Compiled Code Verification – Survey and Prospects

Amitabha Sanyal

Department of Computer Science & Engineering

IIT Bombay

(Copyright ©2004 Amitabha Sanyal)
Acknowledgements

- Achyut Jagtap
- Aditya Kanade
- Abhijat Vichare
- Supratim Biswas
- Uday Khedker
So you thought compilers were always right?

- Programmers place a large amount of trust on the correctness of compilers. Is this trust justifiable?

  Yes, because

- Compiling algorithms are well studied. Have sound theoretical foundations.
- Large parts of compilers are automatically generated by tools which have become trustworthy by usage.

  However . . .
So you thought compilers were always right?

http://gcc.gnu.org/bugzilla/query.cgi
So you thought compilers were always right?

Bug #15549

```c
int lessthan (_Bool b, unsigned char c)
{
    return b < c;
}

int main ()
{
    if (!lessthan(1, 'a'))
        abort ();
}
```

**Problem:** \( b < c \) is compiled as \((b == 0) \& (c != 0)\).

From release of GCC 3.4.0 to 3.4.3, the number of open bugs increased from 1400 to 2115.
Increasing Trustworthiness through Compiler Verification
Increasing Trustworthiness through Compiler Verification

Formalize and verify the following diagram for every source program $P$:

\[
\begin{array}{c}
P \\
s\text{mean} \\
\text{abstract} \\
s\text{mean}(P) \\
\text{comp} \\
\text{comp}(P) \\
t\text{mean}(\text{comp}(P))
\end{array}
\]

- $\text{comp}$ represents the transformations due to
  - a compiler (harder problem), or
  - a model of a compiler (easier).

Is the model faithful?
Compiler Verification

\[ y = 5 \]
\[ x = y + 1 \]

\[ \text{push 5} \]
\[ \text{store a0} \]
\[ \text{push a0} \]
\[ \text{push 1} \]
\[ \text{add} \]
\[ \text{store a1} \]

\[ a0 = 5 \]
\[ a1 = 6 \]
\[ \text{stack} = [...1] \]

\[ \text{Comp} \]
\[ \text{smean} \]
\[ \text{tmean} \]
\[ \text{abstract} \]
Compiler Verification – History

Compiler Verification - Polak

- Verifies an actual compiler and not a model.
- Source language – Pascal.
- Target machine – realistic stack machine.
- Compiler written by author.
- No optimization phase.
- Machine aided proof (Uses Stanford Verifier).
Compiler Verification - Drawbacks

- Complexity.
  1. Requires reasoning about actual compiler implementation.
  2. Requires reasoning about the behaviour of the compiler for an infinite number of programs and their translations.
- Automation - unlikely
- Proof reuse?
Increasing Trustworthiness Through Compile Code Verification
Formalize and verify the following diagram for a given source program:

\[
\begin{align*}
&y = 5 \\
&x = y + 1 \\
&y = 5 \\
&x = 6
\end{align*}
\]

\[
\begin{align*}
push & 5 \\
store & a0 \\
push & a0 \\
push & 1 \\
add & store a1 \\
a0 & = 5 \\
a1 & = 6 \\
stack & = [...]
\end{align*}
\]
Compiled Code Verification

- Less complex
  - Involves reasoning about a given pair of programs
  - The compiler can be made to provide information to help verification.

- Automation - likely.
Compiled code verification - major approaches


A source program...

```
int a[100];
int g;

int foo(int n)
{
    int i = 0;
    while (i < n)
    {
        a[i] = g * i + 3;
        i = i + 1
    }
    return i;
}

int main()
{
    foo(50);
    return 0;
}
```

...and its execution trace.

```
main();
foo(50);
i = 0;
(i < n)
a[i] = g * i + 3;
i = i + 1
(i < n)
a[i] = g * i + 3;
i = i + 1
(i < n)
a[i] = g * i + 3;
i = i + 1
...
...
return i;
return 0;
```
Observation points and observables

- For a given input, the source program will give rise to an execution trace.
- Similarly for the target program.

**Observation points:** Where the correspondence of source and target is ensured.
1. At every assignment statement.
2. At call-return boundaries
3. At print statements

**Observables:** Things which must correspond.
1. Every variable and its value.
2. Arguments and return values
3. Arguments of print statements

**Equivalent traces:** If two traces have the same sequence of observation points, and at each observation point the observables correspond.
**Correctness Criterion**

*Correctness criterion:* For any input, the source and the target should have equivalent execution traces.
The Necula method
The Necula method

An example program

- body of a function
- while → if; repeat
- code hoisting
- strength reduction
Data Space

- Source

**params**

- n
- i
- t
- u
- v

**mem**

- &g
- &a

**locals**
Data Space

- Target

```
regs
r0
r2

mem
&g
&a

bp−8
bp−4

stack
```
Data Mapping

- Mapping
  1. Globals → globals.
  2. Locals and params → stack and regs.
Observation points and Observables

*Observation points*: Call-return boundaries

*Observables*: At call point – arguments and global memory.
   
   At return point – return values and global memory.
Key question

If we want the condition $i = r0$ and $m = m'$ at $b2$ and $b5$ ...
Key question

how should the variables be related at b0 and b4?
How does one express the values of variables at b2 in terms of variables at b0?...

Through symbolic evaluation.
Symbolic Evaluation

\[
\begin{align*}
u &= t \times i \\
v &= u + 3 \\
i &= i + 1
\end{align*}
\]

\[
\begin{align*}
m' &= m \\
i' &= i + 1 \\
t' &= t \\
u' &= t \times i \\
v' &= t \times i + 3
\end{align*}
\]

\[
\begin{align*}
b0(m, i, t, u, v) &
\end{align*}
\]

\[
\begin{align*}
b1(m, i, t, u, v) &
\end{align*}
\]

\[
\begin{align*}
b0(m, i, t) &
\end{align*}
\]
Symbolic Evaluation

\[ b_0(m, i, n) \]

\[ b_0(m, i) \rightarrow i < n \quad b_1(m, 0) : b_2(m, n) \]

\[ b_0(m, i) = b_1(\text{upd}(m, (&a+i), \text{sel}(m, &g)*i + 3), i+1) \]
Generating and Solving Constraints

- Constraints are generated to represent the condition that observables at observation points correspond.
- Solution of constraints actually establishes this correspondence.
Generating and Solving Constraints

\[ b_0(m, n) \quad b_4(m', [BP-4]) \]

Compiled Code Verification – Survey and Prospects : Amitabha Sanyal 30
Generating and Solving Constraints

\[ b_0(m,n) \quad b_4(m', [BP-4]) \]

\[ b_0(m,n) = 0 < n \quad b_2(m,0) : b_3(m,0,n) \]

\[ b_4(m',[BP-4]) = 0 < [BP-4] \quad b_5(m',0) : b_6(m',0,[BP-4]) \]
Generating and Solving Constraints

\[ b_0(m,n) \quad b_4(m', [BP-4]) \]

\[ n = [BP-4] \]

constraints\((b_2(m,0) \quad b_5(m',0))\)

constraints\((b_3(m,0,n) \quad b_6(m',0, [BP-4]))\)

\[ b_0(m,n) = 0 < n \quad b_2(m,0) : b_3(m,0,n) \]

\[ b_4(m',[BP-4]) = 0 < [BP-4] \quad b_5(m',0) : b_6(m',0,[BP-4]) \]
Generating and Solving Constraints

\[ b_0(m, n) \quad b_1(m', [BP-4]) \]

\[ n = [BP-4] \]

constraints\((b_2(m, 0) \quad b_5(m', 0))\)

constraints\((b_3(m, 0, n) \quad b_6(m', 0, [BP-4]))\)

\[ b_2(m, i) \quad b_5(m', r_0) \]
Generating and Solving Constraints

\[ b_0(m,n) \quad b_4(m', \text{[BP-4]}) \]

\[ n = \text{[BP-4]} \]
\[ \text{constraints}(b_2(m,0) \quad b_5(m',0)) \]
\[ \text{constraints}(b_3(m,0,n) \quad b_6(m',0, \text{[BP-4]}) \]

\[ b_2(m,i) \quad b_5(m', r_0) \]

\[ b_2(m,i) = \text{ret}(m,i) \]
\[ b_5(m', r_0) = \text{ret}(m', r_0) \]
Generating and Solving Constraints

\[ b_0(m,n) \quad b_4(m', [BP-4]) \]

\texttt{n = [BP-4]}
\textit{constraints}(b_2(m,0) \quad b_5(m',0))
\textit{constraints}(b_3(m,0,n) \quad b_6(m',0,[BP-4]))

\[ b_2(m,i) \quad b_5(m', r_0) \]

\texttt{m = m'}
\texttt{i = r_0}

\texttt{b_2(m,i) = \textit{ret}(m,i) \quad b_5(m',r_0) = \textit{ret}(m',r_0)}
Generating and Solving Constraints

\[
\begin{align*}
b3(m, i, n) & \quad b6(m', r0, [BP-4]) \\
i & = r0 \\
n & = [BP-4] \\
\text{constraints} & (b2(upd(m, &a+i, \text{sel}(m, &g)*i+3), i+1), \\
& \quad b5(upd(m', &a+r0, 3), r0+1)) \\
\text{constraints} & (b3(upd(m, &a+i, \text{sel}(m, &g)*i+3), i+1, n), \\
& \quad b7(upd(m', &a+r0, 3), r0+1, [BP-4]))
\end{align*}
\]

\[
\begin{align*}
b3(m, i, n) & = i+1 < n \\
\text{if } b3 & \text{ then } b2(upd(m, &a+i, \text{sel}(m, &g)*i+3), i+1) : \\
& \quad b3(upd(m, &a+i, \text{sel}(m, &g)*i+3), i+1, n) \\
b6(m', r0, [BP-4]) & = r0+1 < [BP-4] \\
\text{if } b6 & \text{ then } b5(upd(m', &a+r0, 3), r0+1) : \\
& \quad b7(upd(m', &a+r0, 3), r0+1, [BP-4])
\end{align*}
\]
Generating and Solving Constraints

\[ b3(m, i, n) \]

\[ b7(m', r0, r2, [BP-4], [BP-8]) \]

\[
\begin{align*}
    i &= r0 \\
    n &= [BP-4] \\
    constraints(b2(upd(m, &a+i, sel(m, &g)*i+3), i+1), \] \\
    &\quad b5(upd(m', &a+r0, r2), r0+1) \\
    &\quad constraints(b3(upd(m, &a+i, sel(m, &g)*i+3), i+1, n), \\
    &\quad b7(upd(m', &a+r0, r2), r0+1, r2+[BP-8],[BP-4],[BP-8])) \\
\end{align*}
\]

\[ b3(m, i, n) = i+1 < n ? \]

\[ b2(upd(m, &a+i, sel(m, &g)*i+3), i+1) : \]

\[ b3(upd(m, &a+i, sel(m, &g)*i+3), i+1, n) \]

\[ b7(m', r0, [BP-4]) = r0+1 < [BP-4] ? \]

\[ b5(upd(m', &a+r0, r2), r0+1) : \]

\[ b7(upd(m', &a+r0, r2), r0+1, r2+[BP-8], [BP-4], [BP-8]) \]
Satisfying the Constraints

\[ b_0(m,n) \quad b_4(m', [BP-4]) \]

\[ n = [BP-4] \]

constraints(b2(m,0) \ b5(m',0))

constraints(b3(m,0,n) \ b6(m',0,[BP-4]))

\[ b_2(m,i) \quad b_5(m', r_0) \]

\[ i = r_0 \]

\[ m = m' \]

\[ b_3(m,i,n) \quad b_6(m', r_0, [BP-4]) \]

\[ i = r_0 \]

\[ n = [BP-4] \]

constraints(b2(upd(m,&a+i, sel(m,&g)*i+3), i+1),
           b5(upd(m',&a+r0, 3), r0+1))

constraints(b3(upd(m,&a+i, sel(m,&g)*i+3), i+1,n),
           b7(upd(m',&a+r0, 3), r0+1, [BP-4]))

\[ b_3(m,i,n) \quad b_7(m', r_0, r_2, [BP-4], [BP-8]) \]

\[ i = r_0 \]

\[ n = [BP-4] \]

constraints(b2(upd(m,&a+i, sel(m,&g)*i+3), i+1),
           b5(upd(m',&a+r0, 2), r0+1))

constraints(b3(upd(m,&a+i, sel(m,&g)*i+3), i+1,n),
           b7(upd(m',&a+r0,r2),r0+1,r2+[BP-8],[BP-4], [BP-8]))
Satisfying the Constraints

\[ b_0(m, n) \quad b_4(m', \text{[BP-4]}) \]

\[ n = \text{[BP-4]} \]
constraints\( b_2(m, 0), b_5(m', 0) \)
constraints\( b_3(m, 0, n), b_6(m', 0, \text{[BP-4]}) \)

\[ b_3(m, i, n) \quad b_6(m', r_0, \text{[BP-4]}) \]

\[ i = r_0 \]
\[ n = \text{[BP-4]} \]
constraints\( b_2(\text{upd}(m, &a+i, \text{sel}(m, &g)*i+3), i+1), \]
\[ b_5(\text{upd}(m', &a+r_0, 3), r_0+1) \]
constraints\( b_3(\text{upd}(m, &a+i, \text{sel}(m, &g)*i+3), i+1, n), \]
\[ b_7(\text{upd}(m', &a+r_0, 3), r_0+1, \text{[BP-4]}) \]

\[ b_2(m, i) \quad b_5(m', r_0) \]

\[ i = r_0 \]
\[ m = m' \]

\[ b_3(m, i, n) \quad b_7(m', r_0, r_2, \text{[BP-4], [BP-8]}) \]

\[ i = r_0 \]
\[ n = \text{[BP-4]} \]
constraints\( b_2(\text{upd}(m, &a+i, \text{sel}(m, &g)*i+3), i+1), \]
\[ b_5(\text{upd}(m', &a+r_0, r_2), r_0+1) \]
constraints\( b_3(\text{upd}(m, &a+i, \text{sel}(m, &g)*i+3), i+1, n), \]
\[ b_7(\text{upd}(m', &a+r_0, r_2), r_0+1, r_2+\text{[BP-8]}, \text{[BP-4]}, \text{[BP-8]}) \]
Satisfying the Constraints

\[ b_0(m,n) \quad b_4(m', [BP-4]) \]

\[ n = [BP-4] \]
\[ m = m' \]
\[ constraints(b_2(m,0) \quad b_5(m',0)) \]
\[ constraints(b_3(m,0,n) \quad b_6(m',0,[BP-4])) \]

\[ b_3(m,i,n) \quad b_6(m', r_0, [BP-4]) \]

\[ i = r_0 \]
\[ n = [BP-4] \]
\[ constraints(b_2(upd(m,&a+i, sel(m,&g)*i+3), i+1), \]
\[ b_5(upd(m',&a+r_0, 3), r_0+1)) \]
\[ constraints(b_3(upd(m,&a+i, sel(m,&g)*i+3), i+1, n), \]
\[ b_7(upd(m',&a+r_0, 3), r_0+1, [BP-4])) \]

\[ b_2(m,i) \quad b_5(m', r_0) \]

\[ i = r_0 \]
\[ m = m' \]

\[ b_3(m,i,n) \quad b_7(m', r_0, r_2, [BP-4], [BP-8]) \]

\[ i = r_0 \]
\[ n = [BP-4] \]
\[ constraints(b_2(upd(m,&a+i, sel(m,&g)*i+3), i+1), \]
\[ b_5(upd(m',&a+r_0, r_2), r_0+1) \]
\[ constraints(b_3(upd(m,&a+i, sel(m,&g)*i+3), i+1, n), \]
\[ b_7(upd(m',&a+r_0,r_2),r_0+1,r_2+[BP-8],[BP-4], [BP-8]) \]
[b0(m,n)  b4(m', [BP-4])]

n = [BP-4]
m = m'
constraints(b2(m,0)  b5(m',0))
constraints(b3(m,0,n)  b6(m',0,[BP-4]))

[b3(m,i,n)  b6(m', r0, [BP-4])]
i = r0
n = [BP-4]
constraints(b2(upd(m,&a+i, sel(m,&g)*i+3), i+1),
   b5(upd(m',&a+r0, 3), r0+1))
constraints(b3(upd(m,&a+i, sel(m,&g)*i+3), i+1, n),
   b7(upd(m',&a+r0, 3), r0+1, [BP-4]))

[b2(m,i)  b5(m', r0)]
i = r0
m = m'

[b3(m,i,n)  b7(m', r0, r2, [BP-4], [BP-8])]
i = r0
n = [BP-4]
constraints(b2(upd(m,&a+i, sel(m,&g)*i+3), i+1),
   b5(upd(m',&a+r0, r2), r0+1))
constraints(b3(upd(m,&a+i, sel(m,&g)*i+3), i+1,n),
   b7(upd(m',&a+r0,r2),r0+1,r2+[BP-8],[BP-4],
      [BP-8])}
Satisfying the Constraints

\( b_0(m,n) \) \hspace{1cm} \( b_4(m', \text{[BP-4]}) \)

\[ n = \text{[BP-4]} \]
\[ m = m' \]

\textit{constraints} \((b_2(m,0) \rightarrow b_5(m',0))\)

\textit{constraints} \((b_3(m,0,n) \rightarrow b_6(m',0,\text{[BP-4]})\)

\( b_3(m,i,n) \) \hspace{1cm} \( b_6(m', r_0, \text{[BP-4]}) \)

\[ i = r_0 \]
\[ n = \text{[BP-4]} \]

\textit{constraints} \((b_2(\text{upd}(m, &a + i, \text{sel}(m, &g) \ast i + 3), i + 1), \)
\[ b_5(\text{upd}(m', &a + r_0, 3), r_0 + 1)\]

\textit{constraints} \((b_3(\text{upd}(m, &a + i, \text{sel}(m, &g) \ast i + 3), i + 1, n), \)
\[ b_7(\text{upd}(m', &a + r_0, 3), r_0 + 1, \text{[BP-4]})\)

\( b_2(m,i) \) \hspace{1cm} \( b_5(m', r_0) \)

\[ i = r_0 \]
\[ m = m' \]

\( b_3(m,i,n) \) \hspace{1cm} \( b_7(m', r_0, r_2, \text{[BP-4]}, \text{[BP-8]}) \)

\[ i = r_0 \]
\[ n = \text{[BP-4]} \]

\textit{constraints} \((b_2(\text{upd}(m, &a + i, \text{sel}(m, &g) \ast i + 3), i + 1), \)
\[ b_5(\text{upd}(m', &a + r_0, 3), r_0 + 1)\]

\textit{constraints} \((b_3(\text{upd}(m, &a + i, \text{sel}(m, &g) \ast i + 3), i + 1, n), \)
\[ b_7(\text{upd}(m', &a + r_0, 3), r_0 + 1, r_2 + \text{[BP-8]}, \text{[BP-4]}, \text{[BP-8]})\)
Satisfying the Constraints

\[ b_0(m,n) \quad b_4(m', [BP-4]) \]

\[ n = [BP-4] \]
\[ m = m' \]
\[ constraints(b_2(m,0) b_5(m',0)) \]
\[ constraints(b_3(m,0,n) b_6(m',0,[BP-4])) \]

\[ b_3(m,i,n) \quad b_6(m', r_0, [BP-4]) \]

\[ i = r_0 \]
\[ n = [BP-4] \]
\[ constraints(b_2(upd(m, &a+i, sel(m, &g) \cdot i+3), i+1), b_5(upd(m', &a+r_0, 3), r_0+1)) \]
\[ constraints(b_3(upd(m, &a+i, sel(m, &g) \cdot i+3), i+1, n), b_7(upd(m', &a+r_0, 3), r_0+1, [BP-4])) \]

\[ b_2(m, i) \quad b_5(m', r_0) \]

\[ i = r_0 \]
\[ m = m' \]

\[ b_3(m, i, n) \quad b_7(m', r_0, r_2, [BP-4], [BP-8]) \]

\[ i = r_0 \]
\[ n = [BP-4] \]
\[ constraints(b_2(upd(m, &a+i, sel(m, &g) \cdot i+3), i+1), b_5(upd(m', &a+r_0, r_2), r_0+1)) \]
\[ constraints(b_3(upd(m, &a+i, sel(m, &g) \cdot i+3), i+1, n), b_7(upd(m', &a+r_0, r_2), r_0+1, r_2+[BP-8],[BP-4],[BP-8])) \]
Satisfying the Constraints

\[ b_0(m,n) \quad b_4(m', [BP-4]) \]

\[ n = [BP-4] \]
\[ m = m' \]
\[ \text{constraints}(b_2(m,0) \quad b_5(m',0)) \]
\[ \text{constraints}(b_3(m,0,n) \quad b_6(m',0,[BP-4])) \]

\[ b_3(m,i,n) \quad b_6(m', r_0, [BP-4]) \]

\[ i = r_0 \]
\[ n = [BP-4] \]
\[ m = m' \]
\[ \text{sel}(m,&g) \times i = 0 \]
\[ \text{constraints}(b_2(\text{upd}(m,&a+i, \text{sel}(m,&g) \times i+3), i+1), b_5(\text{upd}(m',&a+r_0, 3), r_0+1)) \]
\[ \text{constraints}(b_3(\text{upd}(m,&a+i, \text{sel}(m,&g) \times i+3), i+1), b_7(\text{upd}(m',&a+r_0, 3), r_0+1, [BP-4])) \]

\[ b_2(m,i) \quad b_5(m', r_0) \]

\[ i = r_0 \]
\[ m = m' \]

\[ b_3(m,i,n) \quad b_7(m', r_0, r_2, [BP-4], [BP-8]) \]

\[ i = r_0 \]
\[ n = [BP-4] \]
\[ m = m' \]
\[ \text{constraints}(b_2(\text{upd}(m,&a+i, \text{sel}(m,&g) \times i+3), i+1), b_5(\text{upd}(m',&a+r_0, r_2), r_0+1)) \]
\[ \text{constraints}(b_3(\text{upd}(m,&a+i, \text{sel}(m,&g) \times i+3), i+1), b_7(\text{upd}(m',&a+r_0, r_2), r_0+1,r_2+[BP-8],[BP-4], [BP-8])) \]
Satisfying the Constraints

\[ b0(m, n) \quad b4(m', [BP-4]) \]

\[ n = [BP-4] \]
\[ m = m' \]
\[ constraints(b2(m, 0) \quad b5(m', 0)) \]
\[ constraints(b3(m, 0, n) \quad b6(m', 0, [BP-4])) \]

\[ b3(m, i, n) \quad b6(m', r0, [BP-4]) \]

\[ i = r0 \]
\[ n = [BP-4] \]
\[ m = m' \]
\[ sel(m, &g) * i = 0 \]
\[ constraints(b2(upd(m, &a + i, sel(m, &g) * (i + 3)), i + 1), \]
\[ b5(upd(m', &a + r0, 3), r0 + 1)) \]
\[ constraints(b3(upd(m, &a + i, sel(m, &g) * (i + 3)), i + 1), \]
\[ b7(upd(m', &a + r0, 3), r0 + 1, [BP-4])) \]

\[ b2(m, i) \quad b5(m', r0) \]

\[ i = r0 \]
\[ m = m' \]

\[ b3(m, i, n) \quad b7(m', r0, r2, [BP-4], [BP-8]) \]

\[ i = r0 \]
\[ n = [BP-4] \]
\[ m = m' \]
\[ sel(m, &g) * i = 0 \]
\[ constraints(b2(upd(m, &a + i, sel(m, &g) * (i + 3)), i + 1), \]
\[ b5(upd(m', &a + r0, 2), r0 + 1)) \]
\[ constraints(b3(upd(m, &a + i, sel(m, &g) * (i + 3)), i + 1), \]
\[ b7(upd(m', &a + r0, r2), r0 + 1, r2 + [BP-8], [BP-4], \]
\[ [BP-8]) \]
Satisfying the Constraints

\[ \begin{align*}
\text{b0}(m, n) & \quad \text{b4}(m', [BP-4]) \\
n = [BP-4] & \quad i = r0 \\
m = m' & \quad m = m' \\
\text{b2}(m, i) & \quad \text{b5}(m', r0) \\
& \quad m = m' \\
\text{b3}(m, i, n) & \quad \text{b6}(m', r0, [BP-4]) \\
i = r0 & \quad i = r0 \\
n = [BP-4] & \quad n = [BP-4] \\
m = m' & \quad m = m' \\
& \quad m = m' \\
\text{sel}(m, &g) \times i = 0 \\
& \quad \text{sel}(m, &g) \times i + 3 = r2 \\
\end{align*} \]
**Demonstrating correctness – Simulation relation**

- $(b_0, b_4)$: $n = [\text{BP}-4]$, $m = m'$
- $(b_2, b_5)$: $i = r_0$, $m = m'$
- $(b_3, b_6)$: $i = 0$, $n = [\text{BP}-4]$, $m = m'$, $\text{sel}(m, &g) \times i = 0$
- $(b_3, b_7)$: $i = 0$, $n = [\text{BP}-4]$, $m = m'$, $\text{sel}(m, &g) \times i + 3 = r_2$
**Simulation Relation**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b0, b4)</td>
<td>( n = \text{BP-4}, m = m' )</td>
</tr>
<tr>
<td>(b2, b5)</td>
<td>( i = r0, m = m' )</td>
</tr>
<tr>
<td>(b3, b6)</td>
<td>( i = 0, n = \text{BP-4}, m = m', ) sel(m, &amp;g) * i = 0</td>
</tr>
<tr>
<td>(b3, b7)</td>
<td>( i = 0, n = \text{BP-4}, m = m', ) sel(m, &amp;g) * i + 3 = r2</td>
</tr>
</tbody>
</table>

**Source**

- **b0**: \( i = 0 \)
- **b1**: \( i < n \)
- **b2**: return i

**Target**

- **b4**: r0 = 0, 0 < [BP-4]
- **b5**: return r0
- **b6**: [BP-8] = [&g] r2 = 3
- **b7**: [&a+r0] = r2 r0 = r0+1 r2 = r2+[BP-8] r0 < [BP-4]
### Simulation Relation

<table>
<thead>
<tr>
<th>Condition</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>((b_0, b_4))</td>
<td>(n = [BP-4], m = m')</td>
</tr>
<tr>
<td>((b_2, b_5))</td>
<td>(i = r_0, m = m')</td>
</tr>
<tr>
<td>((b_3, b_6))</td>
<td>(i = 0, n = [BP-4], m = m', \text{ sel}(m, &amp;g) \ast i = 0)</td>
</tr>
<tr>
<td>((b_3, b_7))</td>
<td>(i = 0, n = [BP-4], m = m', \text{ sel}(m, &amp;g) \ast i + 3 = r_2)</td>
</tr>
</tbody>
</table>

**Source**

- \(i = 0\)
- \(i < n\)
- \(t = [&g]\)
- \(u = t \ast i\)
- \(v = u + 3\)
- \(v = [a + i]\)
- \(i = i + 1\)
- \(r_0 = 0\)
- \(0 < [BP-4]\)
- \([BP-8] = [&g]\)
- \(r_2 = 3\)
- \([&a + r_0] = r_2\)
- \(r_0 = r_0 + 1\)
- \(r_2 = r_2 + [BP-8]\)
- \(r_0 < [BP-4]\)

**Target**

- \(\text{return } r_0\)
- \(\text{return } i\)
Simulation Relation

| (b0, b4) | n=[BP-4], m = m' |
| (b2, b5) | i=r0, m = m' |
| (b3, b6) | i=0, n=[BP-4], m = m', sel(m,&g)*i = 0 |
| (b3, b7) | i=0, n=[BP-4], m = m', sel(m,&g)*i + 3 = r2 |
### Simulation Relation

<table>
<thead>
<tr>
<th>Condition</th>
<th>Source Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(b_0, b_4)$</td>
<td>$n = [BP-4], m = m'$</td>
</tr>
<tr>
<td>$(b_2, b_5)$</td>
<td>$i = r_0, m = m'$</td>
</tr>
<tr>
<td>$(b_3, b_6)$</td>
<td>$i = 0, n = [BP-4], m = m', sel(m, &amp;g) \cdot i = 0$</td>
</tr>
<tr>
<td>$(b_3, b_7)$</td>
<td>$i = 0, n = [BP-4], m = m', sel(m, &amp;g) \cdot i + 3 = r_2$</td>
</tr>
</tbody>
</table>

### Source

1. $i = 0$
2. $i < n$
3. $t = [\&g]$
4. $u = t \ast i$
5. $v = u + 3$
6. $[\&a + i] = v$
7. $i = i + 1$

#### Target

1. $r_0 = 0$
2. $0 < [BP-4]$
3. $[BP-8] = [\&g]$
4. $r_2 = 3$
5. $[\&a + r_0] = r_2$  $r_0 = r_0 + 1$
6. $r_2 = r_2 + [BP-8]$  $r_0 < [BP-4]$
7. $[BP-4]$
Adding procedure calls

### Source

- **main**
  - entry main
  - call foo(i+j)
  - k = returnVal
  - return k

### Target

- **main**
  - entry main
  - foo(n)
  - m=m'
  - n=[BP–4]
  - m=m'

- **foo**
  - entry foo
  - call foo(r0)
  - r1 = returnVal
  - return r1

- **foo**
  - entry foo
  - call foo(i+j)
  - return i

- **foo**
  - entry foo
  - k = returnVal

- **foo**
  - entry foo
  - foo([BP–4])

- **entry main**

#### Equations

- \( i + j = r0 \land m = m' \Rightarrow \{ n = [BP - 4] \land m = m' \} \)
- \( \{ i = r0 \land m = m' \} \Rightarrow k = r1 \land m = m' \)
Heffter-Gawkoski Approach
Heffter-Gawkoski Approach

Source program → Proof generating compiler → Target program

Translation contract
- Specification of source language
- Specification of target language
- Specification of correct compilation

Proof checker

Compiler specific information
- Memory map
- Intermediate representation

Proof script

yes/no
Heffter-Gawkoski Approach

Example Source Program:

\[
\begin{align*}
x &= 7 + x; \\
y &= 8; \\
x + y
\end{align*}
\]

Semantics:

\[
evalprog :: (Sprogram, State) \rightarrow \text{Integer}
\]
Heffter-Gawkoski Approach

Example Target Program:

push 7
load a0
add
store a0
push 8
store a1
load a0
load a1
add

Semantics:

type Store = (Stack, Memory)

exec :: (Tprogram, Store) → Store
### Heffter-Gawkoski Approach

#### Memory map (dmap)

<table>
<thead>
<tr>
<th>x</th>
<th>a0</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>a1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

#### Initialization of memory (initmem)

<table>
<thead>
<tr>
<th>x</th>
<th>a0</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>a1</td>
<td>7</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Heffter-Gawkoski Approach

\[
\text{ctrans} \ sp \ tp \ =_{\text{def}} \ \\
\exists \ dmap \ \forall \ st : \ \text{evalprog} \ sp \ st = \top \ (\text{fst} \ (\text{exec} \ tp \ ([ ]), \text{initmem} \ st \ dmap))
\]

The value of the source program is the same as that left on the top of the stack by the target program.
Heffter-Gawkoski Approach

For:

\[ sp = [x = 7 + x; y = 8] \quad x + y \]
\[ tp = [\text{push 7}; \text{load} \ a0; \text{add}; \text{store} \ a0; \text{push} \ 8; \text{store} \ a1; \text{load} \ a0; \]
\[ \quad \text{load} \ a1; \text{add} \]
\[ dmap = [x \rightarrow a0; \quad y \rightarrow a1] \]

The compiler generates a proof script of:

\[ \forall \text{st} : \text{evalprog} \ sp = \text{top} \ (\text{fst} \ (\text{exec} \ tp \ ([], \text{initmem} \ \text{st} \ dmap))) \]

Proves if \( x = a0 = \alpha \), then both programs return \( \alpha + 15 \)
Heffter-Gawkoski Approach

Proof long and complex.

Can checking of semantic equivalence be replaced by checking of syntactic equivalence?
Heffter-Gawkoski Approach

- $tr$ models the compiler \textit{as closely and conveniently} as possible.
- $tr$ should be easily checkable.
Heffter-Gawkoski Approach

Diagram:

- Correct translation
- Incorrect translation
Heffter-Gawkoski Approach

If we additionally prove a result:

\[(\exists \text{dmap}\forall \text{sp}, \text{tp} : \text{tr sp dmap tp}) \Rightarrow (\text{ctrans sp tp})\]
Heffter-Gawkoski Approach

- Suppose T is the target produced by the compiler for a source S.
- Are S and T related by tr?
  1. Yes – T is a correct compilation of S.
  2. No – Either T is incorrect or false negative due to inadequate model.
Looking Ahead
Towards More Realistic Compiled Code Verifiers

Can Necula be extended to more realistic compilers?

- **Modelling of richer source language features.**
  - Richer data types.
  - Dynamic memory allocation.
  - Aliasing.
  - Expressions with multiple side effects.
  - Functions calls and returns.

- **Realistic modelling of target machine.**
  - Accurate modelling of target data space (aliasing in stack, heap).
  - Addressing modes.
  - Modelling target data types (bit, byte, word, long).
Looking Ahead

- Relating source to the target.
  - Elaborate data mapping
  - Operator mapping

- The constraint solver.
  - Designing an algorithm for constraint solving.
Our goal

- To build a realistic and usable compiled code verifier for MISRA C subset of GCC C.
  1. No mallocs.
  2. no unions.
  3. Multiple side-effects in a expression not allowed.
  4. No pointer arithmetic.
- Use Necula’s method as basis with minimal compiler instrumentation.
Conclusions

- Building a realistic compiled code verification system for a moderately simple language seems possible.