Joint project between the FSL group at UIUC (USA) and the FMSE group at UAIC (Romania)
K Team

UIUC, USA

–  Grigore Rosu (started K in 2003)
–  Cansu Erdogan
–  Dwight Guth
–  David Lazar
–  Patrick Meredith
–  Andrei Stefanescu

Former members

–  Kyle Blocher
–  Peter Dinges
–  Chucky Ellison
–  Mike Ilseman
–  Traian Serbanuta

UAIC, Iasi, Romania

–  Dorel Lucanu
–  Traian Serbanuta
–  Andrei Arusoae
–  Denis Bogdanas
–  Stefan Ciobaca
–  Gheorghe Grigoras
–  Radu Mereuta
–  Raluca Necula
–  Emilian Necula

Former Members

–  Irina Asavoae
–  Mihai Asavoae
Current State-of-the-Art in PL Design, Implementation and Analysis

Consider some programming language, L

• **Formal semantics of L?**
  – Typically skipped: considered expensive and useless

• **Implementations for L**
  – Based on some adhoc understanding of what L is

• **Model checkers for L**
  – Based on some adhoc encodings/models of L

• **Program verifiers for L**
  – Based on some other adhoc encodings/models of L

• ...
Example of C Program

• What should the following program evaluate to?

```c
int main(void) {
    int x = 0;
    return (x = 1) + (x = 2);
}
```

• According to the C “standard”, it is **undefined**

• GCC4, MSVC: it returns 4
  GCC3, ICC, Clang: it returns 3
By April 2011, both Frama-C (with its Jessie verification plugin) and Havoc "prove" it returns 4
A Formal Semantics Manifesto

• Programming languages must have formal semantics! (period)
  – And analysis/verification tools should build on them
    • Otherwise they are adhoc and likely wrong

• Informal manuals are not sufficient
  – Manuals typically have a formal syntax of the language (in an appendix)
  – Why not a formal semantics appendix as well?
Motivation and Goal

We want a semantic framework which makes it easy and fun to define programming languages, no matter how complex or large they are!
The K Framework

k-framework.org

A tool-supported rewrite-based framework for defining programming language design and semantics.
Complete K Definition of KernelC
Complete K Definition of KernelC

Syntax declared using annotated BNF

\[ \text{SYNTAX } \quad \text{Exp} ::= \quad \ldots \quad \text{Exp} = \text{Exp} \ [\text{strict}(2)] \]
Complete K Definition of KernelC

Configuration given as a nested cell structure.
Leaves can be sets, multisets, lists, maps, or syntax

T

k

funs

env

mem

ptr

next

rand

in

out


Complete K Definition of KernelC

Semantic rules given contextually

\[ \text{<k>} X = V \Rightarrow V \ldots \text{<k>}\]

\[ \text{<env>} \ldots X \text{ } \vdash \text{ } (_\uparrow \Rightarrow V) \ldots \text{<env>\]} \]
What does the K Tool Offer?

- Efficient and interactive execution (interpreters)
- State-space exploration (search and model-checking)
- Deductive program verification (in progress)
K Scales

Besides smaller and paradigmatic teaching languages, several larger languages were defined:

• Java 1.4 : by Chen [CAV’06]
• Verilog : by Meredith&Katelman [MEMOCODE’10]
• C : by Chucky Ellison [POPL’12]
etc.
The K Configuration of C

Heap

75 Cells!
Statistics for the C definition

• Total number of rules: ~1200

• Tested on thousands of C programs (several benchmarks, including the gcc torture test, code from the obfuscated C competition, etc.)
  – Passed 99.2% so far!
  – GCC 4.1.2 passes 99%, ICC 99.4%, Clang 98.3% (no opt.)

• The most complete formal C semantics
Next Steps

• Watch the five-minute video demo
• Download the K tool
• Do the K Tutorial
• Define your own language!
• Watch the interview on rewrite-based semantics if you’d like to better understand our viewpoint
• Read select publications for a deeper understanding of K’s foundations