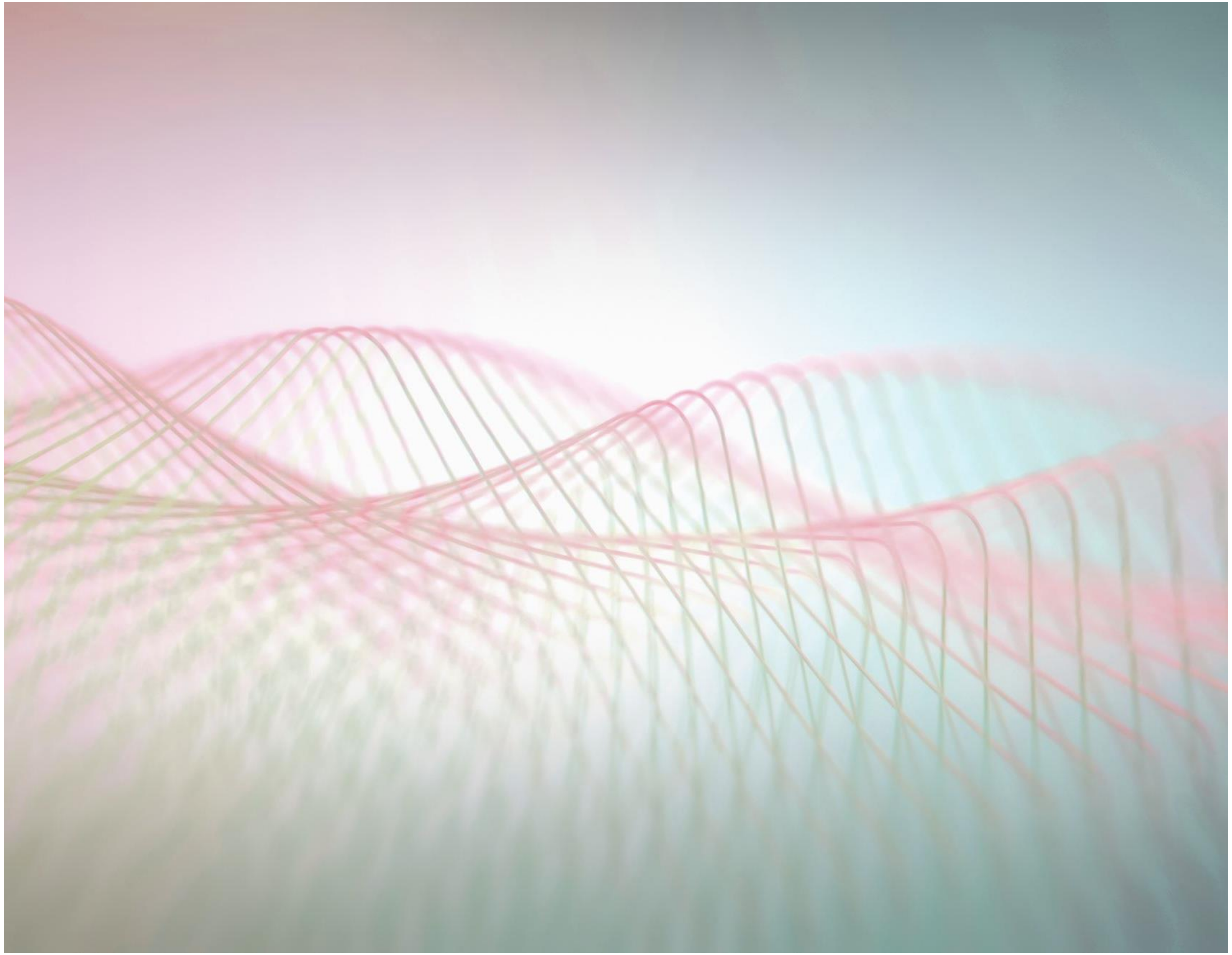
MODEL-BASED SOFTWARE DEVELOPMENT

LECTURE XIII.

OUTLOOK

Dr. Mezei Gergely



SIMULATION

SIMULATION

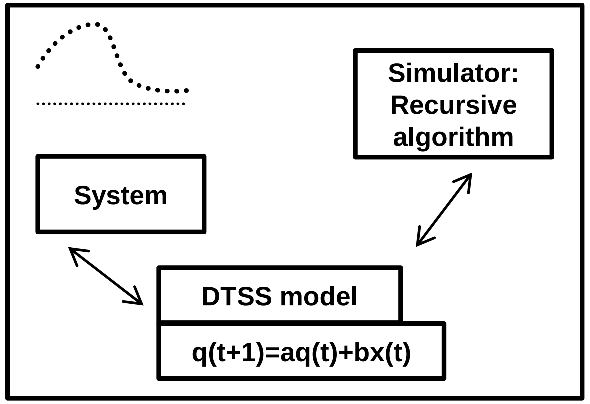
- To describe the dynamic operation of the model
- Basic question: does the system work according to the specification?
- Real systems are expensive to build and test, the models allow: running simulations without or before building the real system
- Proof (mathematical, automated, or other) that the system has the desired properties
- Purposes of use
  - Analysis: The system to be analyzed either exists or is being designed. We try to understand its behavioral characteristics
  - Conclusion: The system exists. We try to deduce its behavior from observations.
  - Design: The system is being designed, in the form in which it is modeled. We would like to make the best possible plan

SIMULATE, …

- …if the physical system is impossible to build...
  - Simulate a collision of galaxies
- … or costly...
  - Aircraft Airflow Simulation
- … or unobservable...
  - Prediction, e.g. virus spread
- … or measurement data-poor environment
  - E.g. adjusting traffic lights
  - It takes a long time to measure the throughput of each tried configuration

SIMULATION FORMALISMS

- DTSS: discrete time
- DESS: continuous time
- DEVS: event-based

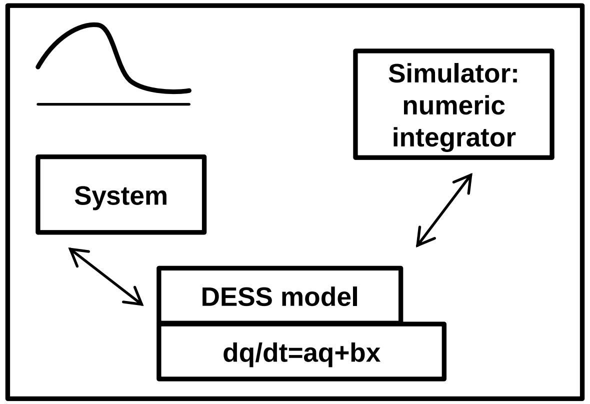


Simulator: Recursive algorithm

System

DTSS model

q(t+1)=aq(t)+bx(t)

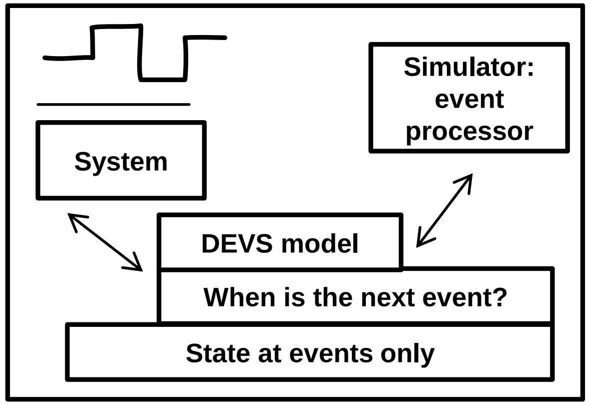


Simulator: numeric integrator

System

DESS model

dq/dt=aq+bx



Simulator: event processor

System

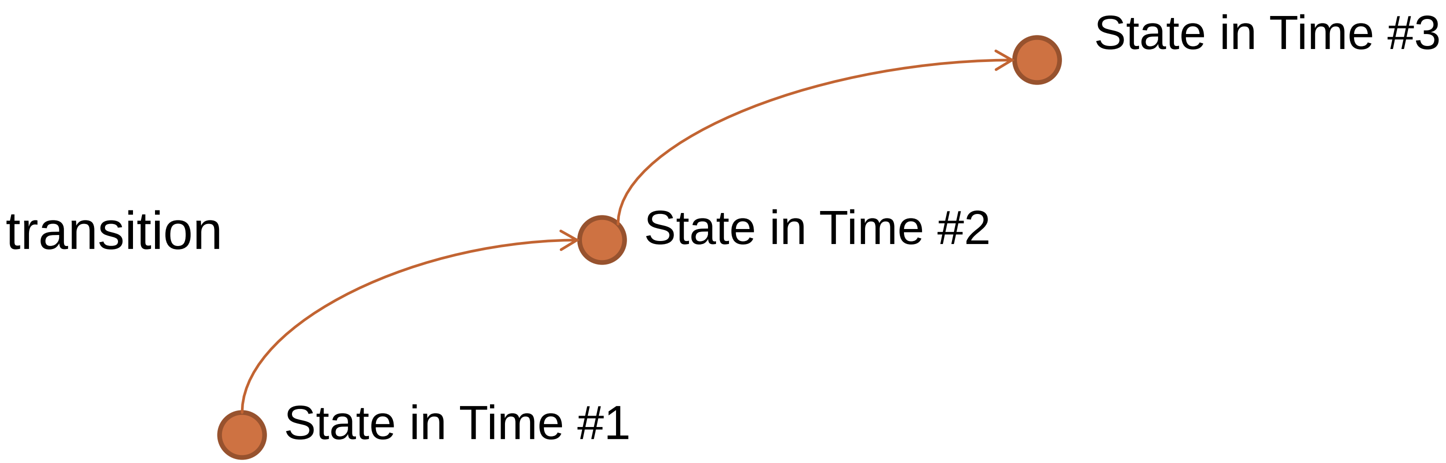
DEVS model

When is the next event?

State at events only

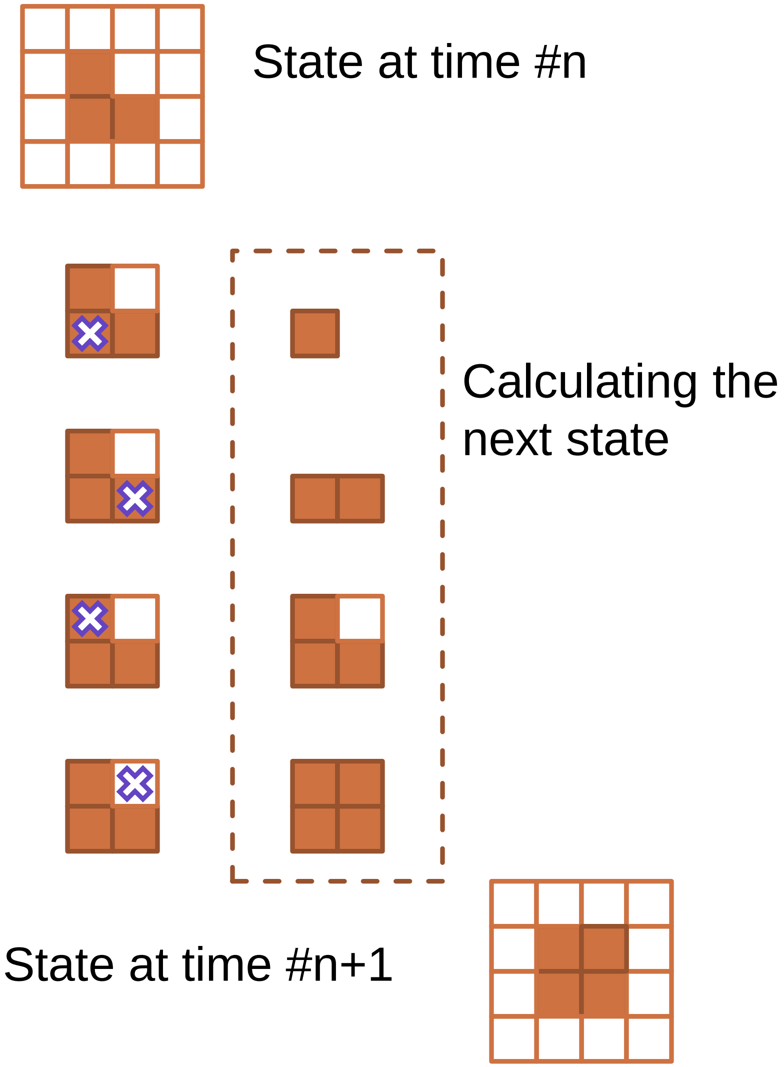
DISCRETE-TIME SYSTEM SPECIFICATION (DTSS)

- Stepwise execution
  - Digital clock, time units
  - Works with any time unit (1 sec, 1 min, …)
- State cannot change during a time unit



EXAMPLE: CELLULAR AUTOMATON (GAME OF LIFE)

- Two dimensional grid (discrete space)
  - All grid cells represent a cell
  - The state of the cell is influenced by its neighbors (diagonal neighbors also count)
  - State examination at time steps
- Rules
  - A cell stays alive if it has 2 or 3 living neighbors
  - If the cell has more than 3 or less then 2 living neighbors, it dies
  - A dead cell come alive if it has exactly 3 living neighbors



State at time #n

Calculating the next state

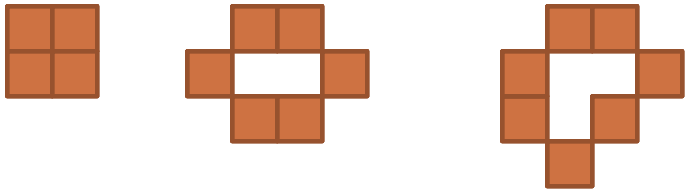
State at time #n+1

BEHAVIORS

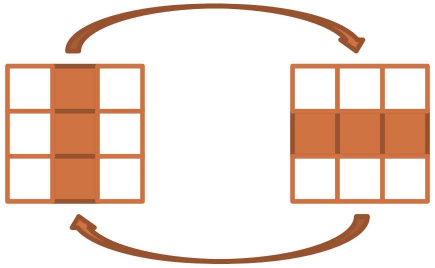
- Behaviors
  1. The pattern dies out quickly
  2. The pattern becomes periodic
  3. Chaotic behavior
  4. Unpredictable, not periodic, but shows regular patterns

EXAMPLE PATTERNS

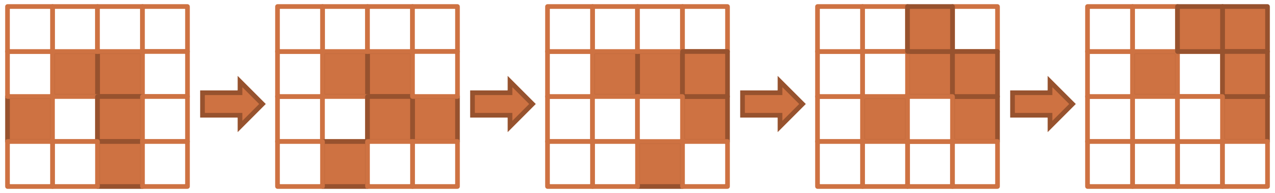
- Stable patterns



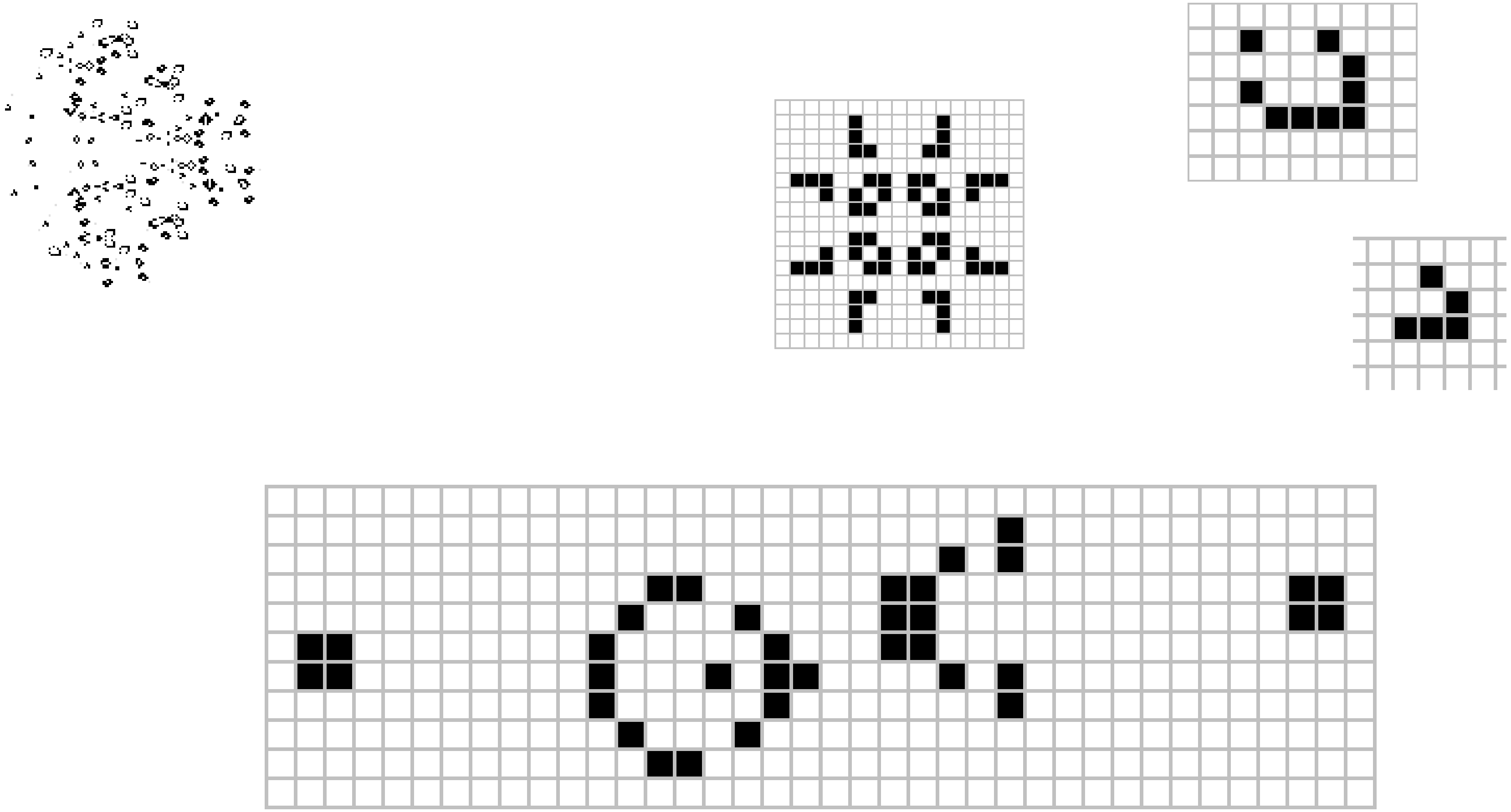
- Oscillating pattern:



- Moving pattern:

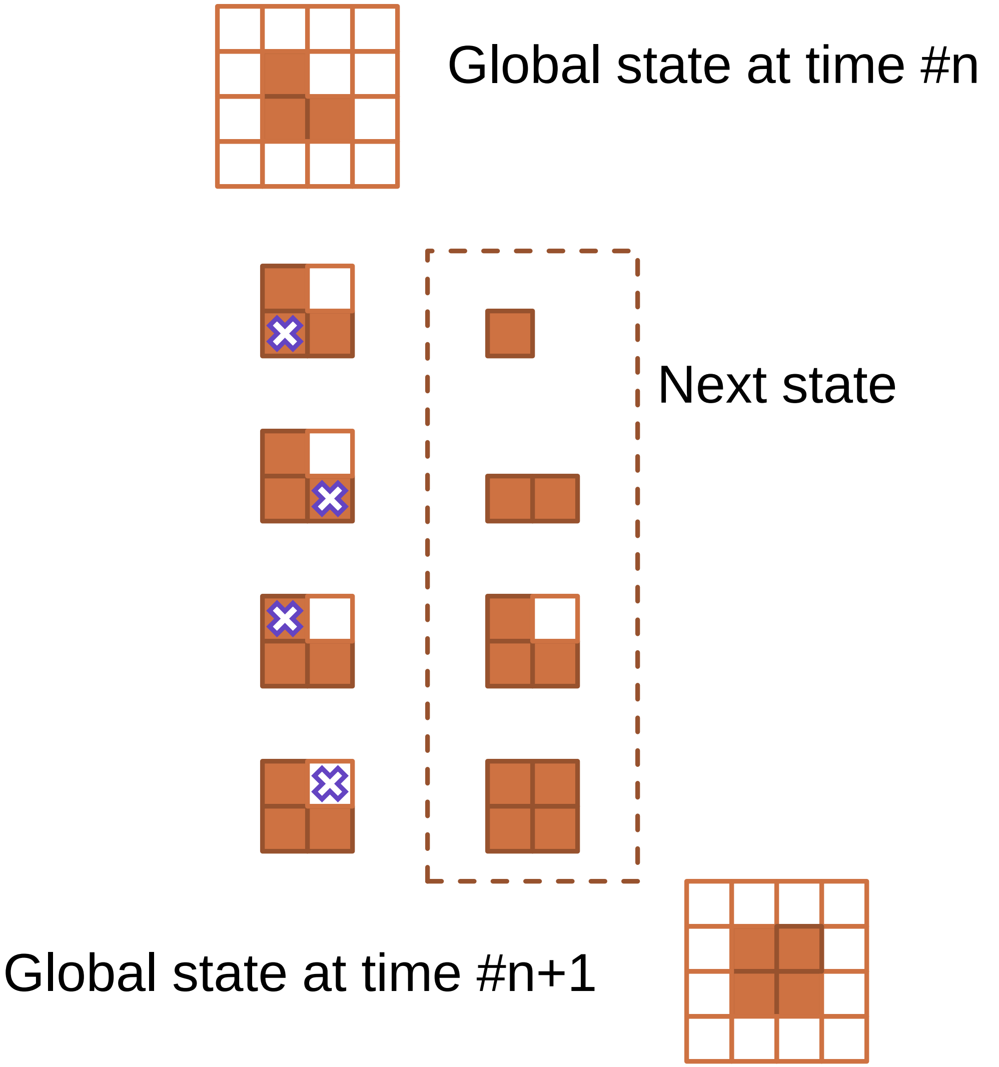


FURTHER EXAMPLES



SIMULATION MECHANISM

1. We examine all cells in all steps
2. We apply the state transition rules for each of them at the same time
3. We save the next state of the cells
4. We update the global state and increment the timer



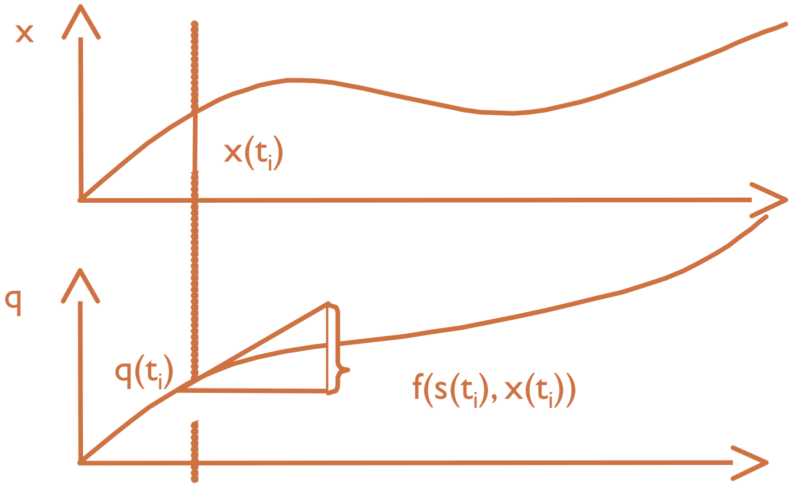
DTSS – STATE CHANGES

- Transition function: $\delta(q, x)$
- Output function: $\lambda(q, x)$

- State trajectory: $q(t+1) = \delta(q(t), x(t))$
- Output trajectory: $y(t) = \lambda(q(t), x(t))$

DIFFERENTIAL EQUATION SYSTEM SPECIFICATION (DESS)

- The time is not discrete, but continous
- We use derivation function instead of a state transition function
  - Instead of the "next state" we focus on the amount of change



DESS – FORMALISM

- Input: dq(t)/dt = x(t)
- Output: y(t) = q(t) (current state)
- We use first-order differential equation for more than one variable:

$$
\begin{gathered}
d\, q_1(t)/dt = f_1(q_1(t), q_2(t), \ldots, q_n(t), x_1(t), x_2(t), \ldots, x_m(t)) \\
d\, q_2(t)/dt = f_2(q_1(t), q_2(t), \ldots, q_n(t), x_1(t), x_2(t), \ldots, x_m(t)) \\
\ldots \\
d\, q_n(t)/dt = f_n(q_1(t), q_2(t), \ldots, q_n(t), x_1(t), x_2(t), \ldots, x_m(t))
\end{gathered}
$$

- Challenge:
  - In a given t_i time, we know only dq_i/dt (steepness)
  - Next point is a t_{i+1} time and an interval $[t_i, t_{i+1}]$ associated
  - *The value of* t_{i+1} must be calculated without knowing the elements of $[t_i, t_{i+1}]$ (only t_i is known)
- We use numerical integration methods

MANUFACTORING

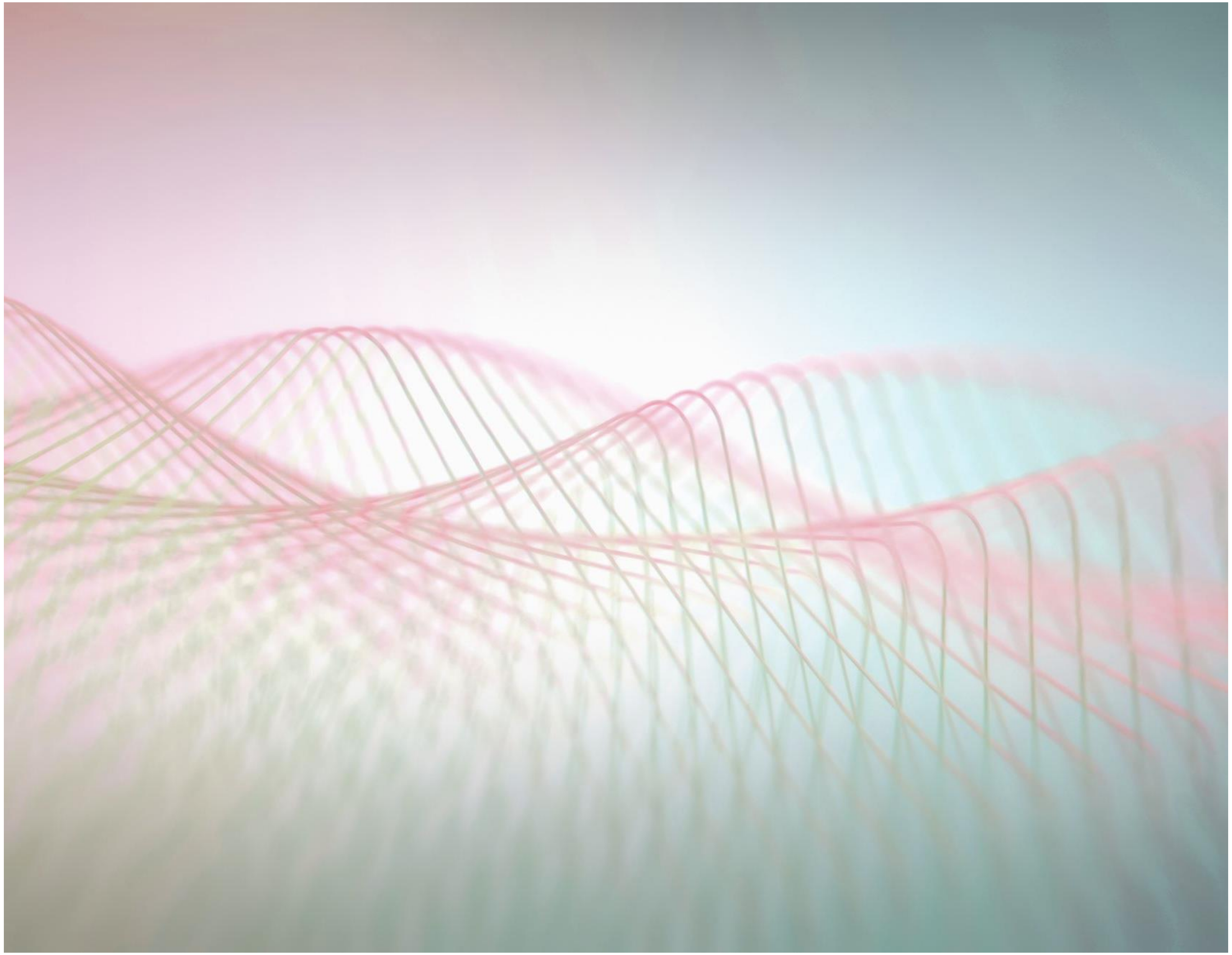
- Tasks
  - Order materials
  - Store materials
  - Manufacture product
  - Quality management
- The process is event-driven
  - Order if the stock is almost empty
  - Quality insurance after manufacture
  - …
- DTSS/DESS
  - Can be used, but not efficient
  - *Event-based timing would be better*

CELLULAR AUTOMATON - ISSUES

- State transition is calculated and applied on all components
  - Most of the cells do not change at all
  - E.g. dead regions will not raise from dead
- Idea: liveness state changes can be handled as events
  - How can we find cells to observe?
  - Basic idea
    - State transition marks cells which change their states
    - Only the neighbor cells of the marked ones are to be observed

DEVS

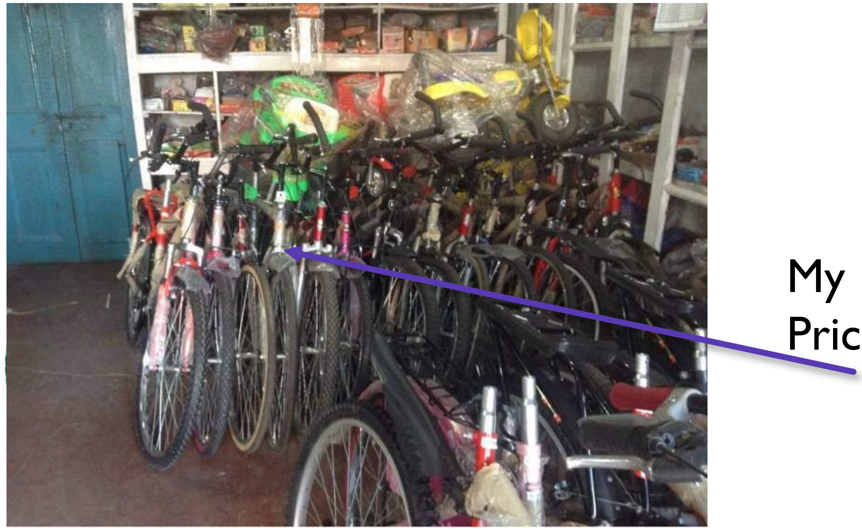
- Discrete EVent System
  - The system is built up from components, which may interact each other
  - The components interact each other via messages referred to as events
  - The system can have different states, which are discrete
  - The state changes are always triggered by events



MULTI-LAYER DYNAMIC METAMODELLING

MULTILAYER METAMODELLING

- The evolution of concepts
  - At the beginning, we have *something…*
  - … refined through several steps….
  - … leading to concrete products



My Black Thunder
Price: 260.000Ft

MULTILAYER METAMODELING

- Modeling the manufacturing workflow
  - Based on prototypes, using many intermediate steps
  - We refine concepts step by step
    - Abstract and concrete components co-exists and my refer to each other
  - “Living”, evolving domain specific languages
    - Agile-style methodology
    - Branching along refinements instead of versioning
  - Validated refinement

DMLA BOOTSTRAP

- Bootstrap – core elements
  - Classifier (class)
  - Slot (field)
  - Operation (method)
  - Annotation (annotation, constraint)
  - Contract (interface)

DMLA - DYNAMISM

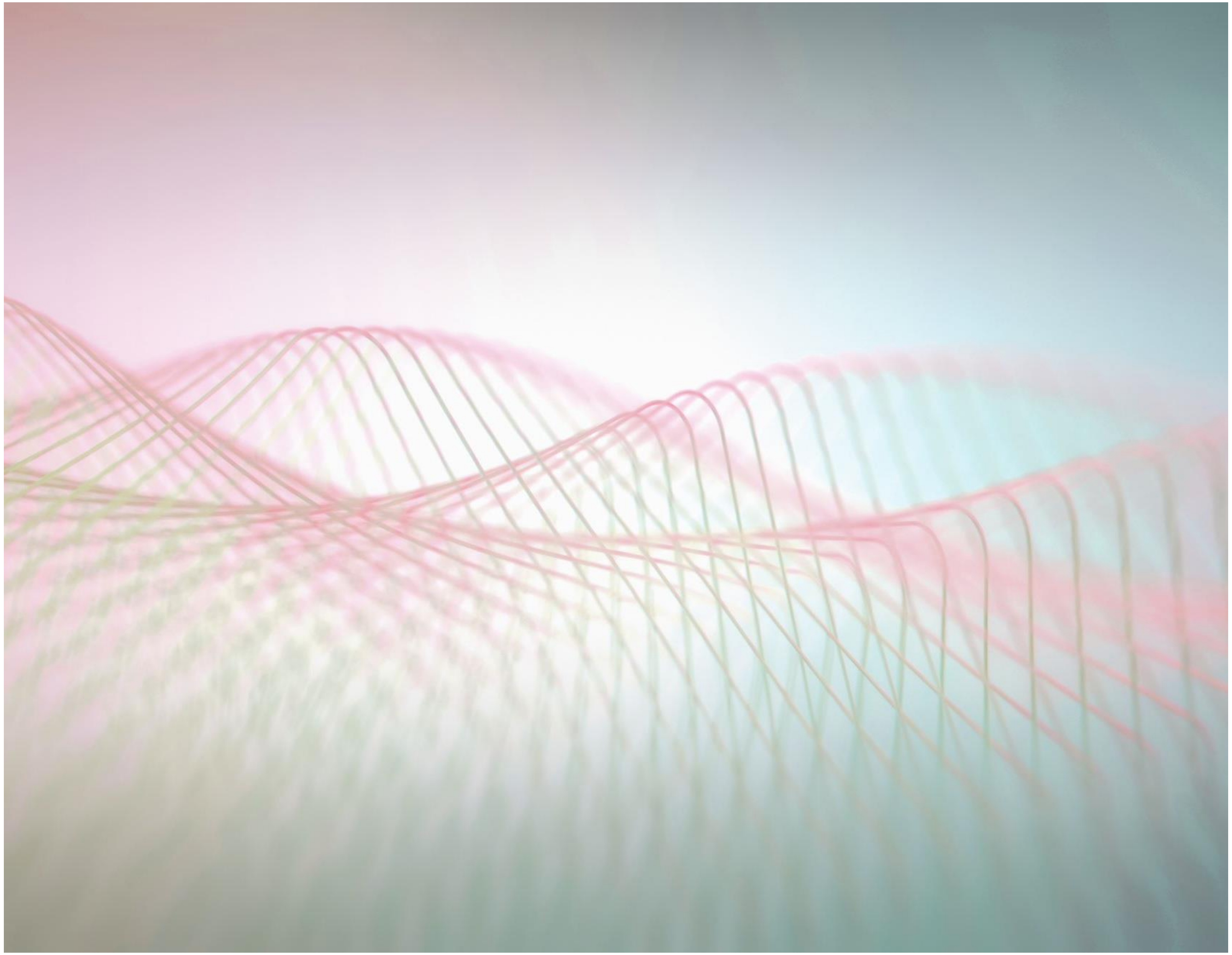
- Customizable type constraint
  - Type-, Cardinality constraint (annotation)
- Customizable validation
  - Refinement rules are not wired
  - Key: operation = AST built from entities
    - Operation is data → can be changed
- Customizable behavior
  - Operation that rewrited another operation
  - (Self-healing code?)

DMLA

- GraalVM + Truffle
  - Interpreter, not compiler (dynamism)
  - Operation ⇔ AST
- Custom, hand written parser, CAAS
- LSP

DMLA - REFINEMENT

- Stepwise refinement
  - Refinement chain for Entities, Slots and Annotations
  - Slot division
- Model is not a snapshot
  - Model contains all design decisions, branches
  - Step back is always possible



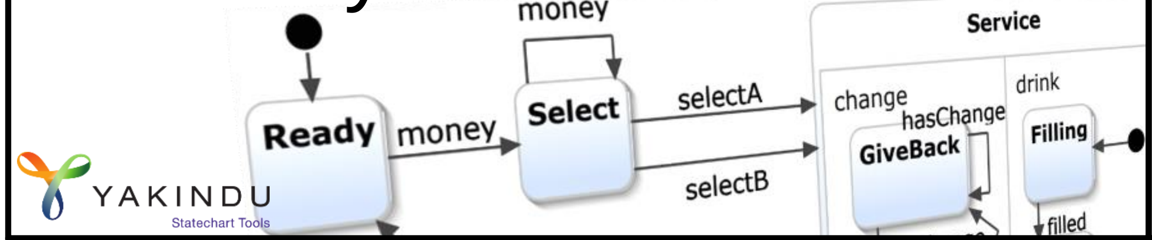
Automated Model Generation Techniques

Chalenges of **CoReDiSc**

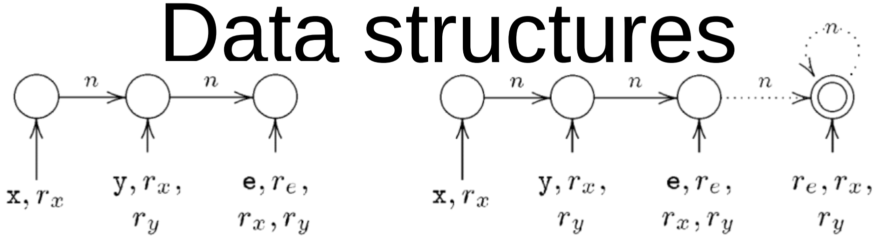
Modeling with Graphs

- Graphs are widely used in software engineering

System models

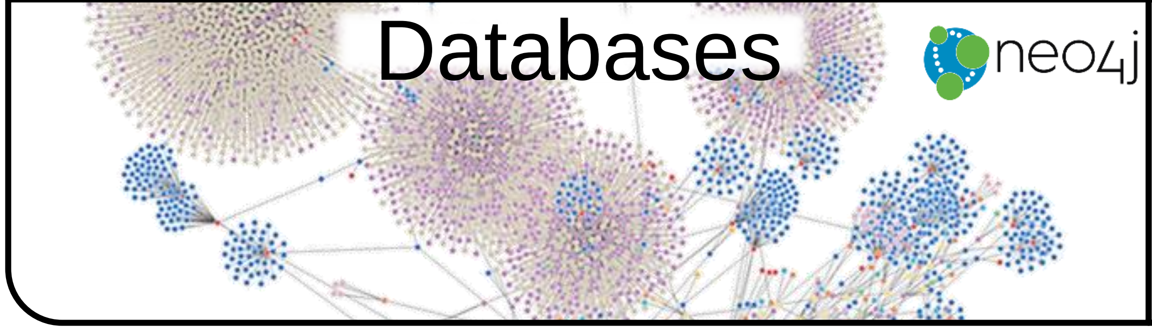


Data structures

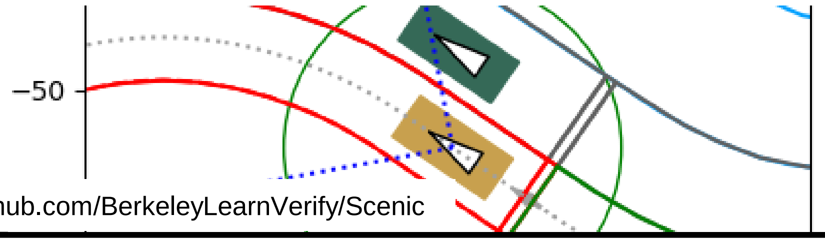


Sagiv, M., Reps, T., & Wilhelm, R. (2002). Parametric shape analysis via 3-valued logic. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, *24*(3), 217-298.

Databases



Scenarios



https://github.com/BerkeleyLearnVerify/Scenic

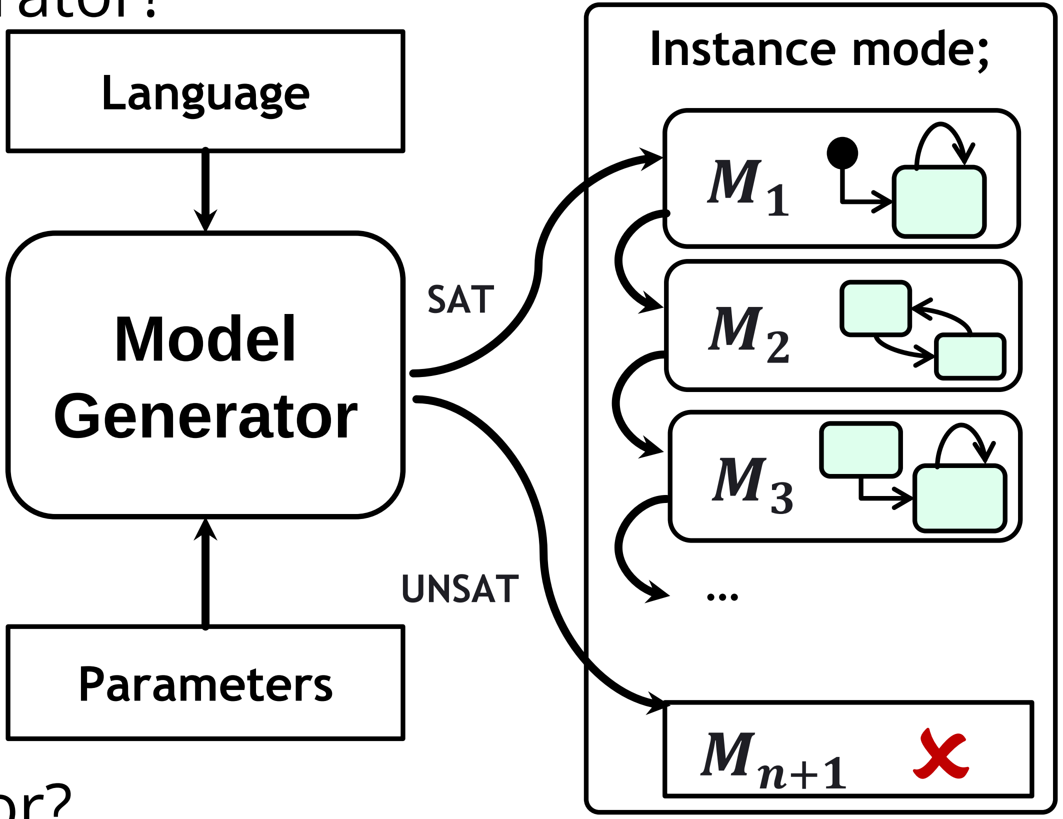
- Testing, benchmarking, generative architecture → graphs

Goal: (Consistent | Realistic | Diverse | Scalable) generation!

Arcitecture of generators

- What is the architecture of an ideal generator?

1. Input: Language specification
   We are generating the instances of this language
2. Input: Parameters
   We can characterize the wanted models
3. Output: Models
   Sequence of models
4. Output: Inconsistency
   If there is no such model, we should prove it



Language

Model Generator

Parameters

SAT

UNSAT

Instance mode;

M_1

M_2

M_3

...

M_{n+1}

- What are the properties of such generator?

Consistency

- A generator is consistent, if it can generate models that satisfies all well-formedness constraints.
- A generator is complete, if it can generate all valid models.

- Well-formedness constraints are complex logic predicates (e.g. OCL or graph patterns)

- We need to use logic solvers

CO

Consistent

Realistic

- A model generator is realistic, if the generated models are indistinguishable from real ones.

CO

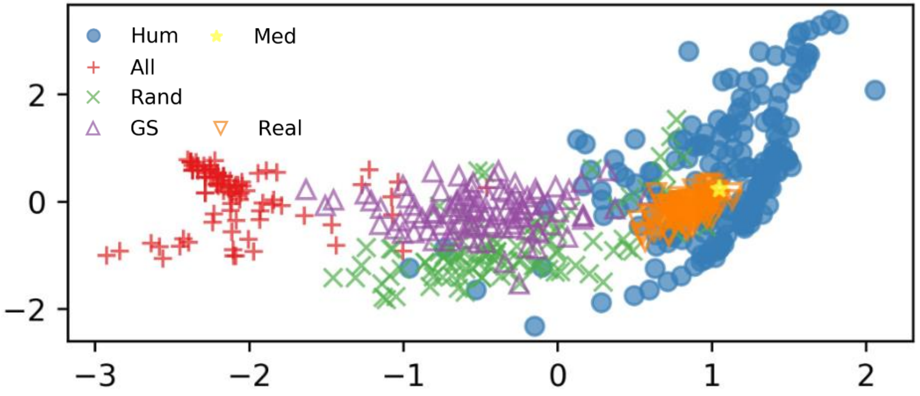
Consistent

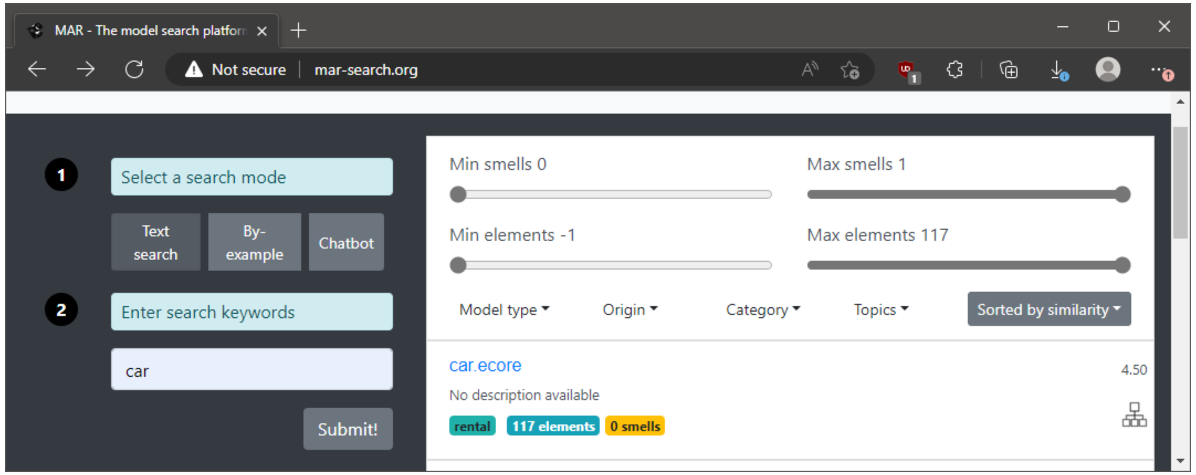
RE

Realistic

Realistic

- How to measure ther ealistic nature
  - Graph Solver: network science (degree distribution), during generation, we measure and control
  - AI-based solutions
- How to get real models?
  - IP issues, model repositories are important (http://mar-search.org/)





1 Select a search mode

Text search | By-example | Chatbot

2 Enter search keywords

car

Submit!

Min smells 0

Max smells 1

Min elements -1

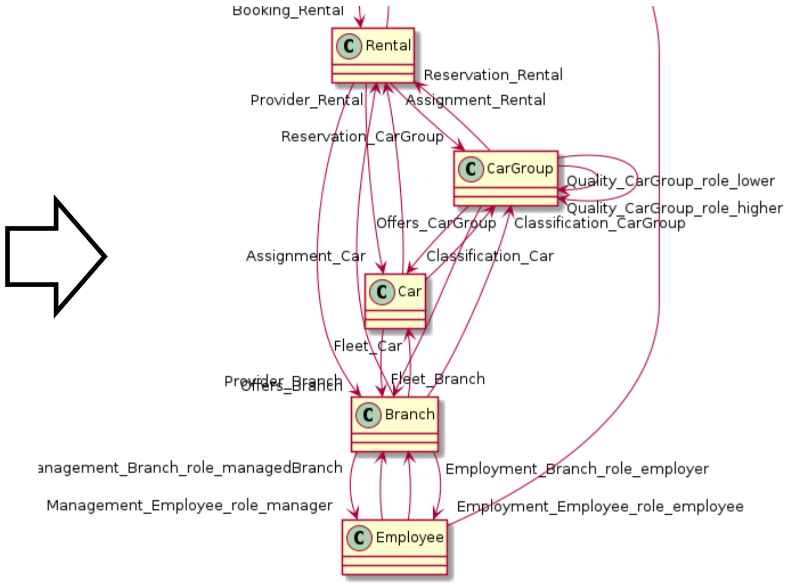
Max elements 117

Model type ▾ Origin ▾ Category ▾ Topics ▾ Sorted by similarity ▾

car.ecore 4.50

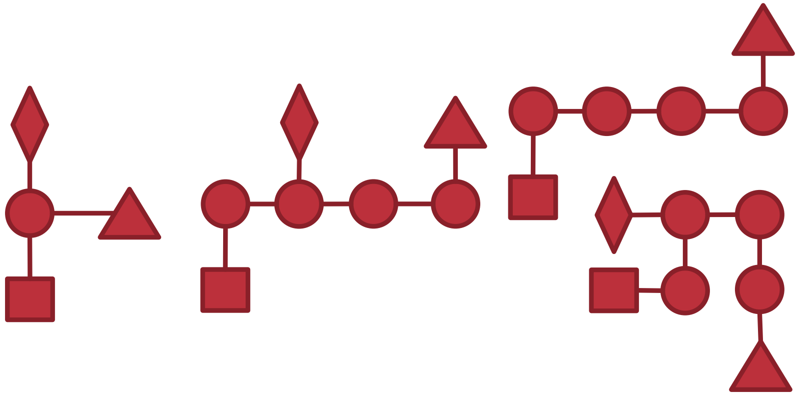
No description available

rental 117 elements 0 smells



Diversity

- Models are not symmetric
- Generated models are different from each other.
- How to measure diversity?
  - E.g., count the different kind of subgraphs



  - The more the better

CO

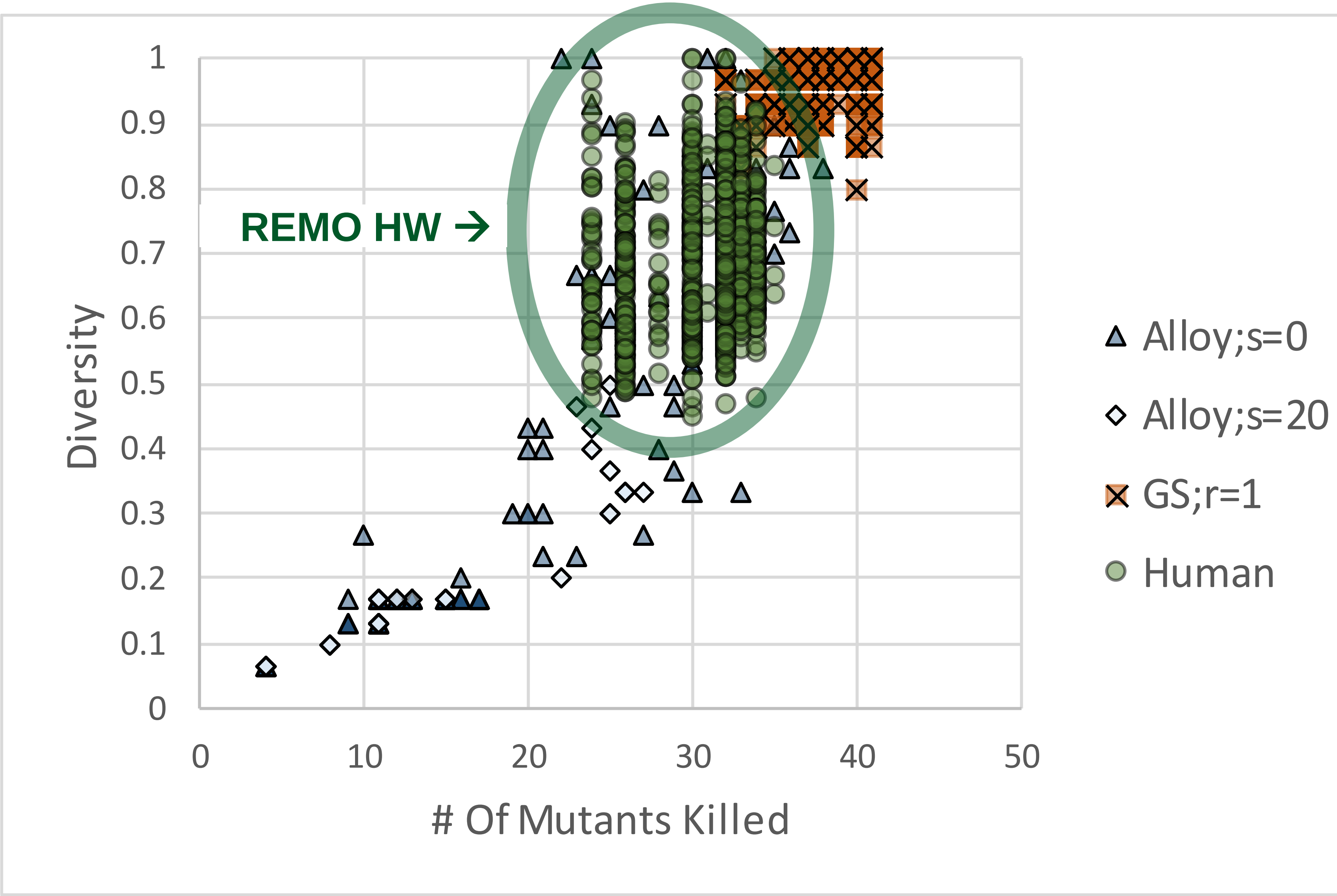
Consistent

RE

Realistic

DI

Diverse



How to compare
diversity
test coverage?

YAKINDU
Statechart Tools

Used different solvers, compared the result

- **Takeaway 1:**
  SAT < Human < Graph Solver
- **Takeaway 2:** Correlation

Scalability

- A generator is scalable, if it is able
  - To generate large models
  - To generate a lot of models

- Co/Re/Di is hard
  - Scalability issues (Sc--)
  - Or reduce others (Co-- Re-- Di--)

CO

Consistent

RE

Realistic

DI

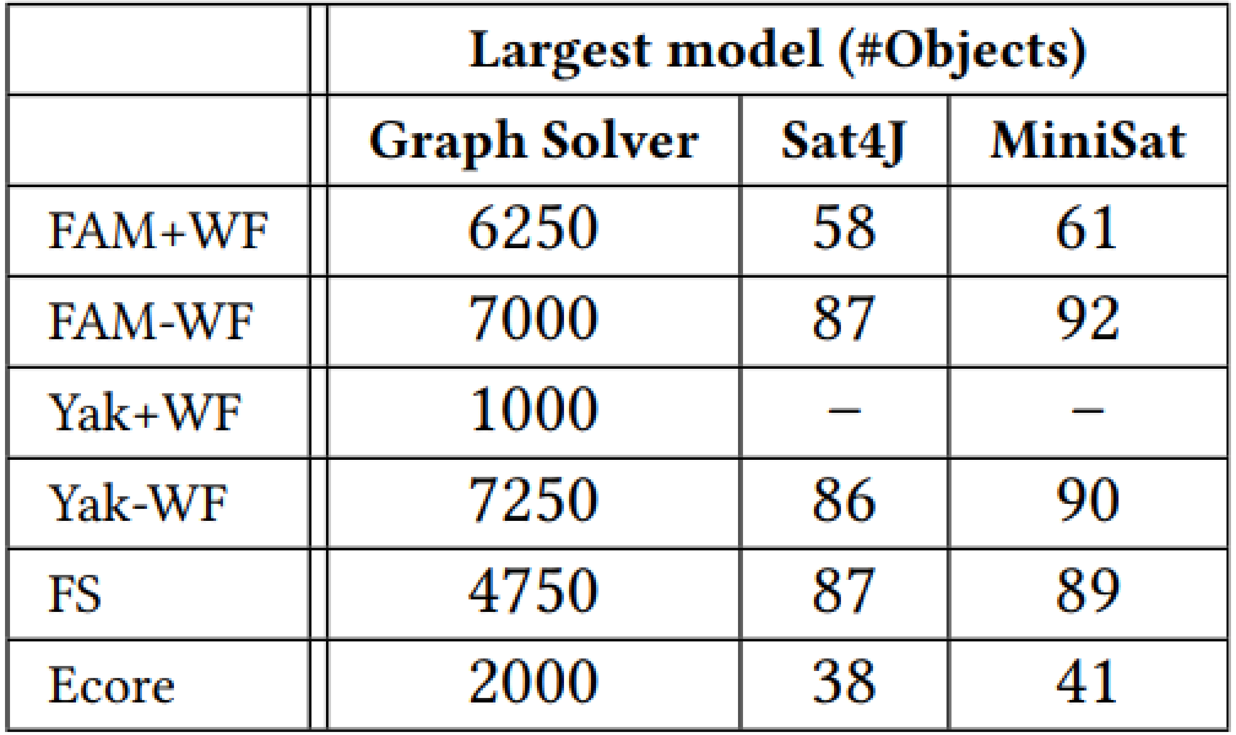
Diverse

SC

Scalable

Some measurements for graph solvers

Max size



	Largest model (#Objects)		
	Graph Solver	Sat4J	MiniSat
FAM+WF	6250	58	61
FAM-WF	7000	87	92
Yak+WF	1000	–	–
Yak-WF	7250	86	90
FS	4750	87	89
Ecore	2000	38	41

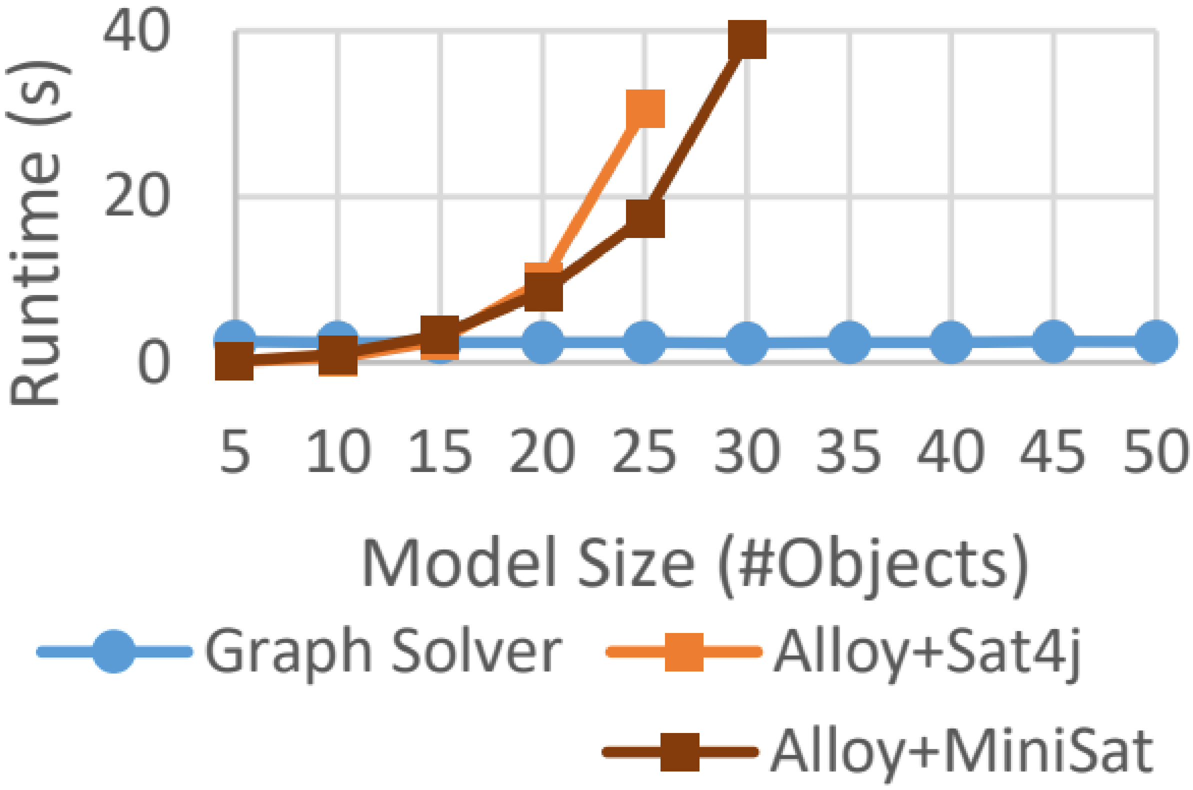
FAM: Industrial, Avionics

FS: File System example of

Yakindu: Industrial, Statemachine

Ecore: Metamodelling language

Runtime comparison

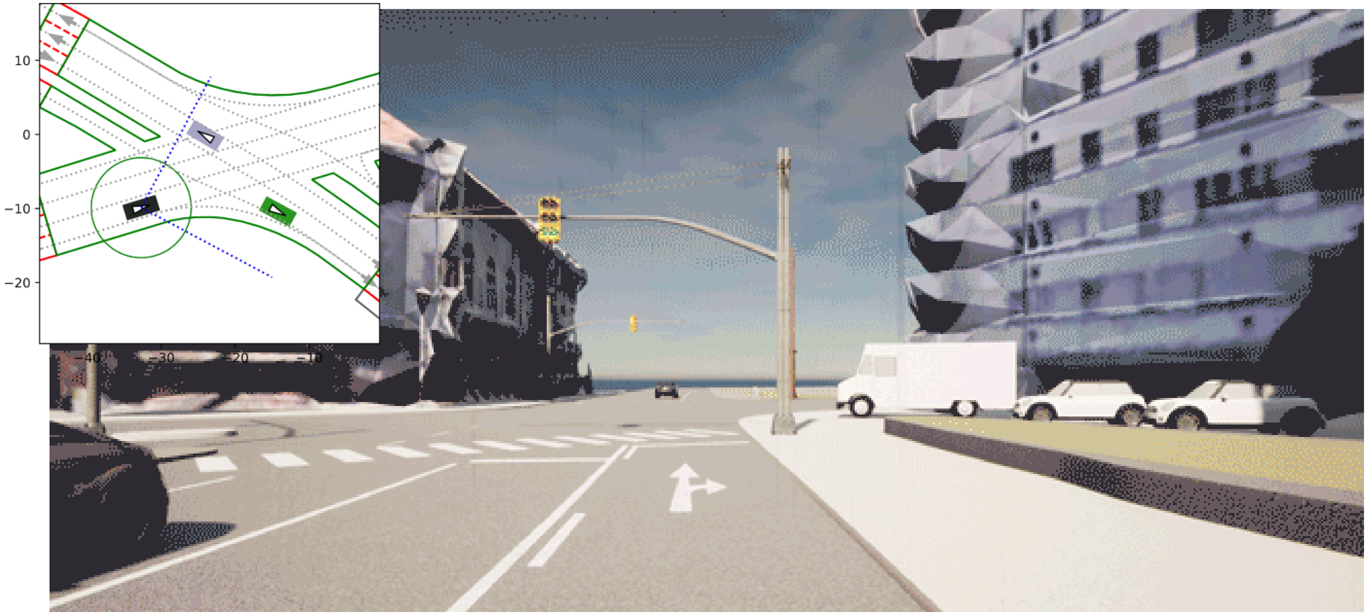


(Consistent | Realistic | Diverse | Scalable) generation!

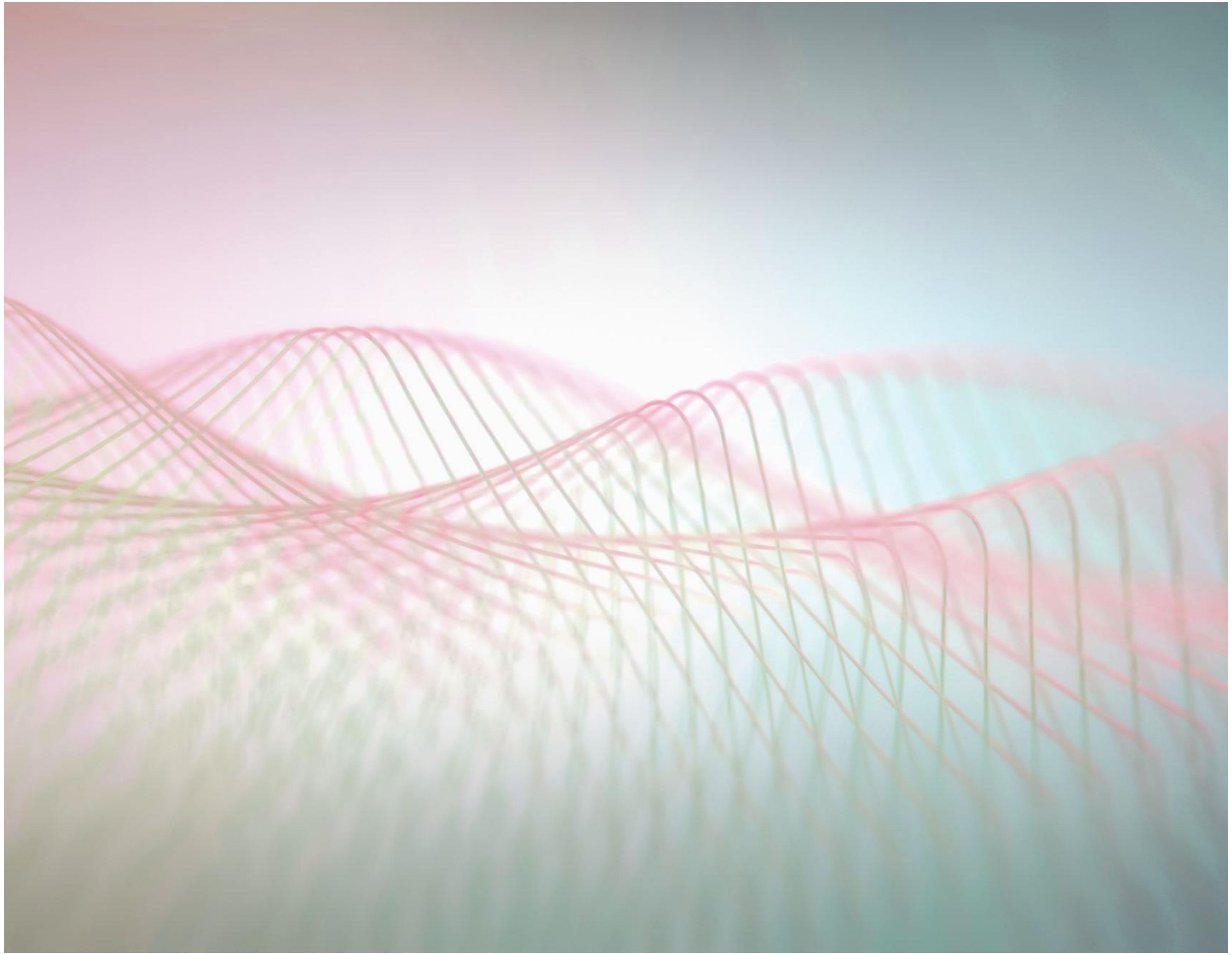
- What combination is needed for different tasks?
  - Test generation
  - Performance evaluation
  - Representative examples
  - Fair homework
  - Cheat detection
  - Optimization

Summary

- Model generation is hard
- Different properties: (Consistent | Realistic | Diverse | Scalable)
- We are developing one!
  - Homeworks
  - Railway
  - Avionic architecture
  - Testing self-driving vehicles
    - ESA
    - NAVY



- Video: https://youtu.be/fUopeDFIUKA
- Open source tool: https://github.com/graphs4value/refinery

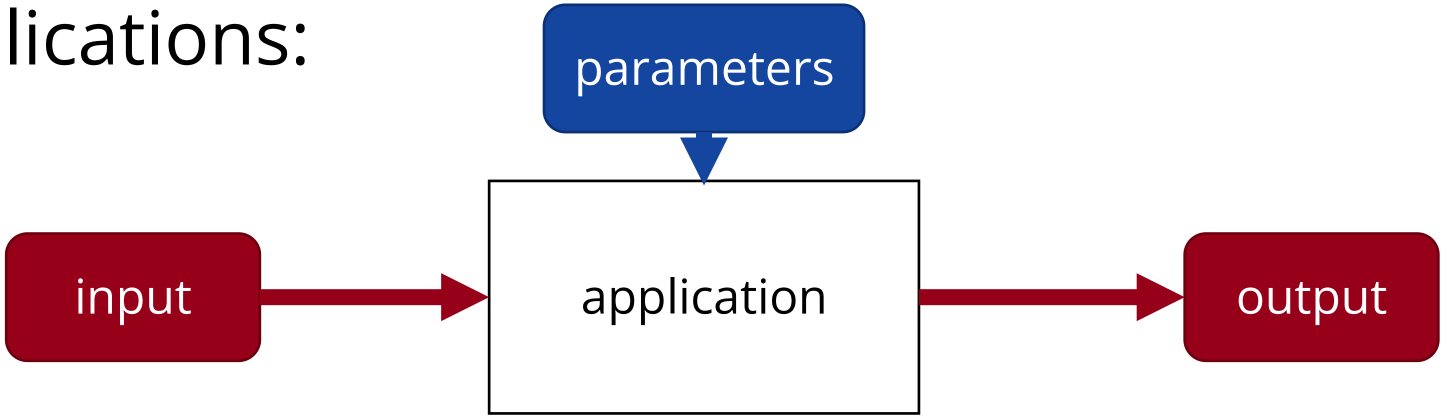


Models and AI

Neuro-Symbolic Reasoning

AI Applications

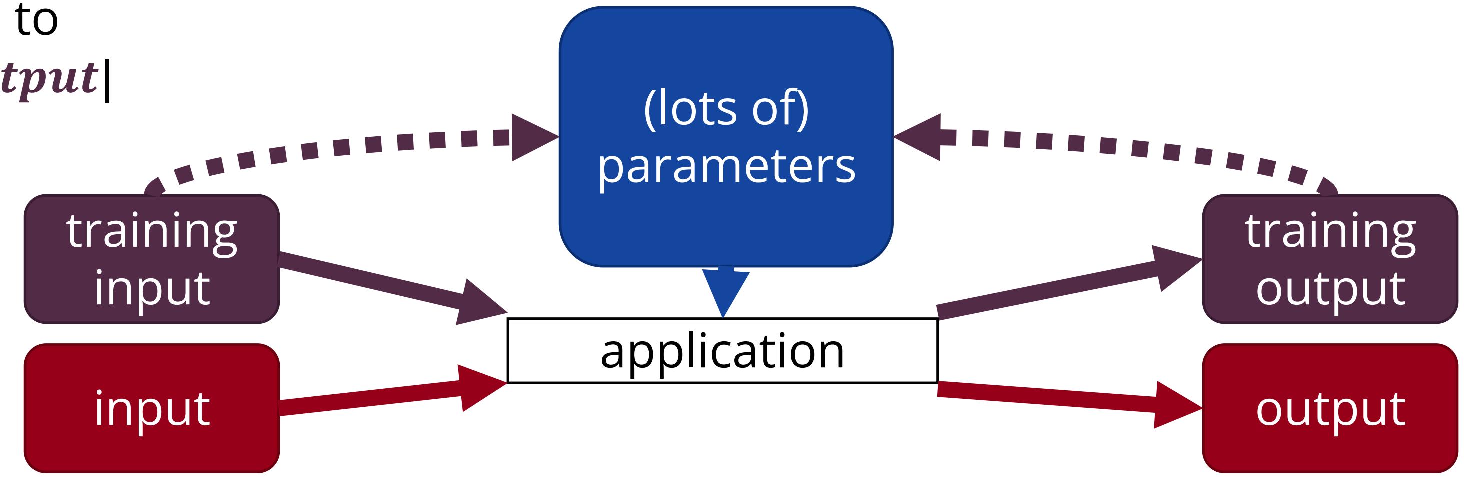
- Normal applications:



- AI apps: requirements + manual work → data + power

- calculate ***parameters*** to
- min $|\boldsymbol{app}(\boldsymbol{input}) - \boldsymbol{output}|$
- wrt. a cost function

If training, validation, and runtime data is similar, it should work.



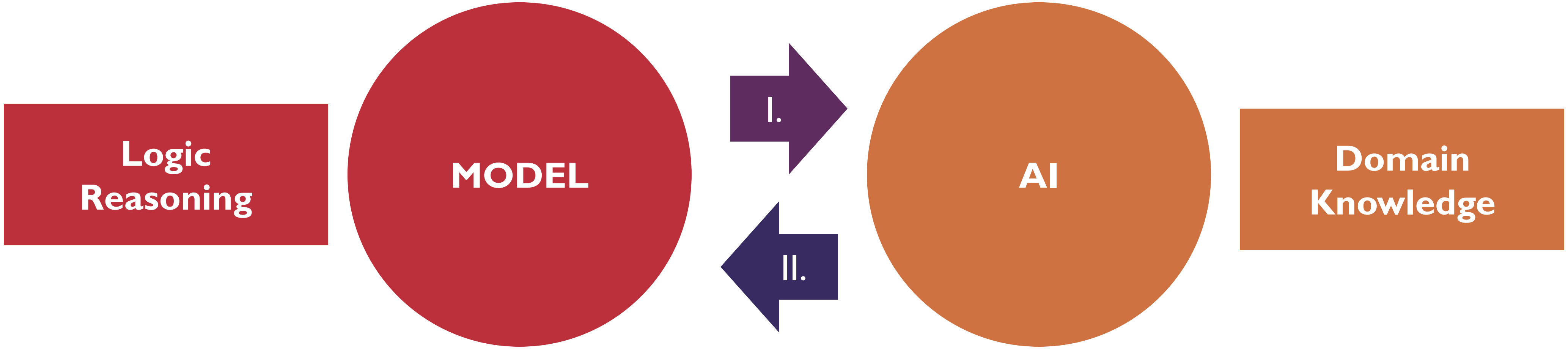
CRITICAL AI APPLICATION

- AI-based applications are widely used
- Even in safety critical domains

- However, AI-based applications
  - do not have the same kind of guarantees for trustworthiness
  - need novel testing/verification techniques

MODELS AND AI APPLICATIONS

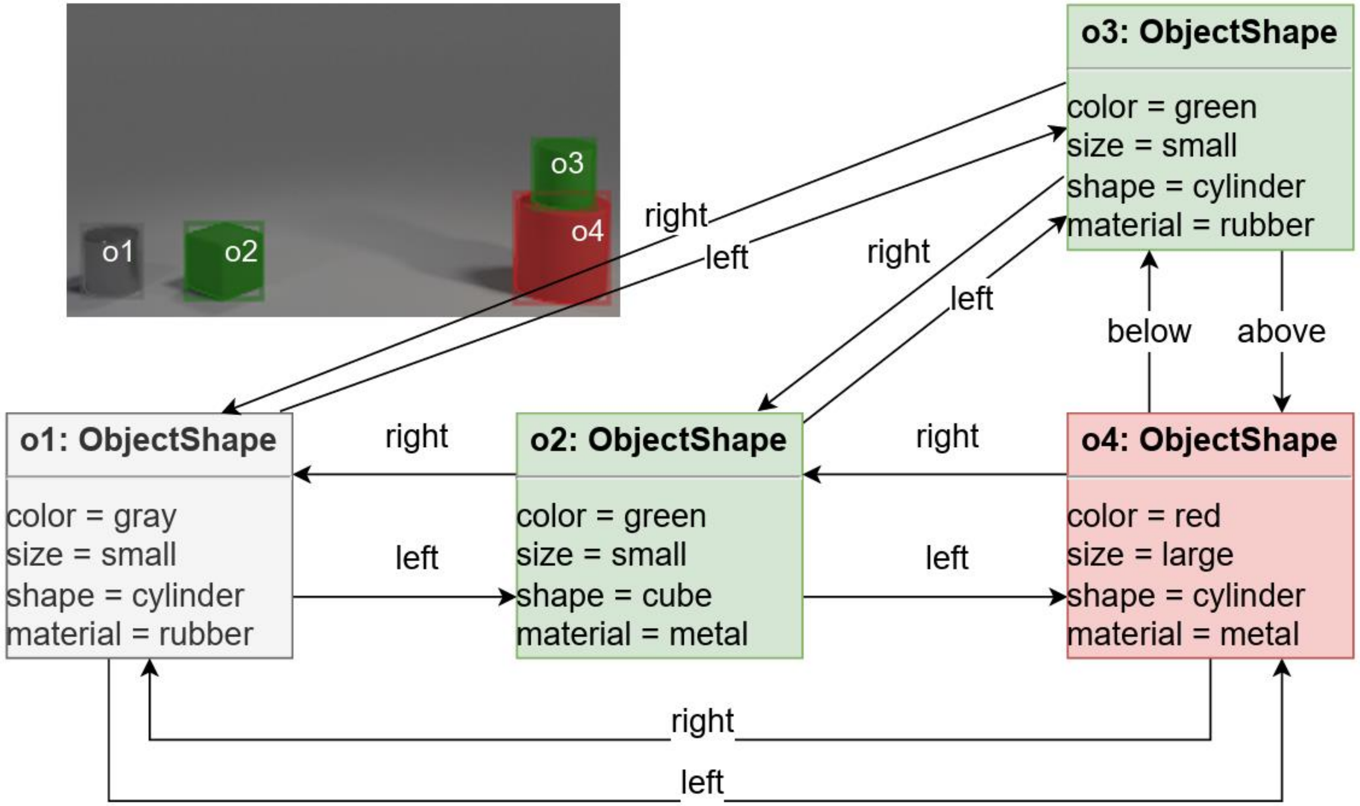
- Model expected behavior
- and model diversity
- to improve the reliability of AI applications



- We can **capture either**: (I.) coverage or (II.) random errors

HOW TO MODEL THE OUTPUT OF THE AI

- **Clevr:** visual question answering
- Ask all the questions
- Store it in a model
- We can answer the questions from the model, if we are smart enough



o1: ObjectShape
color = gray
size = small
shape = cylinder
material = rubber

o2: ObjectShape
color = green
size = small
shape = cube
material = metal

o3: ObjectShape
color = green
size = small
shape = cylinder
material = rubber

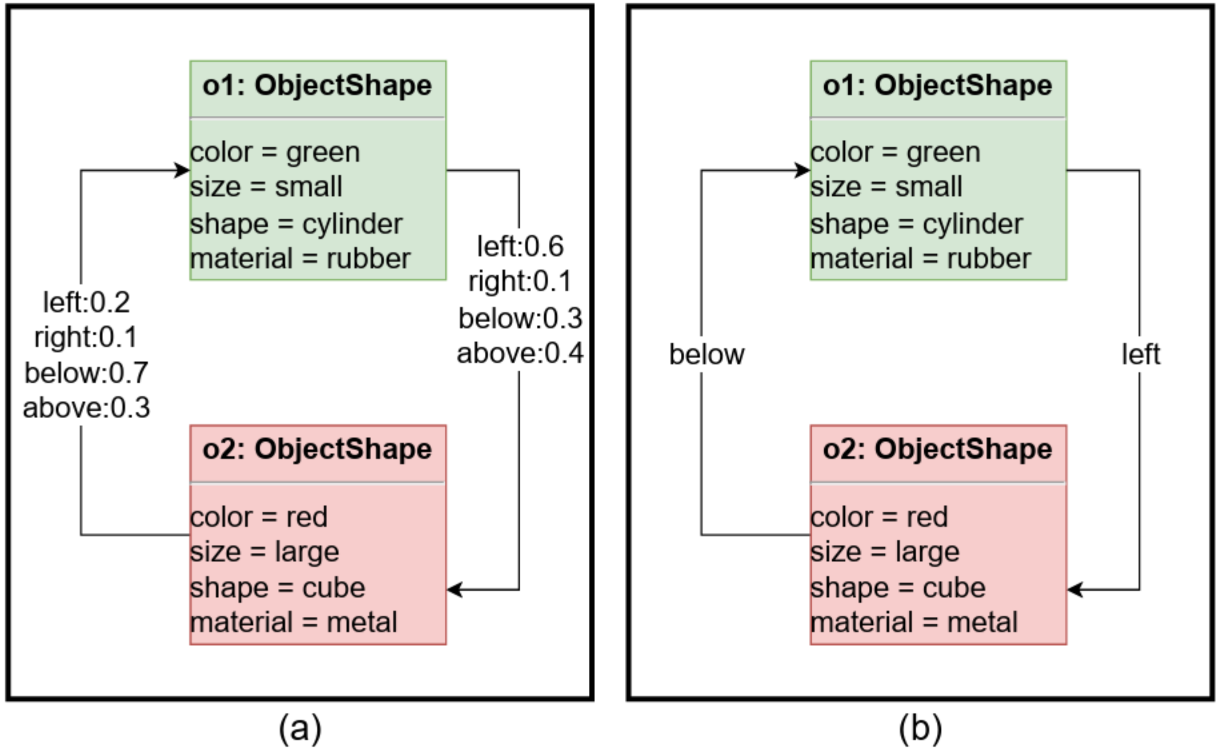
o4: ObjectShape
color = red
size = large
shape = cylinder
material = metal

UNCERTAINTY IN MODELS

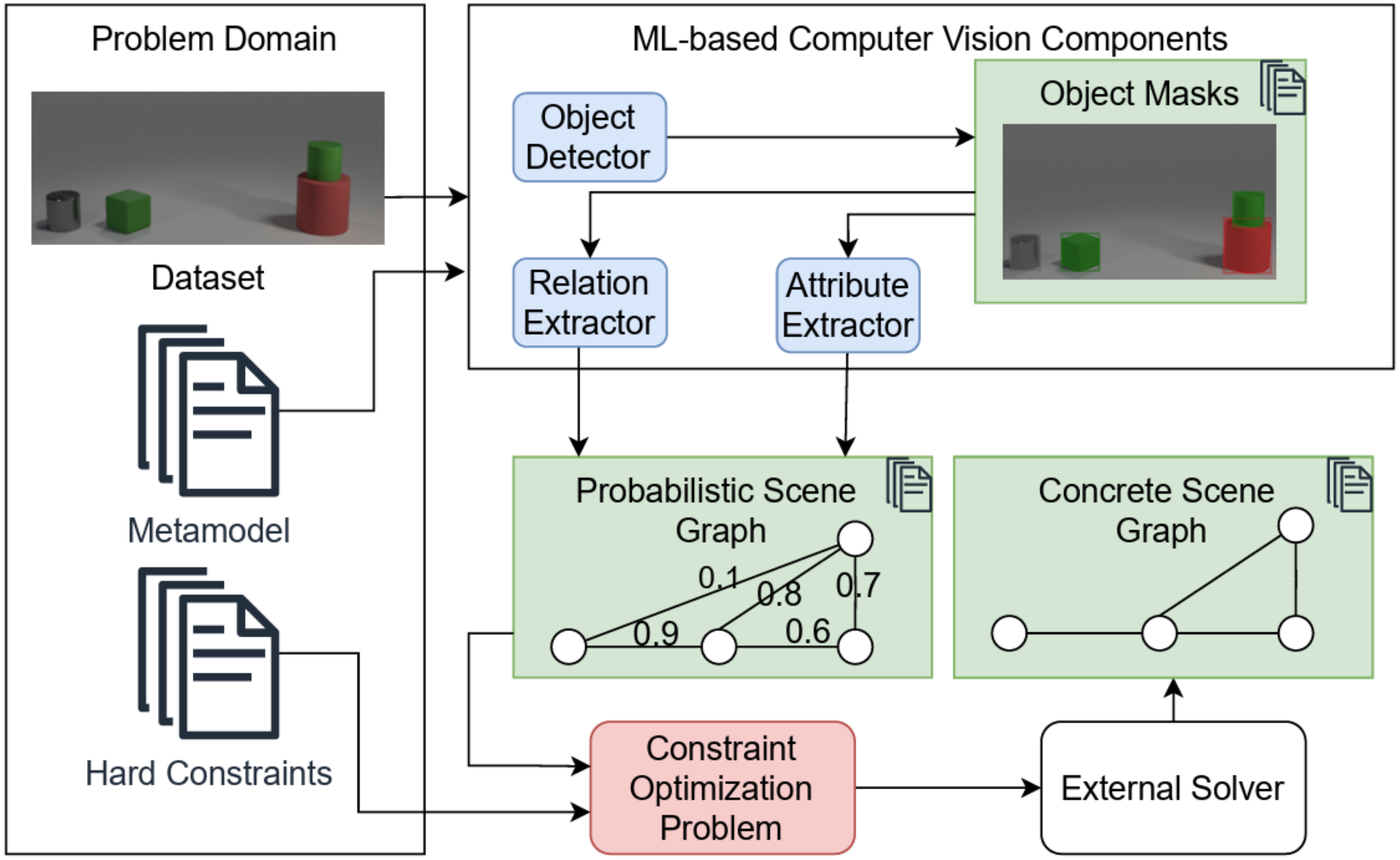
- AI applications cannot do logic reasoning with relations
  - But they can cheat with high probability
  - Lot of logic inconsistencies!
- AI applications provide weights for the answers
  - Those weights can be interpreted as confidence in answer
  - Lots of logic uncertainties!
- store uncertainties in the model explicitly.

partial models:

true | false | error | unknown



AI IMPROVEMENT WITH MODEL GENERATION


Problem Domain

Dataset

Metamodel

Hard Constraints

ML-based Computer Vision Components

Object Detector

Object Masks

Relation Extractor

Attribute Extractor

Probabilistic Scene Graph

Concrete Scene Graph

Constraint Optimization Problem

External Solver

Results:

+ **Consistent** models
+ Extra **constraints**
+ Better **accuracy**

- Lower **performance**

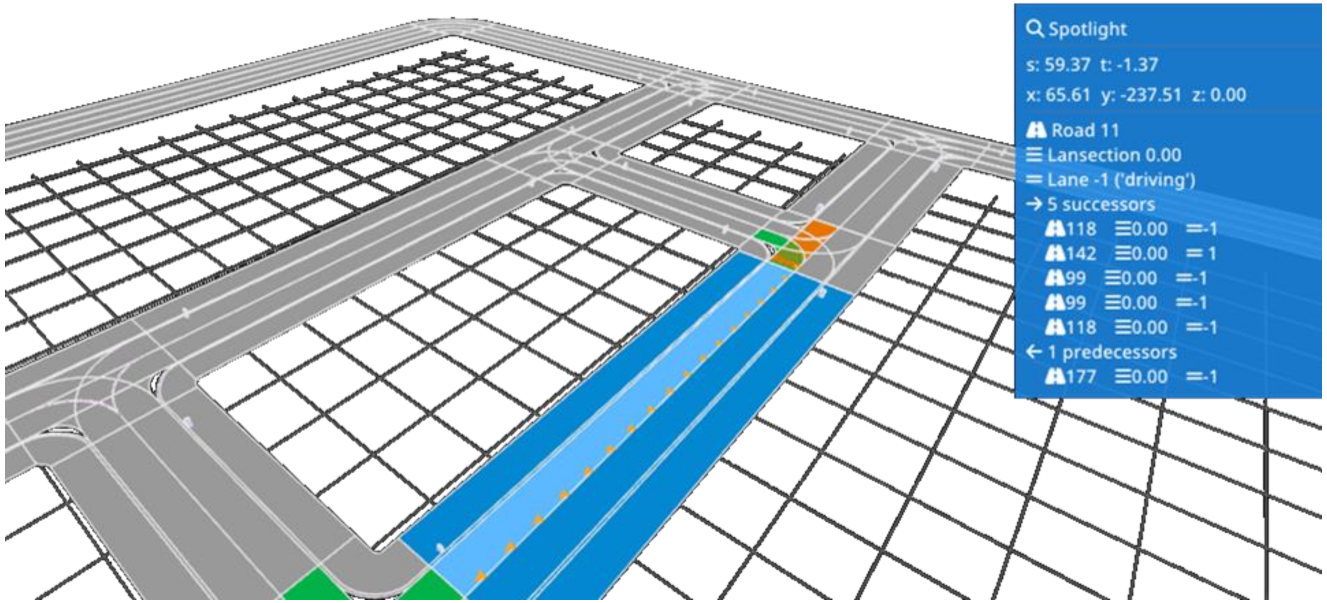
EXAMPLE PROJECT APPLICATION: VAMPIR

Input:

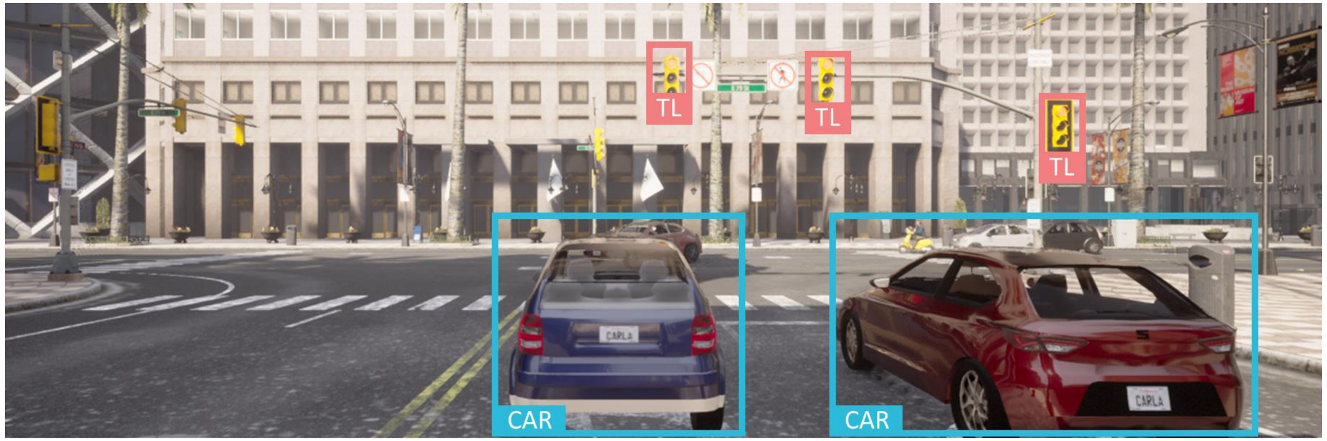
- GNSS position + HD-Map
- AI sensor output

Output:

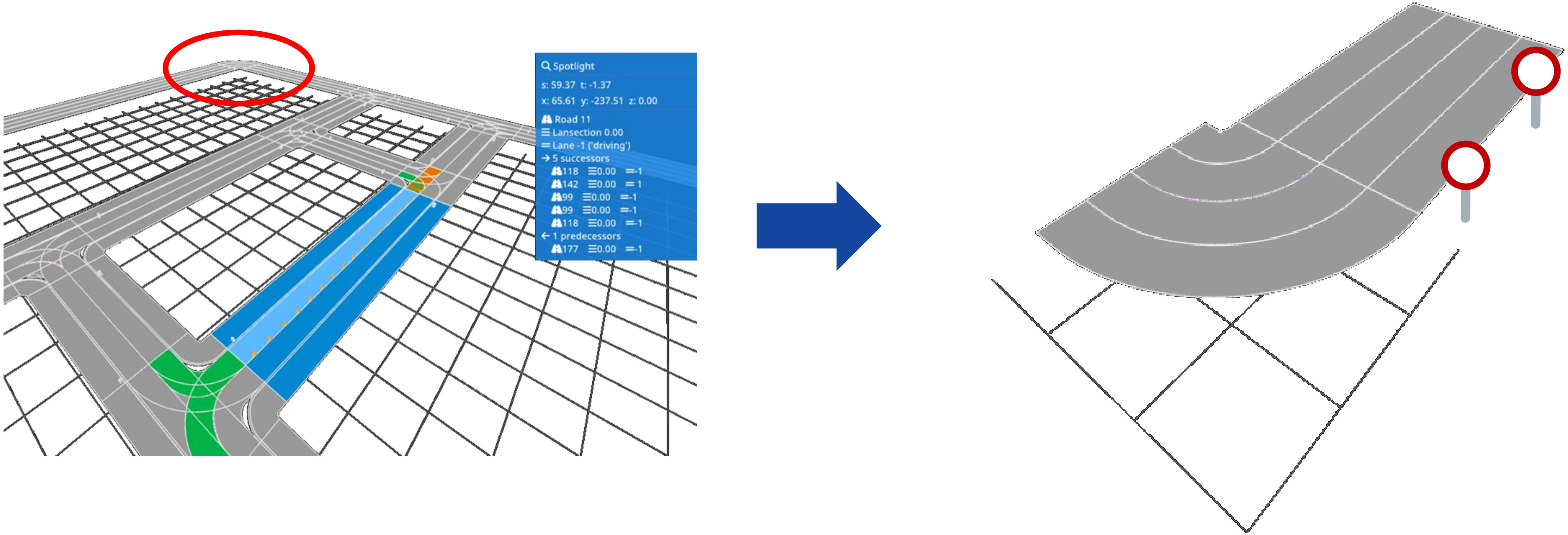
- Level 1 > Scene interpretation
- Level 2 > Safety and accuracy
- Level 3 > Improved GNSS accuracy



```
Spotlight
s: 59.37  t: -1.37
x: 65.61  y: -237.51  z: 0.00
Road 11
≡ Lansection 0.00
= Lane -1 ('driving')
→ 5 successors
  118  ≡0.00  =-1
  142  ≡0.00  = 1
  99  ≡0.00  =-1
  99  ≡0.00  =-1
  118  ≡0.00  =-1
← 1 predecessors
  177  ≡0.00  =-1
```

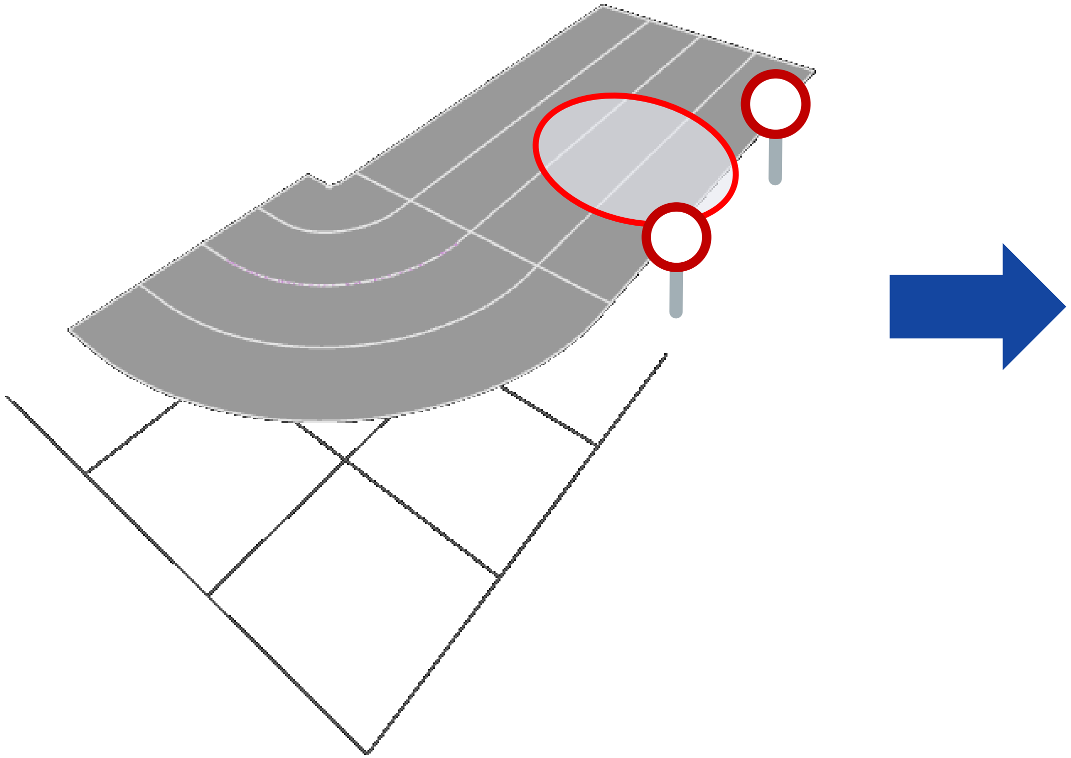


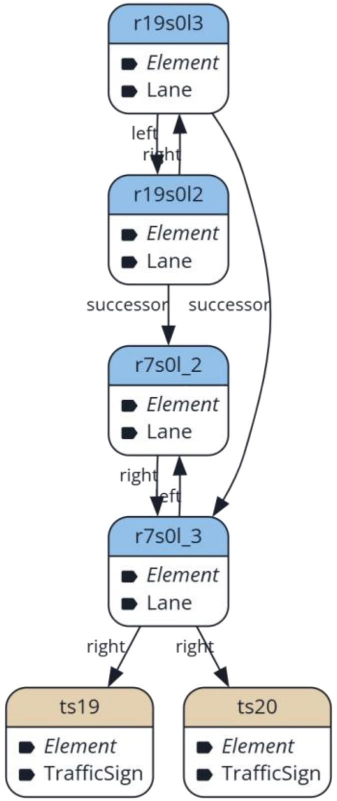
ESA EXAMPLE: MAP EXTRACTION



```
🔍 Spotlight
s: 59.37  t: -1.37
x: 65.61  y: -237.51  z: 0.00
Road 11
≡ Lansection 0.00
= Lane -1 ('driving')
→ 5 successors
   118   ≡0.00   =-1
   142   ≡0.00   = 1
   99   ≡0.00   =-1
   99   ≡0.00   =-1
   118   ≡0.00   =-1
← 1 predecessors
   177   ≡0.00   =-1
```

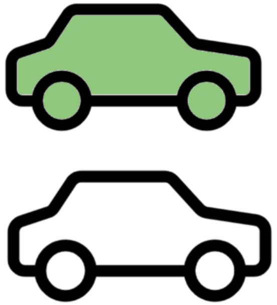
ESA EXAMPLE: MAP TO TOPOLOGY

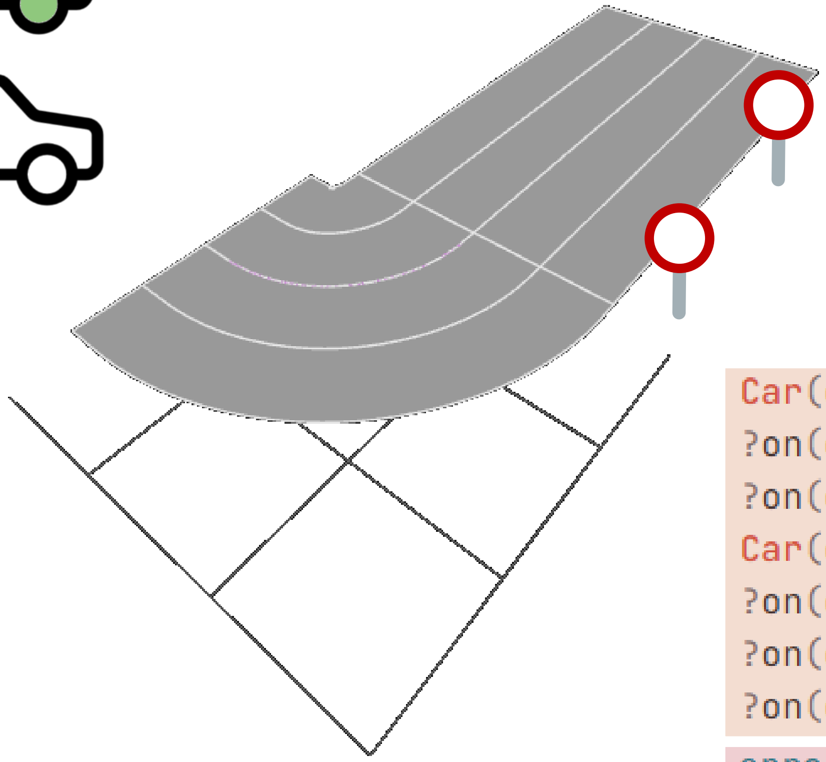




- Capture the topology of the environment
- Objects (lanes, traffic signs) and relations (successor, left, right)

ESA EXAMPLE: AI OBSERVATIONS





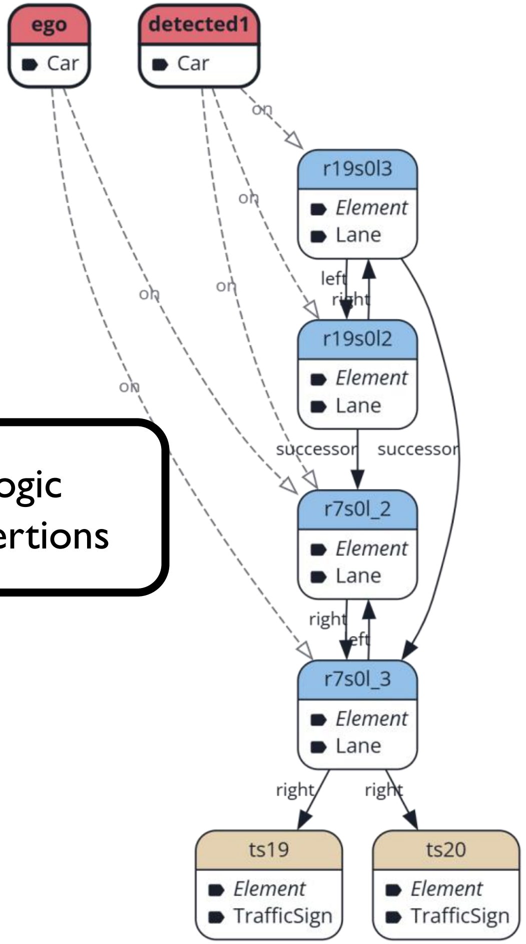
```
Car(ego).
?on(ego,r7s0l_2).
?on(ego,r7s0l_3).
Car(detected1).
?on(detected1,r19s0l2).
?on(detected1,r19s0l3).
?on(detected1,r7s0l_2).
```

logic assertions

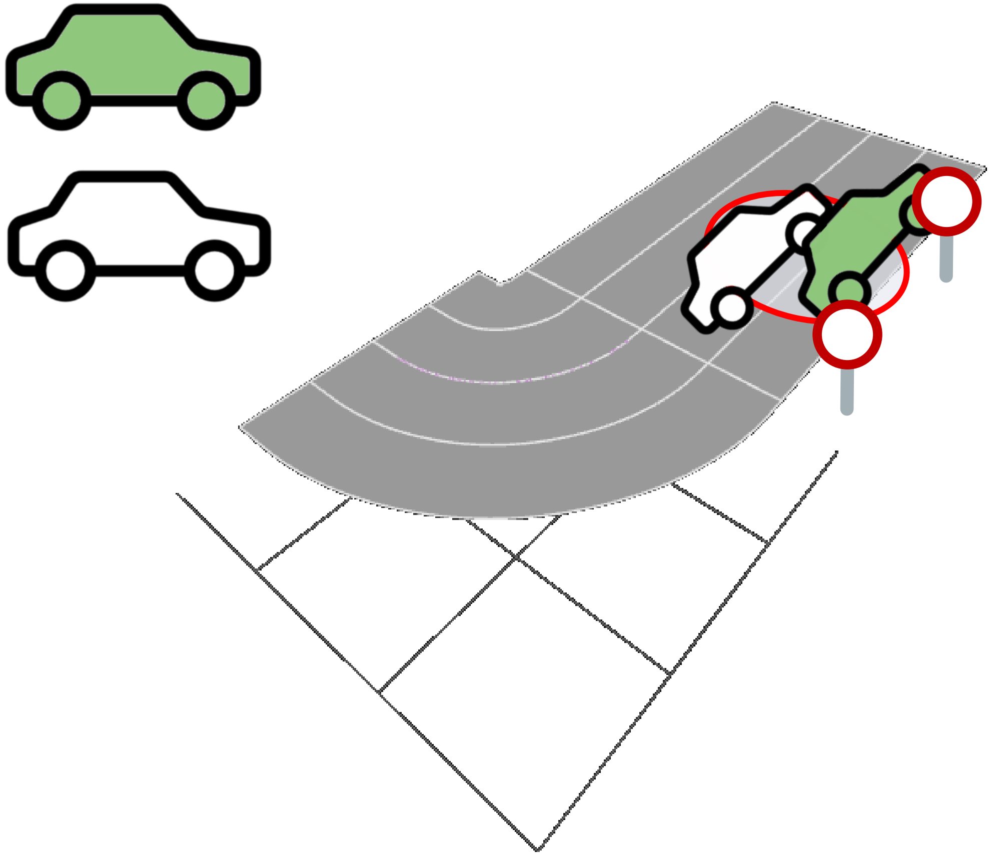
```
error notLeft(l1, l2) <->
    on(ego, l1),
    on(detected1, l2),
    !left(l1,l2).
```

logic predicates

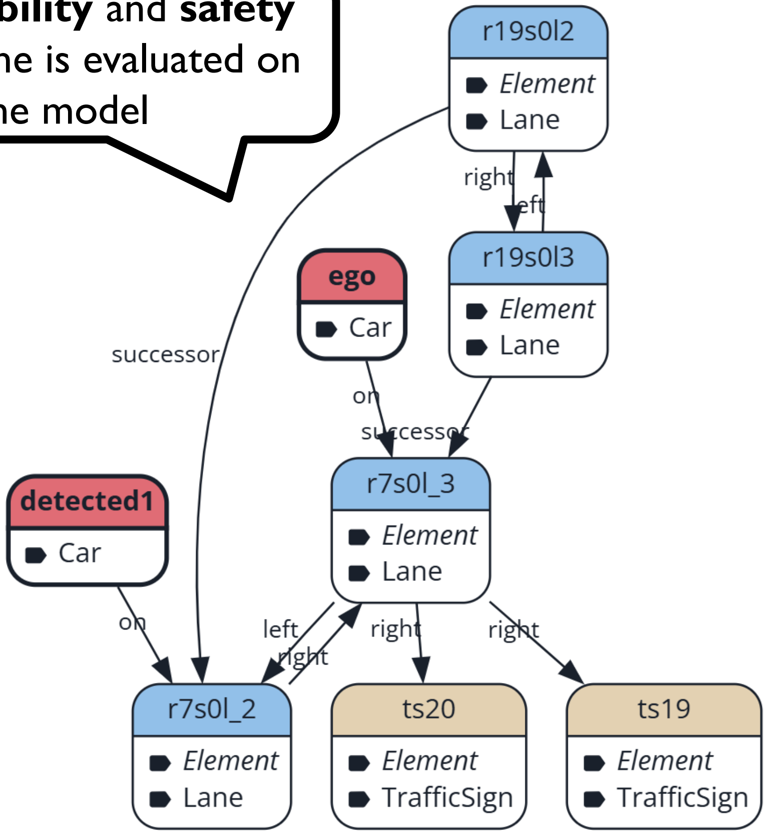
Detected car must be on the left



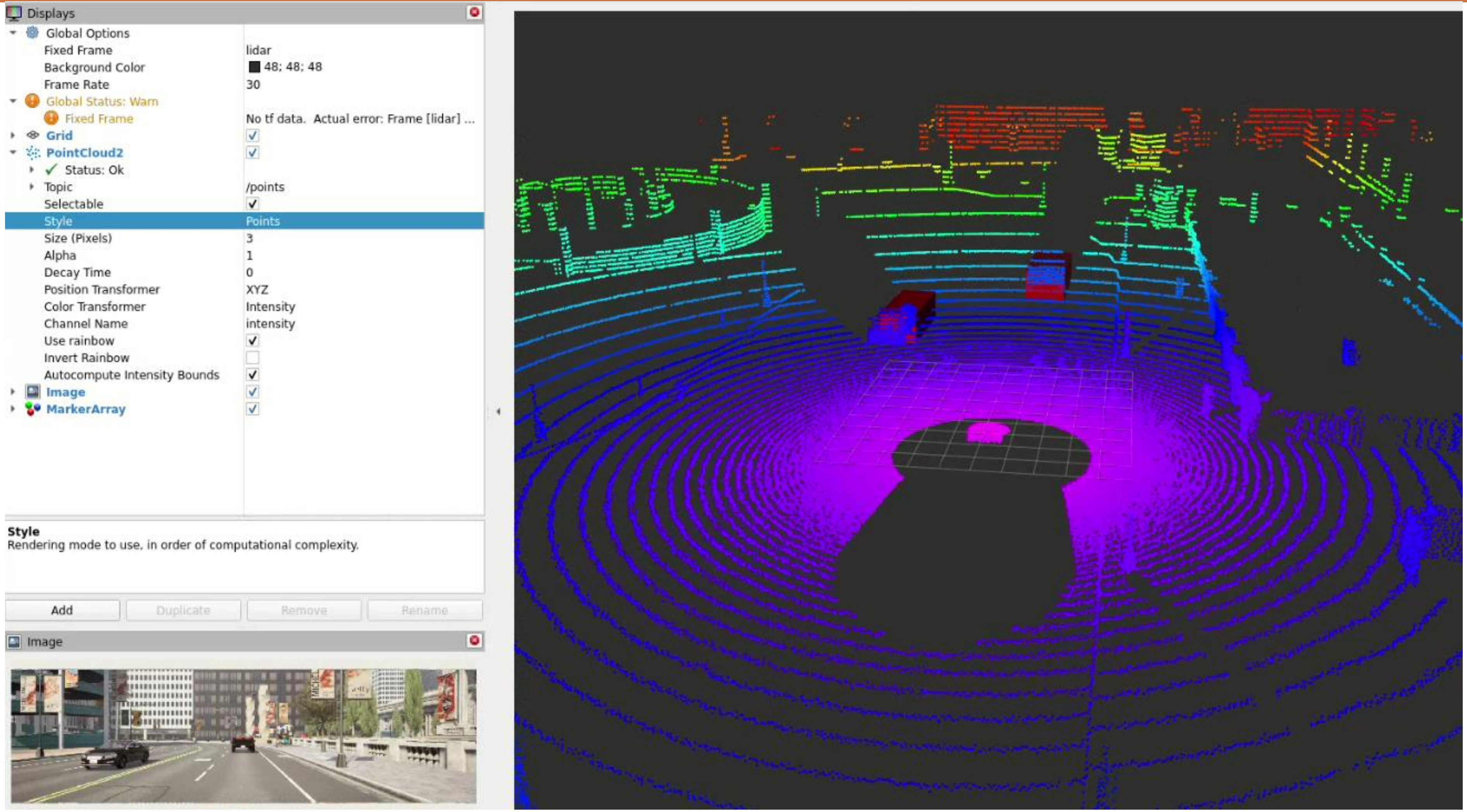
ESA EXAMPLE: SCENE RECONSTRUCTION



The **reliability** and **safety** of the scene is evaluated on the model



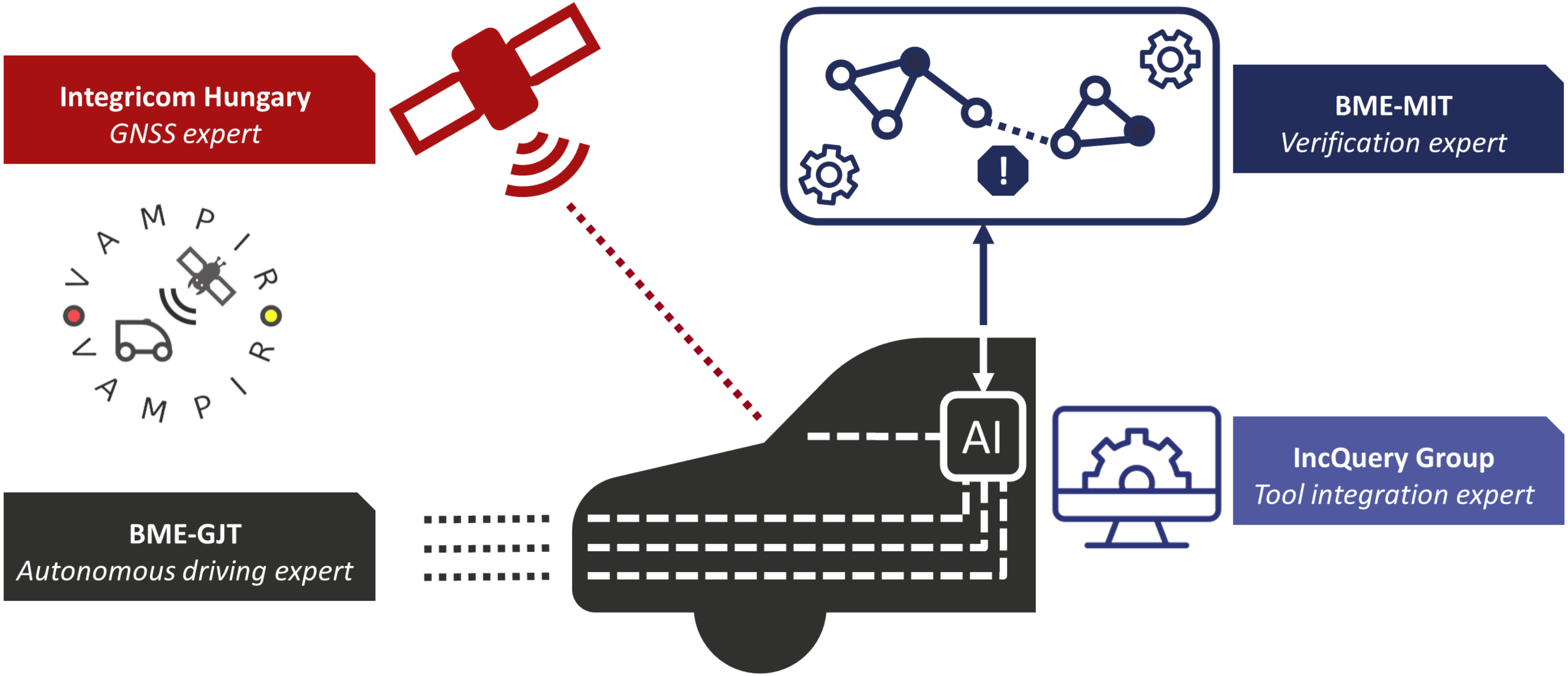
SENSORS



Global Options	
Fixed Frame	lidar
Background Color	■ 48; 48; 48
Frame Rate	30
Global Status: Warn	
Fixed Frame	No tf data. Actual error: Frame [lidar] ...
Grid	☑
PointCloud2	☑
✓ Status: Ok	
Topic	/points
Selectable	☑
Style	Points
Size (Pixels)	3
Alpha	1
Decay Time	0
Position Transformer	XYZ
Color Transformer	Intensity
Channel Name	intensity
Use rainbow	☑
Invert Rainbow	☐
Autocompute Intensity Bounds	☑
Image	☑
MarkerArray	☑

Style
Rendering mode to use, in order of computational complexity.

PARTNERS



Integricom Hungary
GNSS expert

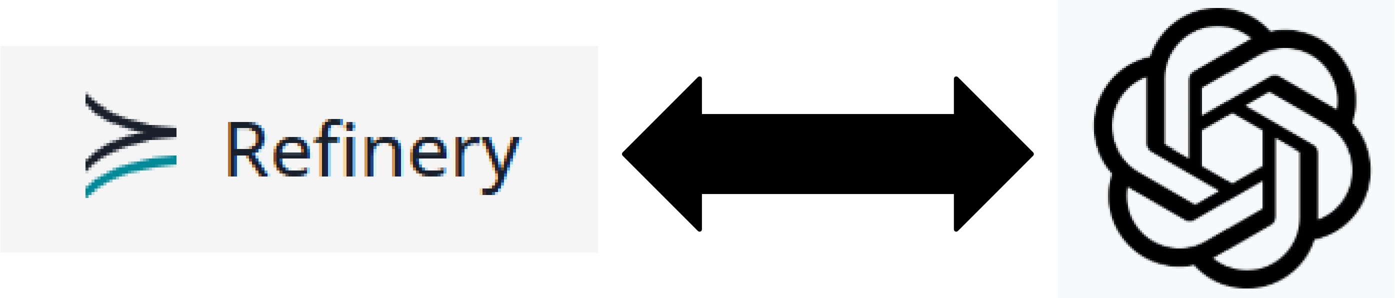
BME-MIT
Verification expert

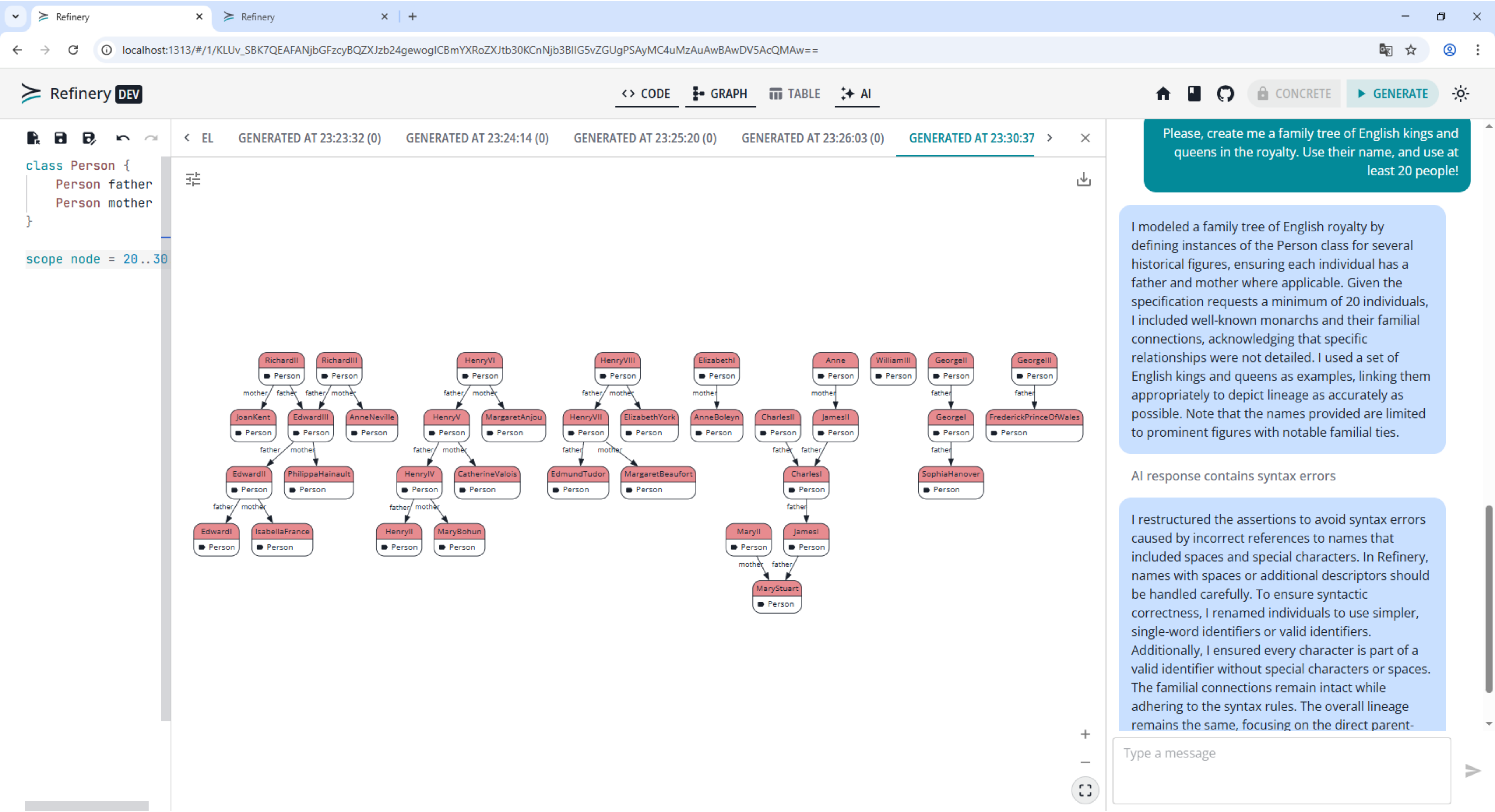
BME-GJT
Autonomous driving expert

IncQuery Group
Tool integration expert

TEXTUAL AI → REFINERY

- **DEMO**





```
class Person {
    Person father
    Person mother
}

scope node = 20..30
```

Please, create me a family tree of English kings and queens in the royalty. Use their name, and use at least 20 people!

I modeled a family tree of English royalty by defining instances of the Person class for several historical figures, ensuring each individual has a father and mother where applicable. Given the specification requests a minimum of 20 individuals, I included well-known monarchs and their familial connections, acknowledging that specific relationships were not detailed. I used a set of English kings and queens as examples, linking them appropriately to depict lineage as accurately as possible. Note that the names provided are limited to prominent figures with notable familial ties.

AI response contains syntax errors

I restructured the assertions to avoid syntax errors caused by incorrect references to names that included spaces and special characters. In Refinery, names with spaces or additional descriptors should be handled carefully. To ensure syntactic correctness, I renamed individuals to use simpler, single-word identifiers or valid identifiers. Additionally, I ensured every character is part of a valid identifier without special characters or spaces. The familial connections remain intact while adhering to the syntax rules. The overall lineage remains the same, focusing on the direct parent-

Thanks for the Attention!