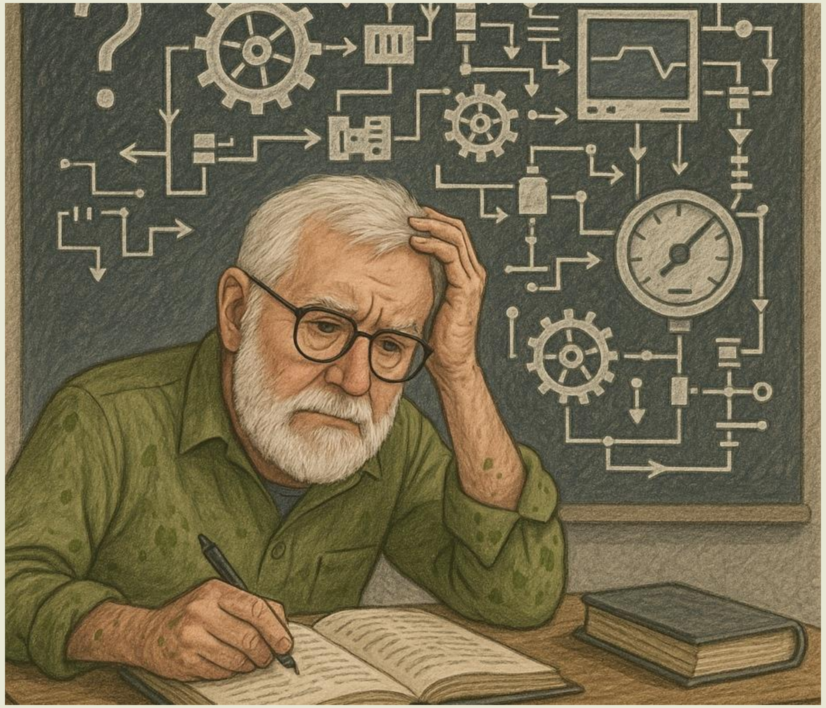
SCC 2025 Annapolis

People struggle to produce good assurance cases…
..would AI do any better?

Robin E Bloomfield
r.e.bloomfield@citystgeorges.ac.uk
(In collaboration with John Rushby, SRI international)
CSR March 2025

Preamble



- Automation is coming to engineering based on Frontier Models
-   Technology and market push
- Assurance cases support decision making
  - https://www.csl.sri.com/users/rushby/assurance2.0
- Need to examine what value we bring
-   For now, justify use of AI
-   Future, justify non-use of AI

Driving on thin ice

- Why did I prefer the “worse” safety report?

Struggles

- Type 1 struggle
  - People *struggle and* ***therefore*** produce good assurance cases and inform decision makers

Type 1 Understanding

- Type 2 struggle
  - People *struggle and* ***despite*** *this* produce good assurance cases and inform decision makers

Type 2 Efficacy

- Type 3 struggle
  - People *struggle and do not* produce good assurance cases and inform decision makers

Type 1 Understanding

Type 1 Struggle
People *struggle and therefore*

Struggle - to try very hard to do something when it is difficult or when there are a lot of problems

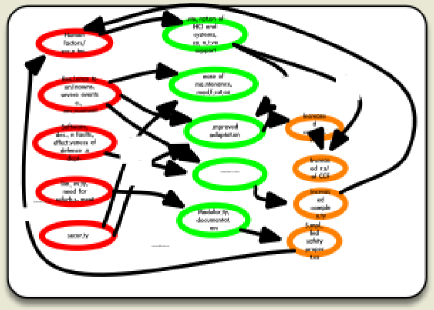
Fundamental Safety Principle FP4

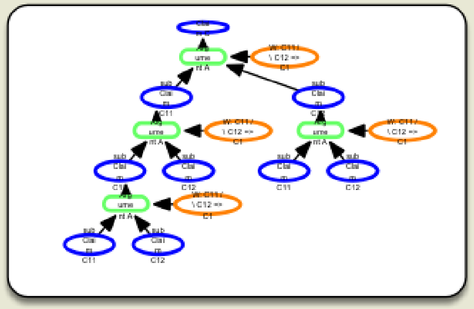
Fundamental principles	Safety assessment	FP.4
Dutyholders must demon ate effective understanding ar control of the hazards posed by a site or facility through a mprehensive and systematic ocess of safety assessment.		

- UK Nuclear Safety Assessment Principles (SAPS)
- Safety and Assurance Cases a mechanism for showing understanding
    - https://www.onr.org.uk/media/34ijvfkc/ns-tast-gd-004.docx
    - https://www.onr.org.uk/publications/regulatory-guidance/regulatory-assessment-and-permissioning/safety-assessment-principles-saps/2014/11/saps-2014/
- Security: hazards --> threats

Development of understanding – system and decision

Mental models

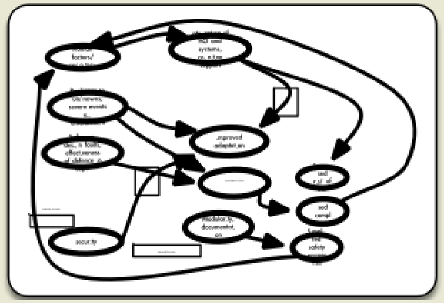


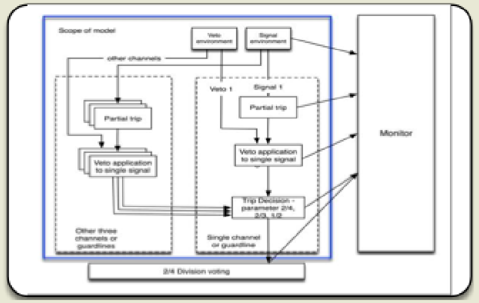


CAE structure

Initial models

Engineering models

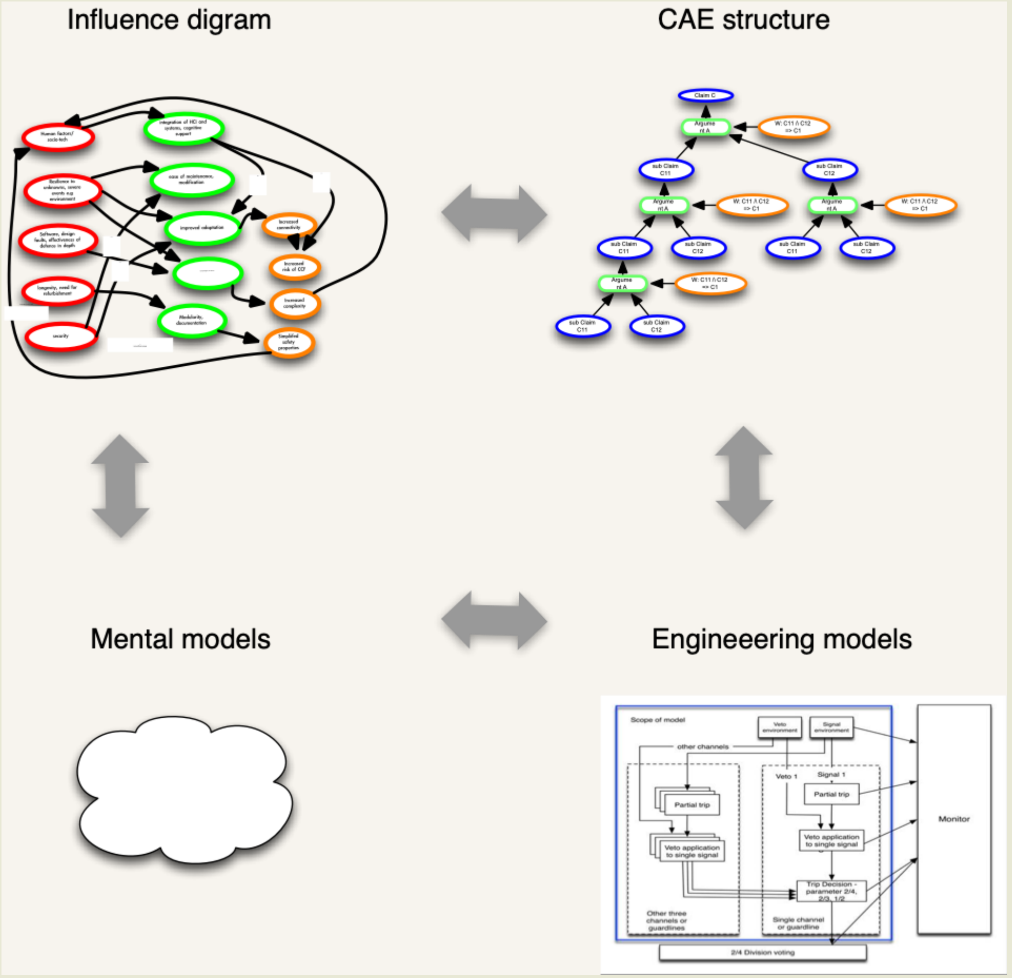




Models can be central to understanding and assurance

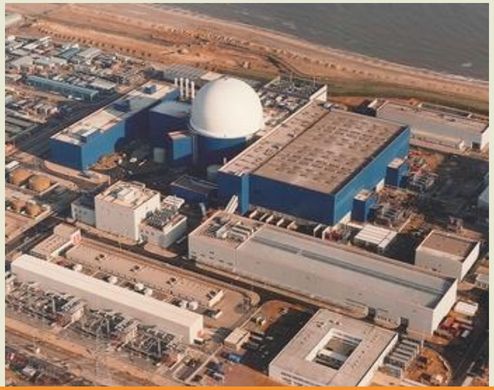
- Understand engineering through models
  - Fundamental to Hazard Analysis
- Engineers use models to change the world
  - Scientists to understand it
  - System and argument models
- Models become embedded in the world
  - Trains vs cars
  - Reactor design
- Fundamental to our understanding of behaviour and risks
  - Adjust the world to make it tractable, adjust model fidelity, abstraction

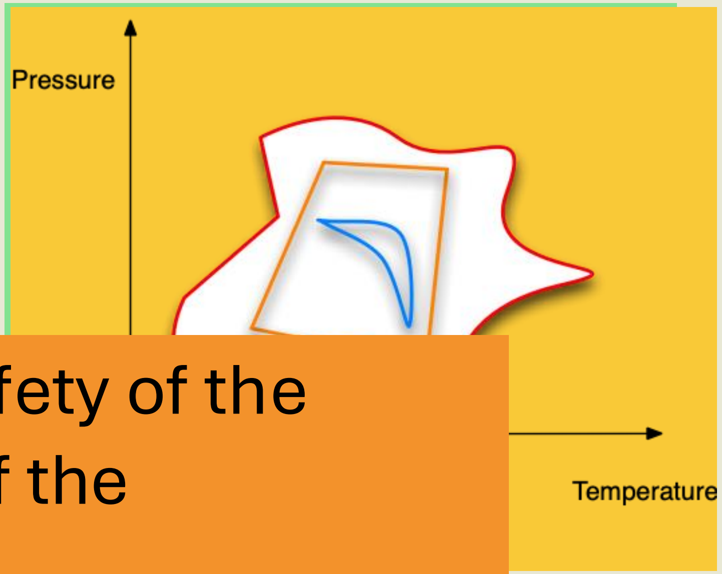
- See
  - Bloomfield, Rushby Assure 2024 paper
  - Edward Lee, Plato and the Nerd



Leverage guards and viability domains

- Simplify world so can model it
- Sensing of the exact state of the word not feasible/expense
- Approximate envelopes of viability domain
- Guards and s
  domain, prot





Guards allow us to understand the safety of the system with only limited knowledge of the components and partial sensing

Not just hazard mitigation
System has a purpose

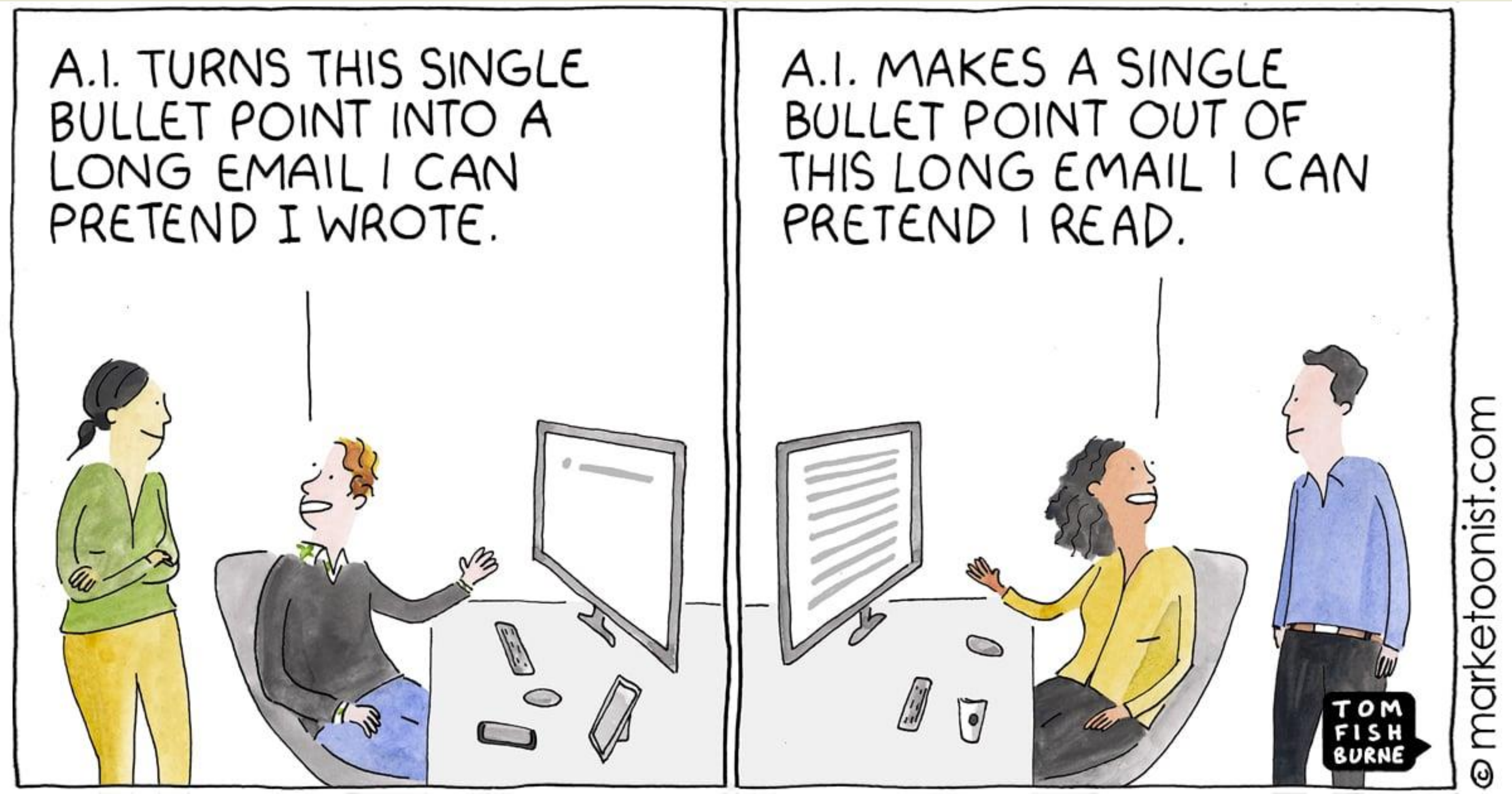
Safety cases

- Models as a basis for recording and demonstrating understanding
    - Predictions, assumptions, limitations, sensitivities, validity…
- Models of the system and the decision
- Safety cases
    - basis for decision making
    - as a model of understanding of support decision
    - a mixed machine/human engineered socio-tech artefact
- Understanding the model, its contextual validity
- (usual quote about all models be wrong but some are useful)

Driving on thin ice

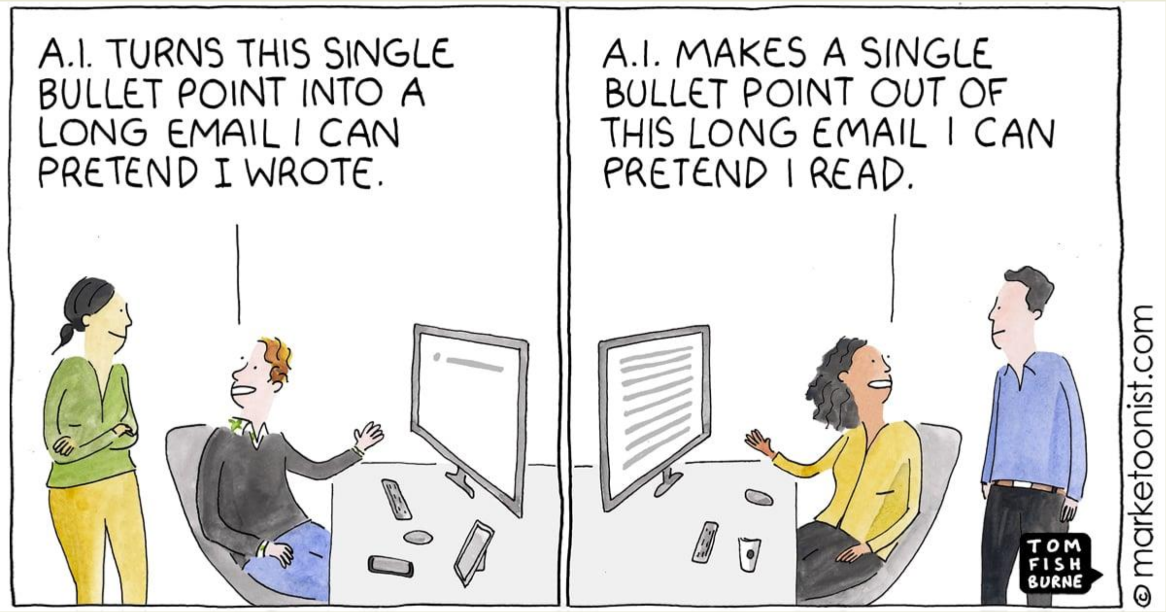
- Why did I prefer the “worse” safety case?

Automation impact



Automation impact

- Type 1 - understanding
- Type 2 - efficiency
  - Interaction
- Role of AI
  - Separate artefact productions from from essential understanding
  - Role of AI in promoting understanding
- Evaluation impact of automation on both



Automation

Don’t we know how to design automated systems?

Why AI automation different

- Starting point
  - Side effects – do we understand what makes good decision now and how impact it
  - Do XYZ inspectors have enough time to do credible job?
  - Automation disappointments – but we have agency
  - Automation bias, adaptation, heterogeneity of users/problems
- Increased AI complexities
  - Technology performative
  - Concurrent institutional and societal change
  - Scheming, misalignment, hallucination
  - Articulate and persuasive, beguiling
- New type of software? New team member? Automation case – now/future?

See
Apollo Research believes that o1-preview has the basic capabilities needed to do simple in-context scheming".. O1 System Cars, OpenAI

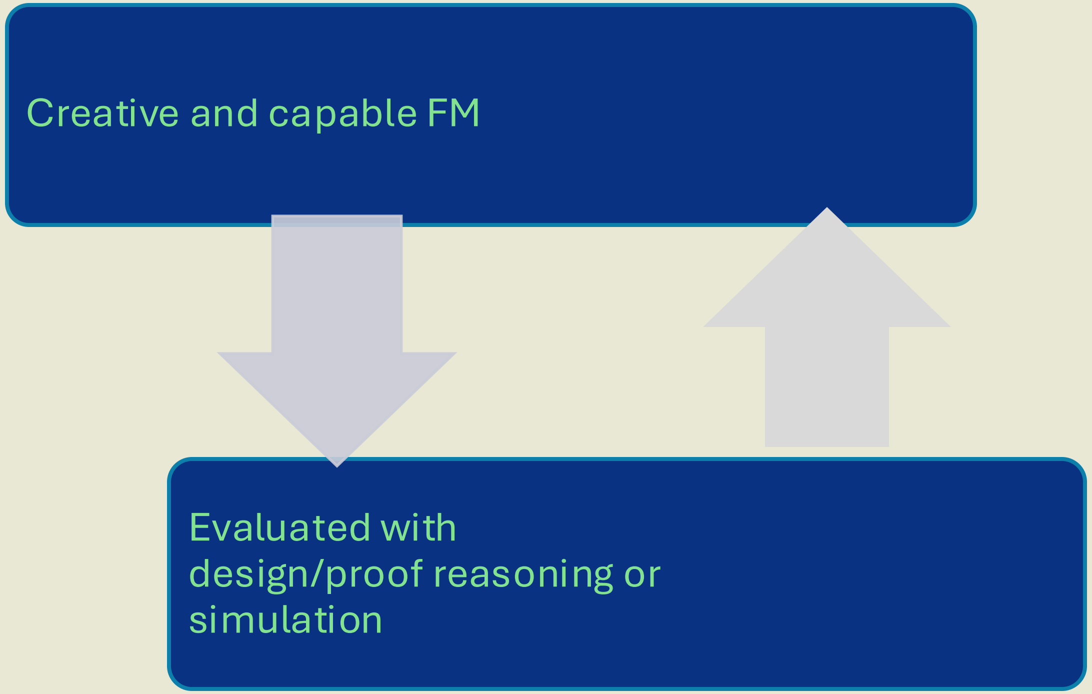
Automation experiments /POC

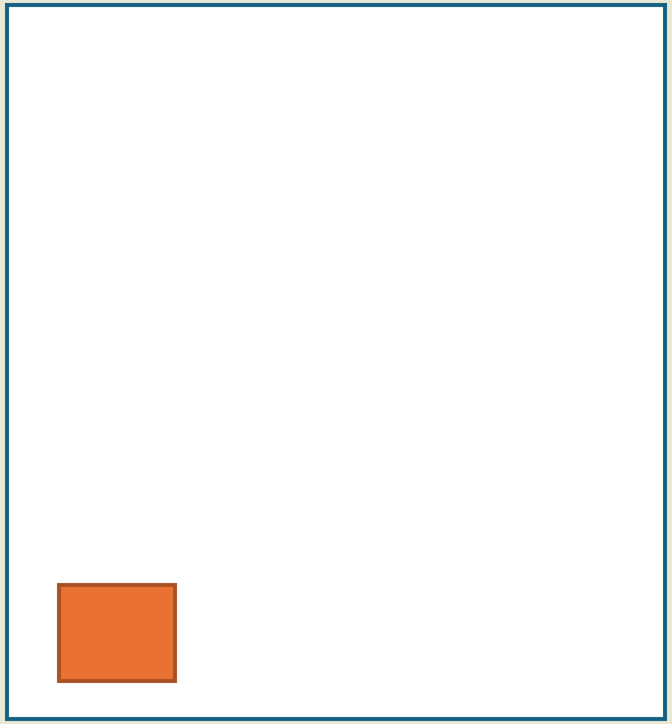
Synthesis

Auto formalisation

Defeaters

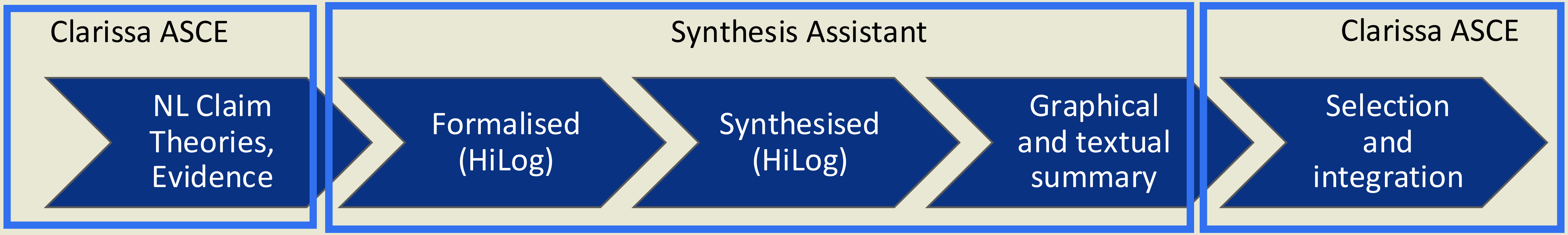
AI more than LLM/ML
AI = LLM/ML + computational reasoning
FM = Frontier Models



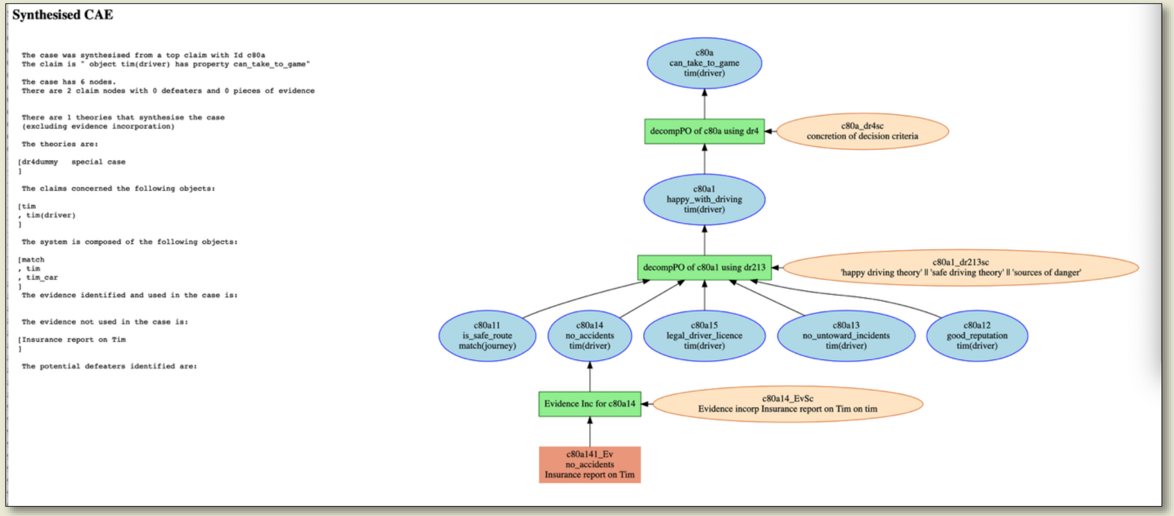


Assurance case synthesis

Synthesis Assistant is a research tool designed to synthesize claims, arguments and evidence structures from a root or top-level claim.

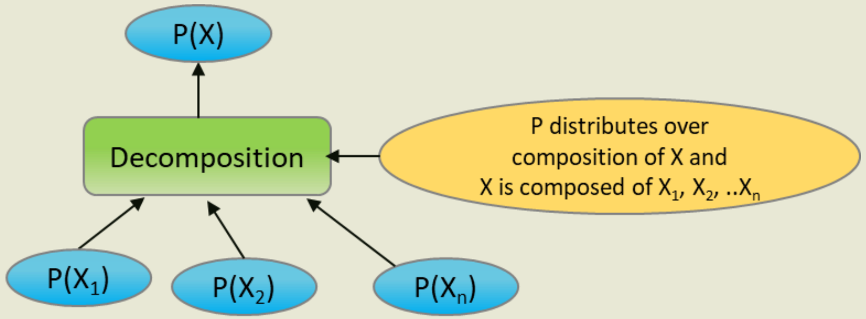


- Given:
  - Top-level claim (defined in ErgoAI or node imported from an ASCE file)
  - Definition of the system structure
  - Possible defeaters
  - Theories used to develop the case
  - Evidences for the case



Theory based synthesis

- Defining a theory in terms of
  - the classes involved - the ontologies
    - kgOnt(?K, ?VSubj1, ?P1, ?VObj1) - identifier is ?K and then read as a knowledge triple of the types/classes, subject-property-object
      - E.g. kgOnt(kgTopConc, sys, meets, RFPreq).
  - the connections between making a CAE Block
    - theoryRuleConnection(kg1, {kg11, kg12}, 'split into funct and non-funct requirements’).
- Define instances of these ontologies
- Define top level claim
- Also define defeaters and evidence



Supporting evaluation and communication

- Shift review effort to
  - Understanding theories
  - Assess their relevance and validity
  - Trust in tools
- Complexity reduction
  - Benefits increase with size of case
  - (Experimentation)
- Generate all cases wrt a constraint
  - Select on cost or some psychological complexity metric
- Checks for
  - Unused evidence, components

Summary of case

```
 The case was synthesised from a top claim with Id c2000
 The claim is " object sys2(sys) has property is_dependable"

 The case has 31 nodes.

 The theories used to synthesise the case are:

tr1   Split of time into epochs of now and future
tr2000   Split of time into epochs of now and future
tr2001   Double decomp Platform Application
tr2001a   Split on hw/sw
tr2004   Model relating confidence in zero defects to reliability
tr2005   Model relating OpEX and defects to reliability
tr2006   Conservative substitution – WCT
tr5   Concretion of timeliness to time response

 The evidence identified and used in the case is:

opex_report
wct_analysis_report


 The potential defeaters identified are:

Known vulnerabilities in this platform
These models have onerous assumptions
wct not feasible for codebase size
```

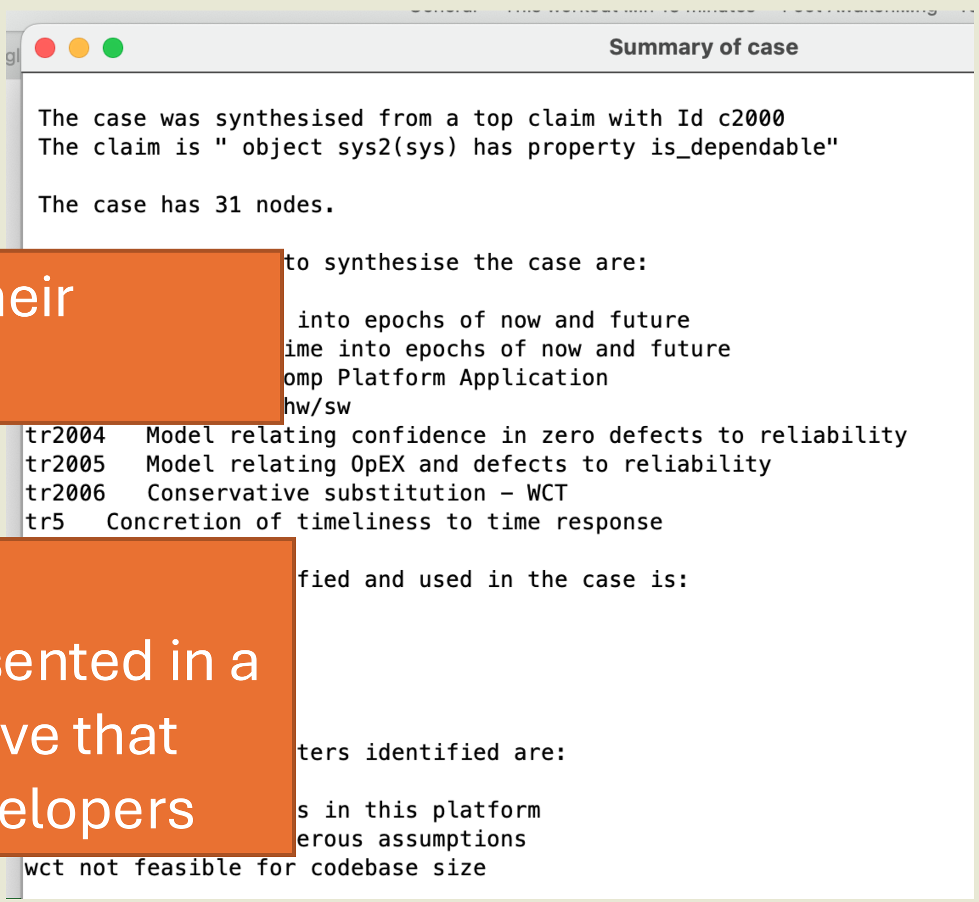
Supporting evaluation and communication

- Shift review effort to
  - Understanding theories
  - Assess their relevance and validity
  - Trust in tools

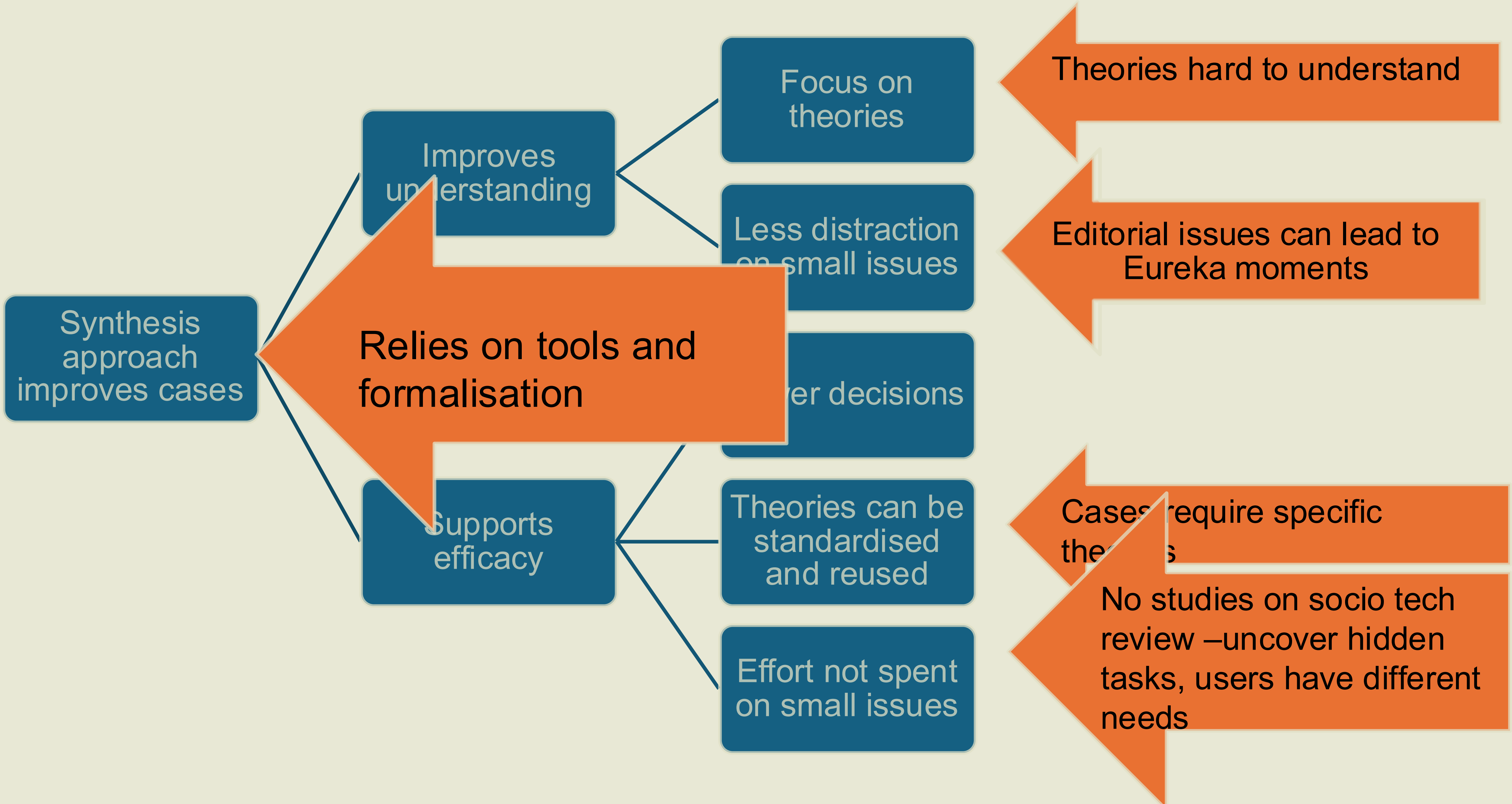
Understand theories and the validity of their application - the rest is “knitting”

- Generate all cases wrt a constraint

Structure of the justification produced automatically but, as always, this is presented in a safety case report that provides a narrative that can be judged by the case users and developers

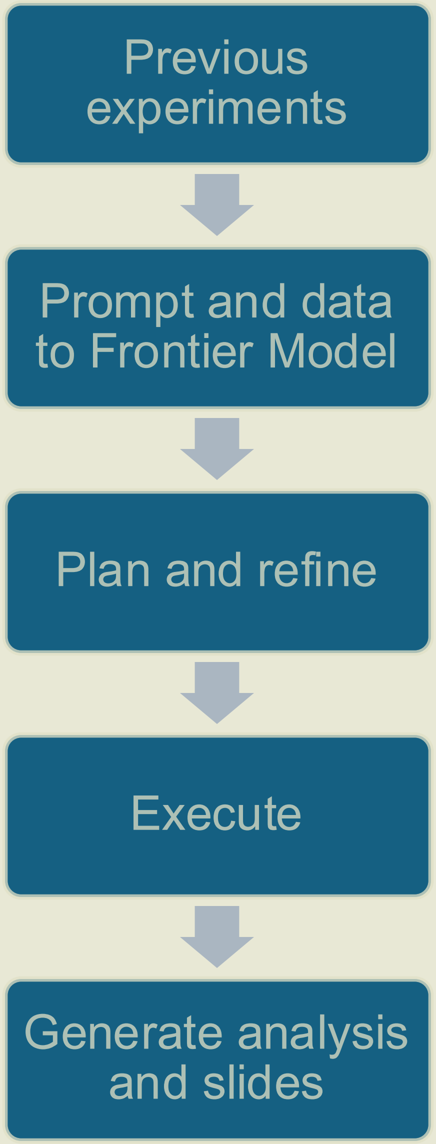


Hypothesis tree



Auto formalization of claims

- Synthesis requires formalisation
- Need to investigate efficacy of formalisation to make approach viable - for tools and experimental evaluation
- Set of real safety case claims, anonymized. No constraints.
- Used Gemini Pro to formalize as Knowledge Triples in logic language



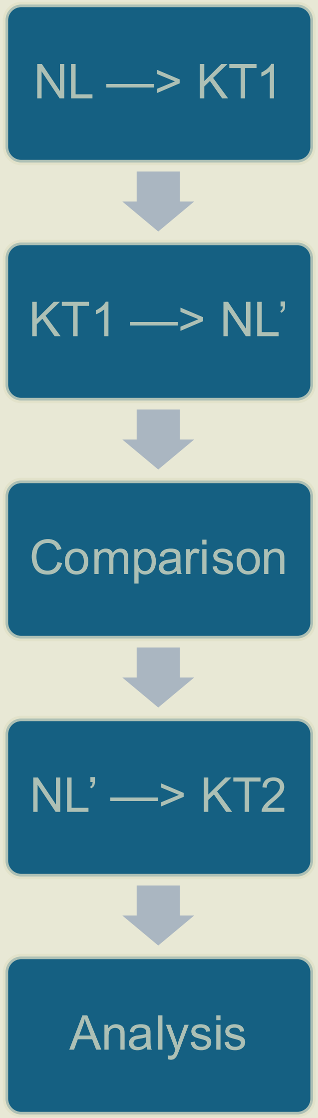
Introduction & Objectives

- Goal: Assess feasibility and limitations of using LLMs (Gemini Pro) for automated formalization of NL assurance claims into Prolog KTs (KT(S, V, O)).
- Why? Potential to reduce manual effort in assurance case development and enable automated reasoning.
- Key Questions:
  - How accurate is the NL -> KT conversion?
  - What types of information are lost or altered?
  - Can back-translation (KT -> NL') effectively detect errors?
  - What are the challenges and future potential?

Methodology

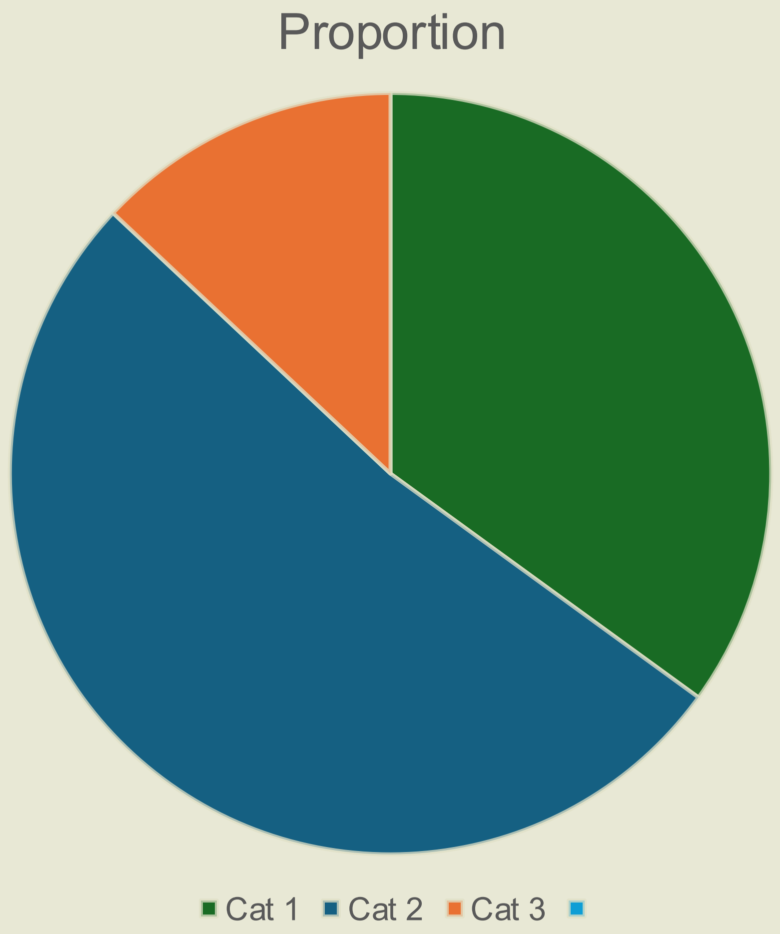
- Input: Corpus of real-world assurance claims
- Process:

1. NL -> KT1: LLM translates NL claim to NL(ID, Text) and KT(ID, S, V, O) (or rule).
2. KT1 -> NL': LLM reconstructs NL phrase (NL') from KT1.
3. Compare & Categorize: Human/machine comparison of original NL vs. reconstructed NL'.
    - Cat1: Direct Match
    - Cat2: Meaning Preserved (Minor diffs)
    - Cat3: Meaning Altered / Lost
4. Analysis: Identify error patterns, limitations.



Categorization Results

- Dataset: 285 unique claims processed.
- Distribution:
  - Category 1 (Direct Match): 101 claims (~35%)
  - Category 2 (Meaning Preserved): 148 claims (~52%)
  - Category 3 (Meaning Altered/Lost): 36 claims (~13%)]
- Observation: LLM handles a majority of claims reasonably well (Cat1 + Cat2 ≈ 87%), especially simple declarative sentences. However, significant issues exist.



Findings: Limitations & Information Loss (Cat 3)

- Loss of Nuance/Detail: SVO reduction discards qualifiers, context.
  - Example (kt_30): "socio-technical-financial-political complexities" -> complexities.
  - Example (kt_71): "internal and external non-nominal conditions" -> postulated_non_nominal_conditions.
  - Example (kt_260): Clause ", necessary to support safe operations," lost.
- Handling of Phrases: LLM infers verbs (is, exists, performed) for NL phrases/titles.
  - Example (kt_151): "Periodic review..." -> ... is, performed. Relational words lost.
- Abbreviations: LLM introduced non-source abbreviations (ALARP, vcd, ssa).
  - Example (kt_160): "as low as reasonably practicable" -> ALARP.
- Complex Clauses: Dependencies, conclusions often over-simplified.
  - Example (kt_218): "has been undertaken and concluded that..." significantly shortened.

Findings: Back-Translation (KT -> NL') Utility

- Effectiveness: Good at highlighting discrepancies, especially major information loss (Cat 3). Differences easily detectable by human comparison.
- Limitation: Primarily validates KT1 == KT2 (consistency of formalization -> reconstruction), not necessarily NL == KT1 (accuracy of initial formalization).
- Potential Hidden Errors:
  - If NL -> KT1 was inaccurate (wrong SVO, wrong inferred verb), but KT1 -> NL' is consistent with the incorrect KT1, the error is masked.
  - Example: If LLM wrongly inferred is_adequate instead of is_performed for kt_190 ("Threat assessment by system architecture"), back-translation would confirm is_adequate, hiding the initial error.

Recommendations & Improvements

- NL Pre-filtering:
  - Check inputs for completeness (declarative sentences).
  - Flag phrases, questions, complex sentences for manual review before formalization.
  - Addresses: Issues like kt_151, kt_152, kt_190.
- Disallow Abbreviations:
  - Explicit prompt instruction: No non-source abbreviations (ALARP, vcd etc).
  - Addresses: Issues like kt_71, kt_160, kt_195.
- Standardize Negation:
  - Provide clear prompt examples for handling 'not', 'no', etc.
  - Prefer negation in verb (is_not_X) or use dedicated wrapper (see next).
  - Addresses: Issues like kt_185, kt_191, kt_192.
- Prompt Engineering & Complex Formalisms:
  - Refine prompts for qualifier/context retention.
  - Explore richer target structures beyond simple KTs (see next).
  - Addresses: Issues like kt_30, kt_218, kt_260.

Conclusion

- LLMs are useful assistants for auto-formalization but not a complete solution yet.
- Simple KTs are insufficient for complex assurance claims; significant information loss occurs.
- Back-translation is a helpful but incomplete validation technique; hidden errors are possible.
- Recommendations: Pre-filtering NL, constraining LLM output (no abbreviations, standard negation), and exploring richer formalisms are key next steps.
- Future: Human-in-the-loop approach combined with improved LLMs targeting more expressive formalisms and using better validation methods.

Auto formalisation
Type1 and Type 2 impact and hypotheses

- If hard to formalize -> complex claims -> Needs attention
  - Negation
  - Additional info (context)
  - Too little info (no verb)
- Detection rates and failure modes established for sample
  - Easy to improve with guidance on claims and preprocessing

- But
  - Loss of nuance in claims might be important
  - Possible divergence between narrative and formalized claim structurers
  - Only translating claims not extracting them from narrative

Defeaters

Defeaters

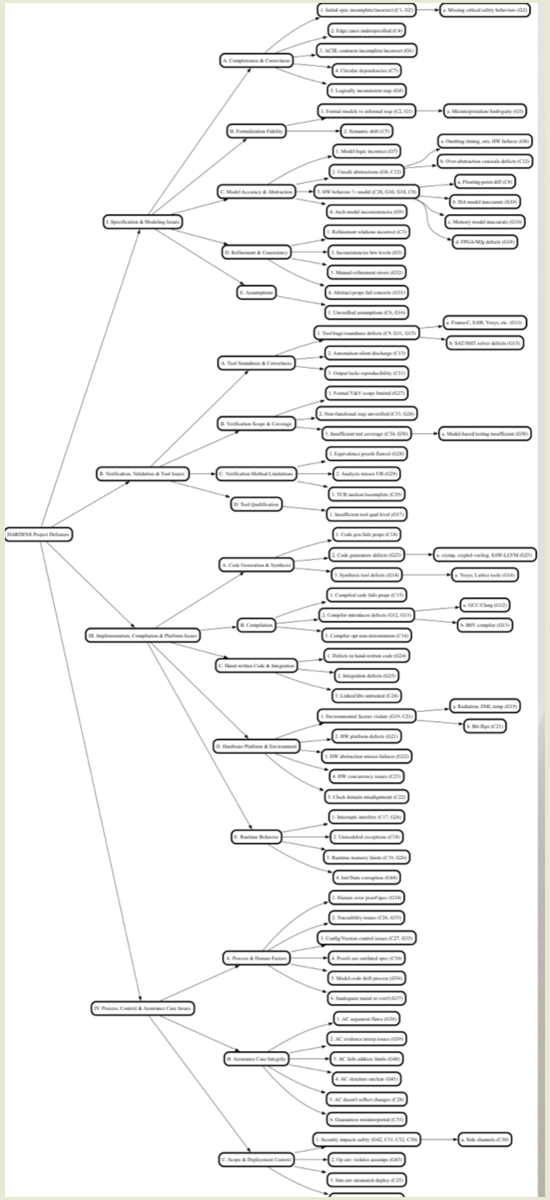
- Trying to break cases basis for indefeasibility, eliminative argumentation
- Hypothesis
  - Searching for defeaters and their mitigation builds understanding
  - Automation can be a colleague and team member in this
- Used ChatGPT, Gemini Pro to extract claims from Hardens project report
- Asked to identify defeaters
- Assessed differences in outputs
  - Identical
  - Specialization/generalization
  - Different
- Speculative “fish tagging” stats to estimate set

Outputs

- Defeater tree

Identical / Very Similar	26. Traceability links between artifacts may be missing or incorrect	33. Traceability links... are missing, incomplete, or incorrect...	Identical concern regarding traceability links.

Specific Version / Generalisation	12. Over-abstraction in model checking may conceal real defects	8. The Cryptol model is an unsafe abstraction, omitting critical timing, environmental, or hardware failure behaviors...	Gemini's point is a specific instance (Cryptol model abstraction) of ChatGPT more general point about over-abstraction concealing defects.

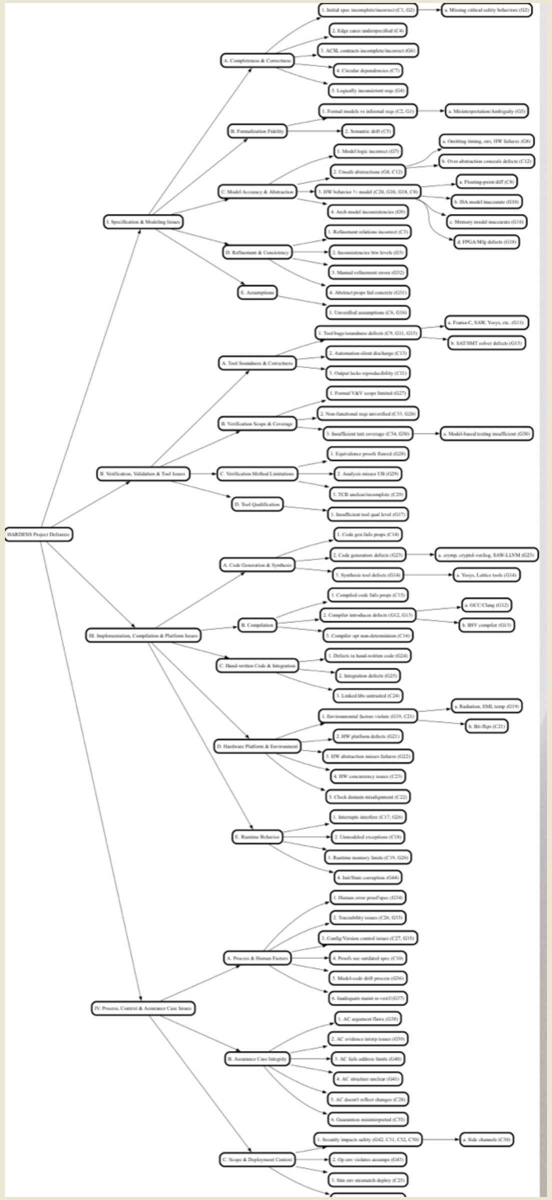


Defeater results

- Both lists identify similar core risks:
  - Tool soundness, formalization accuracy, requirement completeness, refinement soundness, compiler correctness, hardware/environment assumptions, human error, and issues related to security/physical phenomena.
- Scope
  - Gemini explicitly includes several defeaters related to the assurance case itself (logic, evidence interpretation, addressing limitations, clarity) and scope/context (security, operational environment, initialization, configuration).

Difference between models

- Gemini tends to separate concepts like hardware assumptions and model fidelity/abstraction gaps more distinctly, whereas ChatGPT sometimes groups related ideas (e.g., malicious interference and physical faults).
- Gemini includes specific defeaters for verification scope, hand-written code verification, initialization/state corruption, configuration management, and the assurance case logic itself.
- ChatGPT includes specific defeaters for equivalence checking limitations, traceability gaps, lack of version control, stakeholder misunderstanding, and tool-induced false confidence.



Overlap and “fish-tagging” - speculation

- Calculate the capture-recapture estimate based on the first comparison (full ChatGPT list vs full Gemini list),
  - but excluding security-related defeaters and those explicitly categorized under "Assurance Case“
- Using the Lincoln-Petersen index with the adjusted counts:
  - Estimated Total Population (N) = (Adjusted n1 * Adjusted n2) / Adjusted m
  - On overall list 40 estimates ~ 70 population
  - On top 20 “fat fish” estimate ~ 30 population
- Not meaningful but hints of approach

Defeater conclusions

- Dialogue – supports understanding
  - Refining prompts help with clarity
  - Different assumptions about scope revealed (different classes)
  - Interesting detail (different instances)
  - Specialization/instances showed need for prompt tree or classes
  - Diversity/second opinion from using 2 models
- Drawback of defeaters search
  - Anything might be a defeater (see inverted clauses in standards)
  - Context understanding and judgement of model key
  - Did not find what I judged key defeater but found the class

- But …
  - Common issues with training sets lead to lack of independence of sample
  - Reduced scrutiny
  - Temptation
    - How much to validate
  - Distraction/dilution
    - Can displace actual thinking effort
    - Somewhat addictive
  - Swamping
    - Judgement of key issues
  - Scaling
  - Persuasive
  - Need for focus and judgment
    - what are likely in this project

Discussion and conclusions

Summary

- Automation driven by technology push, systematic evaluation and engineering judgement
- Understanding a key safety principle
    - Use of architectures and design to reduce what we need to understand for safety and security
- Propose *models* of the decision and system as the key to producing, documenting and communicating *understanding*
- Clarified role of (Assurance 2.0) cases in *understanding and as a model for decision making*
- Struggle with cases inherent in trying to understand and in dealing with complex documents and issues. Introduced Type 1 and Type 2 struggle.

Analyses tasks and “struggle”

- Isn't this just about automation of tasks... don't we (you) know how to do this?
  - Need role based analysis and hypotheses
  - Who for, Who or what wins/looses, How does it change over time
- More conventional territory but need go beyond AI good/bad as depends on task and context and how that might change and evolve
  - Technology performative, persuasive, scheming, misaligned and insecure
- Need for evaluation

Automation evaluation

- Differentiate between impact on efficacy and understanding
  - Hypothesis tree (and defeaters)
    - Role of theories
      - Reliance on formalisation
- Reported on some preliminary experiments
  - Formalisation reliability and failure modes
  - Defeaters and different LLMs
- Evaluation would require
  - Role based analysis and development of hypotheses
  - Corpus of work to use as basis for evaluation
    - From literature and practice
    - Synthetic – AI generated

Conclusions

- Use of AI on assurance case automation
  - Differentiate between impact on efficacy and understanding
  - If view safety case report merely as an artefact then get a different automation strategy then if we say understanding is key
- Considered role of theories in understanding and dependence on formalisation
- Demonstrated how existing LLM (Frontier Models) can be used to formalise, compare and evaluate formalisation results and defeater discovery
  - Significant increase in LLM capability
- Community should consider need for body of knowledge, challenges, benchmarks for evaluation

Thank you

CITY
ST GEORGE'S
UNIVERSITY OF LONDON