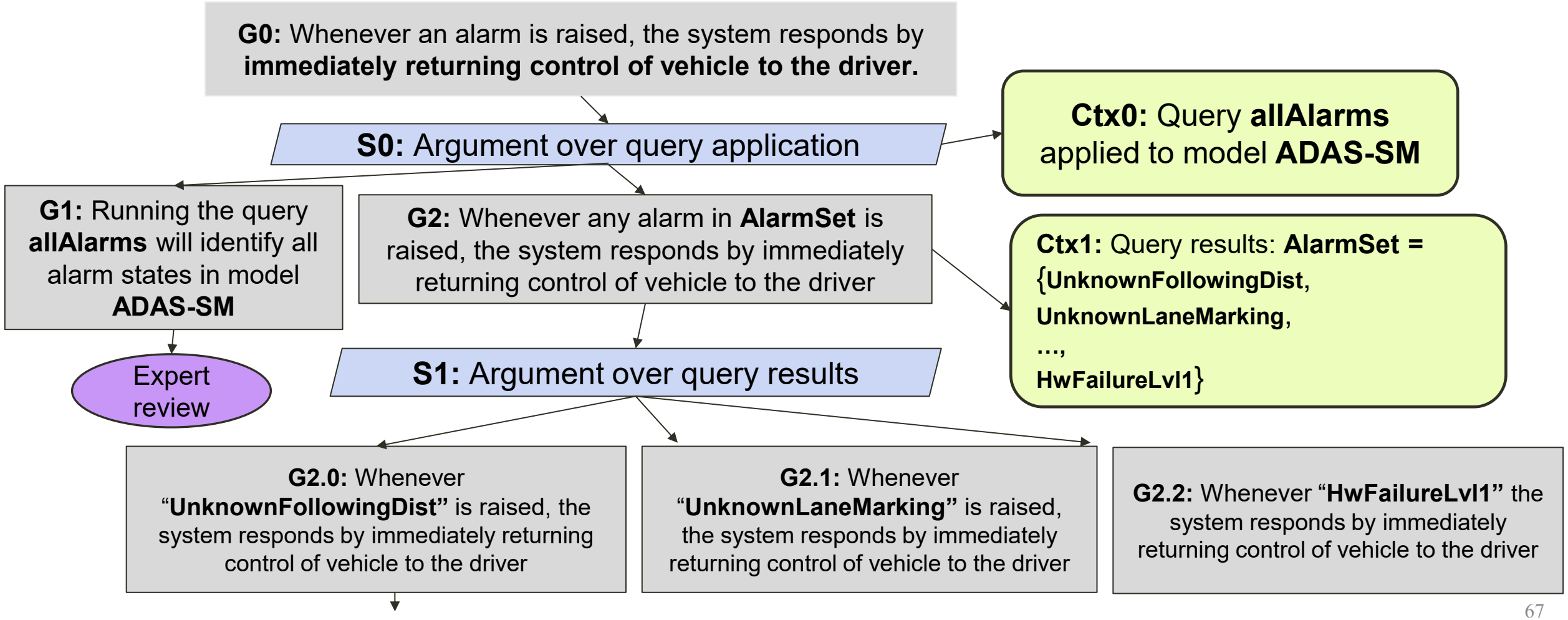






ADAS Assurance Case

Re-run query to generate new set of results to compare against old results



Stage 1: Top-down Impact

Re-run query to generate new set of results to compare against old results. Revise argument

Change:
-- ~~UnknownLaneMarking~~
++ SpeedLimitViolation

G0: Whenever an alarm is raised, the system responds by **immediately returning control of vehicle to the driver.**

S0: Argument over query application

Ctx0: Query **allAlarms** applied to **updated** model **ADAS-SM**

G1: Running the query **allAlarms** will identify all alarm states model **ADAS-SM**

Expert review

G2: Whenever any alarm in **AlarmSet** is raised, the system responds by immediately returning control of vehicle to the driver

S1: Argument over query results

Ctx1: Updated AlarmSet =
{UnknownFollowingDist,
~~**UnknownLaneMarking**~~**,**
...,
HwFailureLvl1
SpeedLimitViolation
}

G2.0: Whenever "**UnknownFollowingDist**" is raised, the system responds by immediately returning control of vehicle to the driver

G2.1: Whenever "**UnknownLaneMarking**" is raised, the system responds by immediately returning control of vehicle to the driver

(Obsolete)

G2.2: Whenever "**HwFailureLvl1**" the system responds by immediately returning control of vehicle to the driver

G2.3: Whenever "**SpeedLimitViolation**" is raised, the system responds by immediately returning control of vehicle to the driver

(New!)

Stage 1: Top-down Impact

Change:
-- ~~UnknownLaneMarking~~
++ SpeedLimitViolation

Revise argument: obsolete alarm + missing new alarm

G0: Whenever an alarm is raised, the system responds by **immediately returning control of vehicle to the driver.**

S0: Argument over query application

Ctx0: Query **allAlarms** applied to **updated** model **ADAS-SM**

G1: Running the query **allAlarms** will identify all alarm states model **ADAS-SM**

G2: Whenever any alarm in **AlarmSet** is raised, the system responds by immediately returning control of vehicle to the driver

Expert review

S1: Argument over query results

Ctx1: Updated AlarmSet =
{UnknownFollowingDist,
~~UnknownLaneMarking~~,
...,
HwFailureLvl1
SpeedLimitViolation
}

G2.0: Whenever "**UnknownFollowingDist**" is raised, the system responds by immediately returning control of vehicle to the driver

G2.2: Whenever "**HwFailureLvl1**" the system responds by immediately returning control of vehicle to the driver

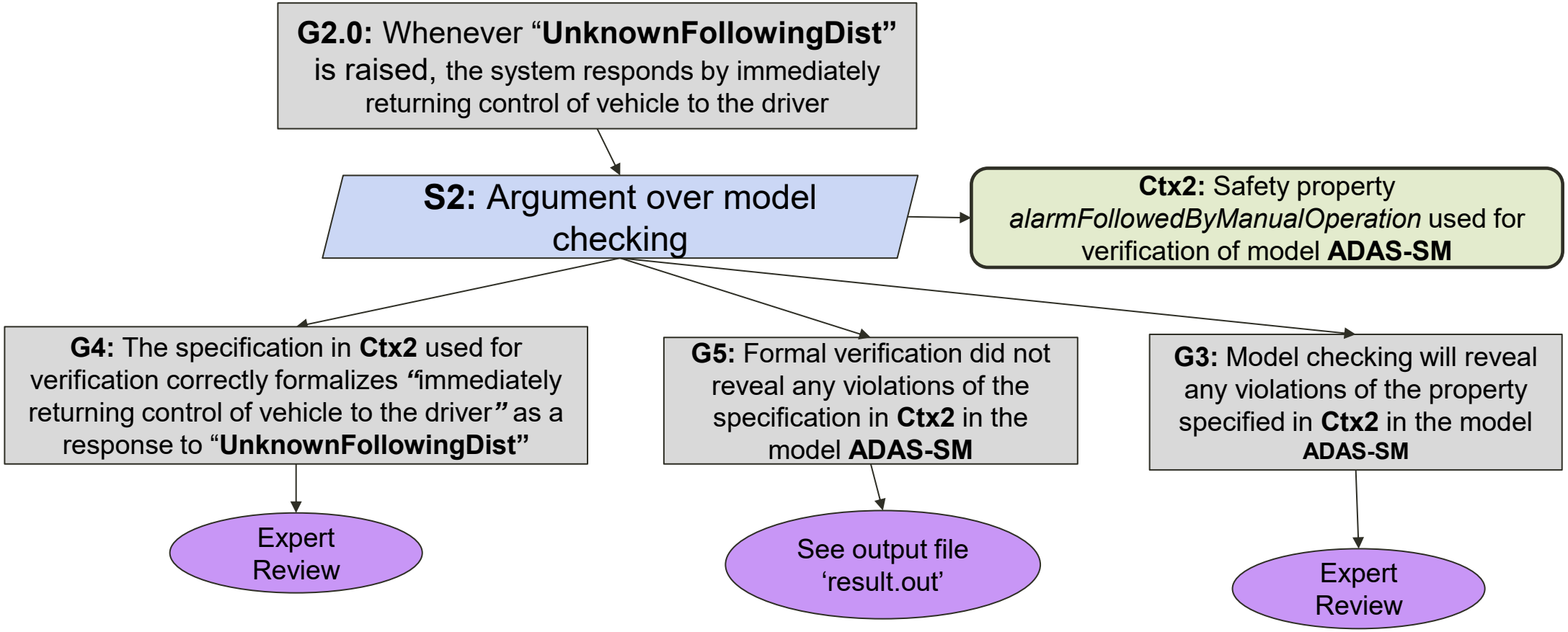
G2.3: Whenever "**SpeedLimitViolation**" is raised, the system responds by immediately returning control of vehicle to the driver

(New!)

Stage 2: Bottom-up Impact

Change:
-- ~~UnknownLaneMarking~~
++ SpeedLimitViolation

Recheck evidence (alternatively: use regression analysis)
Reuse argument (reusable by design)



Tools exist for various kinds of evidence, e.g., testing [1] and model checking [2]

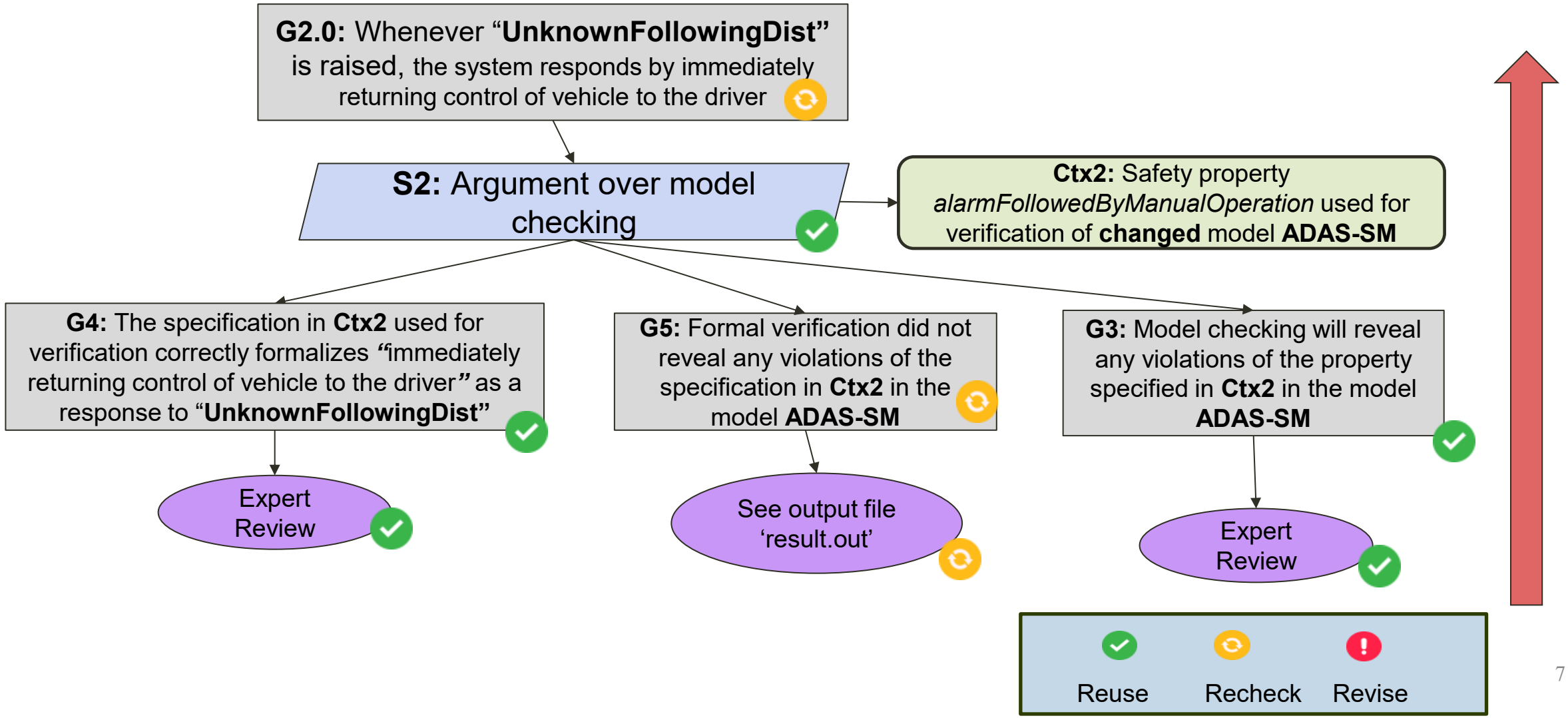
[1] Mora, Federico, et al. "Client-specific equivalence checking." *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 2018.

[2] Menghi, Claudio, et al. "TOrPEDO: witnessing model correctness with topological proofs." *Formal Aspects of Computing*. 2021.

Stage 2: Bottom-up Impact

Change:
-- ~~UnknownLaneMarking~~
++ SpeedLimitViolation

Recheck evidence (alternatively: use regression analysis)
Reuse argument (reusable by design)



Stage 2: Bottom-up Impact

Propagate to the top

Change:
-- ~~UnknownLaneMarking~~
++ SpeedLimitViolation

G0: Whenever an alarm is raised, the system responds by **immediately returning control of vehicle to the driver.**

S0: Argument over query application

Ctx0: Query **allAlarms** applied to **updated** model **ADAS-SM**

G1: Running the **allAlarms** will identify all alarm states in model model **ADAS-SM**

Expert review

G2: Whenever any alarm in **AlarmSet** is raised, the system responds by immediately returning control of vehicle to the driver

Ctx1: Updated AlarmSet =
{UnknownFollowingDist,
~~UnknownLaneMarking~~,
...,
HwFailureLvl1,
SpeedLimitViolation
}

S1: Argument over query results

G2.0: Whenever "**UnknownFollowingDist**" is raised, the system responds by immediately returning control of vehicle to the driver

G2.2: Whenever "**HwFailureLvl1**" the system responds by immediately returning control of vehicle to the driver

G2.3: Whenever "**SpeedLimitViolation**" is raised, the system responds by immediately returning control of vehicle to the driver

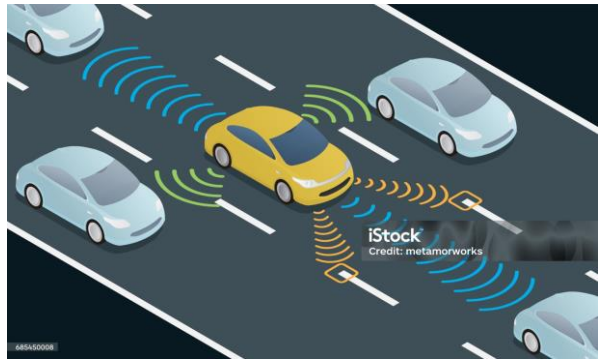
UND

ADAS -- Product Line Version

Representation of a *family* of vehicles with different configurations of ADAS features

Features

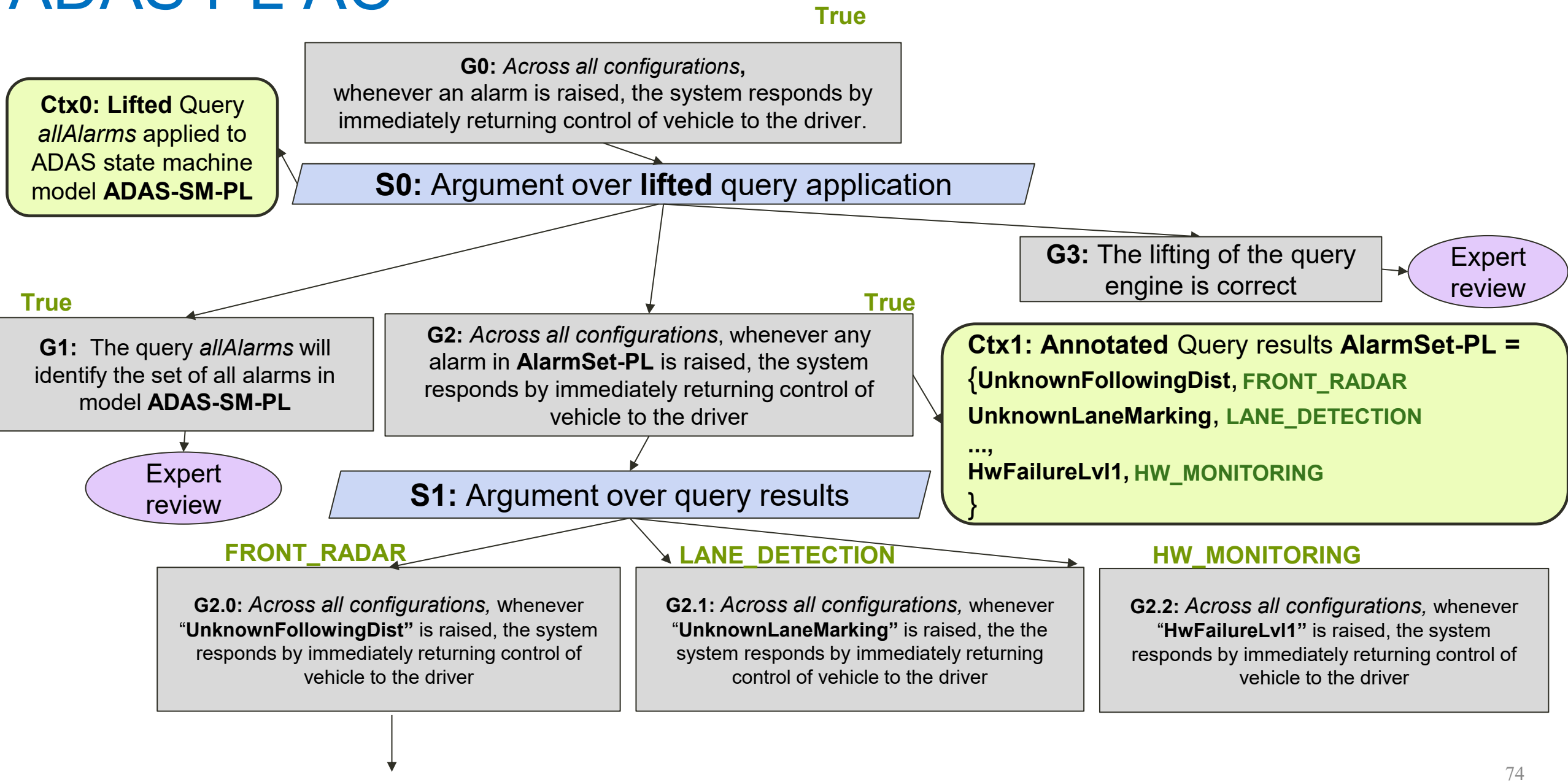
- HW_MONITORING
- LANE_DETECTION
- LANE_CENTERING
- FRONT_RADAR
- ALARM_SYSTEM



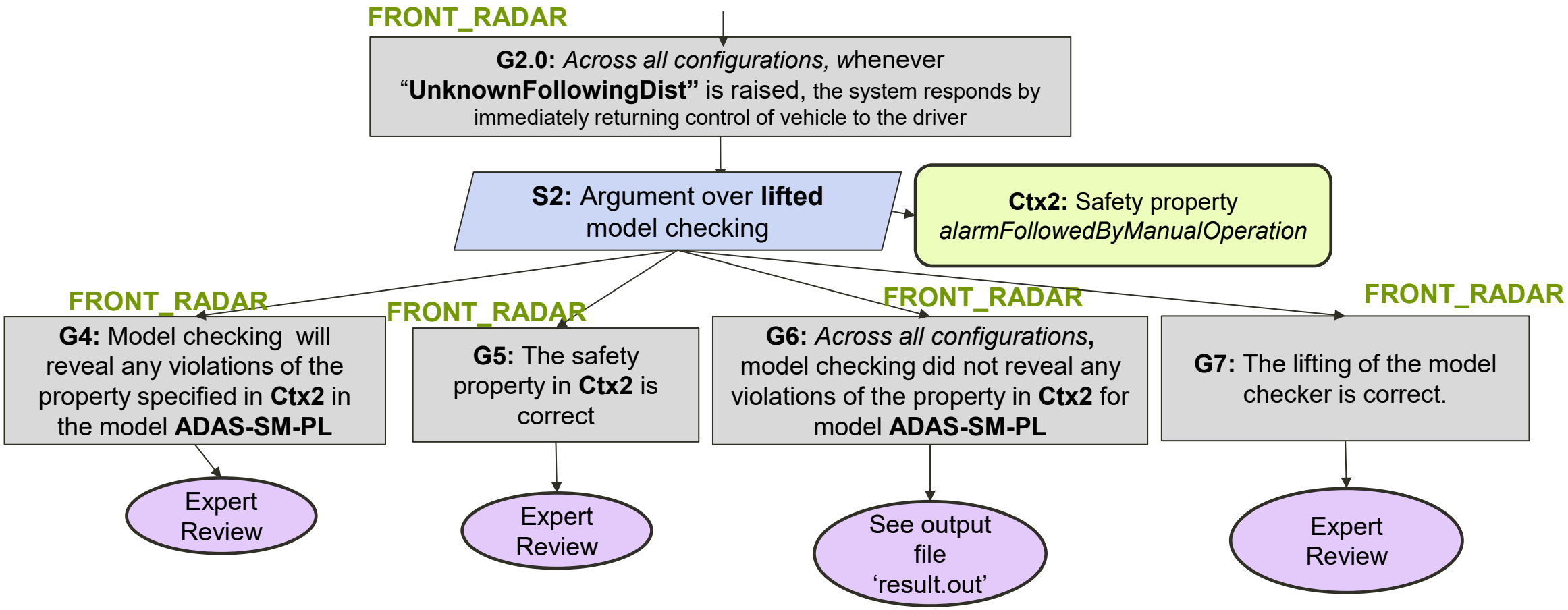
Feature model: HW_MONITORING & (LANE_CENTERING => (LANE_DETECTION & ALARM_SYSTEM))

State machine mode becomes annotated with presence conditions

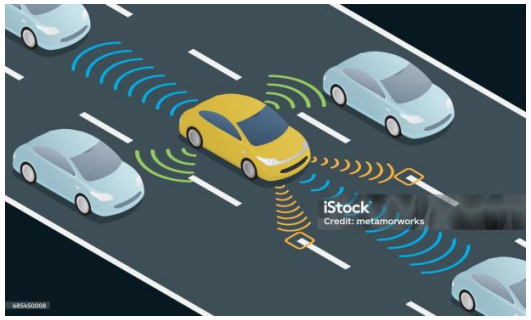
ADAS PL AC


True
G0: Across all configurations, whenever an alarm is raised, the system responds by immediately returning control of vehicle to the driver.
Ctx0: Lifted Query allAlarms applied to ADAS state machine model ADAS-SM-PL
S0: Argument over lifted query application
G3: The lifting of the query engine is correct
Expert review
True
G1: The query allAlarms will identify the set of all alarms in model ADAS-SM-PL
Expert review
True
G2: Across all configurations, whenever any alarm in AlarmSet-PL is raised, the system responds by immediately returning control of vehicle to the driver
Ctx1: Annotated Query results AlarmSet-PL = {UnknownFollowingDist, FRONT_RADAR
UnknownLaneMarking, LANE_DETECTION
...,
HwFailureLvl1, HW_MONITORING
}
S1: Argument over query results
FRONT_RADAR
G2.0: Across all configurations, whenever "UnknownFollowingDist" is raised, the system responds by immediately returning control of vehicle to the driver
LANE_DETECTION
G2.1: Across all configurations, whenever "UnknownLaneMarking" is raised, the the system responds by immediately returning control of vehicle to the driver
HW_MONITORING
G2.2: Across all configurations, whenever "HwFailureLvl1" is raised, the system responds by immediately returning control of vehicle to the driver

ADAS PL AC


FRONT_RADAR
G2.0: Across all configurations, whenever "UnknownFollowingDist" is raised, the system responds by immediately returning control of vehicle to the driver
S2: Argument over lifted model checking
Ctx2: Safety property alarmFollowedByManualOperation
FRONT_RADAR
G4: Model checking will reveal any violations of the property specified in Ctx2 in the model ADAS-SM-PL
Expert Review
FRONT_RADAR
G5: The safety property in Ctx2 is correct
Expert Review
FRONT_RADAR
G6: Across all configurations, model checking did not reveal any violations of the property in Ctx2 for model ADAS-SM-PL
See output file 'result.out'
FRONT_RADAR
G7: The lifting of the model checker is correct.
Expert Review

Running Example (Product Line Changes)



Recall: two dimensions of change of product lines:

- **Structural dimension**: the 150% model and/or presence conditions are modified
- **Variability dimension:** the alphabet of features and/or feature model are modified.

ADAS product line changes:

Structural Dimension: Modification of an existing feature. New states added associated with the **ALARM_SYSTEM** feature that allow users to view alarm history

Variability Dimension: Addition of new feature **ADAPTIVE_CRUISE_CONTROL**

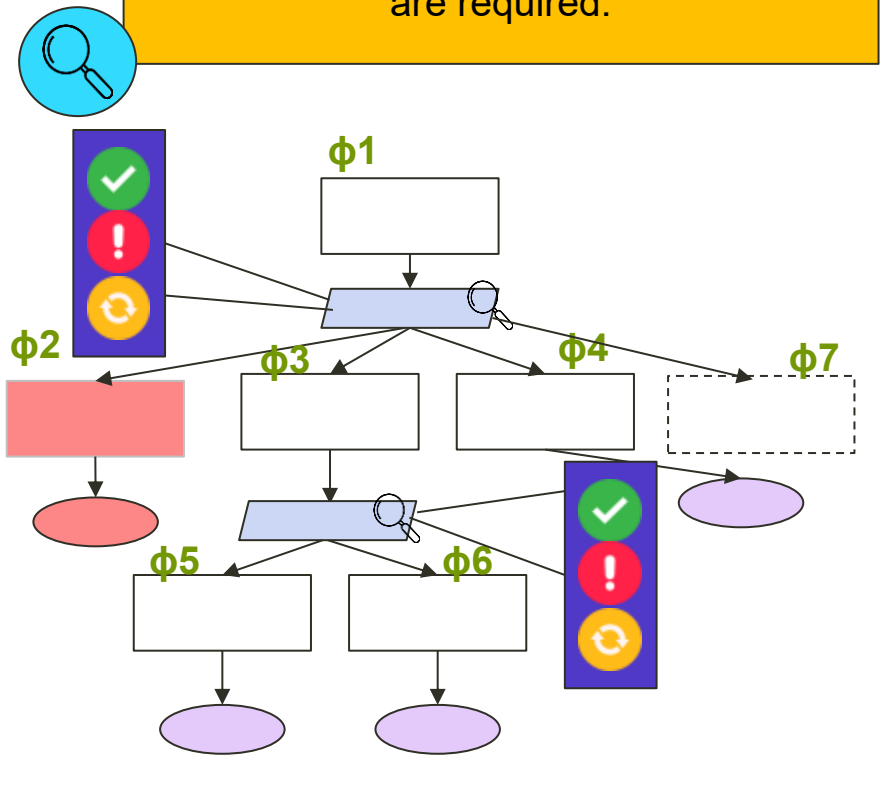
Updated Feature model: **HW_MONITORING &**
(LANE_CENTERING => (LANE_DETECTION & ALARM_SYSTEM)) &
(ADAPTIVE_CRUISE_CONTROL => (FRONT_RADAR & ALARM_SYSTEM))

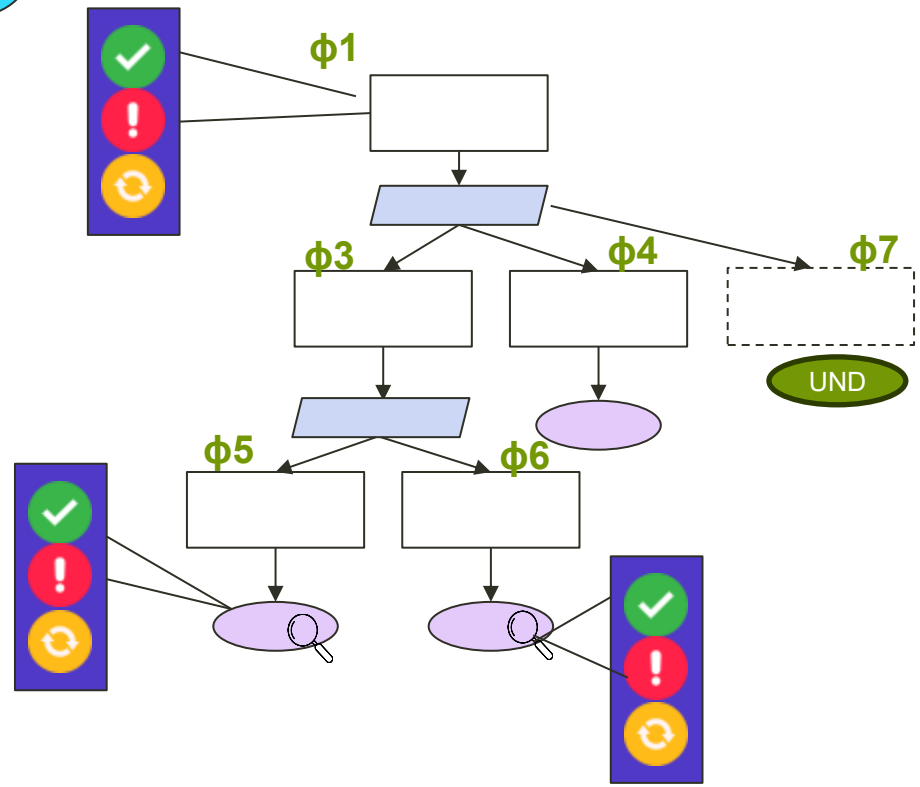
Change:
** Modify elements owned by feature ALARM_SYSTEM
++ New feature ADAPTIVE_CRUISE_CONTROL
++ New model elements for ADAPTIVE_CRUISE_CONTROL
(including new alarm)

Lifted Impact Analysis

Stage 1 (Top-down): For each strategy, identify **for which sets of products** any goals have become obsolete, and **for which sets of products** any new goals are required.

Stage 2 (Bottom-up): For each (non-obsolete) branch, determine the adequacy for each piece of evidence **across all relevant products**, and propagate the results back up through the AC **in a variability-aware fashion.**





Each node gets all three values, with each value corresponding to *a set of products,* represented as a presence condition

Stage 1: Top-Down Impact

Change:
** Modify elements owned by feature ALARM_SYSTEM
++ New feature ADAPTIVE_CRUISE_CONTROL
++ New model elements for ADAPTIVE_CRUISE_CONTROL (including new alarm)

Upon the change, rerun the **lifted** query to produce a set of annotated query results.

True

G0: *Across all configurations*, whenever an alarm is raised, the system responds by immediately returning control of vehicle to the driver.

S0: Argument over **lifted** query application

Ctx0: Lifted Query *allAlarms* applied to **changed** model **ADAS-SM-PL**

True

G1: The query *allAlarms* will identify the set of all alarms in model **ADAS-SM-PL**

Expert review

True

G2: *Across all configurations*, whenever any alarm in **AlarmSet-PL** is raised, the system responds by immediately returning control of vehicle to the driver

Ctx1: Annotated query results:
{**UnknownFollowingDist**, FRONT_RADAR
UnknownLaneMarking, LANE_DETECTION
<...>,
HwFailureLvl1, HW_MONITORING
SpeedLimitViolation, ADAPTIVE_CRUISE_CONTROL
}

S1: Argument over query results

FRONT_RADAR

G2.0: *Across all configurations,* whenever "**UnknownFollowingDist**" is raised, the system responds by immediately returning control of vehicle to the driver

LANE_DETECTION

G2.1: *Across all configurations,* whenever "**UnknownLaneMarking**" is raised, the system responds by immediately returning control of vehicle to the driver

HW_MONITORING

G2.2: *Across all configurations,* whenever "**HwFailureLvl1**" is raised, the system responds by immediately returning control of vehicle to the driver

ADAPTIVE_CRUISE_CONTROL

G2.4: *Across all configurations,* whenever "**SpeedLimitViolation**" is raised, the system responds by immediately returning control of vehicle to the driver

(New)

Stage 1: Top-Down Impact

Change:
** Modify elements owned by feature ALARM_SYSTEM
++ New feature ADAPTIVE_CRUISE_CONTROL
++ New model elements for ADAPTIVE_CRUISE_CONTROL (including new alarm)

Revise argument

True

G0: *Across all configurations*, whenever an alarm is raised, the system responds by immediately returning control of vehicle to the driver.

S0: Argument over **lifted** query application

True

G1: The query *allAlarms* will identify the set of all alarms in model **ADAS-SM-PL**

Expert review

True

G2: *Across all configurations*, whenever any alarm in **AlarmSet-PL** is raised, the system responds by immediately returning control of vehicle to the driver

S1: Argument over query results

Reflects the fact that we are now **missing** a subgoal for the new feature.

! ADAPTIVE_CRUISE_CONTROL

ADAPTIVE_CRUISE_CONTROL

False

FRONT_RADAR

G2.0: *Across all configurations,* whenever "**UnknownFollowingDist**" is raised, the system responds by immediately returning control of vehicle to the driver

LANE_DETECTION

G2.1: *Across all configurations,* whenever "**UnknownLaneMarking**" is raised, the system responds by immediately returning control of vehicle to the driver

HW_MONITORING

G2.2: *Across all configurations,* whenever "**HwFailureLvl1**" is raised, the system responds by immediately returning control of vehicle to the driver

ADAPTIVE_CRUISE_CONTROL

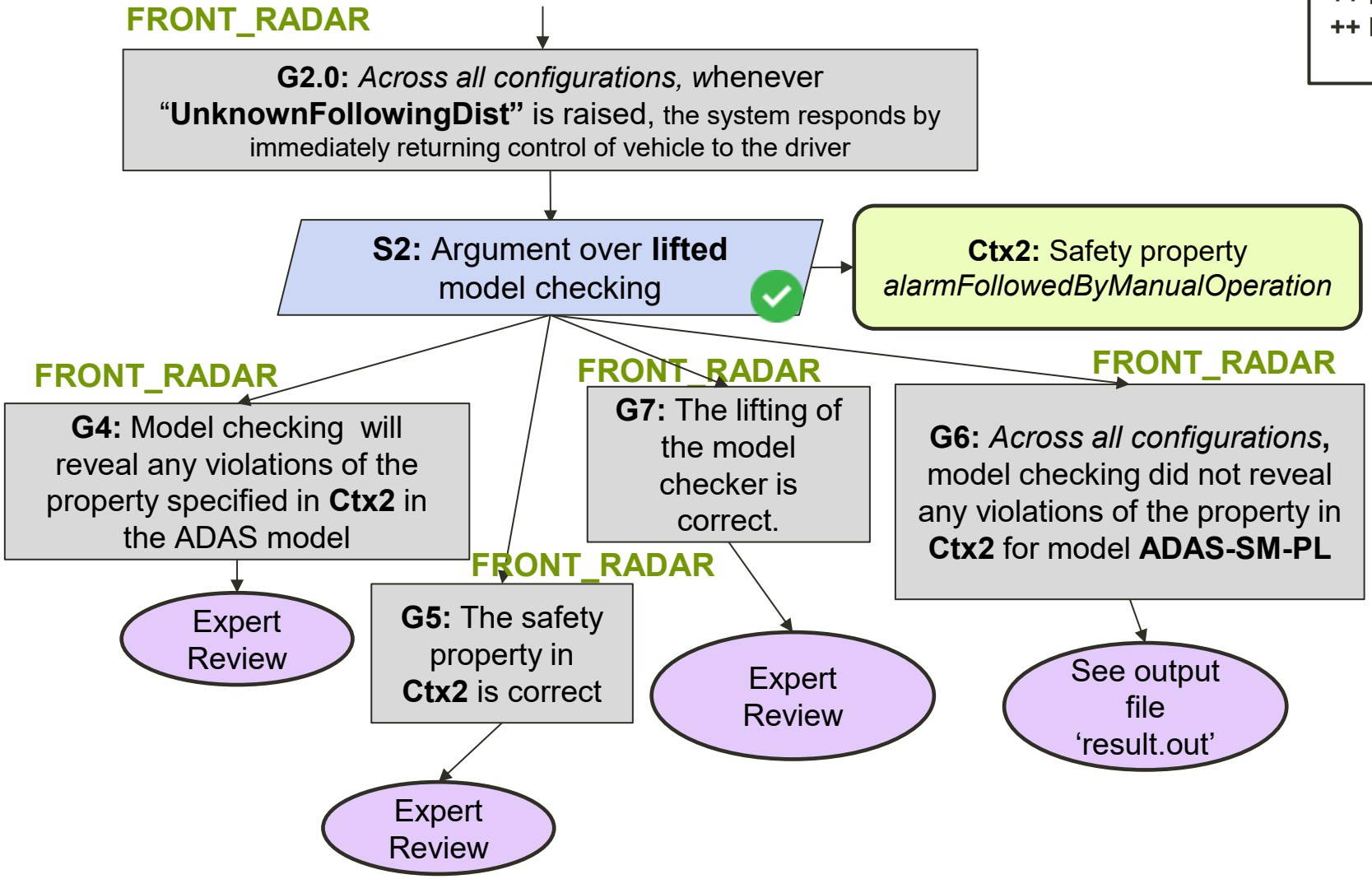
G2.4: *Across all configurations,* whenever "**SpeedLimitViolation**" is raised, the system responds by immediately returning control of vehicle to the driver

(New)

Stage 2: Bottom-up Impact

Recheck evidence (alternative if available: **lifted** regression analysis, e.g., [3])

Change:
** Modify elements owned by feature ALARM_SYSTEM
++ New feature ADAPTIVE_CRUISE_CONTROL
++ New model elements for ADAPTIVE_CRUISE_CONTROL
(including new alarm)



[3] Wang, et al.. "Code-Level Functional Equivalence Checking of Annotative Software Product Lines." *SPLC'*2023.

Stage 2: Bottom-up Impact

Recheck evidence (alternative if available: **lifted** regression analysis, e.g., [3])

FRONT_RADAR

G2.0: *Across all configurations, w*henever "**UnknownFollowingDist"** is raised, the system responds by immediately returning control of vehicle to the driver

- ! ADAPTIVE_CRUISE_CONTROL /\ ! ALARM_SYSTEM
- ADAPTIVE_CRUISE_CONTROL
- ALARM_SYSTEM /\ !ADAPTIVE_CRUISE_CONTROL

S2: Argument over **lifted** model checking

Ctx2: Safety property *alarmFollowedByManualOperation*

Reflects the fact that

1. we have **no** evidence for configurations with the new feature,
2. evidence is **certainly** reusable for existing configurations which are not changed,
3. evidence is **possibly** reusable for existing configurations which are changed.

FRONT_RADAR

G4: Model checking will reveal any violations of the property specified in **Ctx2** in the ADAS model

Expert Review

FRONT_RADAR

G5: The safety property in **Ctx2** is correct

Expert Review

FRONT_RADAR

G7: The lifting of the model checker is correct.

Expert Review

FRONT_RADAR

G6: *Across all configurations*, model checking did not reveal any violations of the property in **Ctx2** for model **ADAS-SM-PL**

See output file 'result.out'

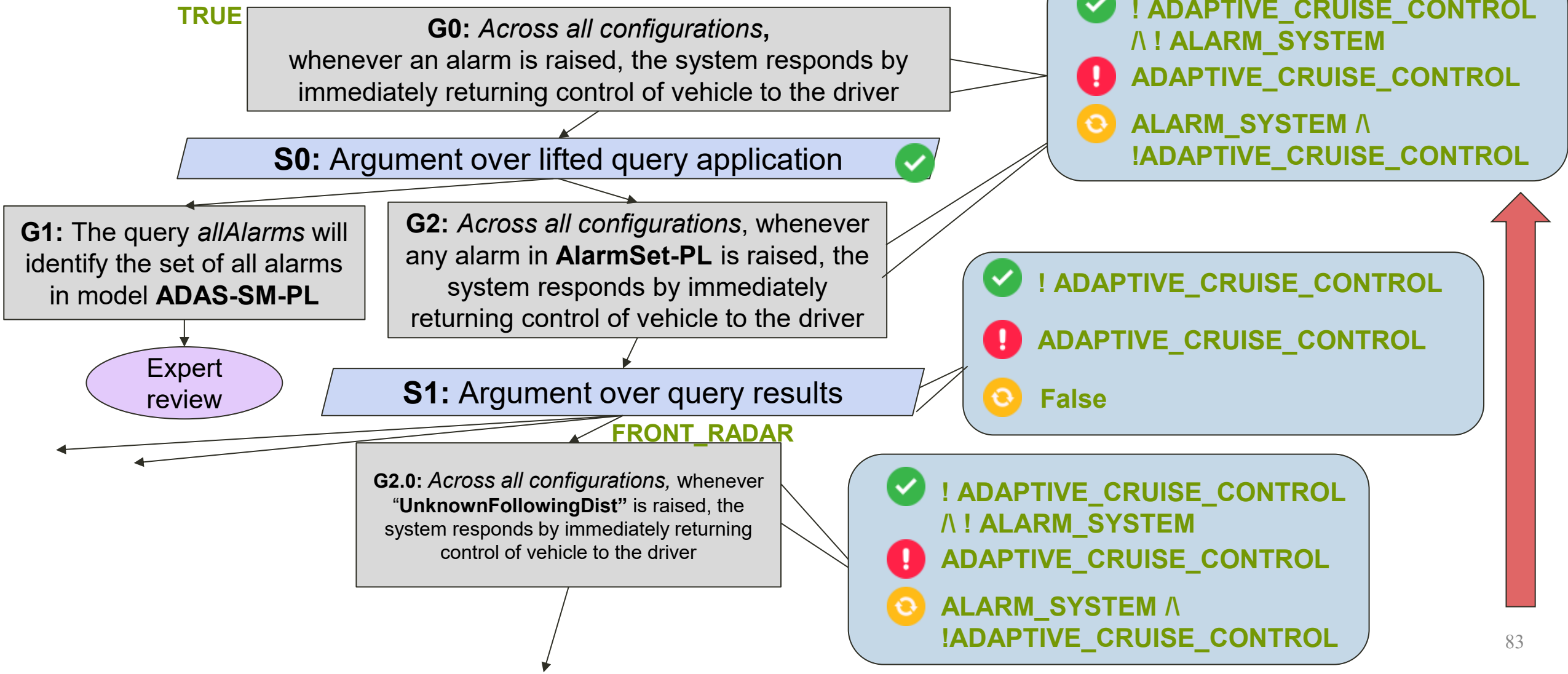
- ! ADAPTIVE_CRUISE_CONTROL /\ ! ALARM_SYSTEM
- ADAPTIVE_CRUISE_CONTROL
- ALARM_SYSTEM /\ !ADAPTIVE_CRUISE_CONTROL

[3] Wang, et al.. "Code-Level Functional Equivalence Checking of Annotative Software Product Lines." *SPLC'*2023.

Stage 2: Bottom-up Impact

Change:
** Modify elements owned by feature ALARM_SYSTEM
++ New feature ADAPTIVE_CRUISE_CONTROL
++ New model elements for ADAPTIVE_CRUISE_CONTROL
(including new alarm)

Need to compose impact results of subgoals and strategy
The impact result produced for leaves subsumes the impact result for the strategy

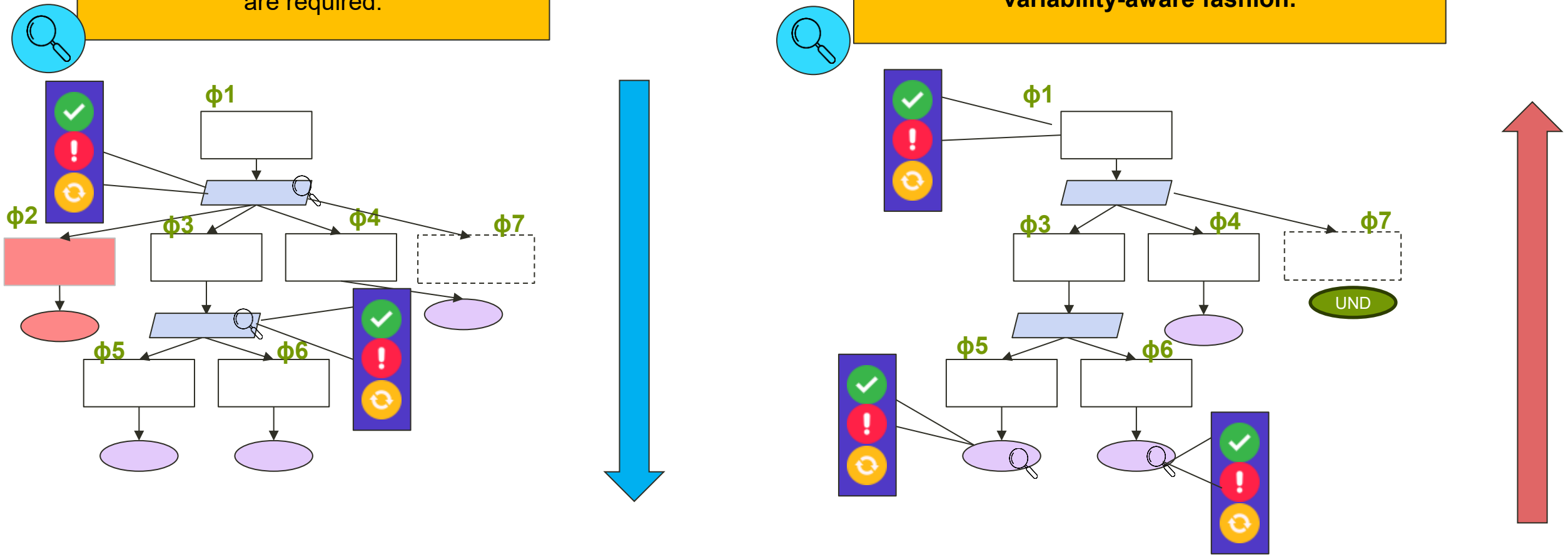


Lifted Impact Analysis

Reuse Recheck Revise

Stage 1 (Top-down): For each strategy, identify **for which sets of products** any goals have become obsolete, and **for which sets of products** any new goals are required.

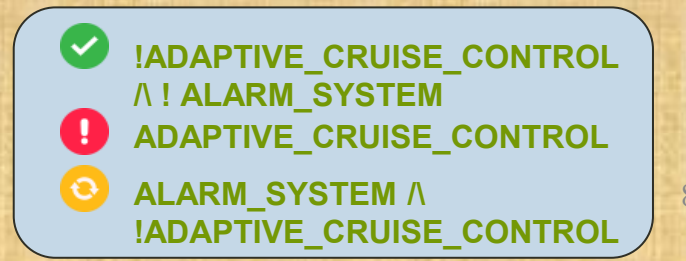
Stage 2 (Bottom-up): For each (non-obsolete) branch, determine the adequacy for each piece of evidence **across all relevant products**, and propagate the results back up through the AC **in a variability-aware fashion.**



Each node gets all three values, with each value corresponding to *a set of products,* represented as a presence condition

Main points

1. Assurance cases combine argument and evidence, allow to contextualize analysis and verification. Need to be reviewable
2. OTA updates yield product lines in time and space which need assuring
3. To assure product lines, reinterpret arguments and evidence to apply to sets of products
4. Assurance cases can be defined using analytic templates which can be lifted, preserving correctness, and composed
5. Change impact analysis can be lifted by associating each AC element with sets of products where it can be reused, rechecked and revised

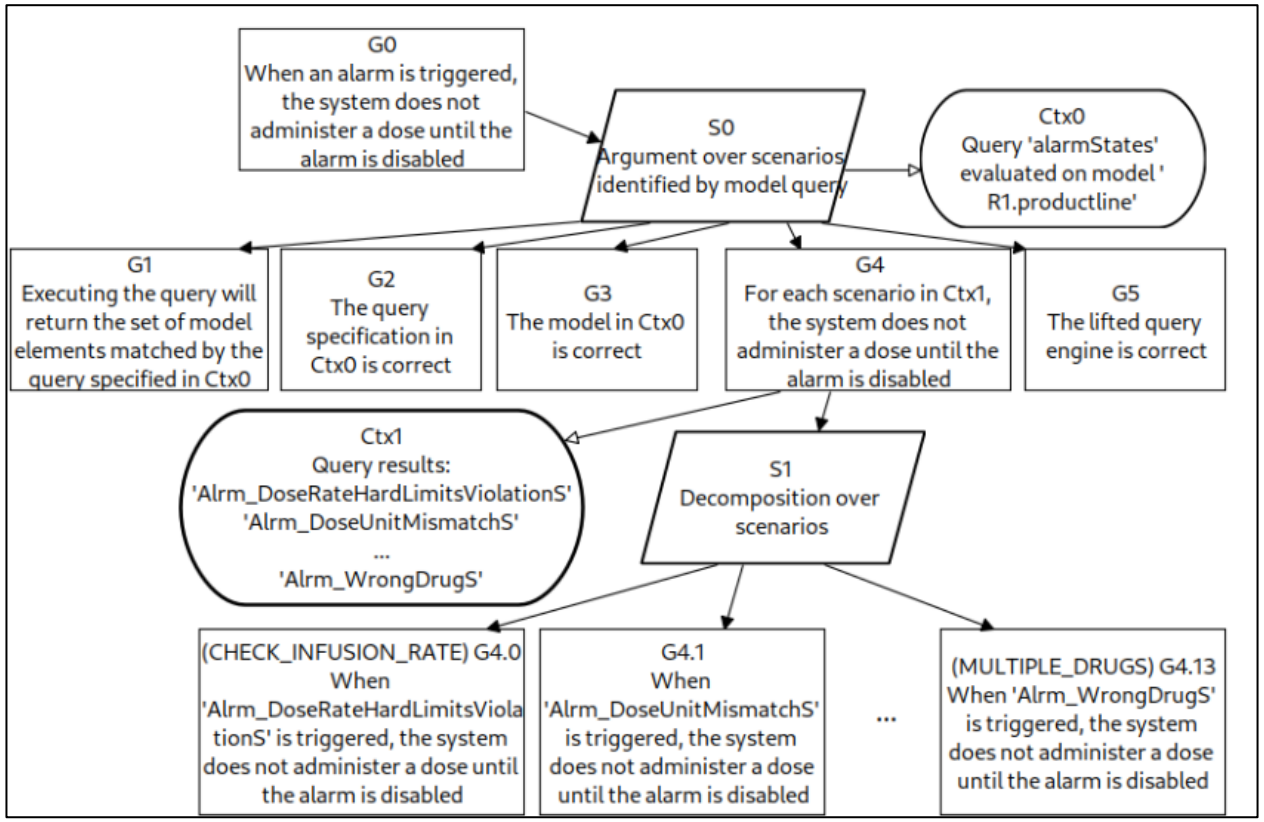


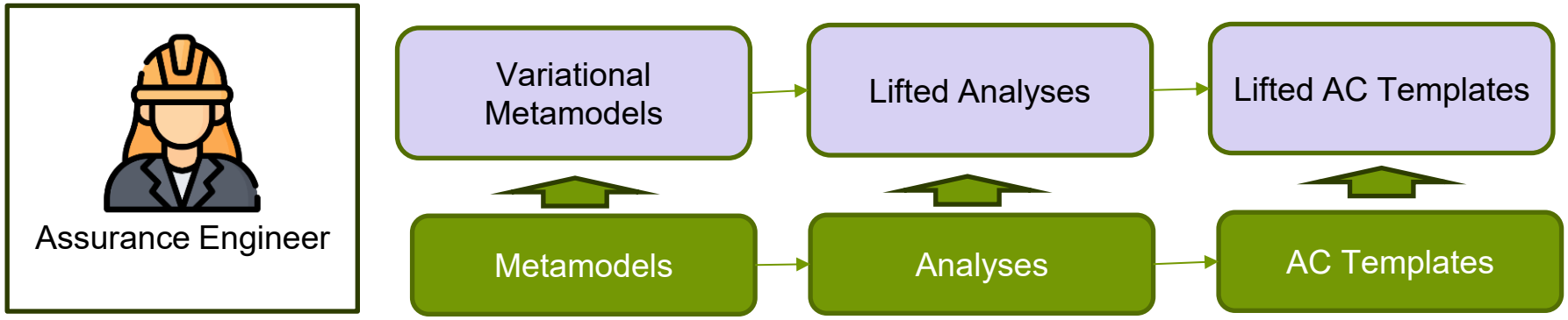
Assuring Product Lines of Complex Systems -- Talk Plan

- Motivation
- Background
  - Assurance
  - Product lines – variability in space and time
- Modeling
  - Variability
  - Product Line Assurance Cases
- Assuring Product Lines
- Assessing Change
- Tooling
- Summary and Next Steps

Tooling: MMINT-A/PL

Extension of MMINT-A: A Tool for Model-Based Assurance Cases





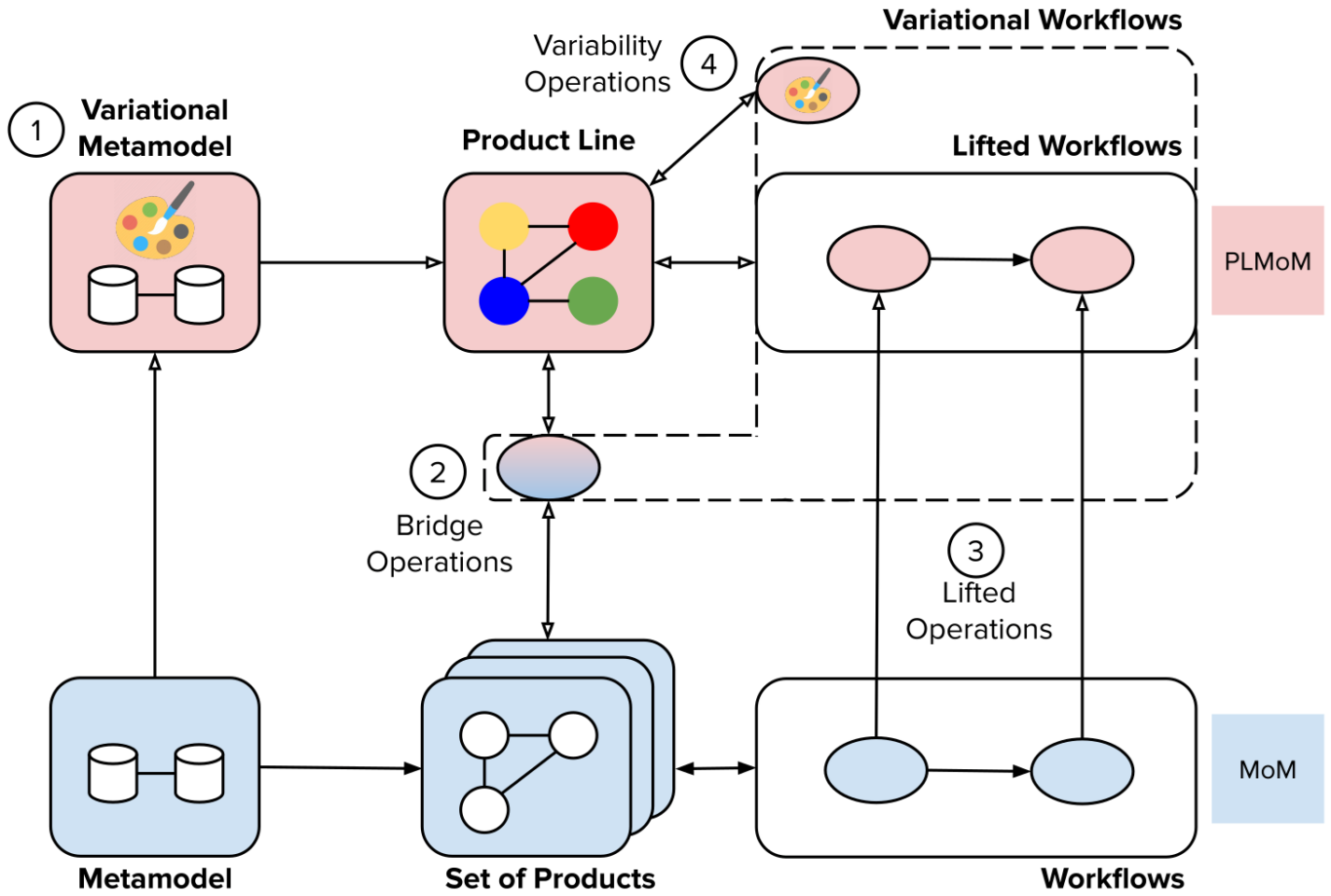
Di Sandro, A., Murphy, L., et al. (2024). MMINT-A: A Framework for Model-Based Safety Assurance. *Science of Comp. Programming*

Di Sandro, A., Murphy, L., Viger, T., Chechik, M. "From Products to Product Lines: A Model-based Methodology", Proceedings of MODELS'24 (Companion), pp. 720-724

Aside: Product vs. PL Management Framework

Components:

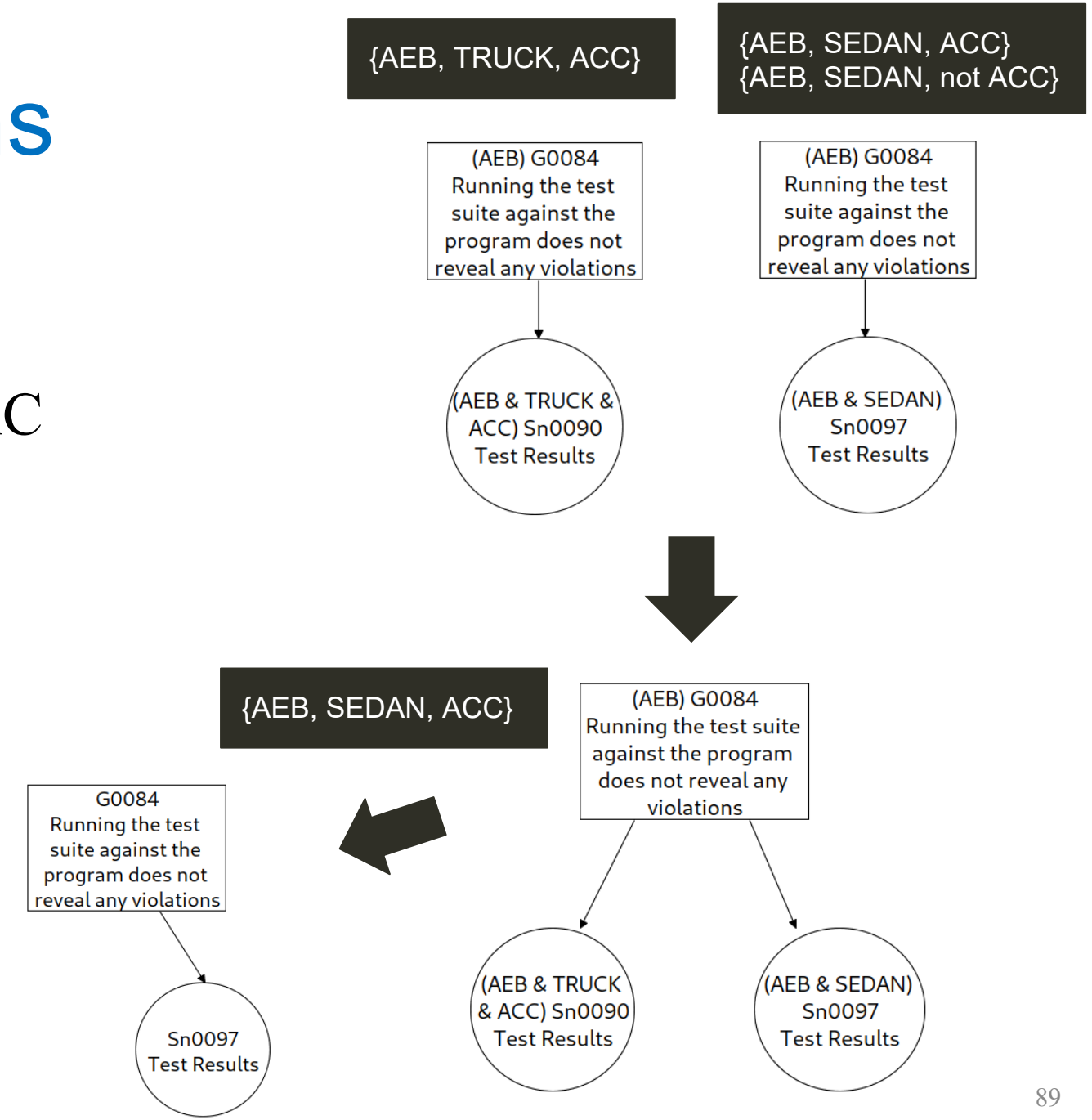
1. Define variability
2. Switch between product and PL levels ("bridge")
3. Lift product operations to PL ("lift")
4. Perform PL-specific operations ("variability" operators)



Di Sandro, A., Murphy, L., Viger, T., Chechik, M. "From Products to Product Lines: A Model-based Methodology", Proceedings of MODELS'24 (Companion), pp. 720-724

Supported Operations

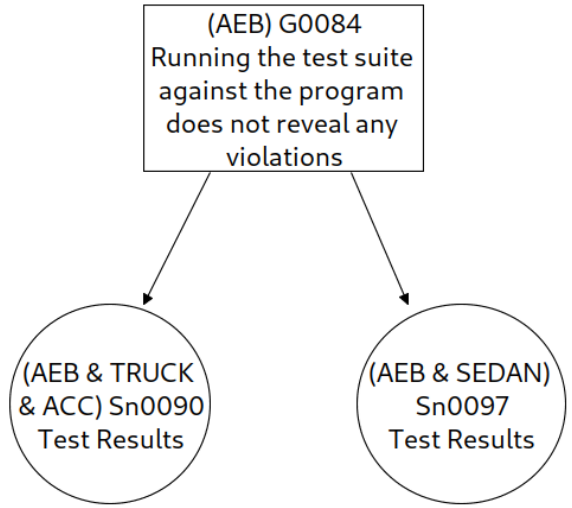
- Create PL AC
- Merge product ACs into PL AC
- Instantiate product AC from PL AC
- Validate AC and PL AC
- Describe change
- Perform change impact analysis



Supported Analyses

```
17 pattern unsupportedGoals(goal: Goal) {
18   0 == count Goal.supportedBy(goal, _);
19 }
```

Testing, FTA, model-checking, query-checking (VQL query language)
Lifted from product- to product-line level



MMINT-A/PL tool prototype:
https://github.com/adisandro/MMINT#mmint-pl

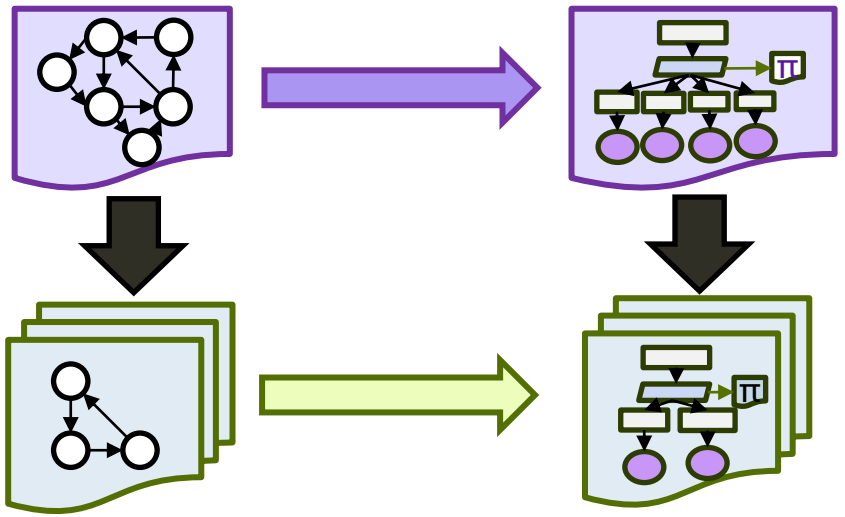
Lifted version
count = 1 {AEB, TRUCK, ACC}
count = 0 {AEB, TRUCK, not ACC}
count = 1 {AEB, SEDAN, ACC}
count = 1 {AEB, SEDAN, not ACC}
Goal G84 is unsupported

Assuring Product Lines of Complex Systems -- Talk Plan

- Motivation
- Background
  - Assurance
  - Product lines – variability in space and time
- Modeling
  - Variability
  - Product Line Assurance Cases
- Assuring Product Lines
- Assessing Change
- Tooling
- Summary, Other Work, Next Steps

Problem: How do we know whether an entire product line is suitable for deployment?

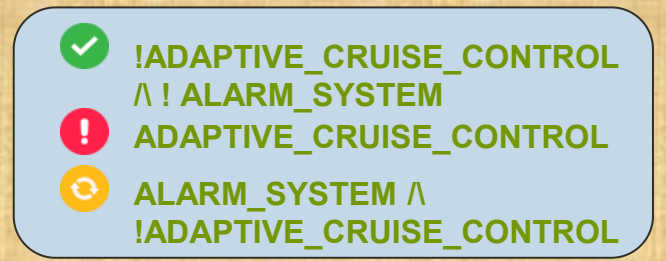
How do we create a rigorous assurance case for an entire product line **without resorting to product-level work**?



A general methodology for lifted AC development (PLACIDUS) with a rigorous formalization of variational GSN + lifting of argument templates

Key Insights

1. Assurance cases combine argument and evidence, allow to contextualize analysis and verification. Need to be reviewable
2. OTA updates yield product lines in time and space which need assuring
3. To assure product lines, reinterpret arguments and evidence to apply to sets of products
4. Assurance cases can be defined using analytic templates which can be lifted, preserving correctness, and composed
5. Change impact analysis can be lifted by associating each AC element with sets of products where it can be reused, rechecked and revised



One More Insight

6. The process requires a collaboration of individuals with different expertise:

Assurance Engineering Management (requirements for assurance process, guidelines on types of arguments / evidence generation to be used)

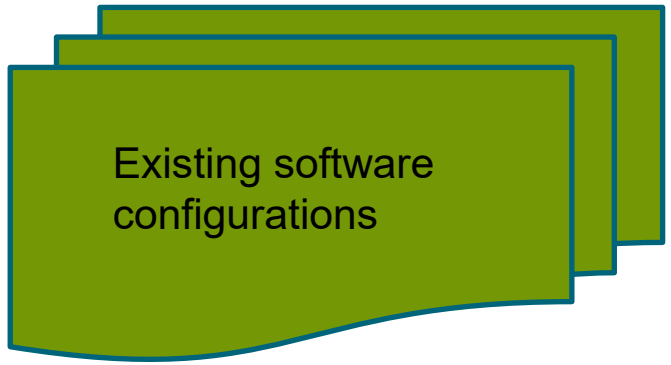
Product Line Expert (modeling, domain expertise)

Assurance Engineer (safety expertise)

Formal Methods Expert (template setup and validation)

Challenges of OTA Assurance for Safety-Critical Product Lines

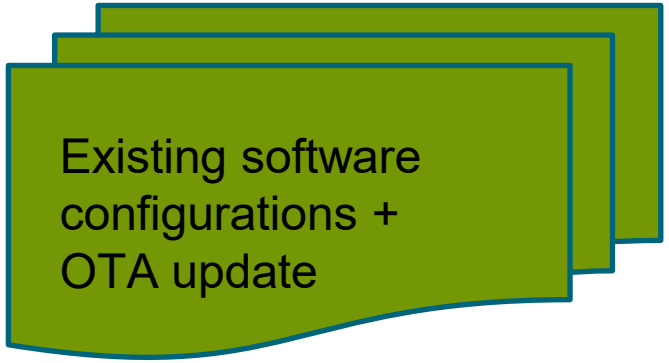
Aim to use the notion of product lines to represent variability in space (different configurations) and time (different updates) and assure them together



How to verify the update?
How to assure the update?
How to test the update?

What information to keep about each feature?
What information to keep about the entire configuration?

so that



… is safe for each configuration
… does not fail in each configuration

What can be proven? How can potential failures be avoided at runtime?

Current and Future Work

- Large scale assurance case development
- Testing for OTA
- Repair of argument, evidence, contracts
- Use of Gen AI for assurance case development and validation
- Static vs. dynamic assurance cases for product lines

Lots more questions – always looking for collaborators!

Acknowledgements

Logan Murphy, Torin Viger, Ali Raeisdanaei, Aren Babikian, Alessio Di Sandro, Prof. Claudio Menghi (Bergamo), Dr. Sahar Kokaly (General Motors), Dr. Ramesh S. (General Motors)

NSERC and General Motors funded this work

UNIVERSITY OF TORONTO

Modeling and verification group

Questions?

UNIVERSITY OF TORONTO