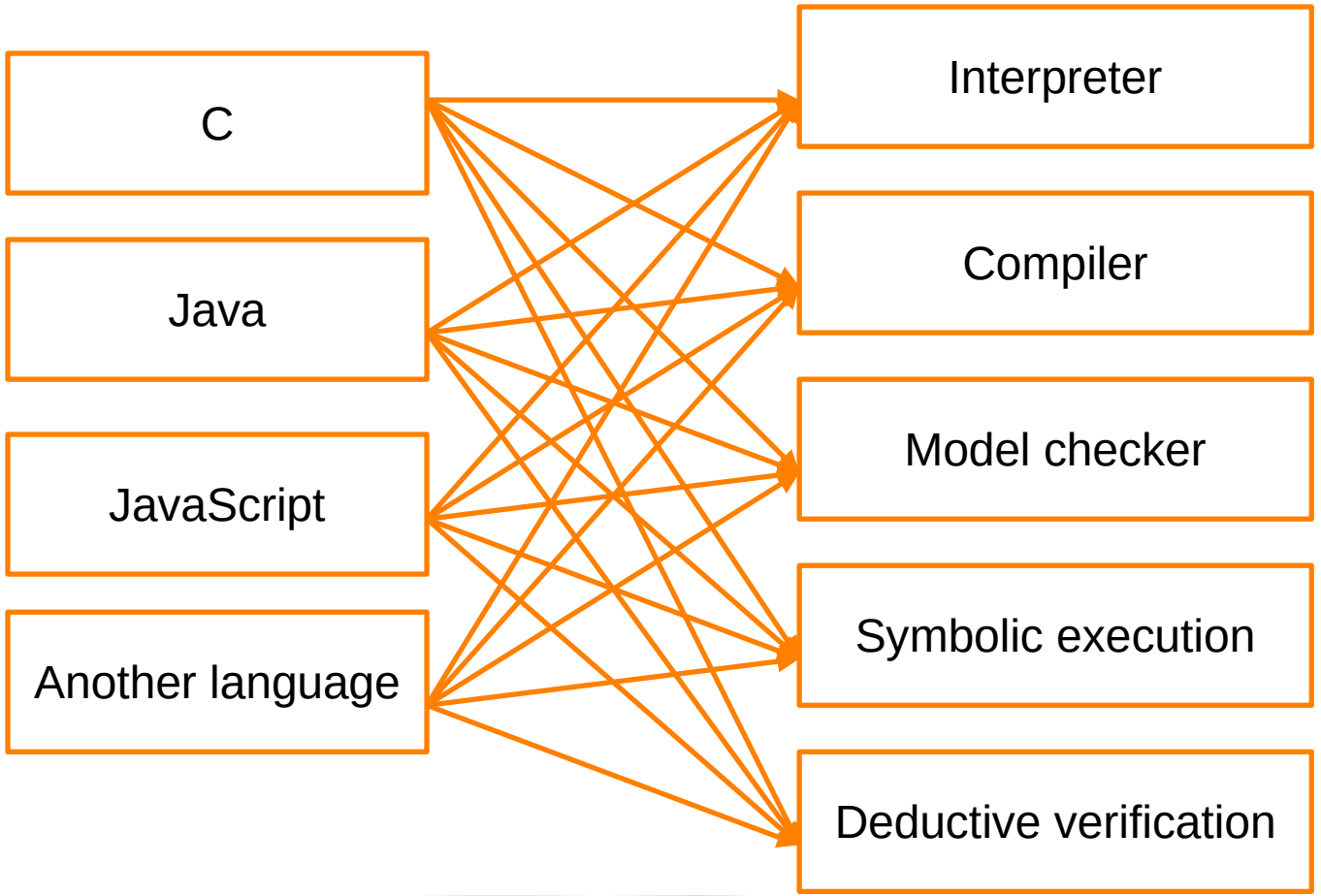
Formal Operational Semantics in Practice
(K-framework and its industrial applications)

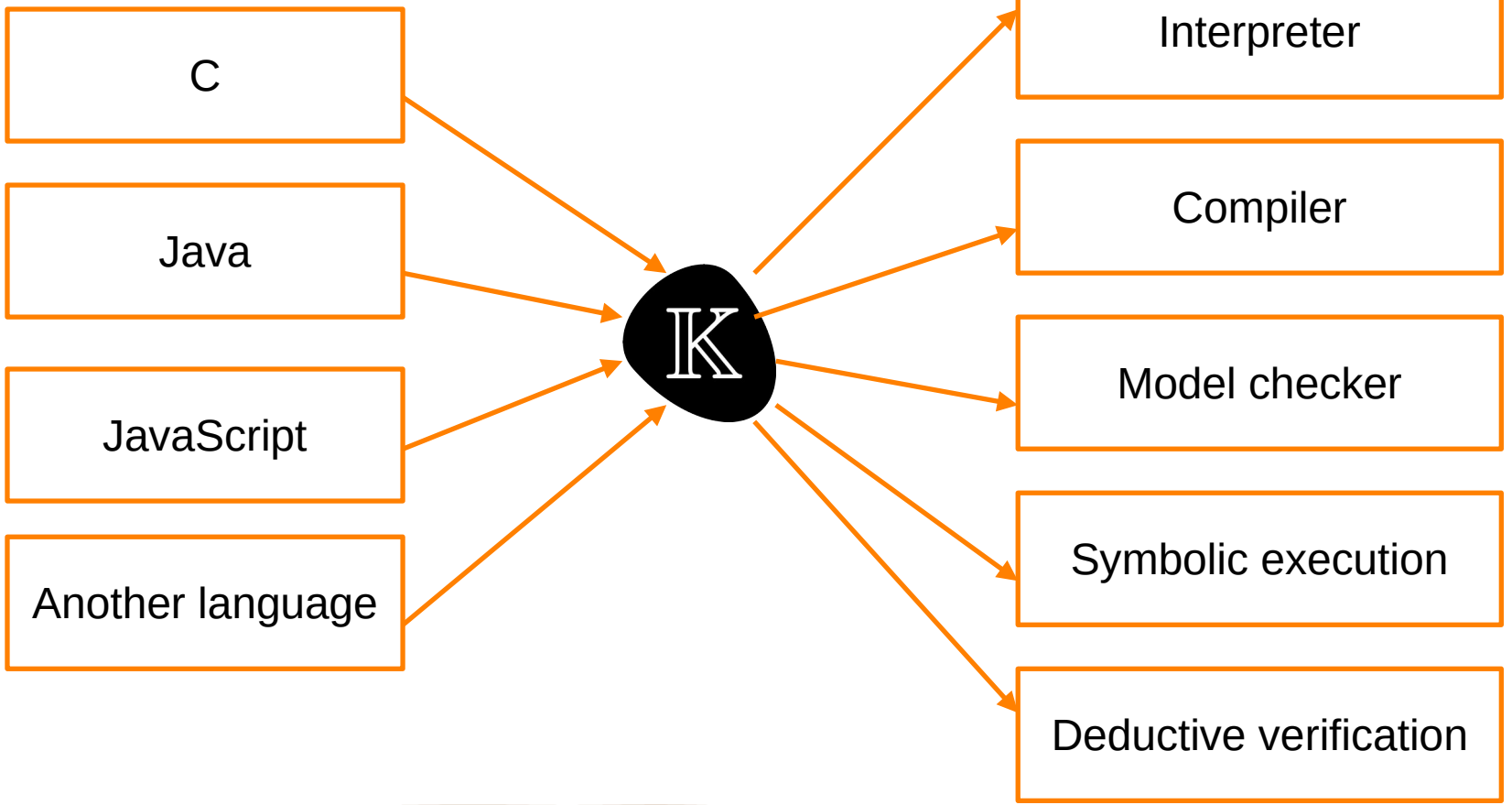
Formal verification is not as formal as one might think

- We use formal methods to prove properties of programs written in languages that are not formally defined.
- Different compilers often has different views on language standard.
- Frama-C has its own understanding of C standard and in some places it is not the same as of GCC, for instance.
- Clang and GCC often disagree between themselves about user program behavior.

A lot of efforts to support tools for each language



Unified approach - K



What is K?

- First of all - it is a formalism for defining an operational semantics
- K-framework is an operational semantics framework
- Based on Matching Logic
- Ideas came from rewriting logic (mostly from Maude)

- Given a K specification user will get:
  - Fast interpreter
  - Symbolic execution engine
  - Model checker
  - Verification engine

Holy Grail

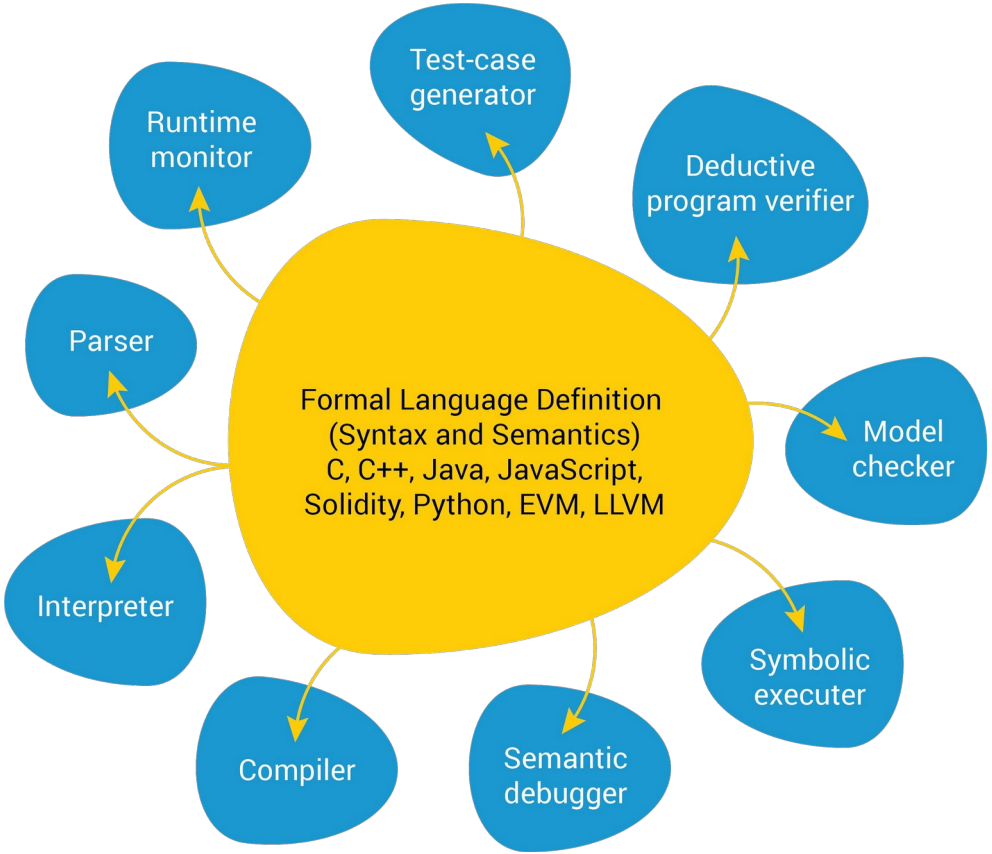
- I think that it won’t be very far from truth if I say that almost every programmer wants to implement his own ideal programming language.
- If a programmer did not even try to implement his own programming language he or she cannot be considered as a true programmer :)
- And a search for ideal programming language lasted and lasted
- And finally it was found! :)

A dream come true

- Implement your own programming language in 30 minutes
- Easiest syntax definition you've ever seen
- Intuitive way of semantics definition (small step structural operational semantics)
- A bunch of tools out of the box

- Tools for free:
  - Fast llvm-based interpreter – start program in your own language instantly
  - Symbolic execution engine – explore properties of programs in your language
  - Prover – prove claims about programs in your own language

The K vision


Test-case generator
Runtime monitor
Deductive program verifier
Parser
Formal Language Definition
(Syntax and Semantics)
C, C++, Java, JavaScript,
Solidity, Python, EVM, LLVM
Model checker
Interpreter
Symbolic executer
Compiler
Semantic debugger

Some history

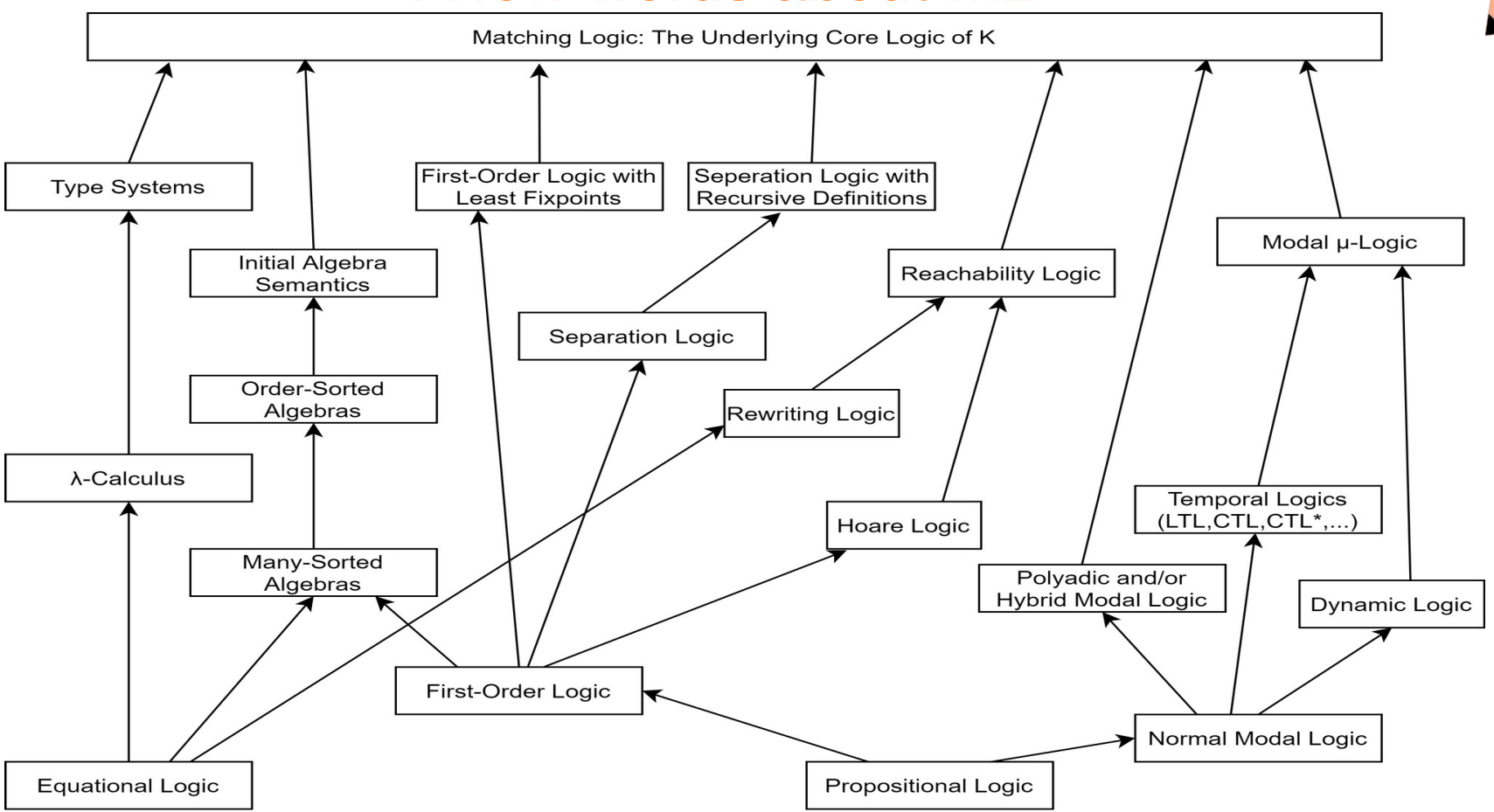
- The inventor and main ideologist is Grigore Rosu, he is an USA scientist
- Almost 20 years ago the story had begun
- Initial implementation of K was in Maude and was done without clear understanding of its theoretical grounds
- Then a long work was started that had led to a new family of logic: Matching Logic and Reachability Logic
- Personal page: https://fsl.cs.illinois.edu/people/grigore-rosu/

K has rigorous foundations

- K is a formalism based on Matching Logic
- Both are mechanized in MetaMath
- There is a ML proof-checker and ITP
- Many important proofs are mechanized

- There are a lot of publications about K, ML, etc
- There is a site, solely dedicated to Matching Logic

A few words about ML



K in practice

- A lot of industrial projects are bases on K-framework:
  - Tools for WASM
  - EVM formalization and toolbox
  - C language semantics and various tools (RV-match, RV)
  - X86_64 ISA formalization
  - Boogie semantics
  - Ocaml semantics
  - and many others
- There are a lot of blog posts related to K usage
- There is a nice website with a bunch of K tutorials, lessons, and videos

A one of many use-cases and success stories

- Formalization of Etherium Virtual Machine
  - Full support of EVM functionality
  - Proving Etherium byte-code program properties
  - Symbolic execution with abstracted environment, abstract state-space exploration
  - Tight integration with Etherium tools, KEVM may be a drop-in replacement of EVM
- KEVM helps to verify a set of complex smart-contracts of Maker DAO
- KEVM in the core of Firefly – a tool for model-checking properties of protocols of smart-contracts, expressed as LTL formulae

K-framework links

- https://kframework.org/ - tutorials, documentation, etc
- https://runtimeverification.com/blog/from-0-to-k-tutorial - a good introduction into K from simple examples to proofs
- https://github.com/kframework - repo with a lot of open-sourced semantics, tools, etc
- https://github.com/kframework/k-exercises
- https://fsl.cs.illinois.edu/publications/rosu-2017-marktoberdorf.pdf
  - introduction into K by examples
- https://runtimeverification.com/blog/category/k - blog posts about K-framework and its applications

Matching Logic, etc links

- http://www.matching-logic.org/ - main website
- https://fsl.cs.illinois.edu/people/grigore-rosu/grigore-rosu-publications.html
  - publications
- https://fsl.cs.illinois.edu/publications/chen-lucanu-rosu-2020-trb.pdf
  - highly recommended intro into ML
- https://fsl.cs.illinois.edu/publications/moore-pena-rosu-2018-esop.pdf
  - method of program verification in ML

Matching Logic, etc links

- https://fsl.cs.illinois.edu/publications/stefanescu-park-yuwen-li-rosu-2016-oopsla.pdf
  - how to achieve language-independent formal verification
- https://fsl.cs.illinois.edu/publications/rosu-2016-rv.pdf - Finite trace LTL