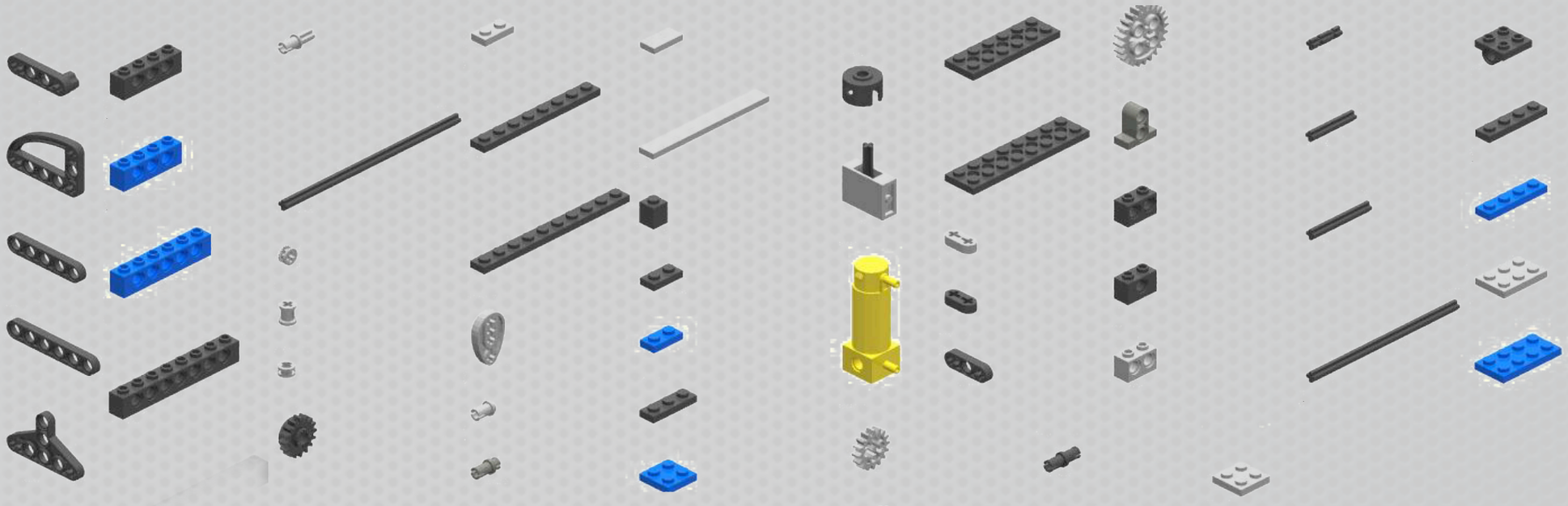
Towards Safer Composition

How to verify lots of similar systems?

Andreas Classen Patrick Heymans Thein Than Tun Bashar Nuseibeh

Software product line engineering

Reusable Assets



Products

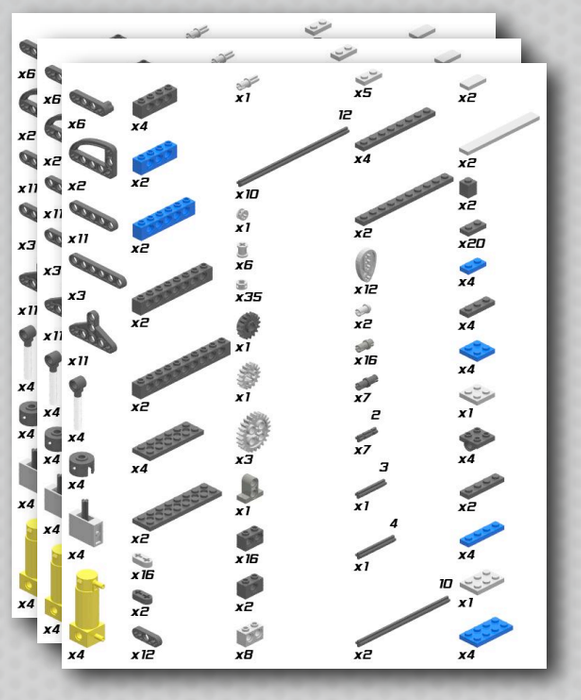
...

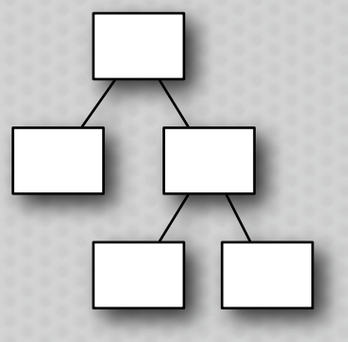
Lots of other combinations.

ICSE/NIER 20/05/09

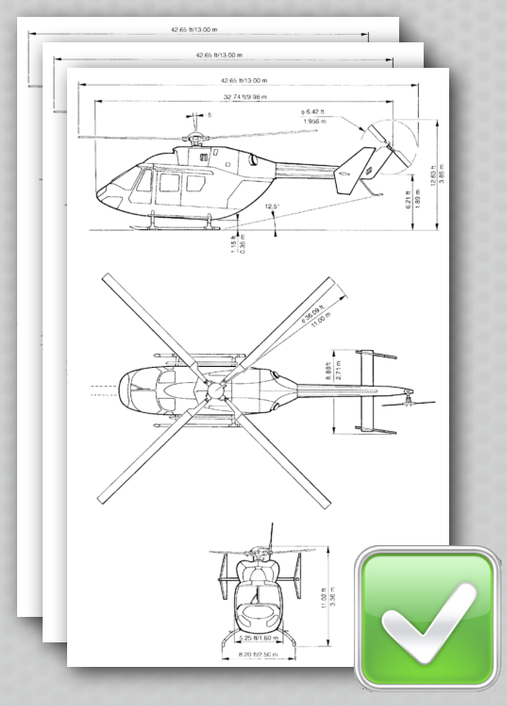
Model-based verification

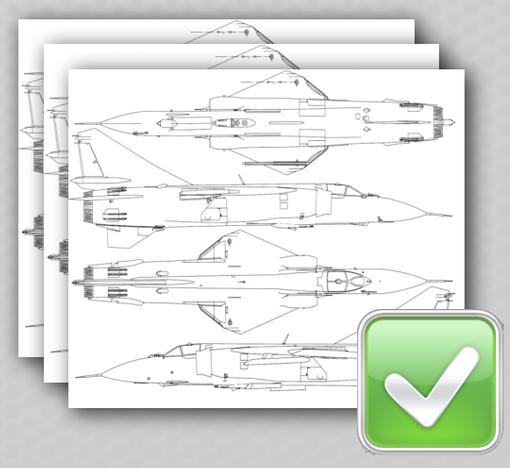
Reusable Assets

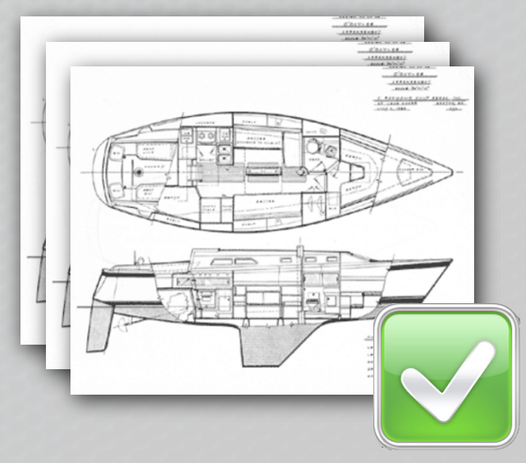




Products



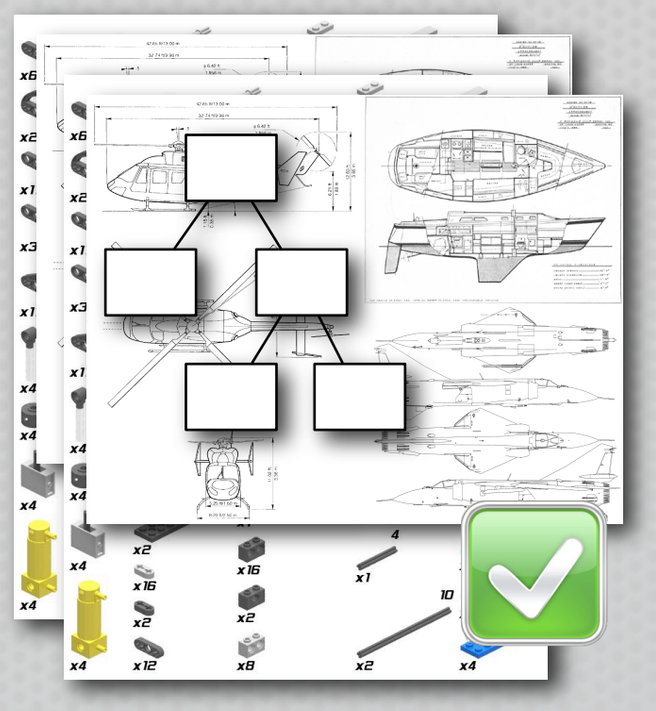




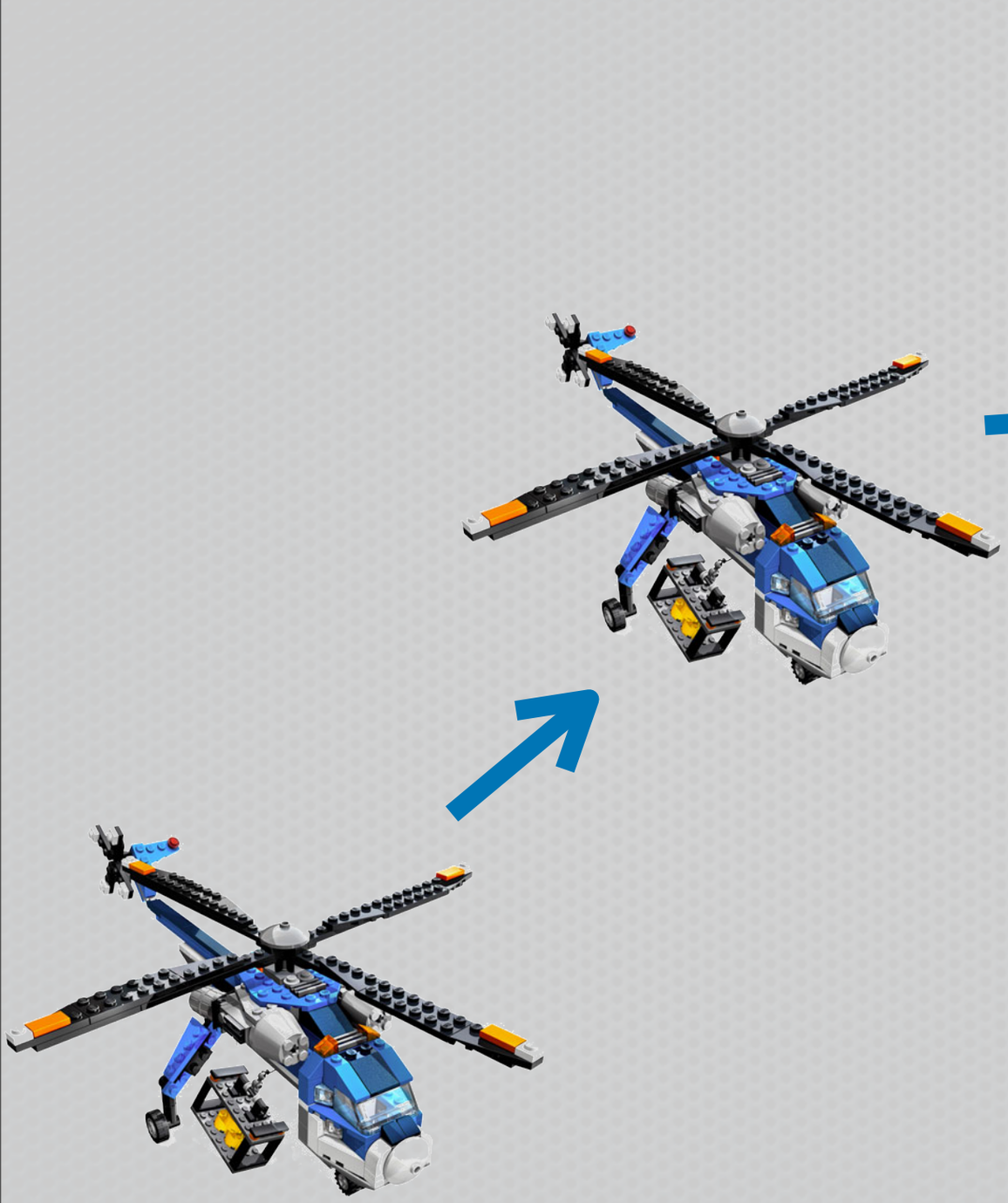
...

Goal: verify generic model

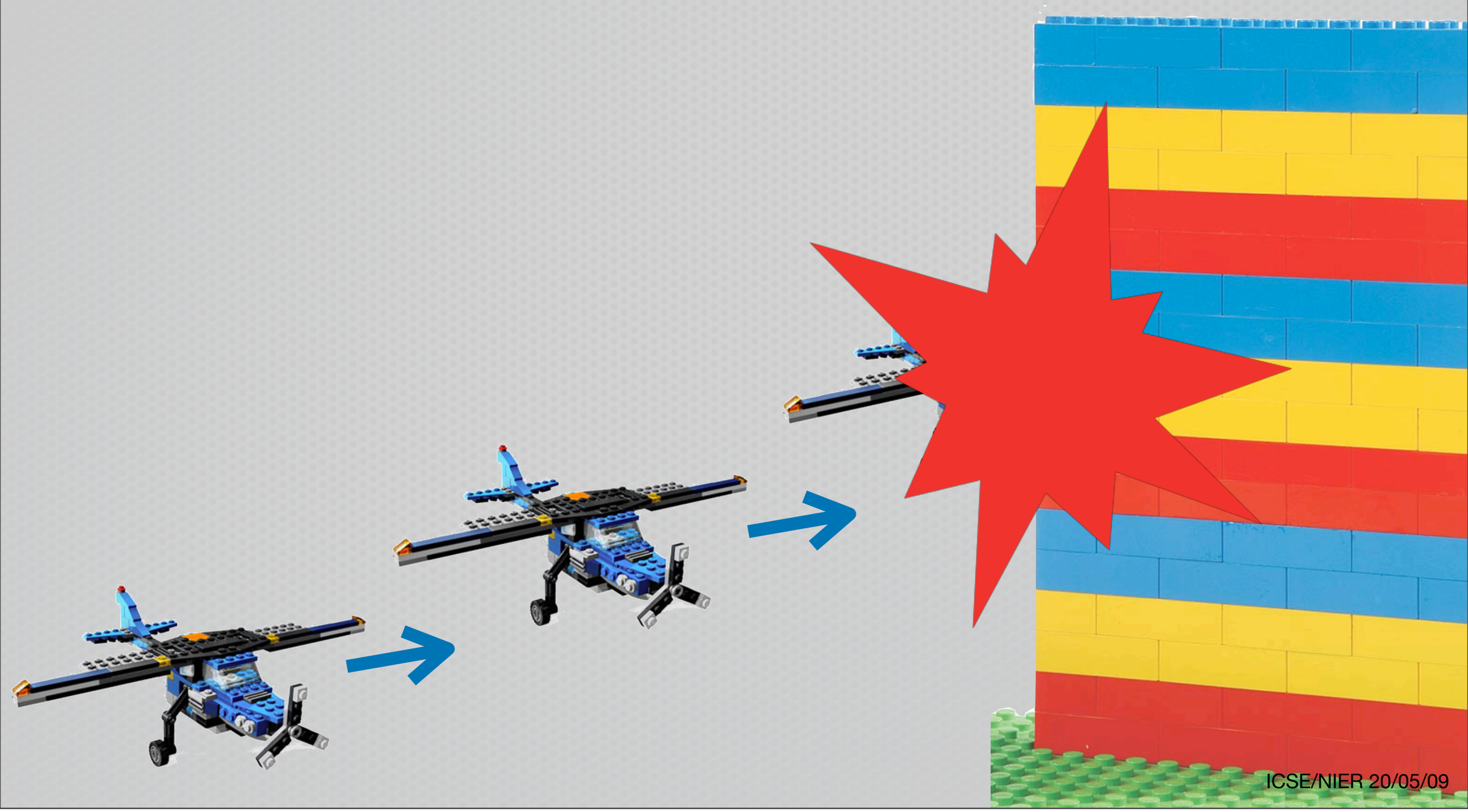
General model



Subject of verification: Behaviour



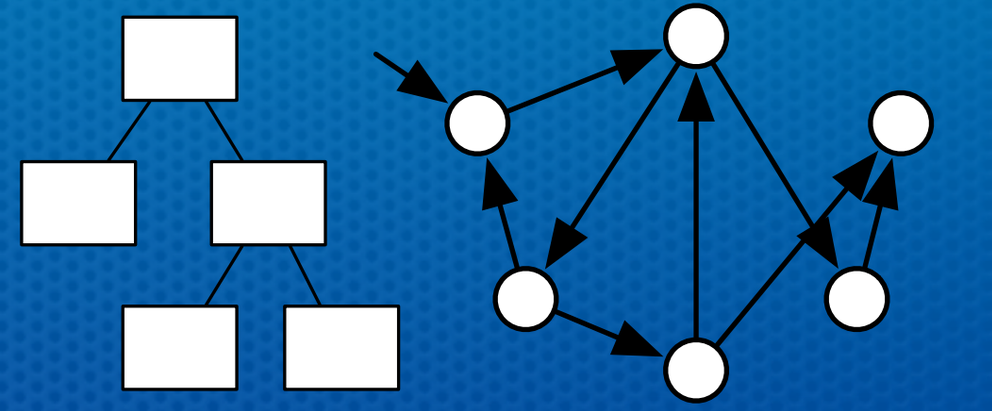
Subject of verification: Behaviour



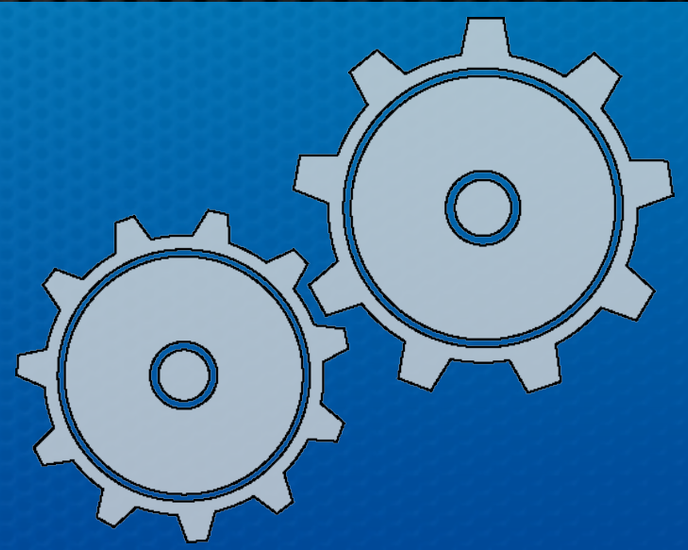
Challenges

Behaviour and variability

Scalable representation



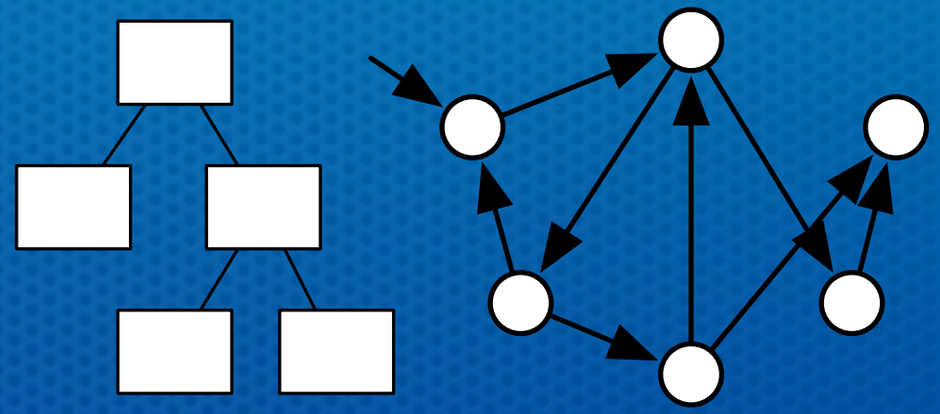
Efficient reasoning



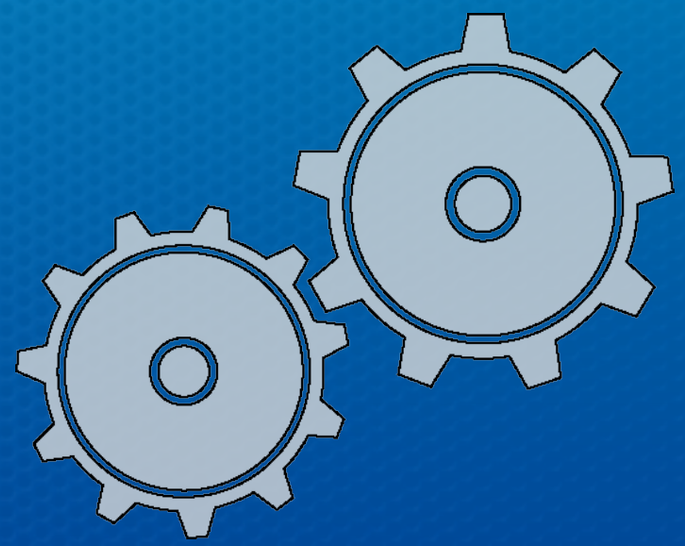
State of the art

Behaviour and variability

[Post2008] [Larsen2007]
[Fischbein2006]
[Prehofer2004]
[Sabetzadeh2007]



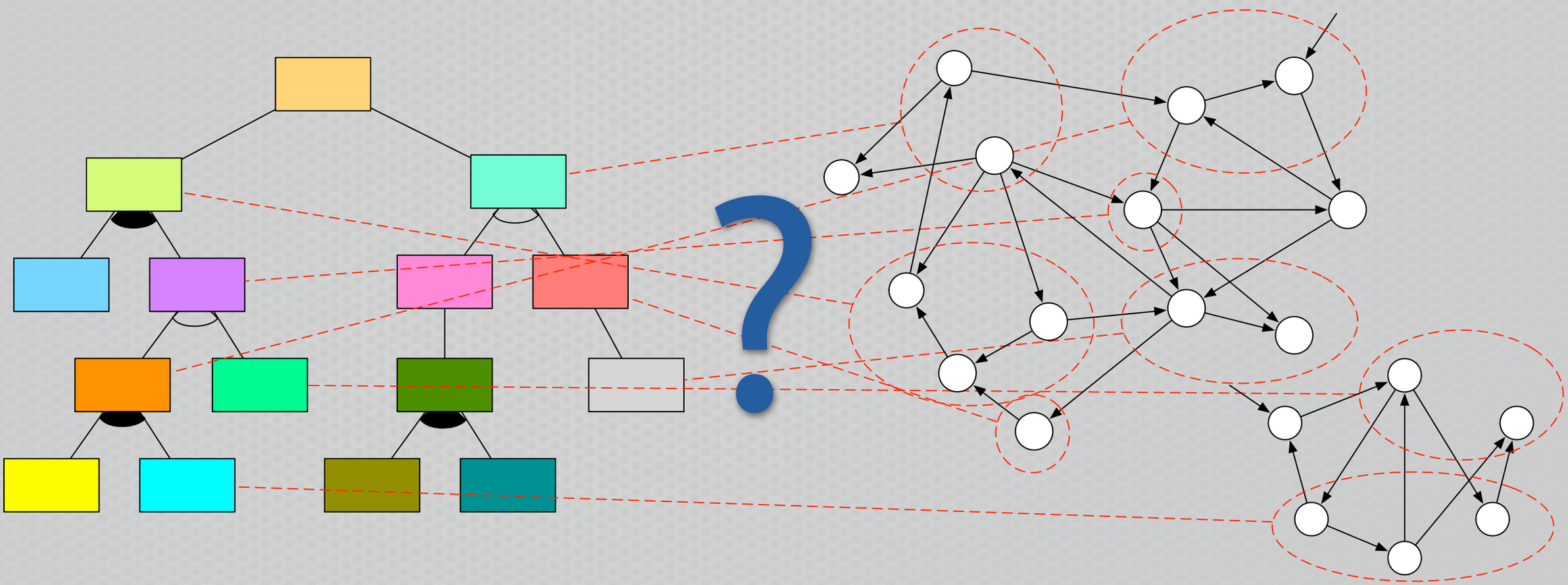
?



Research question 1

High-level family overview

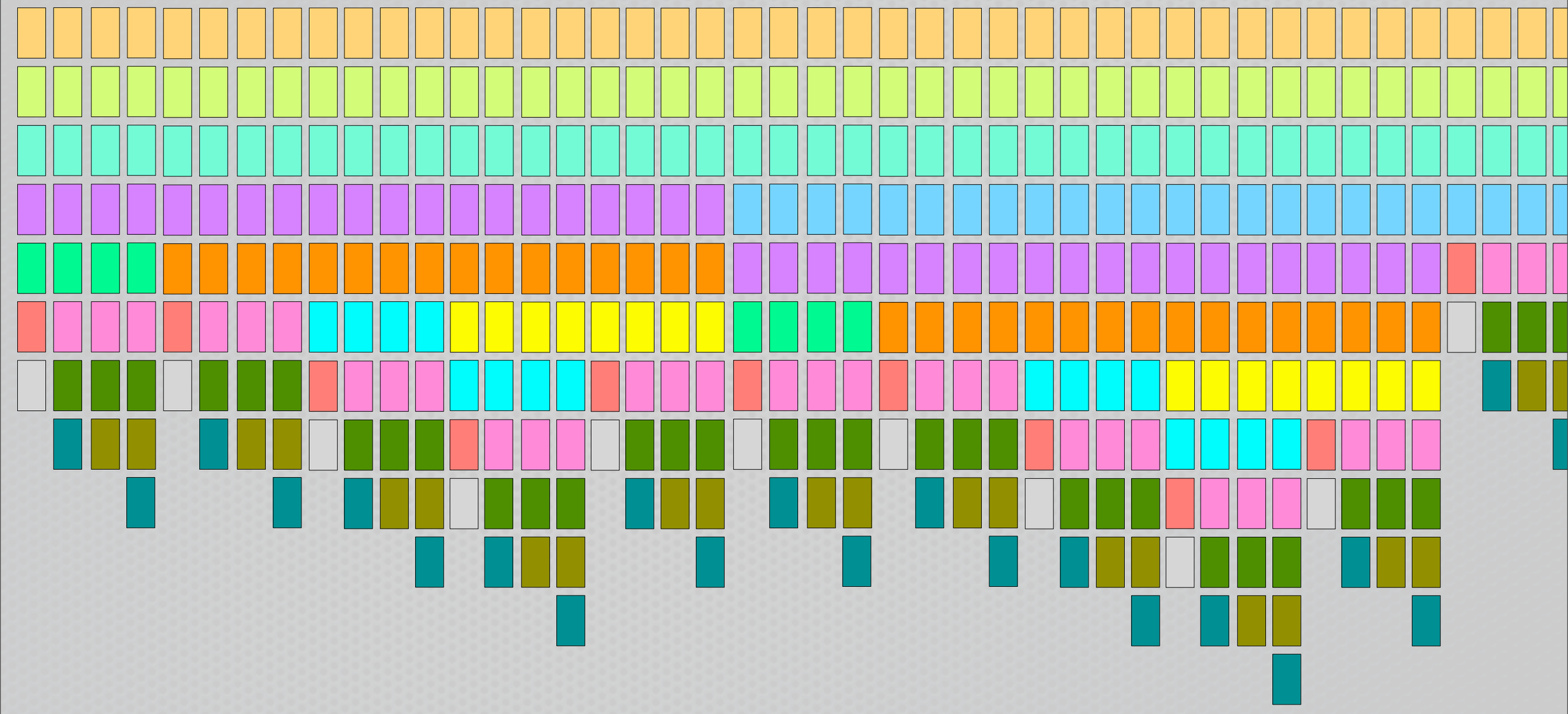
Behavioural model fragments



"How to map behavioural models to feature diagrams"

ICSE/NIER 20/05/09

Research question 2



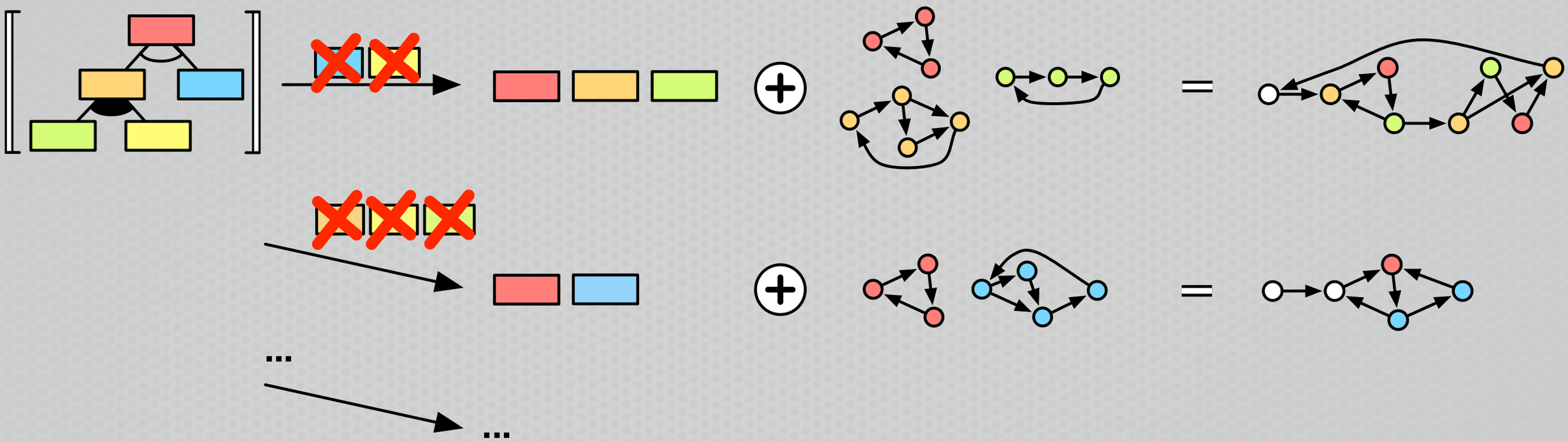
"How to efficiently model-check an exponential number of different products"

ICSE/NIER 20/05/09

“Sub”research questions

- How to represent feature behaviour
  - Which formalism?
  - Which format (single/transformation/..)?
- How to compose feature behaviour
  - How to combine single features?
  - How to encode variability in composition?

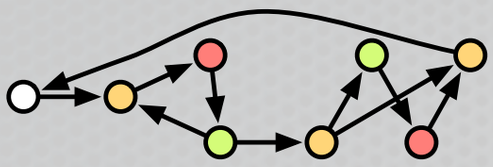
Option A: single merge



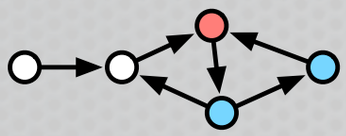
One merge/check per product

Option A: **single** merge, verification

=



=

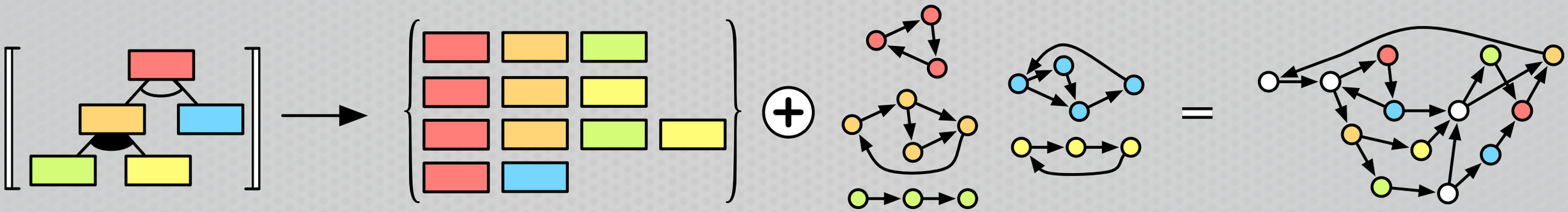


...

...

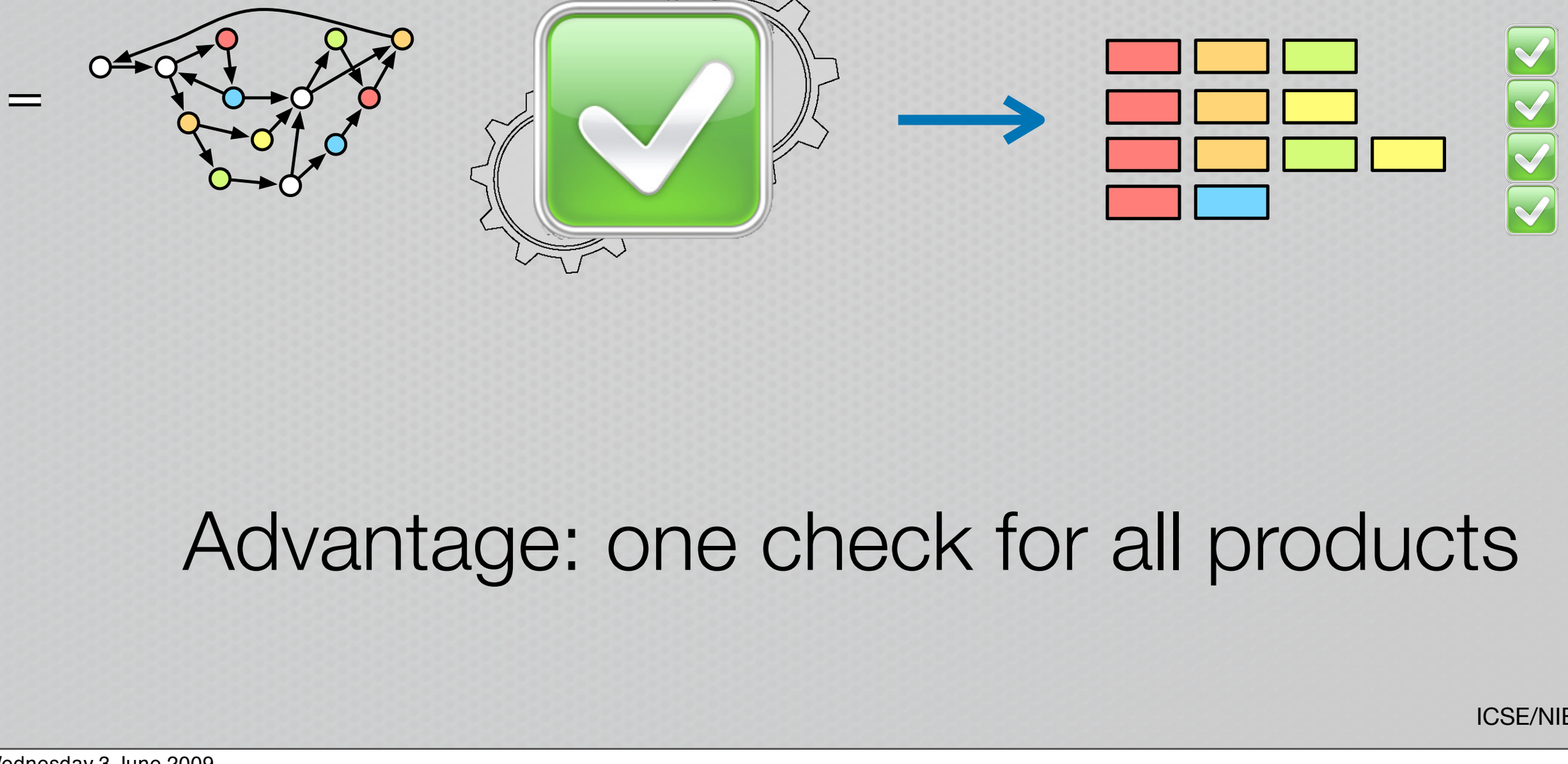
Problem: exponential number explicitly considered

Option B: **multi** merge



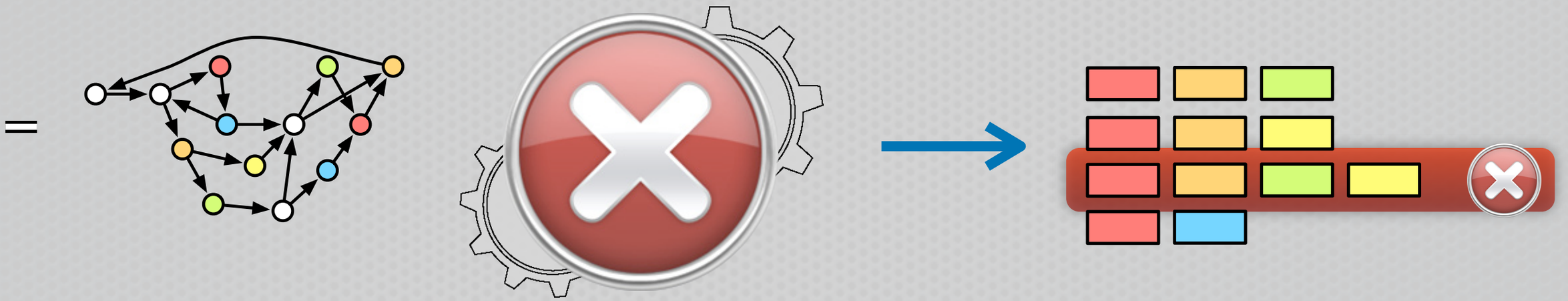
Idea: merge once and take variability into account

Option B: **multi** merge, verification



Advantage: one check for all products

Option B: **multi** merge, verification



Problem: verification not as fine-grained

Current progress

- Proposed multimerge approach
- Proposed enhanced transition systems for product family behaviour
- Algorithm for reachability
- Model-checking LTL safety properties

Ongoing work

- Implement a prototype
  - Symbolic verification
- Extend model-checking to omega-regular properties

Future work

- Define merge operations
- Investigate optimisation techniques

The End

Thank you!