Inferring Automatic Test Oracles

William B. Langdon[1], Shin Yoo[2], and Mark Harman[1]
1: University College London, UK
2. KAIST, Republic of Korea

“Would you still have broken it if I hadn’t said anything?”
- The Oracle

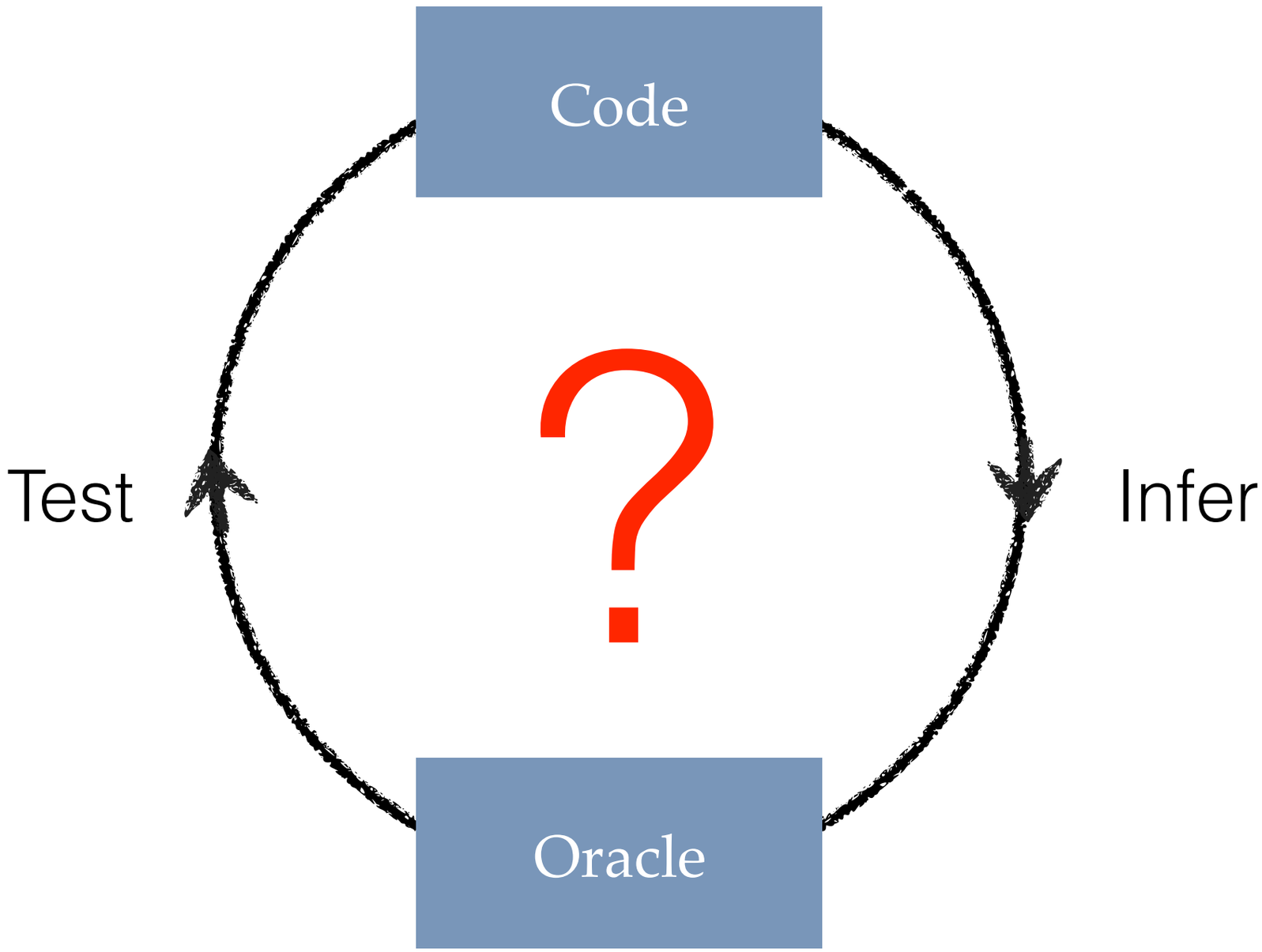
Where do they come from?

- *“Don’t think it twice, it’s alright”*: test lacks explicit oracles, they only check for implicit ones (such as crashes) - we can and should do better.

- *“Nine to five”*: oracles are generated manually - it is hard work and can be error-prone.

- *“Up where we belong”*: oracles are derived from higher-level abstractions, such as formal specifications or (correct) models - elegant in theory, somewhat hard to find in reality.

- ***“Mother and child reunion”***: oracles are learnt from existing code - that works very well in practice, but how does it work in theory?


Code
Test
Infer
?
Oracle

“This is ground control to…”

Definition 2.6 (Ground Truth). *The* ground truth oracle*, $\mathcal{G}$, is a total test oracle that always gives the “right answer”.*

We can now define soundness and completeness of a test oracle with respect to $\mathcal{G}$.

Definition 2.7 (Soundness). *The test oracle D is* sound *iff*

$$D(a) \Rightarrow \mathcal{G}(a).$$

Definition 2.8 (Completeness). *The test oracle D is* complete *iff*

$$\mathcal{G}(a) \Rightarrow D(a).$$

E. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo. The oracle problem in software testing: A survey. IEEE Transactions on Software Engineering, 41(5):507–525, May 2015.

Some other sources

- *“(Ch—Ch—Ch—) Changes”*: oracles can be generated using metamorphic relations - we are yet to know how generalised it can be.

- *“Daddy’s Car”*: AI has been used to to learn oracles - still in the very early stage.

- *“Me, myself, and I”*: N-version computing can be used to formulate oracles - human based N-version computing has been shown to be too expensive.

Multiplicity

- The risk shared by many of the existing oracle inference techniques is the fact that they tend to learn from a single datapoint, i.e., the current implementation.

- The second batch techniques share multiplicity: relationships between multiple test cases, multiple datapoint used to train the AIs, and multiple programs.

What We Suggest

- We can exploit advances in AI/ML to learn or at least better approximate **the ground truth** (or, dare we say, *testing common sense*).

- **Automated N-version computing** and automated oracle inference will, and should, go hand in hand.

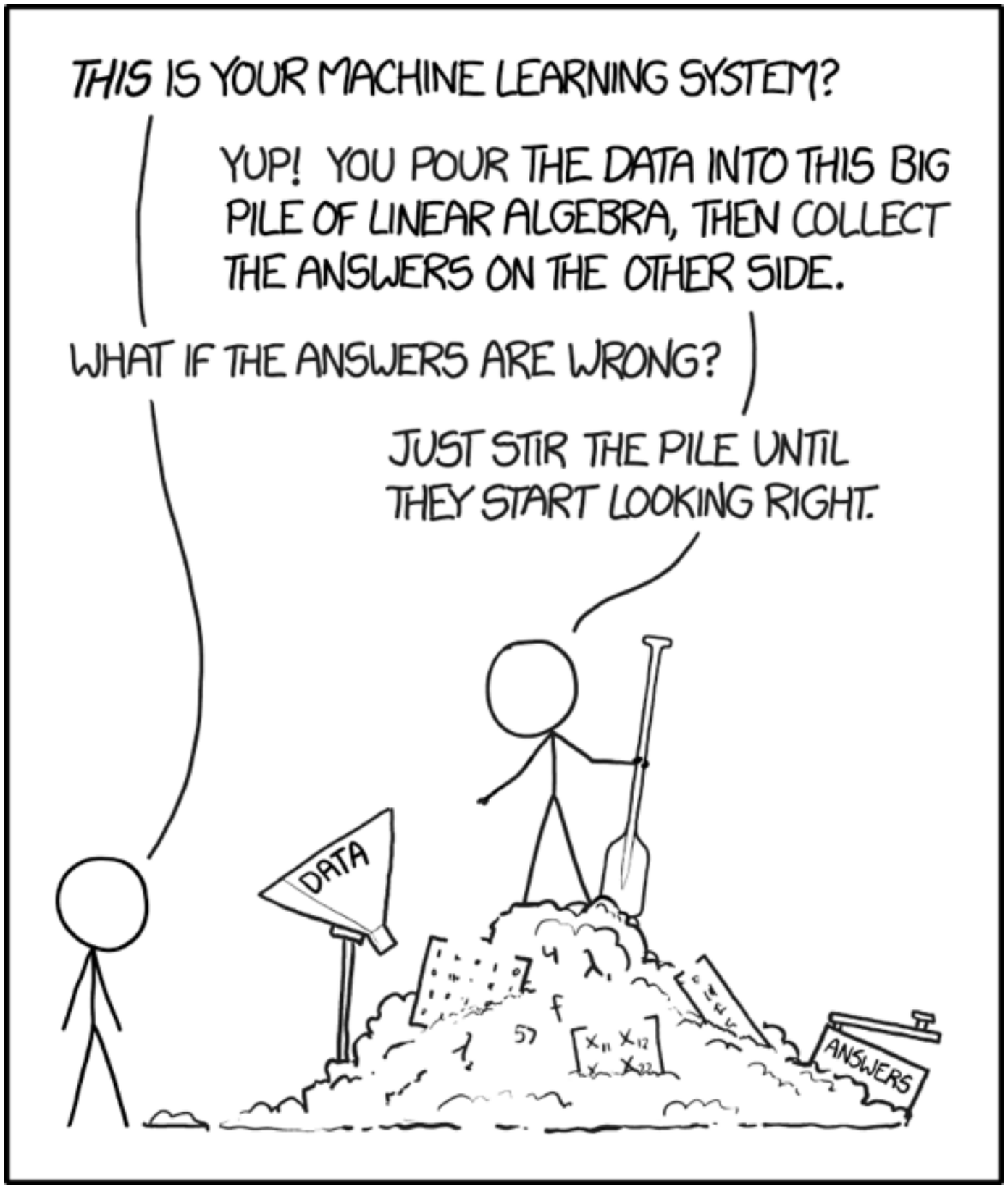
Learning the Common Sense

- Why stop at learning oracles from your own project? Learn from the entire open source.
  - *"(…in certain contexts) it seems that getting* `null` *return values is not a good thing."*
  - *"given the name* `getAge()`*, a negative return value appears to be suspicious."*
  - *"**normally** we should not really encounter OutOfMemoryException, so this looks wrong."*

“Thick as a Brick”

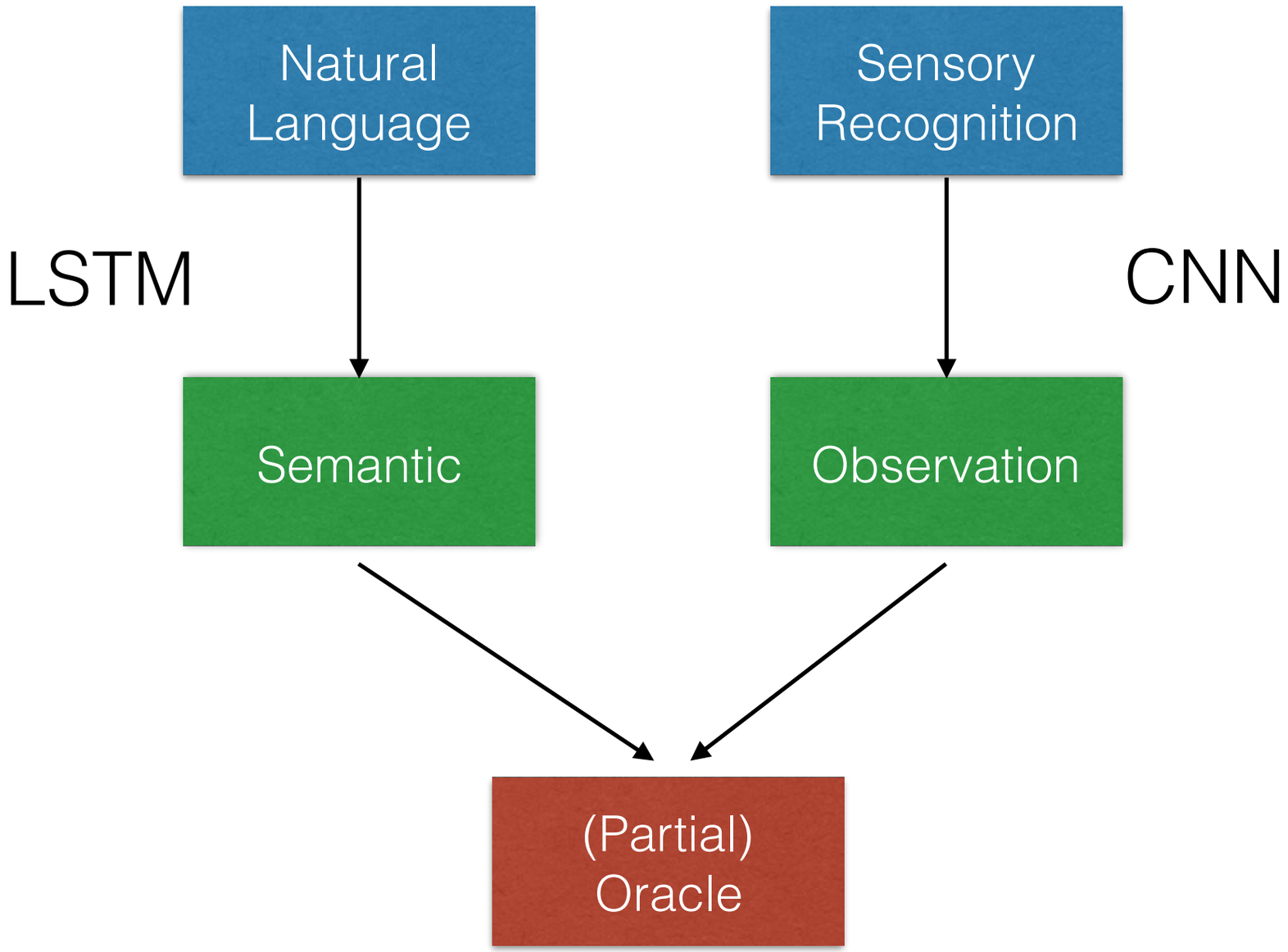
- All of these require simultaneously learning at multiple abstraction levels: natural language, formal semantic, code, etc.

- Deep Neural Networks are making strides in learning at multi-level representations: perhaps they can help?


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Where DNNs can already help

- System level testing, especially at GUI level, is more about high-level observations than low-level, internal program semantics.
  - *“If I click a **menu button** in the app, a **menu** should appear.”*
  - *“If I **log-in**, I should be greeted with the **members landing page**.”*
  - *“If I click the **play button**, a **video** should start.”*


Natural Language
Sensory Recognition
LSTM
CNN
Semantic
Observation
(Partial) Oracle

From N-Version Programming to N-Version Testing/Oracle

- AKA “Sorry, Robert!”: how well N-version programming can be done is directly related to how hard certain requirements are to meet - this connects N-version programming to N-version testing/oracle.

R. Feldt. Generating diverse software versions with genetic programming: and experimental study. IEE Proceedings - Software, 145(6):228–236, 1998.

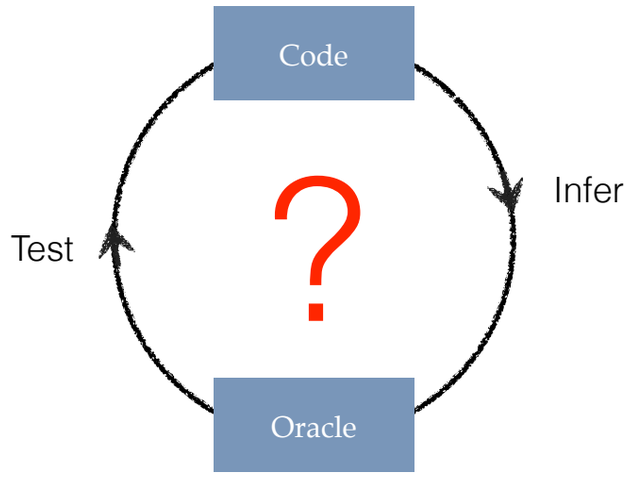
Issues

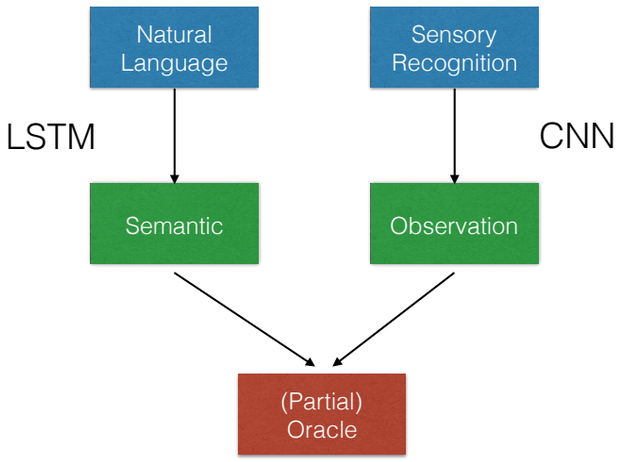
- Human Maintainability: Will it be legible?
- Liability: Will we be responsible? If not, who?
- Acceptability: Will we be using it?
- Eventually, all of these are related to how much we can embrace partial oracles or, even partial correctness.

Benchmarks

- Uh… well… hmmm. Yes.
- It takes a lot of work, but the entire community can benefit from a good benchmark.
- Accept that benchmarks never come one-size-fits-all.
- More context-aware, qualitative approaches are needed.
- We need to plan ahead.

“Would you still have broken it if I hadn’t said anything?”
- The Oracle





http://bit.ly/sbst2017playlist

Please consider submitting your work to ICST 2018 :)

- Double Blind / Full Length Industry Track Papers
- General Chair: Hans Hansson (Mälardalen)
- Program Co-chairs: Robert Feldt (Chalmers), Shin Yoo (KAIST)
- Abstract due by 5th October 2017
- Full Research Papers due by 12th October 2017