runtime verification

Language-agnostic Program Verification

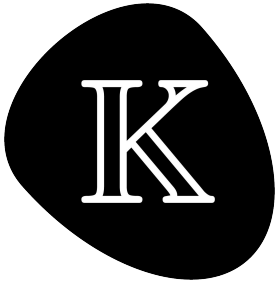
WG3 Kick-off Meeting

Juan Conejero Rodríguez

Runtime Verification Inc.

runtime verification

- Software quality assurance company
- Uses formal methods to perform security audits
- Team behind the maintenance of the K framework



The K framework

runtime verification

➔ **Mission**

Accessible trustworthy computing

➔ **Vision**

Generate implementations and tools automatically from formal specifications, correct by construction

➔ **K**

Semantic framework for design, implementation and formal reasoning

runtime verification

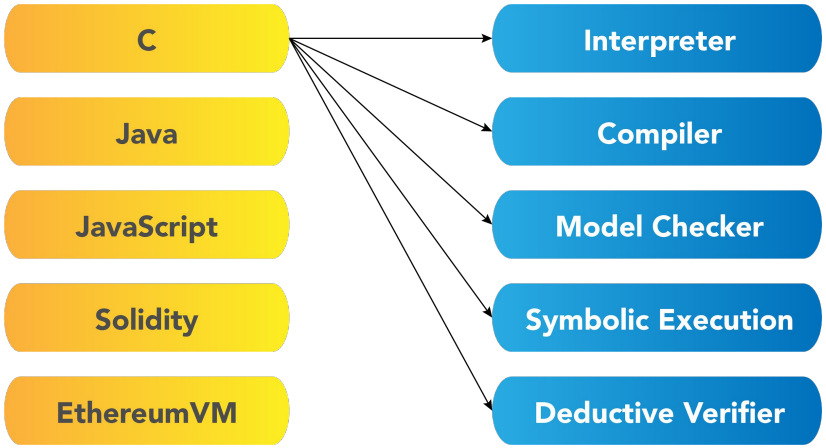
K Vision

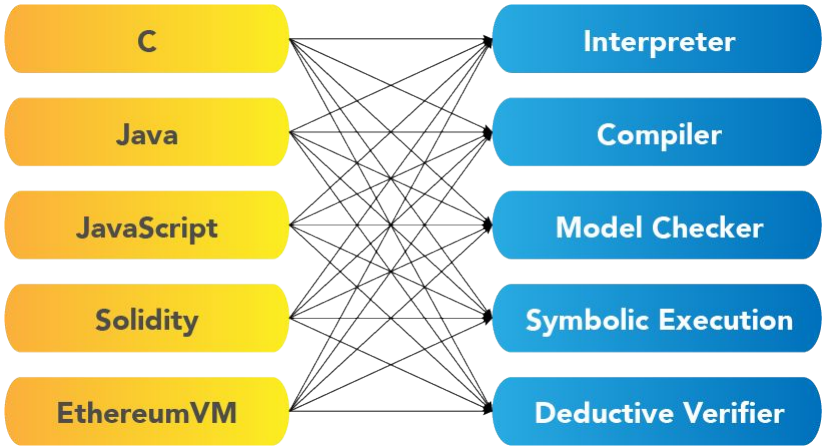
Semantics based tooling

The Problem: Crafting Many Tools

runtime verification

- How many tools is too many tools?



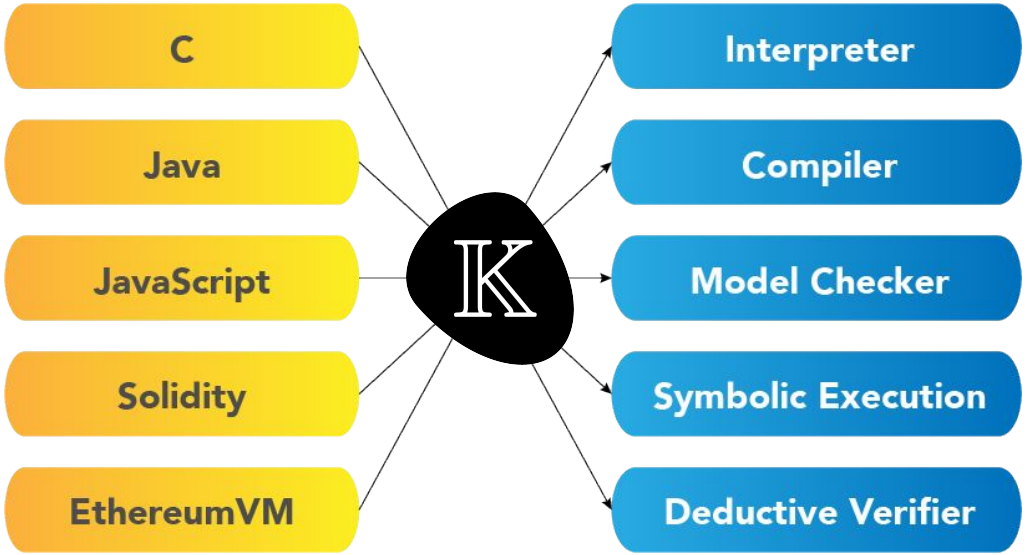


- Quite a bit of repeated effort.

The K Approach

runtime verification

- Develop each language and each tool once:



- Save on the implementation effort!
- Updates to tools benefit *all* the languages!

What is K?

runtime verification

- K is a *formal semantics framework*.
  - Specify your language or system in the K modelling language.
  - K derives a bunch of tools for you from this specification.
  - Notice the interpreter: executable semantics.

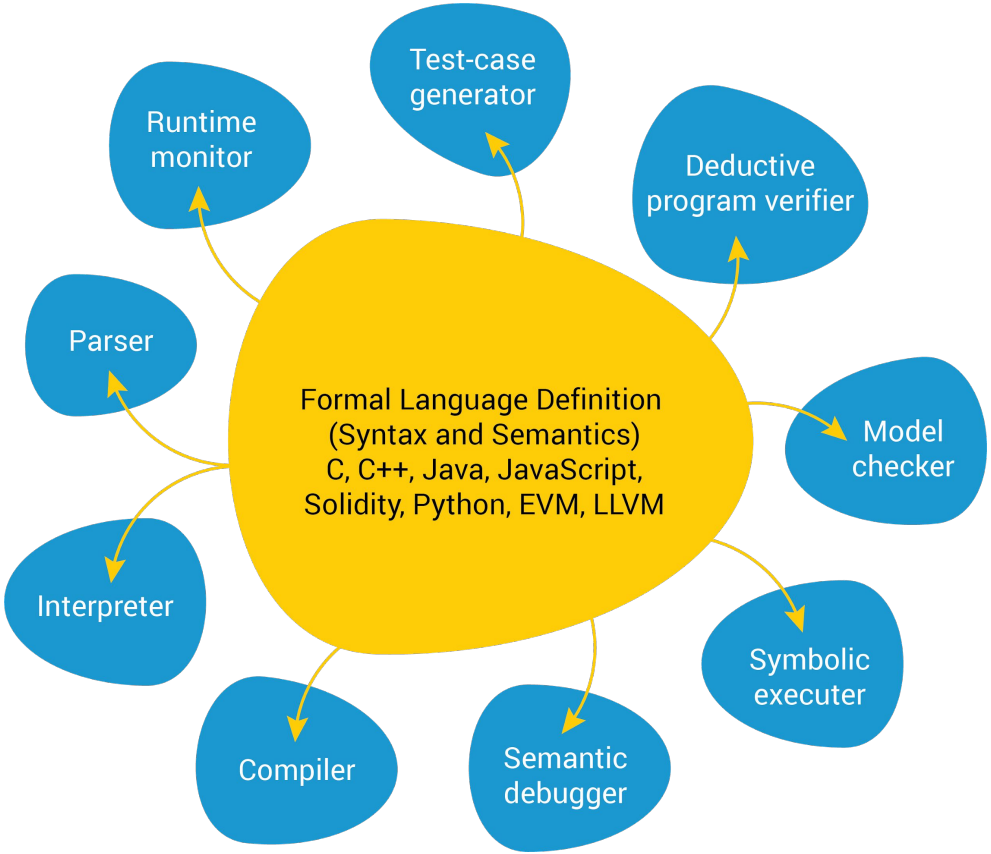
- K foundation: Matching Logic.
  - FOL + μ variant for specifying and reasoning about programs & PL.
  - Generalizes several important logical frameworks.
  - Specifications as rewrite rules.

- All of the tools built by RV are powered by K. Smart contract verification offered by RV is done with K.

- Webpages: https://kframework.org http://www.matching-logic.org/ https://runtimeverification.com/ https://fsl.cs.illinois.edu/

K Vision in the Real World

runtime verification



K in (the) Action

runtime verification

- K as a frontend
  - ➢ Concise and powerful language to express semantics
  - ➢ Write semantics in K, prove properties in an ITP
  - ➢ This would allow to prove deeper properties of PL

- K as a backend
  - ➢ Use K as a solver for ITP
  - ➢ Possible downside: there are more efficient solvers out there

- Back and forth between K and an ITP
  - ➢ Specify semantics and properties in K
  - ➢ Produce Dedukti proofs from K proofs
  - ➢ Double check in Dedukti every K proof
  - ➢ Send to an ITP the statements that K cannot prove

Our perspective of the Action

runtime verification

- Make ITPs capable of meeting industry standards
  - ➢ Ease of use
  - ➢ Wide range of applicable scenarios
  - ➢ Plenty of tools available

- K can deliver important key stones in this direction
  - ➢ Import semantics of already formalized languages in K
  - ➢ Ease the definition of future semantic definitions in Dedukti & ITPs
  - ➢ Strengthen K by making an ITP available for hard statements

Previous attempts to formalize K

runtime verification

- Maude (1st version of K)
- Isabelle
- Coq

Final version: Matching logic

Bring both proof systems together

runtime verification

$$\frac{}{\vdash_{\Sigma,\mathcal{R}} [\,] \text{ well-formed}}\ \text{(empty)}$$

$$\frac{\Gamma \vdash_{\Sigma,\mathcal{R}} A : s}{\vdash_{\Sigma,\mathcal{R}} \Gamma, x : A \text{ well-formed}}\ \text{(decl)}$$

$$\frac{\vdash_{\Sigma,\mathcal{R}} \Gamma \text{ well-formed}}{\Gamma \vdash_{\Sigma,\mathcal{R}} \texttt{TYPE} : \texttt{KIND}}\ \text{(sort)}$$

$$\frac{\vdash_{\Sigma,\mathcal{R}} \Gamma \text{ well-formed} \quad \vdash_{\Sigma,\mathcal{R}} A : s}{\Gamma \vdash_{\Sigma,\mathcal{R}} c : A}\ \text{(const)} \quad c : A \in \Sigma$$

$$\frac{\vdash_{\Sigma,\mathcal{R}} \Gamma \text{ well-formed}}{\Gamma \vdash_{\Sigma,\mathcal{R}} x : A}\ \text{(var)} \quad x : A \in \Gamma$$

$$\frac{\Gamma \vdash_{\Sigma,\mathcal{R}} A : \texttt{TYPE} \quad \Gamma, x : A \vdash_{\Sigma,\mathcal{R}} B : s}{\Gamma \vdash_{\Sigma,\mathcal{R}} \Pi x : A, B : s}\ \text{(prod)}$$

$$\frac{\Gamma \vdash_{\Sigma,\mathcal{R}} A : \texttt{TYPE} \quad \Gamma, x : A \vdash_{\Sigma,\mathcal{R}} B : s \quad \Gamma, x : A \vdash_{\Sigma,\mathcal{R}} t : B}{\Gamma \vdash_{\Sigma,\mathcal{R}} \lambda x : A, t : \Pi x : A, B}\ \text{(abs)}$$

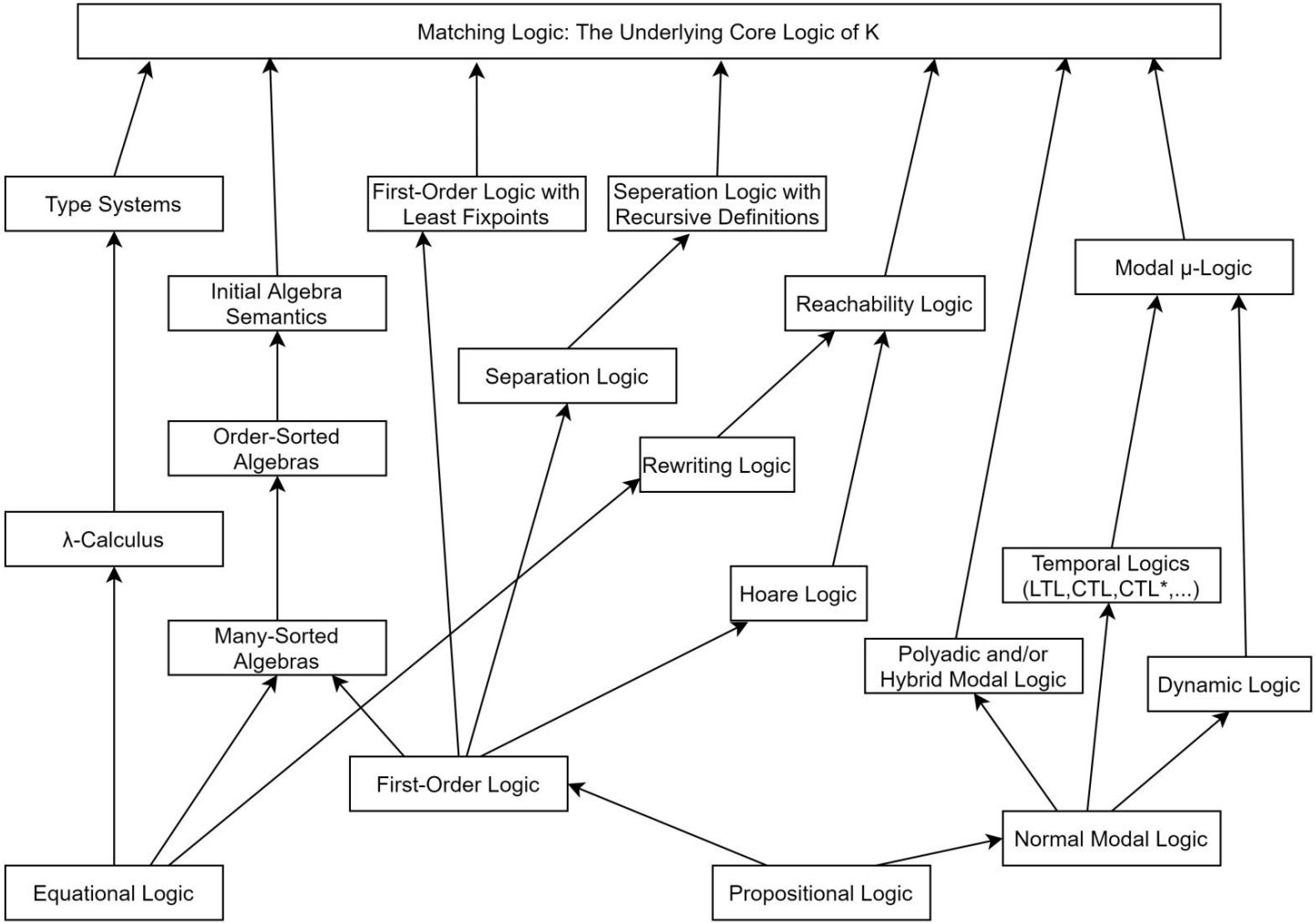
$$\frac{\Gamma \vdash_{\Sigma,\mathcal{R}} t : \Pi x : A, B \quad \Gamma \vdash_{\Sigma,\mathcal{R}} u : A}{\Gamma \vdash_{\Sigma,\mathcal{R}} t\ u : (u/x)B}\ \text{(app)}$$

$$\frac{\Gamma \vdash_{\Sigma,\mathcal{R}} t : A \quad \Gamma \vdash_{\Sigma,\mathcal{R}} B : s}{\Gamma \vdash_{\Sigma,\mathcal{R}} t : B}\ \text{(conv)} \quad A \equiv_{\beta\mathcal{R}} B$$

FOL Rules	(Propositional 1)	$\varphi \to (\psi \to \varphi)$
	(Propositional 2)	$(\varphi \to (\psi \to \theta)) \to ((\varphi \to \psi) \to (\varphi \to \theta))$
	(Propositional 3)	$((\varphi \to \bot) \to \bot) \to \varphi$
	(Modus Ponens)	$\dfrac{\varphi \quad \varphi \to \psi}{\psi}$
	($\exists$-Quantifier)	$\varphi[y/x] \to \exists x.\, \varphi$
	($\exists$-Generalization)	$\dfrac{\varphi \to \psi}{(\exists x.\, \varphi) \to \psi}\ x \notin FV(\psi)$
Frame Rules	(Propagation$_\bot$)	$C[\bot] \to \bot$
	(Propagation$_\vee$)	$C[\varphi \vee \psi] \to C[\varphi] \vee C[\psi]$
	(Propagation$_\exists$)	$C[\exists x.\, \varphi] \to \exists x.\, C[\varphi]$ with $x \notin FV(C)$
	(Framing)	$\dfrac{\varphi \to \psi}{C[\varphi] \to C[\psi]}$
Fixpoint Rules	(Substitution)	$\dfrac{\varphi}{\varphi[\psi/X]}$
	(Prefixpoint)	$\varphi[(\mu X.\, \varphi)/X] \to \mu X.\, \varphi$
	(Knaster-Tarski)	$\dfrac{\varphi[\psi/X] \to \psi}{(\mu X.\, \varphi) \to \psi}$
Technical Rules	(Existence)	$\exists x.\, x$
	(Singleton)	$\neg(C_1[x \wedge \varphi] \wedge C_2[x \wedge \neg\varphi])$

Kuestions?

runtime verification


Matching Logic: The Underlying Core Logic of K
Type Systems
First-Order Logic with Least Fixpoints
Seperation Logic with Recursive Definitions
Modal μ-Logic
Initial Algebra Semantics
Reachability Logic
Separation Logic
Order-Sorted Algebras
Rewriting Logic
λ-Calculus
Temporal Logics (LTL,CTL,CTL*,...)
Hoare Logic
Many-Sorted Algebras
Polyadic and/or Hybrid Modal Logic
Dynamic Logic
First-Order Logic
Normal Modal Logic
Equational Logic
Propositional Logic