

***Using GSN and ASCE for  
Presenting Formal Program  
Verification Information***

**Nurlida Basir**

nb206r@ecs.soton.ac.uk

DSSE Group, ECS,  
University of Southampton

# Automated Code Generation

---

- Commercial generators (e.g. Real-Time Workshop) are “black boxes”
  - Difficult to understand of what and how code is being generated
- Generator typically not qualified
  - No guarantee that output is safe
- Generator assurance techniques:
  - Correct-by-construction – difficult to implement and extend
  - Testing - time-consuming and expensive process
  - Code review - time-consuming and laborious

# Formal Program Verification

---

- Product-oriented assurance approach
  - Certify each and every generated program
- Hoare-style program verification
  - For each statement a corresponding Hoare rule is given
  - A VCG applies the rules to a program, which produces a number of logical statements or proof obligations
  - **Correctness = VCs proven**

# Formal Software Safety Certification

---

- Follows similar lines as proof carrying code
- Property-based verification - safety property
  - Exact characterization of property stating that “something bad never happens”
  - Important requirement of high-assurance software
  - Different safety properties are formalized by different safety policies in a form of Hoare logic
    - highly dependent on the actual program and the properties being proven
    - maintaining safety environment and producing the appropriate safety obligations of the program
- Formal **proofs as evidence** for assurance claims

# Formal Program Verification Issues

---

- Formal proofs by themselves are no panacea
  - Difficult to relate to the code
  - More opaque or cluttered with technical detail - difficult to understand
- Certification tools
  - Complex systems: unforeseen interactions
  - Hide the complexity of the detailed steps of the underlying strategy: provide only pass or fail result

## **Solution: How?**

---

- Additional pieces of evidence need to be included
- Traceability between the proofs, the certified program and the certification tools is important
- How to comprehensively show that the generated programs satisfy a safety property of interest based on the generated proofs?
- How to establish trust in these proofs?

# Safety Case

---

- “A structured argument, supported by a body of evidence, that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment” [Bishop & Bloomfield '98]
- Why argument?
  - To communicate the relationship between the evidence and objectives
  - To assert that the truth of the claim is a logical consequence of the evidence
  - To critically evaluate the validity of claim

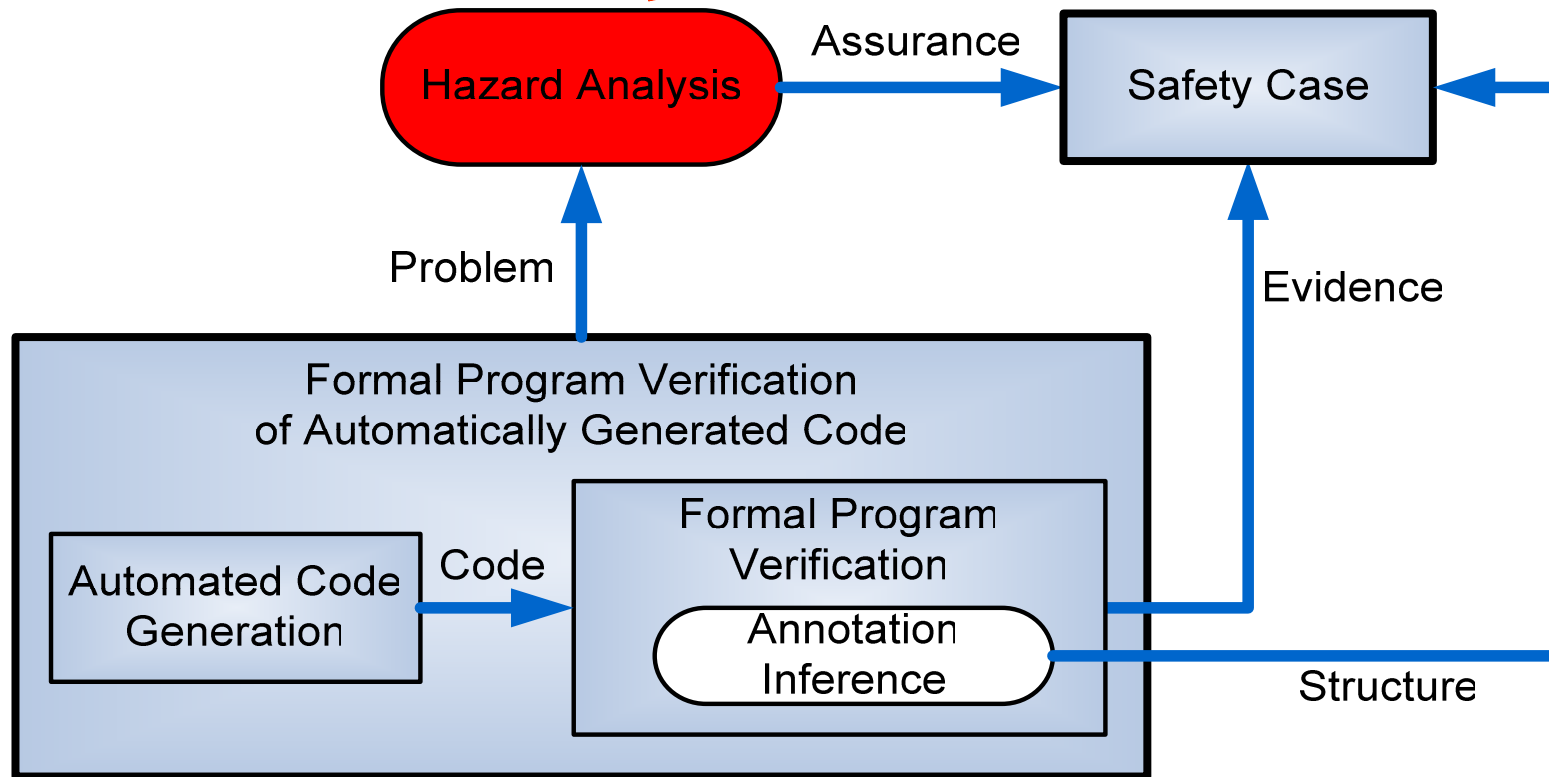
## **ASCE-GSN v3.5**

---

- ASCE is a flexible and extensible graphical hypertext system for developing and managing safety and assurance documents
- Availability of plugins – provide additional functionality within ASCE
- A graphical tool for safety cases construction - explicit representation of the logical flow of the safety argument

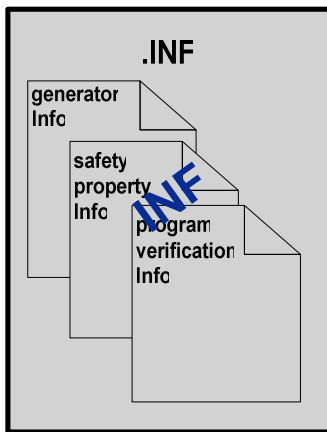
# Solution: From Proofs to Safety Cases

Undetected violation of the given safety property



# Safety Case Generation

Generator & FSSC



Prolog



Doctype



Safety Case Construction

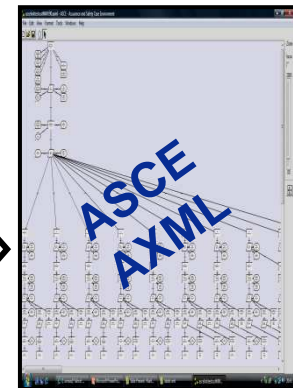
XSLT



Doctype



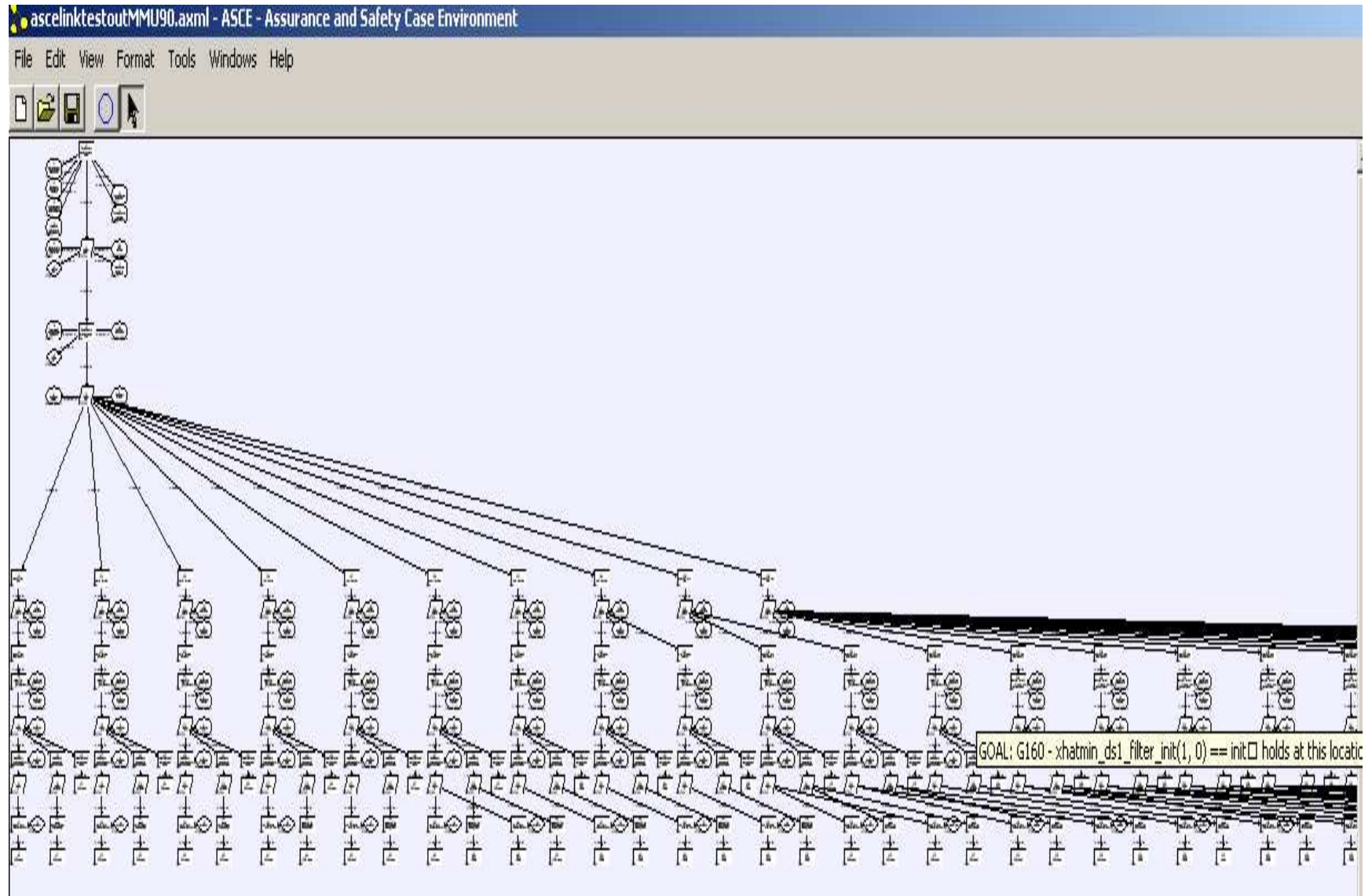
Java



Doctype



# Program Safety Case

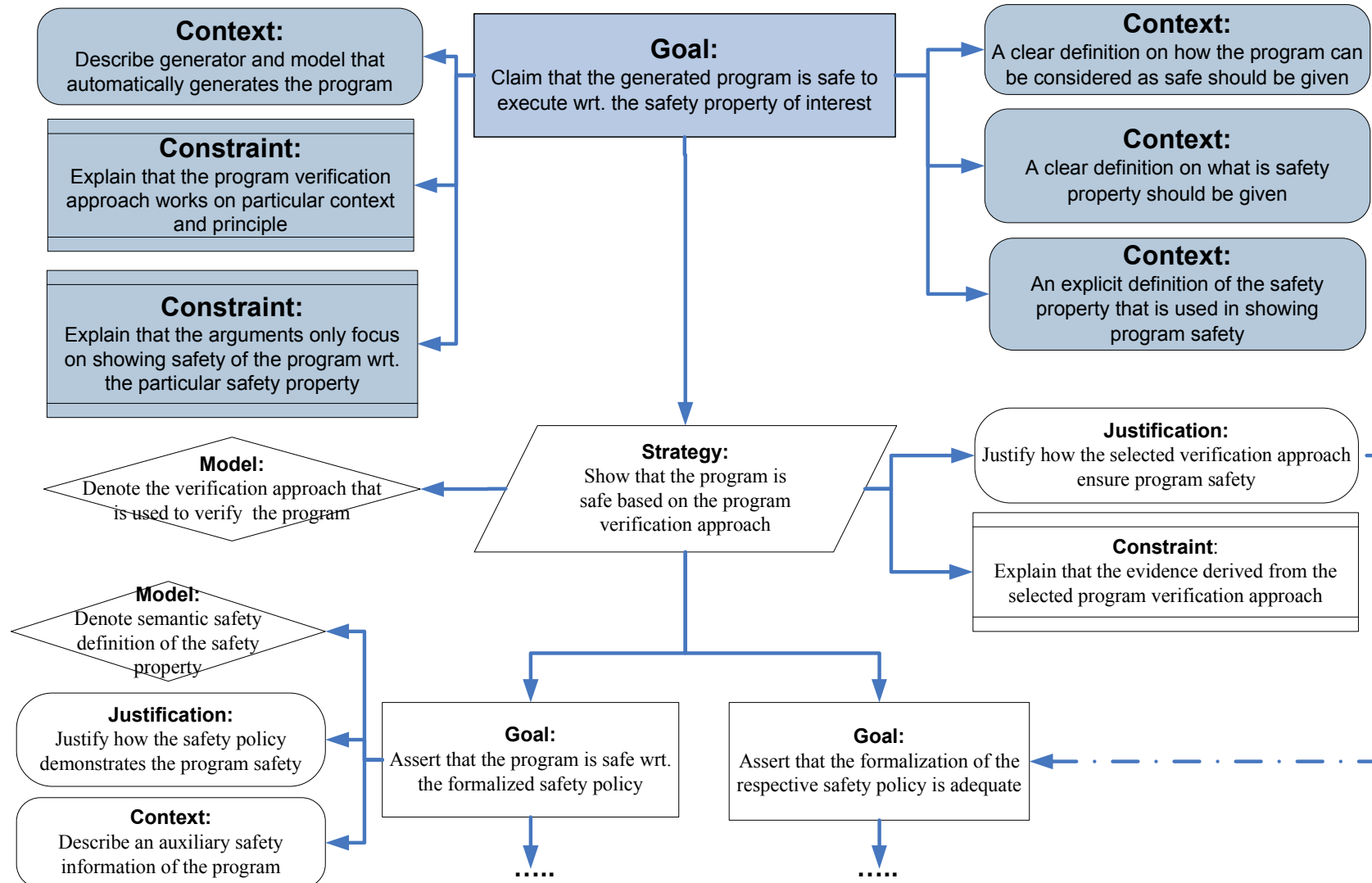


# Program Safety Case

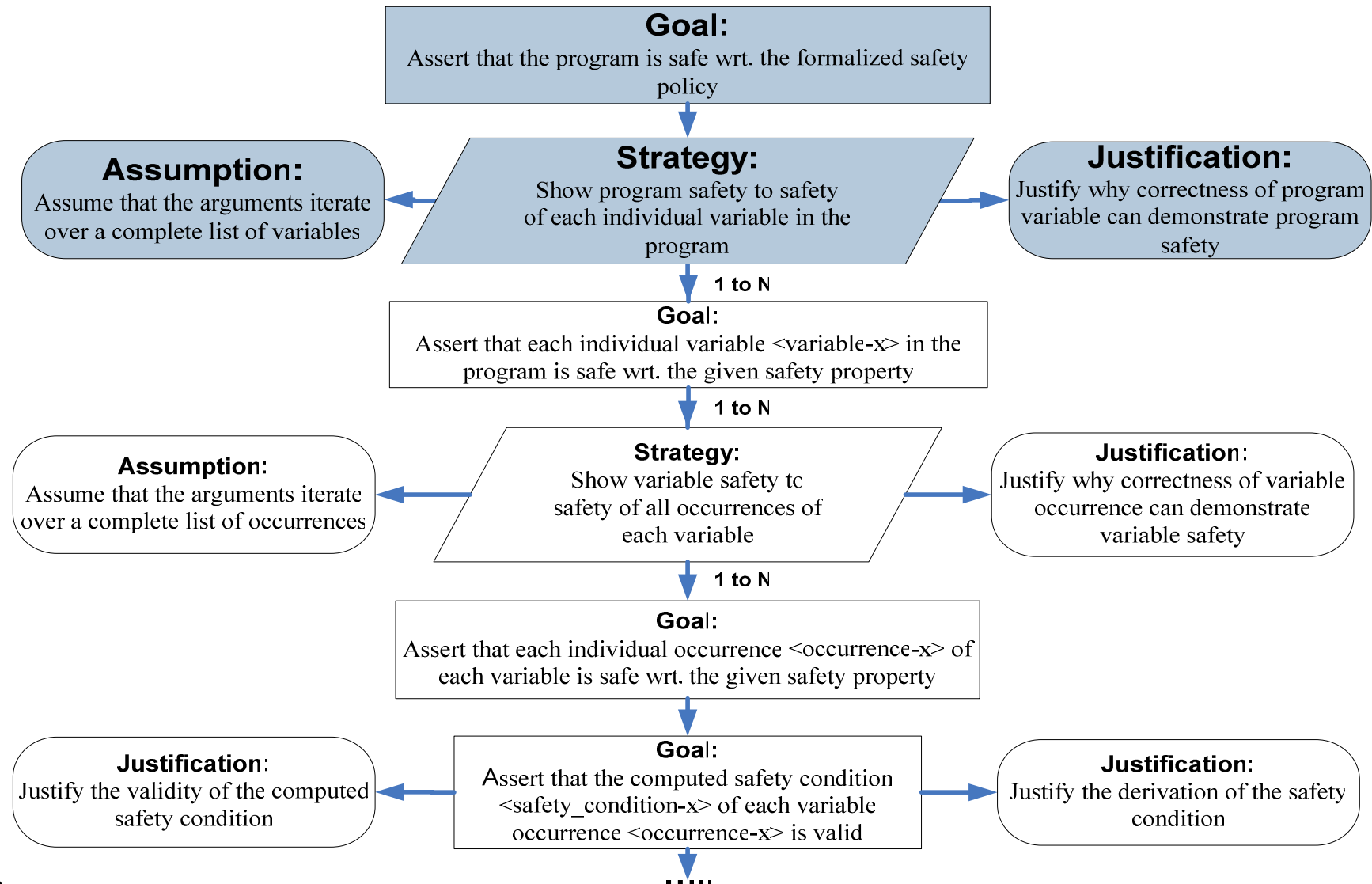
---

- Need structure: multi-tiered argument
- Upper tier:
  - Generic: argumentation structure is independent of the given safety property and program
  - Notion of safety and the formal definitions for the given safety property
- Two lower tiers:
  - Safety of the program as governed by the property
  - Constructed individually to reflect the program structure

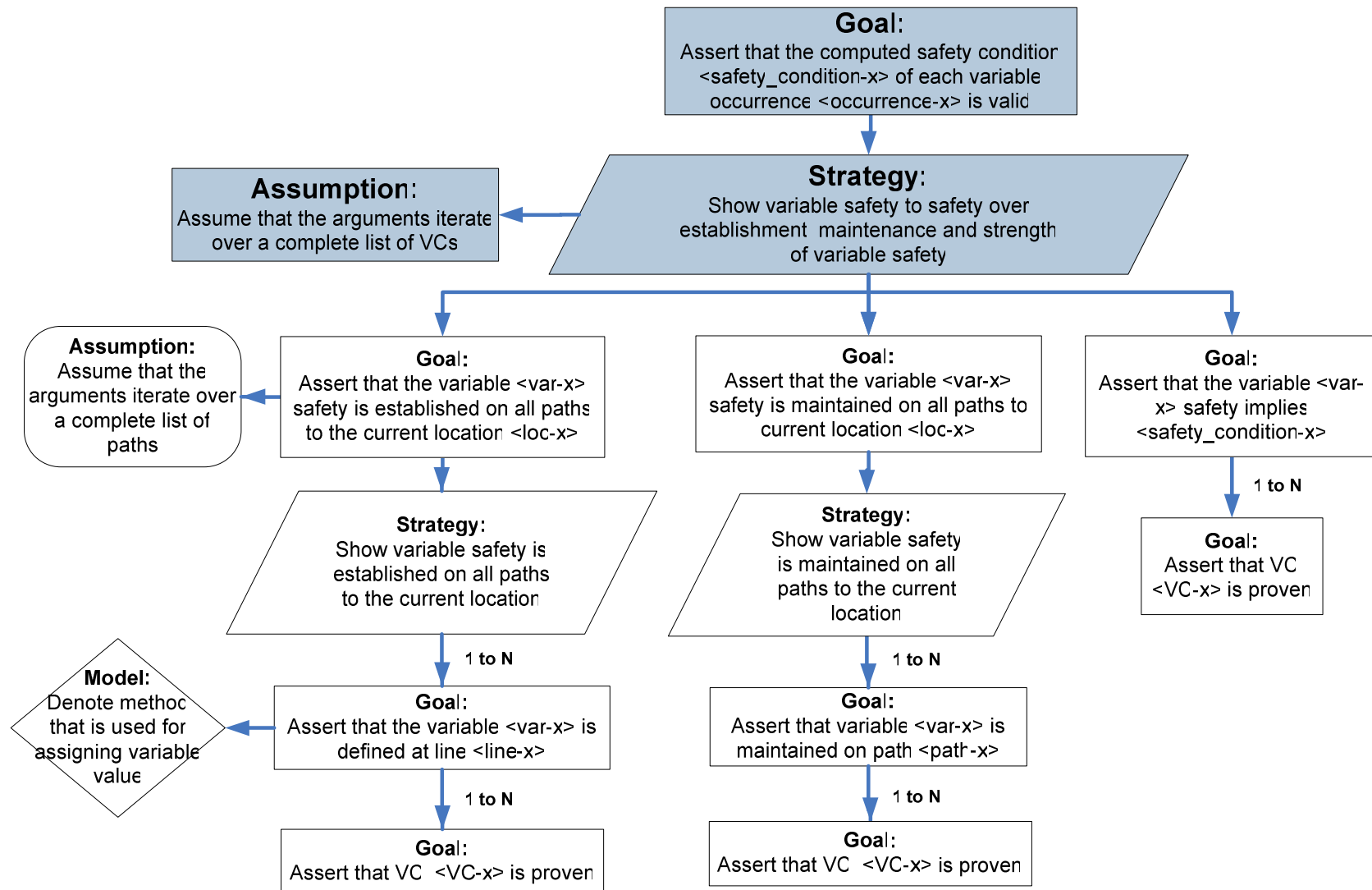
# Tier I: Explaining the Safety Framework



# Tier II: Arguing over the Variables



# Tier III: Arguing over the Paths



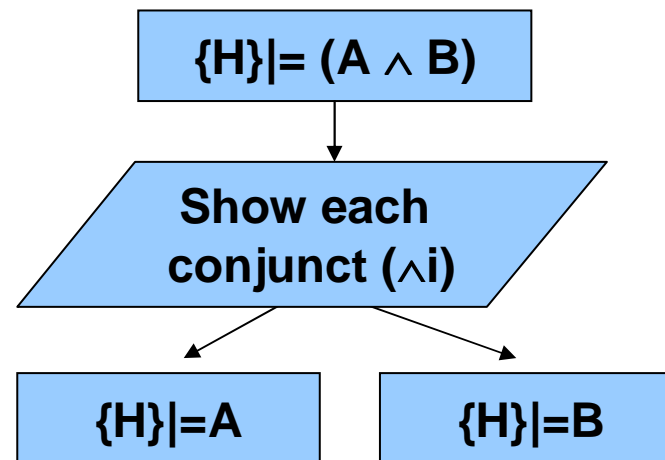
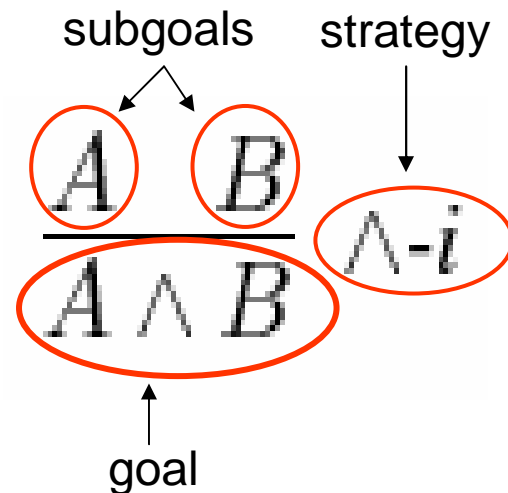
# Natural Deduction Proof Safety Case

---

- Formal proofs as evidence for assurance claims
  - typically constructed by ATPs
- Natural deduction is a collection of proof rules that
  - manipulate logical formulas and transform premises into conclusions
  - closer to human reasoning
- Construct safety cases
  - reveal the underlying argumentation structure and top-level assumptions
  - correspond to formal proofs found by ATPs
  - tiling the natural deduction proof tree with corresponding safety case fragments

# Natural Deduction Proof Safety Case

- Natural deduction proofs as safety cases
  - the original theorem to be proved as the top goal
  - the deductive reasoning as subgoals
  - the applied inference rules as strategies to derive the goals
- Example ( $\wedge$ -rule):





## Current Work

---

- Automatically derives init-before-use and frame safety case from information collected during formal certification phase
- Automatically derives natural deduction proof safety case from proof founds by Muscadet prover
- Safety case template for natural deduction rules
- Integrated with ASCE v3.5 tool

## Future Work

---

- Complementary safety cases that argues safety of the program, certification framework and components
- Additional structuring/scoping - reduce complexity
- More tools integration
  - Application to commercial code generators/projects
- Evaluation of generated safety cases

# Suggestions

---

- Plugin - safety case template
- Enormous safety cases – need structuring
  - feature to decompose into modules
  - feature to support a vertical collapse
  - feature to hide supporting nodes (justification, assumption, context, model)
- Collapse subcases – as a permanent feature

## Conclusions

---

- Formal methods can provide high levels assurance of code safety
- Generic safety case
  - structured reading guide for the program and the safety proofs
  - clearly communicates
    - which claims are actually proven
    - which assumptions and reasoning principles both the claims and the proofs rest
- First step towards a fully-fledged software certificate management system
- Establish trust and confidence in safety of the formal proofs and generated code