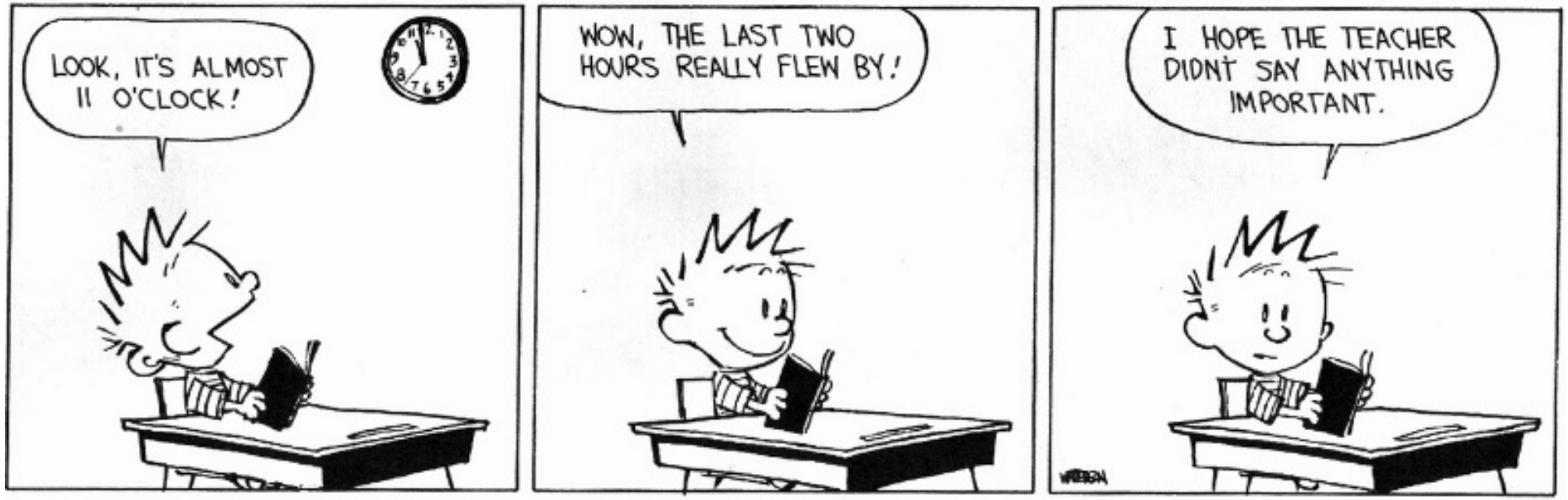
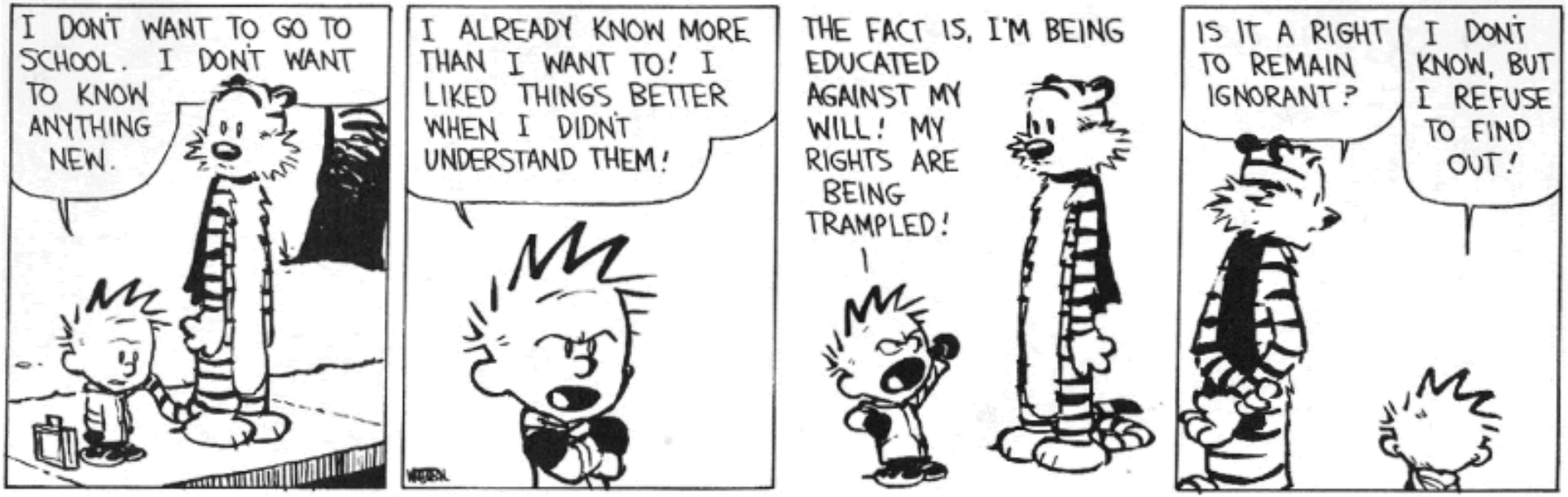


What Have We Learned?



Semantics = "Meaning"

- **Operational Semantics**
  - Large-Step - common in papers, very easy
    - $< e, \sigma > \Downarrow v \qquad < c, \sigma > \Downarrow \sigma'$
  - Small-Step - common in papers
    - $< c, \sigma > \rightarrow < c', \sigma' >$
  - Contextual - model heap, threads, global stuff
    - $(\text{if } \bullet \text{ then } 1+2+3 \text{ else } c_2)\ [\ \text{true}\ ] \rightarrow (\bullet+3)\ [1+2]$
- **Denotational Semantics**
- **Axiomatic Semantics**
  - Verification Condition Generation
    - PCC, SLAM, ESC/Java, etc.

Lambda Calculus = "Model"

- Model of programming and computation
  - Encodings - true, if, +, pairs, …
- Lambda **Type Systems**
  - Simply-Typed - $\Gamma \vdash x + 3 : int$
  - Recursive Types - α list = $\mu t.\ () + (\alpha \times t)$
  - Subtypes - coercion, OO
  - Imperative Types - references, exceptions
  - Second-Order Types - length : α list $\rightarrow$ int
  - Dependant Types - array-of-length(x)
  - Linear Types - cannot leak resources

Proof And Meaning

- **Structural Induction** proof technique
  - **if $\Gamma_0 \vdash e : \tau$ and $\langle e, \sigma_0 \rangle \Downarrow v$ then $v \in |\tau|$**
  - "Type Safety", "Subject Reduction"
- **Abstract Interpretation** analysis framework
  - $\alpha(3)$ = "positive"; $x \in \gamma(\alpha(x))$
- **Automated Theorem Proving**
  - Cooperating Decision Procedures

Advanced Models

- Pi Calculus model of concurrency
  - Synchronous message passing, 1st-class channels
- Region-Based Memory Management
  - As safe as garbage collection, faster than malloc
- Sigma Calculus model of objects
  - Method invocation is primal
  - (ok, we didn't see this …)

Bug-Finding

- Software Model Checking
  - Linear temporal logic, “eventually”
  - State space exploration
- SLAM project for finding bugs
  - convert c program to boolean program with axiomatic semantics and theorem provers
  - model check boolean program
  - explore counterexample with symbolic execution (operational semantics)
- Cooperative Bug Isolation

Conclusions

- **PL is the topic of ultimate mastery**
- Theory, practice, models, engineering, proofs and impact

Common PL Research

- **Design** – evaluate language features with formal semantics and type systems
- **Analysis** – evaluate existing programs for correctness or other properties
- **Implementation** – build scalable systems, work on real-world code