BREKER™ THE LEADER IN PORTABLE STIMULUS

Using Planning Algorithms to Verify AI/ML Designs

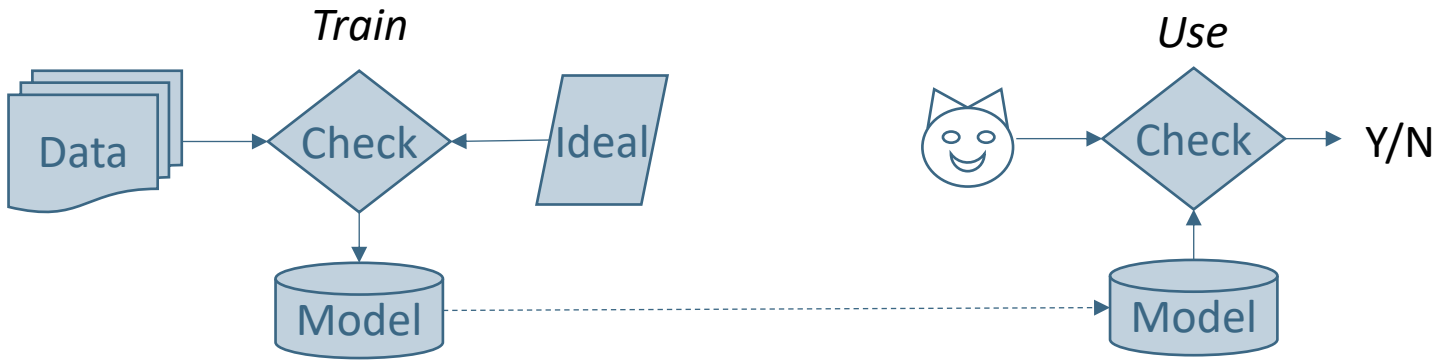
Adnan Hamid, CEO/CTO Breker Systems

DVClub April 2020

Agenda

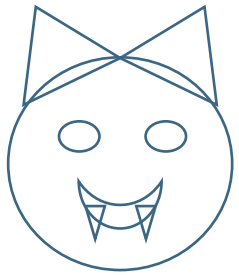
- What is Machine Learning
- ML Hardware Architectures
- Verification Challenges for ML Hardware
- Planning Algorithms and Verification
- ML Planning Model and Results

Machine Learning (ML)

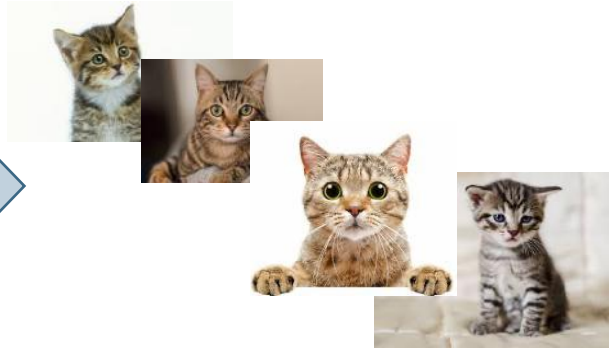


Machine Learning for a "Cat Recognition System"

Supervised
Start with some basic rules of ideal image and iterate on data



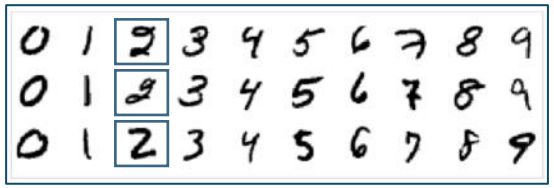
Un-supervised
Input lots of data so system can "learn" cat attributes



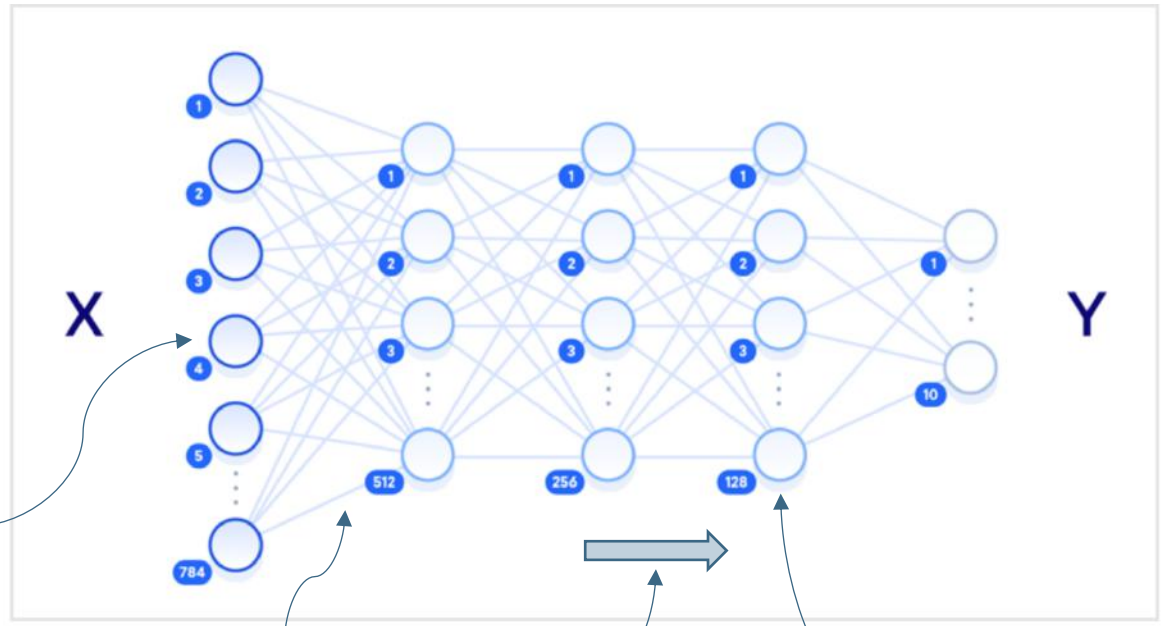
ML - How Does it Work

Example: Number Set Pattern Recognition

Convolutional Neural Network (CNN)



Pixel maps



0	0
0	1
1	2
0	3
0	4
0	5
0	6
0	7
0	8
0	9

Successively apply data and adjust coefficients to reduce error, until high-probability "model" is achieved

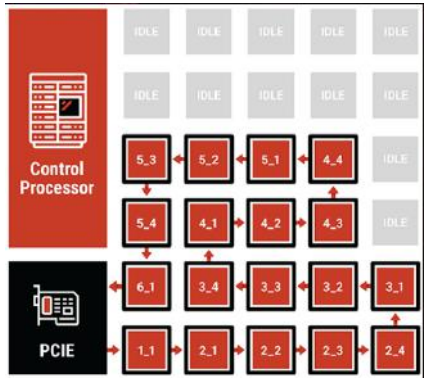
Fully-connected Neural Network

Multiply Accumulate Processor (often floating point)

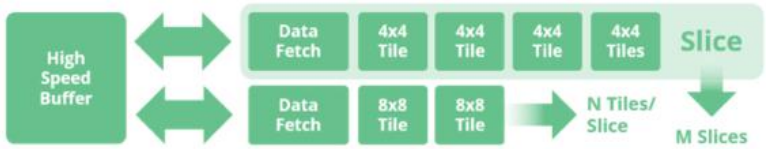
"Deep" learning refers to number of "hidden" element layers

Courtesy: Ellie Birbeck, DigitalOcean

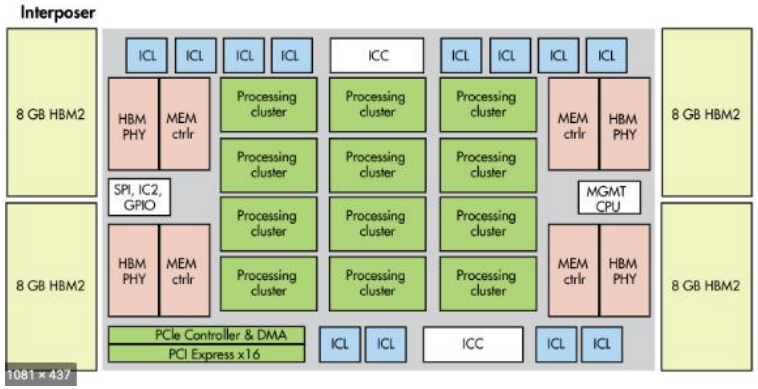
Lots of Implementation Options



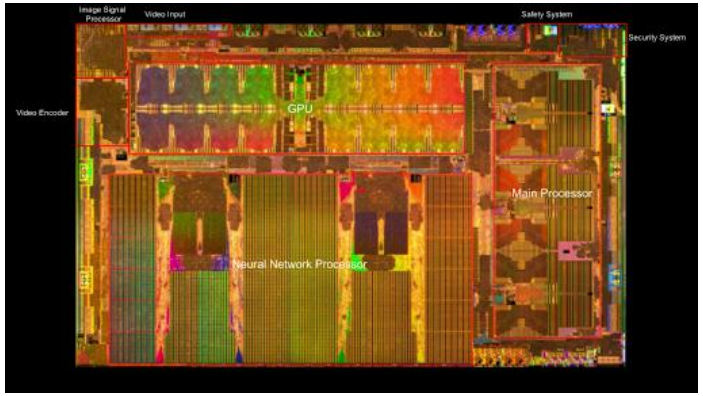
Mythic



Wave



Intel



Tesla

All designs will only run the output of a known compiler

Maybe Verification is not an Issue ?

- Only have to run the output of a known compiler
  - Fix any HW issues in SW ?
- Network of identical tiles
  - Easy to verify repetitive logic ?

Maybe Verification is Mission Critical ?

- Full design too big to simulate / emulate
- One tile too small to verify fully

- Lots of complex control logic
- Lots of complex routing and scheduling

- Compilers can only generate a small "normal" case
- Impossible to stress corner cases for control and routing

Planning Algorithms and Verification

- **What is AI Planning?**

  *Planning is a long-standing sub-area of Artificial Intelligence (AI). Planning is the task of finding a procedural course of action for a declaratively described system to reach its **goals** while **optimizing overall performance** measures. Automated planners find the transformations to apply in each given state out of the possible transformations for that state. In contrast to the classification problem, planners provide guarantees on the solution quality.*

  Courtesy: IBM Watson

- Start with the desired outcome to check
- Solve for a plan of steps that will achieve that outcome

Constrained Random Numbers for Verification

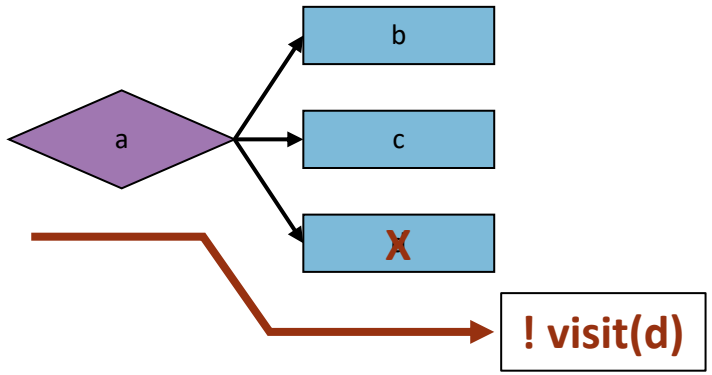
- Randomize values subject to constraints

```
rand int x;
constraint { x < 5 };
```

Planning Algorithms Adds Constrained Random Control Flow

- Randomize control flow subject to constraints

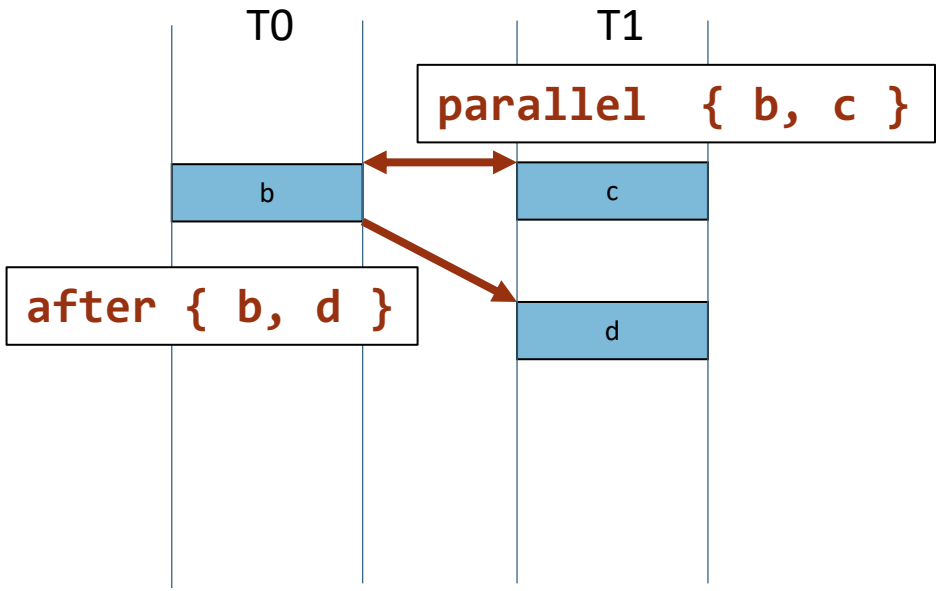
```
class A : public action {
   B b; C c; D d;

   void body() {
      select { b, c, d};
   }
};

constraint c { ! visit(d) };
```



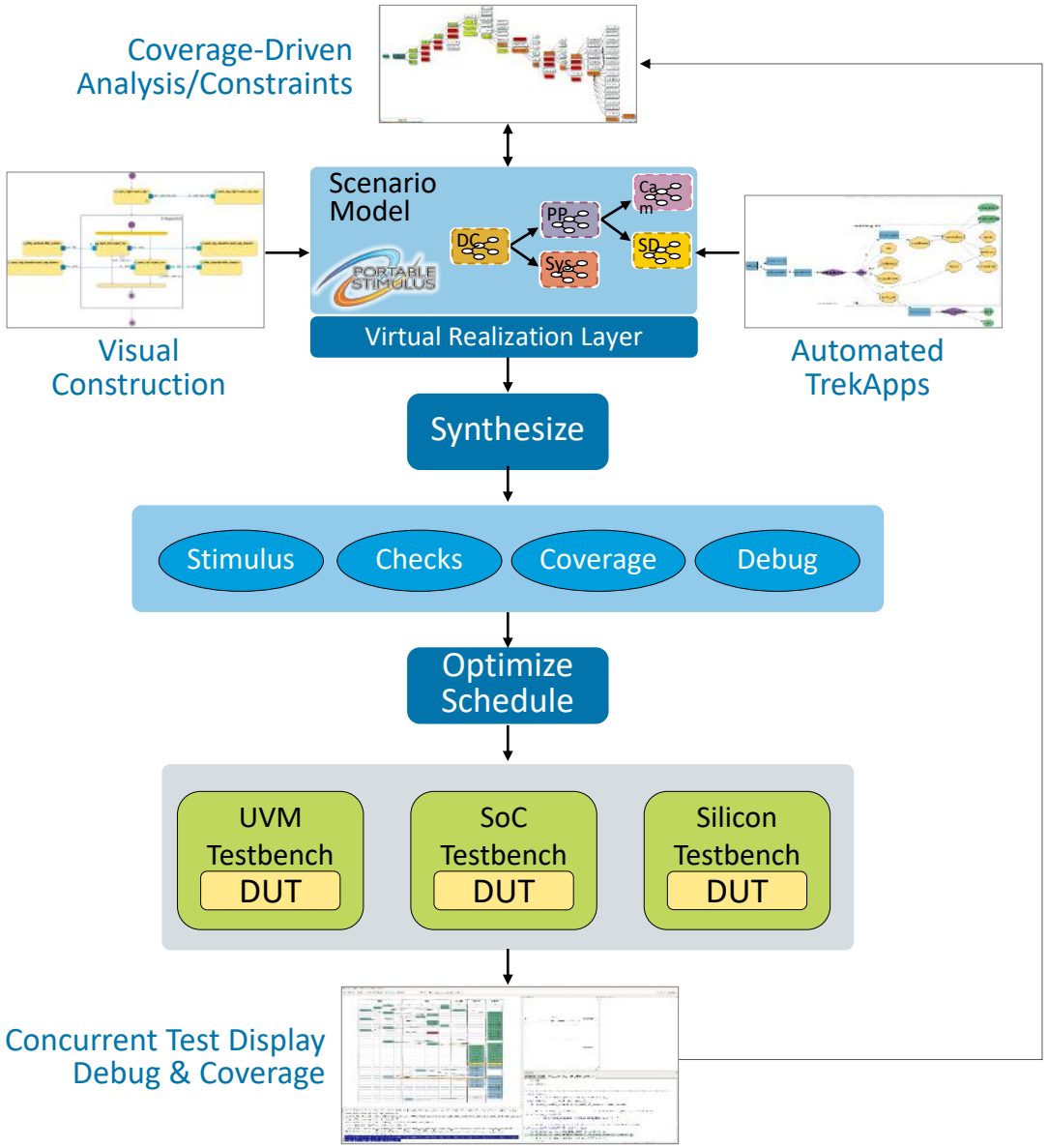
Planning Algorithms Adds Constrained Random Scheduling

- Randomize scheduling subject to constraints

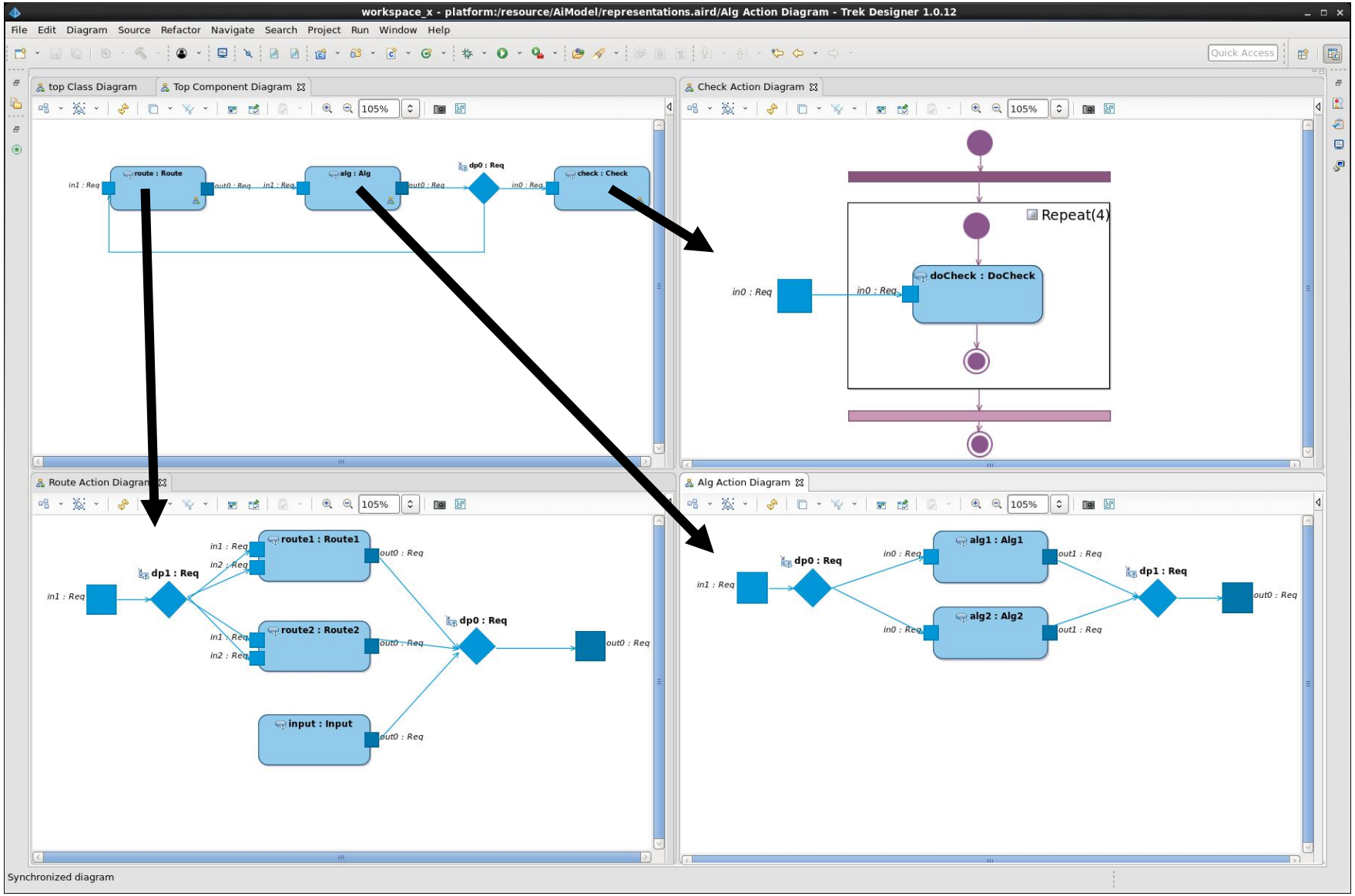
```
class A : public action {
   B b; C c; D d;

   void body() {
      schedule { b, c, d};
   }
};

constraint c1 { parallel {b, c} };
constraint c1 { after {b, d} };
```



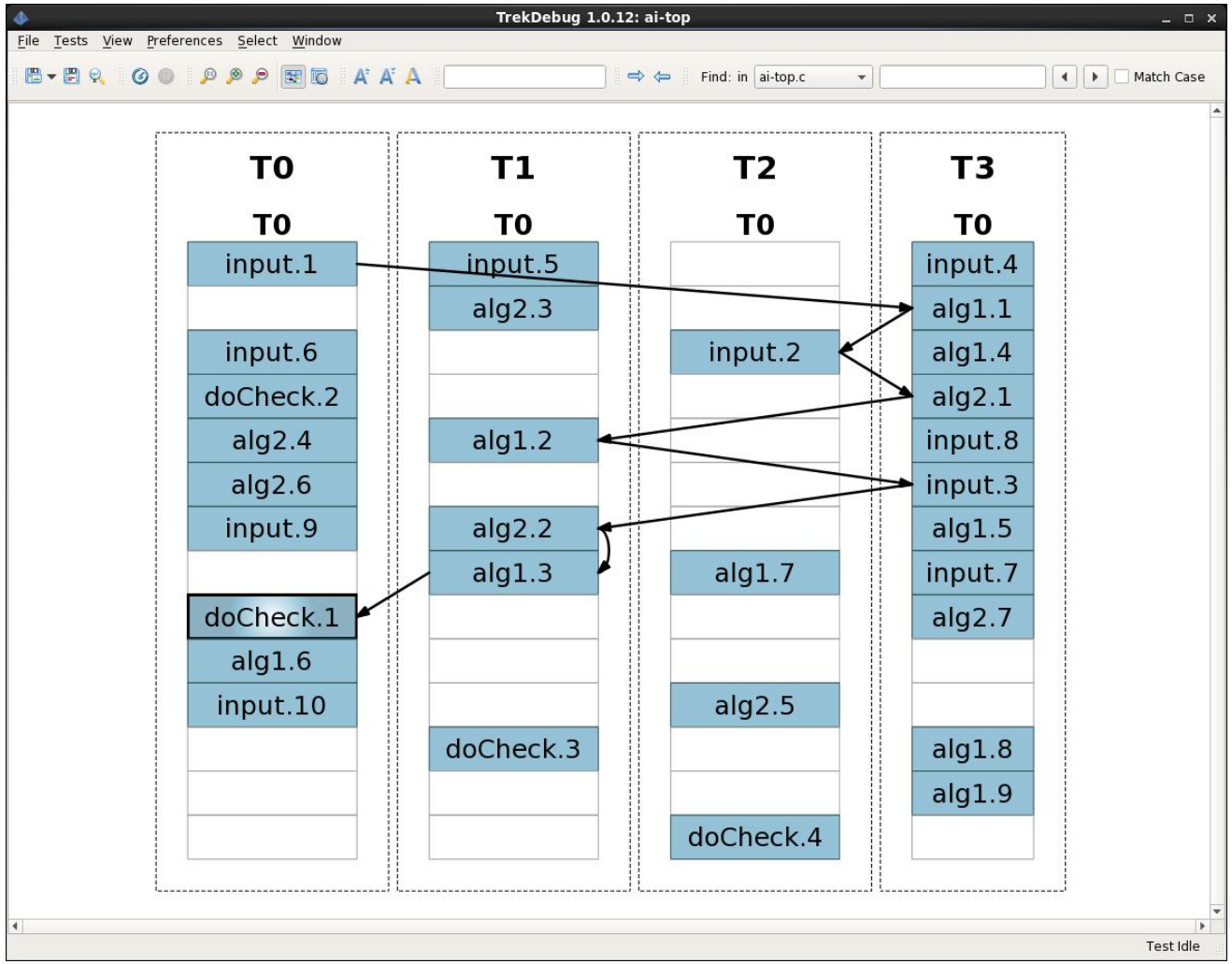
Once We Have Intent Model, We Synthesize It



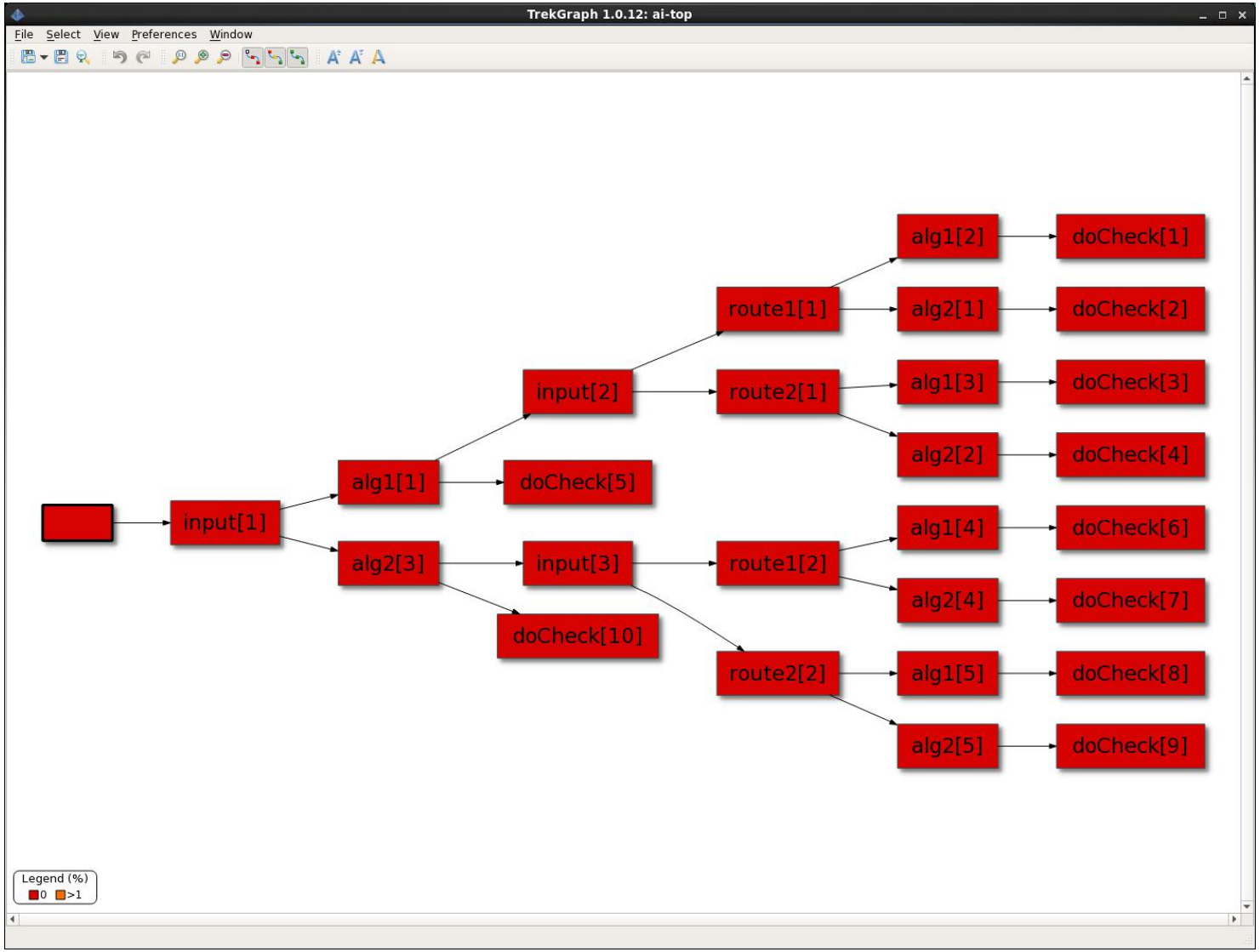
Modeling the Verification Planning Space



Example Schedule of Operations



Path Analysis across Three Stages

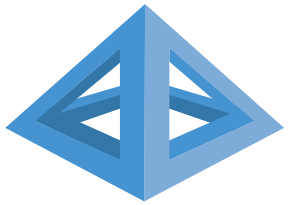


Summary

- Many HW architectures for ML
- All share difficult verification challenges
- Planning algorithms provide a scalable solution
- Reuse from simulation to emulation to silicon

For more Information:

www.brekersystems.com



Thanks for Listening!
Any Questions?

For more Information:

www.brekersystems.com