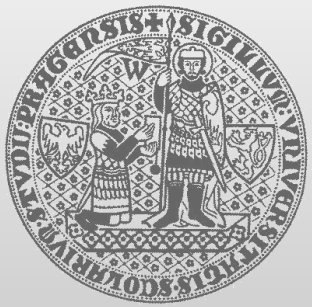
Specifying Semantics of Programming Languages in K Framework

http://d3s.mff.cuni.cz

David Hauzar

hauzar@d3s.mff.cuni.cz



CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Overview

k-framework
The K Semantic Framework

- University of Illinois at Urbana-Champaign
- Scheme, Verilog, Python, C, ...

c-semantics
Semantics of C in the K Framework

- An Executable Formal Semantics of C with Applications
- Chucky Ellison, Grigore Rosu
- POPL 2012

Weverca

- WEb VERifiCAtion for PHP
- GAUK project

Department of Distributed and Dependable Systems


k-framework
The K Semantic Framework

K Framework - Overview

- Executable
  - Interpreter, Model checker, Deductive reasoning
  - Static analyser
- Rewriting-style semantics
  - Uses Maude engine (developed since 80')
  - Use of OCAML (fast execution) and CoQ (verification) planned

http://k-framework.org/

Department of Distributed and Dependable Systems D3S

K Framework - Example

Original language syntax	K Strictness	K Semantics
Pgm ::= var List{Id}; Stmt		`<k>var xl:ListItem; s:Stmt => s</k>` `  <state>xl |-> 0</state>`

Stmt ::= Id = AExp
 	
| Stmt; Stmt
| if BExp
then Stmt else Stmt
| while BExp do Stmt
 	
| spawn Stmt
	[strict(2)]
	
	[strict(1)]
		`<k>x:Id = i:Int => . _</k>`
		`  <state>_ x |-> i _</state>`
		`s1:Stmt; s2:Stmt => s1 ~> s2`
		`if true then s1:Stmt else _ => s1`
		`if false then _ else s2:Stmt => s2`
		`while B:BExp do S:Stmt =>`
		`  if B then (S ; while B do S) else .`
		`<k>spawn s:Stmt => . _</k> . => <k>s</k>`

BExp ::= Bool		

AExp ::= Int
| Id
 	
| AExp + AExp
	
	[strict]
		`<k>x:Id => i _</k>`
		`  <state>_ x |-> i _</state>`
		`i1:Int + i1:Int => i1 +Int i2`

KResult ::= Int | Bool		

```
Configuration ≡    <k*>K</k*>
                   <state>Map{Id |-> Int}</state>
```

Department of Distributed and Dependable Systems D3S


c-semantics
Semantics of C in the K Framework

C Semantics

First complete formal semantics for C

Conforming

- Must accept all portable programs, but can also accept non portable programs

Freestanding

- All language features except comlex numbers
- Subset of the standard library

Extensively tested

- E. g. against GCC torture tests (1093 test programs, 776 standard compliant. Of those 770 passed)
- Better results than Clang or GCC

http://code.google.com/p/c-semantics/

Department of Distributed and Dependable Systems D3S

C Semantics – Features

	Definition							
Feature	GH	CCR	CR	No	Pa	BL	Le	ER
Bitfields	●	◐	○	○	◐	○	○	●
Enums	◐	●	○	○	●	○	○	●
Floats	○	○	○	○	◐	●	●	●
Struct/Union	●	●	●	◐	●	●	●	●
Struct as Value	○	○	○	●	○	○	○	●
Arithmetic	◐	●	●	○	●	●	●	●
Bitwise	○	●	○	○	●	●	●	●
Casts	◐	◐	○	◐	◐	●	●	●
Functions	●	●	◐	●	●	●	●	●
Exp. Side Effects	●	●	○	●	●	○	●	●
Variadic Funcs.	○	○	○	○	○	○	○	●
Eval. Strategies	○	◐	○	●	●	○	●	●
Concurrency	○	○	○	○	○	○	○	◐
Break/Continue	◐	●	◐	●	●	●	●	●
Goto	◐	○	○	○	●	○	●	●
Switch	◐	●	○	○	●	◐	◐	●
Longjmp	○	○	○	○	○	○	○	●
Malloc	○	○	○	○	○	○	○	●

●: Fully Described
◐: Partially Described
○: Not Described

GH denotes *Gurevich and Huggins* (1993),
CCR is *Cook, Cohen, and Redmond* (1994),
CR is *Cook and Subramanian* (1994),
No is *Norrish* (1998),
Pa is *Papaspyrou* (2001),
BL is *Blazy and Leroy* (2009),
Le is *Leroy* (unpublished, 2010), and
ER is *Ellison and Roşu* (our work).

Department of Distributed and Dependable Systems D3S

C Semantics – Implementation

- 75 cells
- 150 syntactic operators
- 5900 source lines of semantics
- 1200 rules
  - 80 rules for statements
  - 160 rules for expressions
  - 500 rules for declarations and types
  - 115 rules for standard library
  - …
- 6 person-months

Department of Distributed and Dependable Systems D3S

C Semantics – Existing Analysis / Verification tools

Existing C analysis/verification tools

- Lint/Purify/Coverity/Valgrind
- Blast
- Havoc
- Slam
- VCC
- …

Based on approximative models of C

- Hard to argue for the soundness of the tools

Department of Distributed and Dependable Systems D3S

C Semantics – Problems of Approximative models (1)

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

Undefined according to C standard

GCC4, MSVC returns 4
GCC3, ICC, Clang returns 3

Frama-C „proves“ it returns 4

Department of Distributed and Dependable Systems D3S

C Semantics – Problems of Approximative models (2)

```
int main(void) {
  "foo"[0] = 'x';
  return "foo"[0];
}
```

Undefined according to C standard

GCC4	doesn‘t compile
ICC, Clang	segmentation fault

Frama-C „proves“ it returns ‚x‘

C Semantics – Problems of Approximative models (3)

```
int r;

int f(int x) {
  return (r = x);
}

int main(void) {
  f(1) + f(2); return r;
}
```

Defined (Could return 1 or 2)

GCC, ICC, MSVC, Clang returns 2

Both Frama-C and Havoc „prove“ it can only return 2

C Semantics – Built tools

Use an explicit and testable definition to build tools that conform to this semantics

Semantics-Based Analysis Tools

- Interpreter
- State-space explorer
- LTL Model-checker
- Debugger
- Program verifier

Department of Distributed and Dependable Systems D3S

C Semantics – Interpreter

```
#include <string.h>
int main(void) {
  char dest[5], src[5] = "hello";
  strcpy(dest, src);
}
```

```
ERROR! KCC encountered an error
while executing this program.
Description: Reading outside the
bounds of an object.
File: buggy_strcpy.c
Function: strcpy
Line: 4
```

Department of Distributed and Dependable Systems D3S

C Semantics – State-space Search (1)

```
int denominator = 5;
int setDenominator(int d) {
  return denominator = d;
}
int main(void) {
  return setDenominator(0) + (7 / denominator);
}
```

```
2 solutions found
---------------------------------
Solution 1
Program got stuck
File: eval_order.c
Line: 8
Description: Division by 0.
---------------------------------
Solution 2
Program completed successfully
Return value: 1
```

Department of Distributed and Dependable Systems D3S

C Semantics – LTL-Based Model Checking

```
typedef enum {green, yellow, red} state;
state lightC = green; state lightW = red;
int changeC() {
  switch (lightC) {
    case(green): lightC = yellow; return 0;
    case(yellow): lightC = red; return 0;
    case(red):
      if (lightW == red) { lightC = green; } return 0;
}}
...
int main(void) { while(1) { changeC() + changeW(); } }
```

```
changeC, changeW => lightC=yel, lightW=red
changeW, changeC => lightC=red, ligh
changeC, changeW => lightC=gre, ligh
```

changeC(); changeW()

```
#pragma __ltl safety: [] (lightC == red \/ lightW == red)
#pragma __ltl progressC: [] <> (lightC == green)
```

Department of Distributed and Dependable Systems D3S

C Semantics – Deductive Verification

```
void listNode* reverse(struct listNode *x) {
  struct listNode *p;
  struct listNode *y;
  p = 0;
  while (x) {
    y = x->next;
    x->next = p;
    p = x;
    x = y;
  }
}
```

```
rule <k> $ => return p; </k>
  <heap>... list(x,A) => list(p,rev(A)) ...</heap>
```

$ is the body of the function

Department of Distributed and Dependable Systems D3S

Weverca

Weverca

- Security Analysis of PHP applications
  - Flow of sensitive data into critical commands

Weverca – Overview

Department of Distributed and Dependable Systems D3S

Weverca – Work in Progress

Summary

k-framework
The K Semantic Framework

- Defining semantics of programming languages

c-semantics
Semantics of C in the K Framework

- Semantics-based analysis tools

Weverca

- Static analyzer in K

Department of Distributed and Dependable Systems D3S