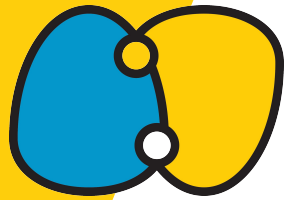
runtime verification



K IDE

via Language Server Protocol

Radu Mereuta

Runtime Verification Inc

Overview

runtime verification

- Motivation
- Proposal
- Features
- Workplan

Motivation

runtime verification

- Speed up the onboarding process for the curious K developers, and help expand the K community
- K has been around for more than a decade and used for:
  - Teaching
  - Defining real world programming languages
- IDE benefits:
  - Beginners - for softening the learning curve
  - Experienced - semantics developers with navigation tools and rapid feedback

Motivation

runtime verification

Thread # k-public

David Brandt APP Nov 9th at 11:43

4. What are the ecosystem benefits of using K to describe a language, as opposed to rolling ones own. For example, would doing so lead to having a VS Code code-completion plugin out-of-the-box for the described language?

1 reply

Also sent to the channel

bruce.collie 12 days ago

We currently generate the following core tools automatically from a K definition of a given language (plus a few others):

- Parser
- Fast concrete interpreter
- Symbolic execution engine
- Formal deductive verification system / proofs

We're working on implementing a semantic debugger at the moment; the ecosystem benefit of using K is that you can generate all of these things from the **same** source definition. We don't currently support code-completion, but it would *hypothetically* be possible to do so using K.

Writing a fast interpreter or rolling your own proof engine is hard if you're trying to do it ad-hoc for your own language! Using K means that you can just focus on the semantics of your language, and leave the boilerplate up to us.

Motivation

runtime verification

File Edit Selection View Go Run Terminal Help

EXTENSIONS: MARKE...

k

K 2K 5
K language support for VSC
Lucian Cumpata

Jupyter Ke... 43.7M 3.5
Jupyter keymaps for noteb...
Microsoft Install

Kubernetes 2.7M 4.5
Develop, deploy and debu...
Microsoft Install

test.k | domains.md | Extension: K

K v0.1.5
Lucian Cumpata | 2,277 | ★★★★★(1)
K language support for VSC
This extension is deprecated. Use the K framework extension instead.

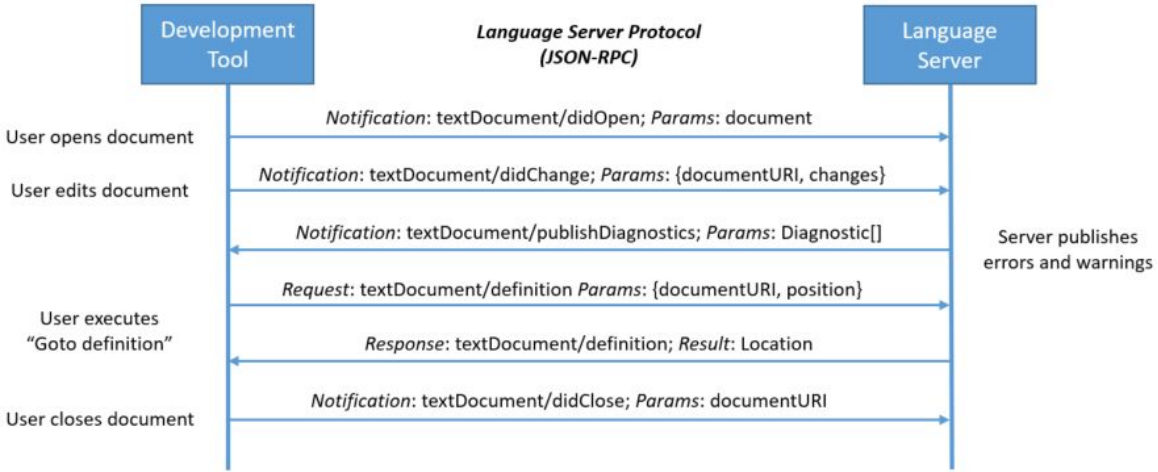
DETAILS FEATURE CONTRIBUTIONS

K for Visual Studio Code

2277 downloads since 2017

Proposal

- Add K framework IDE support via Language Server Protocol
- LSP is an open protocol between IDEs and a language sever
- The **language server** (what we need to implement)
  - Supported by multiple IDEs (VSC, IntelliJ, NeoVim, emacs)
  - It is for IDEs what K is for programming languages



Features - syntax highlighting

test-pr.yml | action.yml | imp.k M

k-distribution > pl-tutorial > 1_k > 2_imp > lesson_3 > imp.k

```
// Copyright (c) 2014-2019 K Team. All Rights Reserved.
require "domains.md"

module IMP-SYNTAX
  imports DOMAINS-SYNTAX
  syntax AExp  ::= Int | Id
                 | "-" Int
                 | AExp "/" AExp              [left, strict]
                 | "(" AExp ")"               [bracket]
                 > AExp "+" AExp              [left, strict]
  configuration <T color="yellow">
                  <k color="green"> $PGM:Pgm </k>
                  <state color="red"> .Map </state>
                </T>
```

Visual Studio Code syntax highlighting added by PumpkinDemo and polished by Virgil, Radu.
Based on regex

Features - on text error reporting

runtime verification

test.k 1 | kast.md | domains.md

test.k

```
4
5   module TEST
6     imports TEST-SYNTAX
7     imports INT
8     configurationn <k> $PGM:Int </k>
9     syntax Exp ::= affunction(Int)
10
```

PROBLEMS 1 | OUTPUT | DEBUG CONSOLE | TERMINAL

test.k 1

Encountered <LOWER_ID>. Outer Parser [Ln 8, Col 3]

Was expecting one of: ["rule", "context", "configuration", "claim", "syntax", "endmodule", "imports", "import"]

Features - goto definition

```
module TEST
  imports TEST-SYNTAX
  imports INT
  configuration <k> $PGM:Int </k>
  syntax Exp ::= affunction(Int)

  rule affunctio  Int "+Int" Int                    [function,
                  latex({#1}\mathrel{+_{\scriptstyle\it Int}

  rule <k> 1 => 2:Int +Int 3 ...</k>
endmodule
```

Ctrl+Click - goto definition

```
1052        Int "+Int" Int                [function, total, klabel(_+Int_), symbol, left,
domains.md ~/work/k/k-distribution/target/release/k/include/kframework/builtin - References (31)

1152
1153  module INT-SYMBOLIC [symbolic]
1154    imports INT-COMMON
1155    imports INT-SYMBOLIC-KORE
1156    imports private BOOL
1157
1158    // Arithmetic Normalization
1159    rule I +Int 0 => I [simplification]
1160    rule I -Int 0 => I [simplification]
1161
1162    rule X modInt N => X requires 0 <=Int X andBool X <Int N [simpl
1163    rule X   %Int N => X requires 0 <=Int X andBool X <Int N [simpl
1164
1165    // Bit-shifts
1166    rule X <<Int 0 => X [simplification]

I1 -Int (I2 +Int C) => (I1 -Int I2) -Int C [cor
Int C) => (I1 -Int I2) +Int C [concrete(I1, I
C -Int (I2 +Int I3) [concrete(I2, I3), symb
Int) => (bitRangeInt(I, IDX, LEN) +Int (1
((I1 %Int absInt(I2)) +Int absInt(I2)) %Int
String) => 1 +Int countAllOccurrences(s
Source, findString(Source, ToCount, 0)
Source, findString(Source, ToReplace, 0
Source, findString(Source, ToReplace, 0
substrString(S,findChar(S, Delimiters, 0
substrString(S,findChar(S, Delimiters, 0
test.k                                  1
k> 1 => 2:Int +Int 3 ...</k>
```

Shift+F12 - find occurrences

Features - run program step by step

runtime verification

kframework.kore.json | ieee754.k | imp.k | def

home > radu > work > k > k-distribution > tests > regression-new > imp-llvm > imp.k

```
62
63    rule if (true)  S else _ => S
64    rule if (false) _ else S => S
65
66    rule while (B) S => if (B) {S while (B) S} else {
67
68    rule <k> int (X,Xs => Xs);_ </k> <state> Rho:Map
69      requires notBool (X in keys(Rho))
70    rule int .Ids; S => S  [structural]
71  endmodule
72
```

sum.imp

home > radu > work > k > k-distribution > tests > regres

```
1   // This program calculates in sum
2   // the sum of numbers from 1 to n
3
4   int n, sum;
5   n = 100;
6   sum = 0;
7   while (!(n <= 0)) {
8     sum = sum + n;
9     n = n + -1;
10  }
11
12  // sum should be 5050 when n is 1
13
```

VSC - *not actually implemented

Feature list

runtime verification

- Phase 1 - LSP
  - Syntax highlighting (done: PumpkinDemo, Virgil, Radu)
  - Go to definition and find usages *(done)*
  - Highlight error messages on code - **as you type** *(partially done)*
  - Code completion and hints
- Phase 2 - Debugger
  - Run step by step through a program
    - Split view: highlight rule matched, highlight part of the program
    - Highlight differences in the configuration, hide unchanged parts
- Phase 3 - Prover
  - Still thinking
    - Visualizer for the prover

Workplan - Phase 1 (kompile)

runtime verification

- Implement the Language Server skeleton and flush out the architecture
- Add syntax highlighting
- Integrate error reporting directly in text (for syntax errors)
- Implement code completion and hints
  - Based on the K AST
- Reparse rules as you type
- Code folding
- Navigation
  - Goto definition and find occurrences
  - Goto implementation

Workplan - Phase 2 (krun)

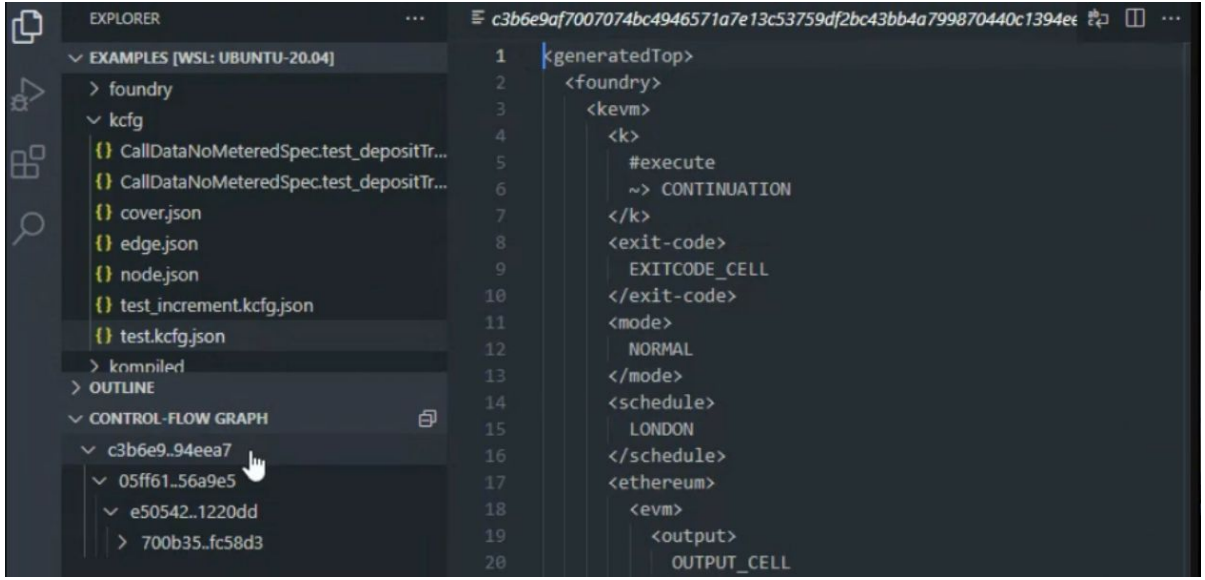
runtime verification

- Visual Debugger
  - With the Debug Adapter Protocol
  - Integrate with GDB - because it’s simple.
  - Pyk library
  - Split view - highlight program location and rule being applied
  - Custom view for the configuration
    - Hide unchanged parts and highlight differences (diff)
    - Remember which cells I folded last time (asked by Raoul and Andrei)
    - Modify the configuration and continue?

Workplan - Phase 3 (kprove)

runtime verification

- Visual Prover
  - Integrate with pyk *(in development)*
  - Build a kcfg visualizer
  - Integrate with the configuration custom view



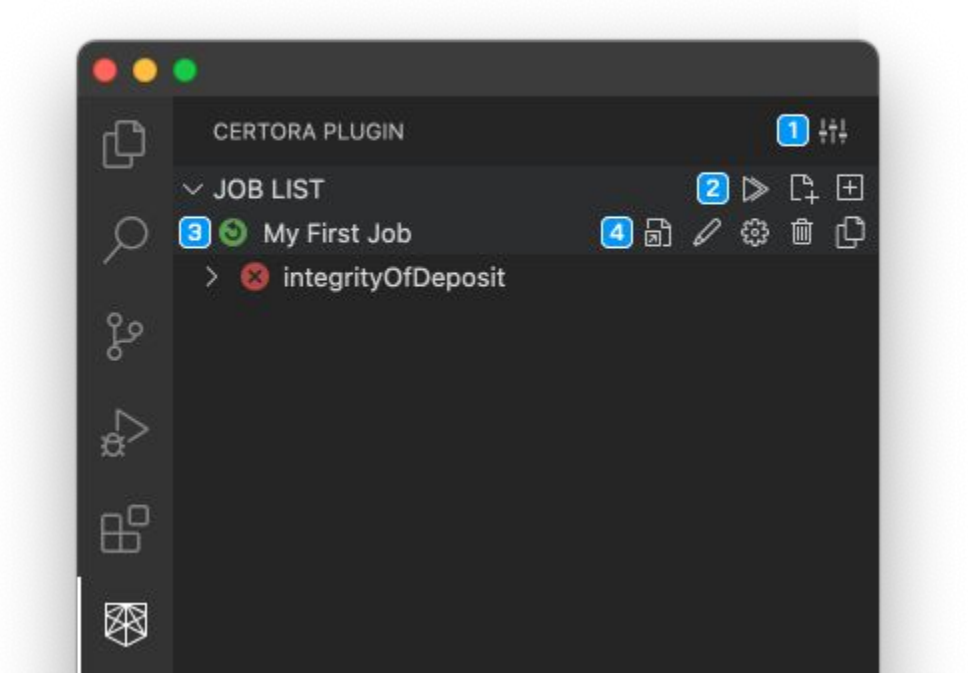
Kcfg view made by Raoul

Other ideas

runtime verification

- Language specific IDE
  - Alongside the K semantics we can offer IDE integration based on K
    - Syntax highlighting - easy manual work
    - Debugger - customized view of the K debugger
    - Prover - customized views
      - EVM bytecode decompilation and source tracking
    - Extra features
- Web integration
  - VSCode can run in a browser

Competition - Certora VSCode



More Info

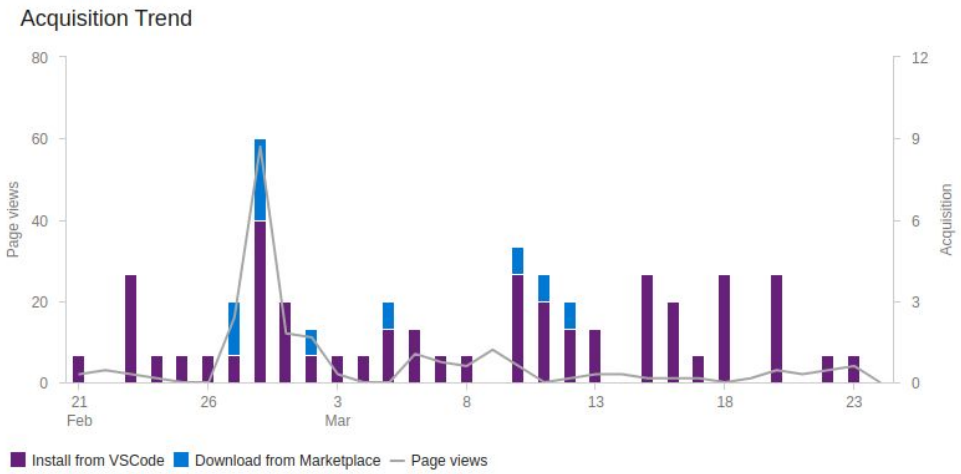
Version	0.1.2
Released on	1/30/2022, 8:39:10 PM
Last updated	1/25/2023, 4:47:59 PM
Publisher	Certora
Unique Identifier	Certora.vscode-certora-prover

~60 downloads in 2 weeks

https://marketplace.visualstudio.com/items?itemName=Certora.vscode-certora-prover

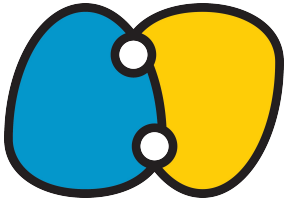
Acquisition

- ~100 downloads in 2 months
- No promotion



https://marketplace.visualstudio.com/items?itemName=RuntimeVerification.k-vscode
https://open-vsx.org/extension/RuntimeVerification/k-vscode

runtime verification



Thanks for the suggestions

https://github.com/runtimeverification/k-editor-support