

Verification of "C" with

Results of the Diploma Thesis

Christoph Gladisch

Universität Koblenz-Landau

Goals

1. What are the major differences between C and JavaCard?
2. How can special features of C be formalized and handled by a calculus?
3. How can the proposed solutions be implemented into the KeY-system?

Results

1. Pointer operators and Deep copy assignments (and dynamic memory)
2. Full definition of a logic and calculus for the verification of C0
3. Implementation of a prototype

Investigation of ANSI C

The Specification of ANSI C ...

- ...gives room for interpretation and customization of a C implementation
- ...is hard to understand in detail

Investigation of MISRA C

- Restrictions on the construction of expressions (evaluation order, types)
- No direct or indirect recursive functions
- Single point of exist at the end of the function
- No dynamic memory (allocation, deallocation)
- Pointerarithmethic restricted to use for array indexing
- No goto, continue, comma-operator, unions

Investigation of MISRA C

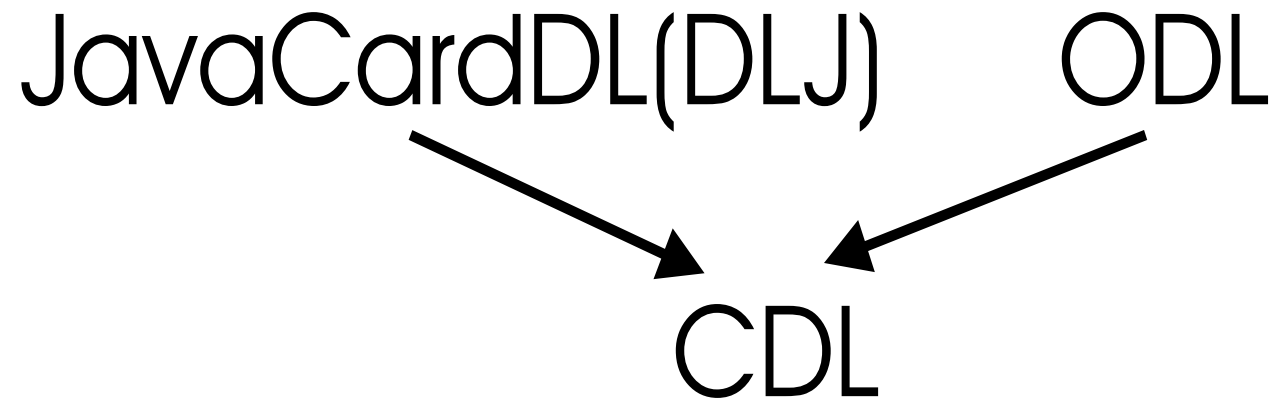
Open questions

- What is the explicit specification of MISRA C?
- Does MISRA C eliminate all ambiguities?
- Should a verification system for MISRA C rely on MISRA C conformance of a program or should this conformance be verified?

C0 Overview

- No side effects in expression, no ambiguities of evaluation
- Size of arrays is statically fixed
- No pointers to local memories
- Two's complement representation
- Strictly typed

CDL



Syntax, Semantic, Calculus and other issues for

- First-order Logic
- Dynamic Logic
- KeY-Updates
- Refinement of integer arithmetic for C0
- Handling of pointeroperators
- Deep copy updates
- C0 Expressions and Statements

Integer datatype refinement

$$X \text{ op } Y \rightsquigarrow \left(\begin{array}{l} \text{if } \mathit{in}_T(X) \wedge \mathit{in}_T(Y) \wedge \mathit{in}_T(X \text{ logOp } Y) \text{ then} \\ \quad X \text{ logOp } Y \\ \text{else if } \neg \mathit{in}_T(X) \vee \neg \mathit{in}_T(Y) \text{ then} \\ \quad X \text{ logOp } Y \\ \text{else overflow}(X, Y, 'o') \text{ fi fi} \end{array} \right)$$

Pointeroperators

Program	Logic
<code>a</code>	?
<code>&a</code>	?
<code>*a</code>	?
<code>a.m</code>	?
<code>&a.m</code>	?

Representation of program variables

	Mono	Dual	Accessor path
pointer a	a	$\&a$	$\&(a, \#)$
value a	$v a$	$v a$	$v(a, \#)$
pointer $a . m$	$\bullet(v a, m)$	$\&(\&a, m)$	$\&(a, \bullet(m, \#))$
value $a . m$	$v(\bullet(v a, m))$	$v(v a, m)$	$v(a, \bullet(m, \#))$
$a . m_1 . \dots . m_n$

Conversion of Expressions

$$X \rightsquigarrow \begin{cases} v(X), \text{ if } X : \text{simple local variable} \\ v(\text{obj}_T(n)), \text{ if } X : \text{simple global variable} \\ v(A, \text{aconv}(B)), \text{ if } X = A.B : \text{structure member} \\ v(A, I), \text{ if } X = A[I] \text{ is array access} \\ \&(A), \text{ if } X = \&A \\ *(A), \text{ if } X = *A \end{cases}$$

Deep copy assignments and aliasing

$a, b \in \mathbf{C}$ struct	$a, b \in \mathbf{C}$ struct pointer	$a, b \in \mathbf{Java}$ class
$b.m = d;$	$b \rightarrow m = d;$	$b.m = d;$
$a = b;$	$a = b;$	$a = b;$
$b.m = e;$	$b \rightarrow m = e;$	$b.m = e;$
$a.m \doteq d$	$a \rightarrow m \doteq e$	$a.m \doteq e$

Deep copy updates

Deep copy update

$$\{a := b\}$$

Unfolded quantified update

$$\text{for } x_1 \dots \text{for } x_n \ a.x_1 \dots x_n := b.x_1 \dots x_n$$

14 Rules for handling deep copy updates

Semantic of C0 Statements 1

$sp_l(l\text{expr}=r\text{expr})t \Leftrightarrow$

- $\text{val}_l(s, \text{isdef}(l\text{expr})) = \text{true}$
- $\text{val}_l(s, \text{isdef}(r\text{expr})) = \text{true}$
- $t = s \triangleleft \text{val}_l(s, \{l\text{expr} := r\text{expr}\})$

Semantic of C0 Statements 2

$sp_l(l\text{expr}=r\text{expr})t \Leftrightarrow$

if $\text{val}_l(s, \text{isdef}(l\text{expr})) = \text{true}$ and

$\text{val}_l(s, \text{isdef}(r\text{expr})) = \text{true}$ then

$t = s \triangleleft \text{val}_l(s, \{l\text{expr} := r\text{expr}\})$

otherwise

$t = s$ with $t.\text{err} = 1$

Implementation

Documentation of parts of the KeY-system

- .key-files
- Main application classes
- SourceElements
- Types, Sorts, Declarations
- Operators and Terms
- Update-related classes
- Meta constructs
- Typeconverter

Implementation

The Prototype

- JavaC
- Conversion of program variables to terms
- Unfolding
- Dynamic memory allocation and deallocation
- (Call by reference)

Conclusion

- Dynamic Logic and Calculus for verification of C0 is provided
- The thesis can be use as a reference for different concepts used in KeY
- A parser and an infrastructure for C is requiried in the KeY-system
- A formalization of MISRA C is required, CDL provides a good starting point