# Smart Processing of Function Calls to Achieve Efficient Simulation Code

9th OpenModelica Annual Workshop

Jan Hagemann, Patrick Täuber, Bernhard Bachmann

Department of Engineering and Mathematics
University of Applied Sciences Bielefeld, Germany

06/02/2017

# Table of Content

# Introduction

**Common Subexpression Elimination**:

- An optimization method that stores recurrent expressions in temporary variables.

```
1 x := a + b;
2 y := 2*(a + b);
```

```
1 cse1 := a + b;
2 x := cse1;
3 y := 2*cse1;
```

⇒ Analyses have shown that especially encapsulating of function calls has positive effects in simulation time.

# Introduction

**Common Subexpression Elimination**:

- An optimization method that stores recurrent expressions in temporary variables.

```
1  x := a + b;
2  y := 2*(a + b);
```

```
1  cse1 := a + b;
2  x := cse1;
3  y := 2*cse1;
```

⇒ Analyses have shown that especially encapsulating of function calls has positive effects in simulation time.

# Introduction

**Common Subexpression Elimination**:

- An optimization method that stores recurrent expressions in temporary variables.

```
1  x := a + b;
2  y := 2*(a + b);
```

```
1  cse1 := a + b;
2  x := cse1;
3  y := 2*cse1;
```

⇒ Analyses have shown that especially encapsulating of function calls has positive effects in simulation time.

# Introduction

**Common Subexpression Elimination**:

- An optimization method that stores recurrent expressions in temporary variables.

```
1  x := a + b;
2  y := 2*(a + b);
```

```
1  cse1 := a + b;
2  x := cse1;
3  y := 2*cse1;
```

⇒ Analyses have shown that especially encapsulating of function calls has positive effects in simulation time.

# Motivation

Following call illustrates the advantage of code motion:

- $f_1(f_2(x), f_3(time))$

  is part of an algebraic loop, with $x$ as iteration variable

- $cse_1 = f_1(cse_2, cse_3)$

  $cse_2 = f_2(x)$

  $cse_3 = f_3(time)$

$\Rightarrow$ $f_3$ can be moved out the loop because it is not dependent on variables.

$\Rightarrow$ $f_3$ can be evaluated once and it is not necessary to calculate it in each iteration step.

# Motivation

Following call illustrates the advantage of code motion:

- $f_1(f_2(x), f_3(time))$
  is part of an algebraic loop, with $x$ as iteration variable
- $cse_1 = f_1(cse_2, cse_3)$
  $cse_2 = f_2(x)$
  $cse_3 = f_3(time)$

$\Rightarrow$ $f_3$ can be moved out the loop because it is not dependent on variables.

$\Rightarrow$ $f_3$ can be evaluated once and it is not necessary to calculate it in each iteration step.

# Motivation

Following call illustrates the advantage of code motion:

- $f_1(f_2(x), f_3(time))$
  is part of an algebraic loop, with $x$ as iteration variable
- $cse_1 = f_1(cse_2, cse_3)$
  $cse_2 = f_2(x)$
  $cse_3 = f_3(time)$

$\Rightarrow$ $f_3$ can be moved out the loop because it is not dependent on variables.

$\Rightarrow$ $f_3$ can be evaluated once and it is not necessary to calculate it in each iteration step.

# Motivation

Following call illustrates the advantage of code motion:

- $f_1(f_2(x), f_3(time))$
  is part of an algebraic loop, with $x$ as iteration variable
- $cse_1 = f_1(cse_2, cse_3)$
  $cse_2 = f_2(x)$
  $cse_3 = f_3(time)$

$\Rightarrow$ $f_3$ can be moved out the loop because it is not dependent on variables.

$\Rightarrow$ $f_3$ can be evaluated once and it is not necessary to calculate it in each iteration step.

# *wrapFunctionCalls* in OpenModelica

**Aim:** Elimination of function calls by storage in temporary variables.

Particularities that have to be observed:

- Nested function calls (line 8)
- Function calls equal simple variables (line 9)
- Function calls return tuples (line 10)
- Detection of function calls which are not identical at first sight

```
1  model wfc
2    function foo
3      input Real x1, x2;
4      output Real y1, y2, y3;
5    [...] end foo;
6    Real a, b, x;
7  equation
8    a = sin(foo(x, x)) + 5.0;
9    x = sin(cos(time));
10   (,b,) = foo(x, sin(cos(time)));
11 end wfc;
```

# *wrapFunctionCalls* in OpenModelica

**Aim:** Elimination of function calls by storage in temporary variables.

Particularities that have to be observed:

- Nested function calls (line 8)
- Function calls equal simple variables (line 9)
- Function calls return tuples (line 10)
- Detection of function calls which are not identical at first sight

```
1  model wfc
2    function foo
3      input Real x1, x2;
4      output Real y1, y2, y3;
5    [...] end foo;
6    Real a, b, x;
7  equation
8    a = sin(foo(x, x)) + 5.0;
9    x = sin(cos(time));
10   (,b,) = foo(x, sin(cos(time)));
11 end wfc;
```

# *wrapFunctionCalls* in OpenModelica

**Aim:** Elimination of function calls by storage in temporary variables.

Particularities that have to be observed:

- Nested function calls (line 8)
- Function calls equal simple variables (line 9)
- Function calls return tuples (line 10)
- Detection of function calls which are not identical at first sight

```
1  model wfc
2    function foo
3      input Real x1, x2;
4      output Real y1, y2, y3;
5    [...] end foo;
6    Real a, b, x;
7  equation
8    a = sin(foo(x, x)) + 5.0;
9    x = sin(cos(time));
10   (,b,) = foo(x, sin(cos(time)));
11 end wfc;
```

# *wrapFunctionCalls* in OpenModelica

**Aim:** Elimination of function calls by storage in temporary variables.

Particularities that have to be observed:

- Nested function calls (line 8)
- Function calls equal simple variables (line 9)
- Function calls return tuples (line 10)
- Detection of function calls which are not identical at first sight

```
1  model wfc
2    function foo
3      input Real x1, x2;
4      output Real y1, y2, y3;
5    [...] end foo;
6    Real a, b, x;
7  equation
8    a = sin(foo(x, x)) + 5.0;
9    x = sin(cos(time));
10   (,b,) = foo(x, sin(cos(time)));
11 end wfc;
```

# *wrapFunctionCalls* in OpenModelica

**Aim:** Elimination of function calls by storage in temporary variables.

Particularities that have to be observed:

- Nested function calls (line 8)
- Function calls equal simple variables (line 9)
- Function calls return tuples (line 10)
- Detection of function calls which are not identical at first sight

```
1  model wfc
2    function foo
3      input Real x1, x2;
4      output Real y1, y2, y3;
5    [...] end foo;
6    Real a, b, x;
7  equation
8    a = sin(foo(x, x)) + 5.0;
9    x = sin(cos(time));
10   (,b,) = foo(x, sin(cos(time)));
11 end wfc;
```

# *wrapFunctionCalls* in OpenModelica

The algorithm is divided into four parts:

I. **Analysis**
   - Traverse equations top-down
   - Collect all function calls
   - Create a CSE-variable if necessary.

II. **Dependencies**
   - Mark the dependencies between the different function calls

III. **Substitution**
   - Traverse equations again, now bottom-up
   - Replace the function call with the CSE-variable

IV. **Creation of CSE-Equations**
   - Add them to the equation system

# *wrapFunctionCalls* in OpenModelica

The algorithm is divided into four parts:

I. **Analysis**
   - Traverse equations top-down
   - Collect all function calls
   - Create a CSE-variable if necessary.

II. **Dependencies**
   - Mark the dependencies between the different function calls

III. **Substitution**
   - Traverse equations again, now bottom-up
   - Replace the function call with the CSE-variable

IV. **Creation of CSE-Equations**
   - Add them to the equation system

# *wrapFunctionCalls* in OpenModelica

The algorithm is divided into four parts:

I. **Analysis**
   - Traverse equations top-down
   - Collect all function calls
   - Create a CSE-variable if necessary.

II. **Dependencies**
   - Mark the dependencies between the different function calls

III. **Substitution**
   - Traverse equations again, now bottom-up
   - Replace the function call with the CSE-variable

IV. **Creation of CSE-Equations**
   - Add them to the equation system

# *wrapFunctionCalls* in OpenModelica

The algorithm is divided into four parts:

I. **Analysis**
   - Traverse equations top-down
   - Collect all function calls
   - Create a CSE-variable if necessary.

II. **Dependencies**
   - Mark the dependencies between the different function calls

III. **Substitution**
   - Traverse equations again, now bottom-up
   - Replace the function call with the CSE-variable

IV. **Creation of CSE-Equations**
   - Add them to the equation system

# *wrapFunctionCalls* in OpenModelica

The algorithm is divided into four parts:

I. **Analysis**
   - Traverse equations top-down
   - Collect all function calls
   - Create a CSE-variable if necessary.

II. **Dependencies**
   - Mark the dependencies between the different function calls

III. **Substitution**
   - Traverse equations again, now bottom-up
   - Replace the function call with the CSE-variable

IV. **Creation of CSE-Equations**
   - Add them to the equation system

# *wrapFunctionCalls* in OpenModelica

I. **Analysis**

$a = sin(foo(x, x)) + 5.0$

$x = sin(cos(time))$

$(, b, ) = foo(x, sin(cos(time)))$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# *wrapFunctionCalls* in OpenModelica

I. **Analysis**

$a = sin(foo(x, x)) + 5.0$

$x = sin(cos(time))$

$(, b, ) = foo(x, sin(cos(time)))$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# *wrapFunctionCalls* in OpenModelica

I. **Analysis**

$a = sin(foo(x, x)) + 5.0$

$x = sin(cos(time))$

$(, b, ) = foo(x, sin(cos(time)))$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(foo(x, x))$ | { } |
| | | | | |
| | | | | |
| | | | | |

# *wrapFunctionCalls* in OpenModelica

I. **Analysis**

$a = sin(foo(x, x)) + 5.0$

$x = sin(cos(time))$

$(, b, ) = foo(x, sin(cos(time)))$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(foo(x, x))$ | { } |
| $foo(x, x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x, x)$ | { } |
| | | | | |
| | | | | |
| | | | | |

# *wrapFunctionCalls* in OpenModelica

I. **Analysis**

$a = sin(foo(x, x)) + 5.0$

$x = sin(cos(time))$

$(, b, ) = foo(x, sin(cos(time)))$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(foo(x, x))$ | { } |
| $foo(x, x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x, x)$ | { } |
| | | | | |
| | | | | |
| | | | | |

# *wrapFunctionCalls* in OpenModelica

I. **Analysis**

$a = sin(foo(x, x)) + 5.0$

$x = sin(cos(time))$

$(, b, ) = foo(x, sin(cos(time)))$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(foo(x, x))$ | { } |
| $foo(x, x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x, x)$ | { } |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(\cos(time))$ | { } |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | { } |
| | | | | |

# *wrapFunctionCalls* in OpenModelica

I. **Analysis**

$a = sin(foo(x, x)) + 5.0$

$x = sin(cos(time))$

$(, b, ) = foo(x, sin(cos(time)))$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(foo(x, x))$ | { } |
| $foo(x, x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x, x)$ | { } |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(\cos(time))$ | { } |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | { } |
| | | | | |

# *wrapFunctionCalls* in OpenModelica

I. **Analysis**

$a = sin(foo(x, x)) + 5.0$

$x = sin(cos(time))$

$(, b, ) = foo(x, sin(cos(time)))$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(foo(x, x))$ | { } |
| $foo(x, x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x, x)$ | { } |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(\cos(time))$ | { } |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | { } |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $foo(x, \sin(\cos(time)))$ | { } |

## II. **Dependencies**

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x,x))$ | 1 | $cse_1$ | $\sin(foo(x,x))$ | { } |
| $foo(x,x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x,x)$ | { } |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(\cos(time))$ | { } |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | { } |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $foo(x, \sin(\cos(time)))$ | { } |

## II. **Dependencies**

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x,x))$ | 1 | $cse_1$ | $\sin(foo(x,x))$ | $\{\ \}$ |
| $foo(x,x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x,x)$ | $\{1\}$ |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(\cos(time))$ | $\{\ \}$ |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | $\{\ \}$ |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $foo(x, \sin(\cos(time)))$ | $\{\ \}$ |

## II. **Dependencies**

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x,x))$ | 1 | $cse_1$ | $\sin(foo(x,x))$ | $\{\ \}$ |
| $foo(x,x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x,x)$ | $\{1\}$ |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(\cos(time))$ | $\{\ \}$ |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | $\{3\}$ |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $foo(x, \sin(\cos(time)))$ | $\{\ \}$ |

## II. Dependencies

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x,x))$ | 1 | $cse_1$ | $\sin(foo(x,x))$ | $\{\ \}$ |
| $foo(x,x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x,x)$ | $\{1\}$ |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(\cos(time))$ | $\{5\}$ |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | $\{3\}$ |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $foo(x, \sin(\cos(time)))$ | $\{\ \}$ |

## II. Dependencies

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x,x))$ | 1 | $cse_1$ | $\sin(foo(x,x))$ | $\{\ \}$ |
| $foo(x,x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x,x)$ | $\{1\}$ |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(\cos(time))$ | $\{5\}$ |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | $\{3, 5\}$ |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $foo(x, \sin(\cos(time)))$ | $\{\ \}$ |

# *wrapFunctionCalls* in OpenModelica

## III. Substitution

$a = sin(foo(x, x)) + 5.0$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(foo(x, x))$ | $\{\ \}$ |
| $foo(x, x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x, x)$ | $\{1\}$ |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(\cos(time))$ | $\{5\}$ |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | $\{3, 5\}$ |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $foo(x, \sin(\cos(time)))$ | $\{\ \}$ |
| | | | | |
| | | | | |
| | | | | |

# *wrapFunctionCalls* in OpenModelica

## III. Substitution

$a = sin(cse_2) + 5.0$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(cse_2)$ | { } |
| $foo(x, x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x, x)$ | { } |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(\cos(time))$ | {5} |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | {3, 5} |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $foo(x, \sin(\cos(time)))$ | { } |
| $\sin(cse_2)$ | 1 | | | |
| | | | | |
| | | | | |

# *wrapFunctionCalls* in OpenModelica

## III. Substitution

$a = cse_1 + 5.0$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(cse_2)$ | $\{\ \}$ |
| $foo(x, x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x, x)$ | $\{\ \}$ |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(\cos(time))$ | $\{5\}$ |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | $\{3, 5\}$ |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $foo(x, \sin(\cos(time)))$ | $\{\ \}$ |
| $\sin(cse_2)$ | 1 | | | |

# *wrapFunctionCalls* in OpenModelica

## III. Substitution

$a = cse_1 + 5.0$

$x = sin(cos(time))$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(cse_2)$ | $\{\ \}$ |
| $foo(x, x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x, x)$ | $\{\ \}$ |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(\cos(time))$ | $\{5\}$ |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | $\{3, 5\}$ |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $foo(x, \sin(\cos(time)))$ | $\{\ \}$ |
| $\sin(cse_2)$ | 1 | | | |
| | | | | |
| | | | | |

## III. Substitution

$a = cse_1 + 5.0$

$x = sin(cse_3)$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x,x))$ | 1 | $cse_1$ | $\sin(cse_2)$ | $\{\ \}$ |
| $foo(x,x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x,x)$ | $\{\ \}$ |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(cse_3)$ | $\{5\}$ |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | $\{\ \}$ |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $foo(x, \sin(cse_3))$ | $\{\ \}$ |
| $\sin(cse_2)$ | 1 | | | |
| $\sin(cse_3)$ | 3 | | | |

## III. Substitution

$a = cse_1 + 5.0$

$x = x$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(cse_2)$ | { } |
| $foo(x, x)$ | 2 | $(cse_2, \_, \_)$ | $foo(x, x)$ | { } |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(cse_3)$ | { } |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | { } |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $foo(x, x)$ | { } |
| $\sin(cse_2)$ | 1 | | | |
| $\sin(cse_3)$ | 3 | | | |
| | | | | |

## III. Substitution

$a = cse_1 + 5.0$

$x = x$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(cse_2)$ | { } |
| $foo(x, x)$ | 2 | $(cse_2, b, \_)$ | $foo(x, x)$ | { } |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(cse_3)$ | { } |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | { } |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $(cse_2, b, \_)$ | { } |
| $\sin(cse_2)$ | 1 | | | |
| $\sin(cse_3)$ | 3 | | | |
| | | | | |

## III. Substitution

$a = cse_1 + 5.0$

~~$x = x$~~

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(cse_2)$ | { } |
| $foo(x, x)$ | 2 | $(cse_2, b, \_)$ | $foo(x, x)$ | { } |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(cse_3)$ | { } |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | { } |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $(cse_2, b, \_)$ | { } |
| $\sin(cse_2)$ | 1 | | | |
| $\sin(cse_3)$ | 3 | | | |
| | | | | |

## III. Substitution

$a = cse_1 + 5.0$

~~$x == x$~~

$(, b, ) = foo(x, sin(cos(time)))$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x,x))$ | 1 | $cse_1$ | $\sin(cse_2)$ | { } |
| $foo(x,x)$ | 2 | $(cse_2, b, \_)$ | $foo(x,x)$ | { } |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(cse_3)$ | { } |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | { } |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $(cse_2, b, \_)$ | { } |
| $\sin(cse_2)$ | 1 | | | |
| $\sin(cse_3)$ | 3 | | | |

## III. Substitution

$a = cse_1 + 5.0$

~~$x == x$~~

$(, b, ) = foo(x, sin(cse_3))$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(cse_2)$ | { } |
| $foo(x, x)$ | 2 | $(cse_2, b, \_)$ | $foo(x, x)$ | { } |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(cse_3)$ | { } |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | { } |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $(cse_2, b, \_)$ | { } |
| $\sin(cse_2)$ | 1 | | | |
| $\sin(cse_3)$ | 3 | | | |
| $foo(x, \sin(cse_3))$ | 5 | | | |

## III. Substitution

$a = cse_1 + 5.0$

~~$x == x$~~

$(, b, ) = foo(x, x)$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x,x))$ | 1 | $cse_1$ | $\sin(cse_2)$ | { } |
| $foo(x,x)$ | 2 | $(cse_2, b, \_)$ | $foo(x,x)$ | { } |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(cse_3)$ | { } |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | { } |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $(cse_2, b, \_)$ | { } |
| $\sin(cse_2)$ | 1 | | | |
| $\sin(cse_3)$ | 3 | | | |
| $foo(x, \sin(cse_3))$ | 5 | | | |

## III. Substitution

$a = cse_1 + 5.0$

~~$x == x$~~

$(, b, ) = (cse_2, b, )$

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x, x))$ | 1 | $cse_1$ | $\sin(cse_2)$ | { } |
| $foo(x, x)$ | 2 | $(cse_2, b, \_)$ | $foo(x, x)$ | { } |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(cse_3)$ | { } |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | { } |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $(cse_2, b, \_)$ | { } |
| $\sin(cse_2)$ | 1 | | | |
| $\sin(cse_3)$ | 3 | | | |
| $foo(x, \sin(cse_3))$ | 5 | | | |

## III. Substitution

$a = cse_1 + 5.0$

~~$x \equiv x$~~

~~$(\_, b, \_) \equiv (cse_2, b, \_)$~~

| **Hash Table** | | **Array+** | | |
|---|---|---|---|---|
| Call | Int | CSE | Call | Int.list |
| $\sin(foo(x,x))$ | 1 | $cse_1$ | $\sin(cse_2)$ | { } |
| $foo(x,x)$ | 2 | $(cse_2, b, \_)$ | $foo(x,x)$ | { } |
| $\sin(\cos(time))$ | 3 | $x$ | $\sin(cse_3)$ | { } |
| $\cos(time)$ | 4 | $cse_3$ | $\cos(time)$ | { } |
| $foo(x, \sin(\cos(time)))$ | 5 | $(\_, b, \_)$ | $(cse_2, b, \_)$ | { } |
| $\sin(cse_2)$ | 1 | | | |
| $\sin(cse_3)$ | 3 | | | |
| $foo(x, \sin(cse_3))$ | 5 | | | |

## IV. **Create CSE-equations**

| **Array+** | | |
|---|---|---|
| CSE | Call | Int.list |
| $cse_1$ | $\sin(cse_2)$ | $\{\ \}$ |
| $(cse_2, b, \_)$ | $foo(x, x)$ | $\{\ \}$ |
| $x$ | $\sin(cse_3)$ | $\{\ \}$ |
| $cse_3$ | $\cos(time)$ | $\{\ \}$ |
| $(\_, b, \_)$ | $(cse_2, b, \_)$ | $\{\ \}$ |

$\Rightarrow$ The complexity of the algorithm is $O(n)$, where $n$ is the number of equations.

## IV. **Create CSE-equations**

| **Array+** | | |
|---|---|---|
| CSE | Call | Int.list |
| $cse_1$ | $\sin(cse_2)$ | $\{\ \}$ |
| $(cse_2, b, \_)$ | $foo(x, x)$ | $\{\ \}$ |
| $x$ | $\sin(cse_3)$ | $\{\ \}$ |
| $cse_3$ | $\cos(time)$ | $\{\ \}$ |
| ~~$(\_, b, \_)$~~ | ~~$(cse_2, b, \_)$~~ | ~~$\{\ \}$~~ |

$\Rightarrow$ The complexity of the algorithm is $O(n)$, where $n$ is the number of equations.

# *wrapFunctionCalls* in OpenModelica

## IV. **Create CSE-equations**

| **Array+** | | |
|---|---|---|
| CSE | Call | Int.list |
| $cse_1$ | $\sin(cse_2)$ | { } |
| $(cse_2, b, \_)$ | $foo(x, x)$ | { } |
| $x$ | $\sin(cse_3)$ | { } |
| $cse_3$ | $\cos(time)$ | { } |
| ~~$(\_, b, \_)$~~ | ~~$(cse_2, b, \_)$~~ | ~~{ }~~ |

```
1  model wfc_result
2    function foo
3      [...]
4    end foo;
5    Real a, b, x;
6    Real cse1, cse2, cse3;
7  equation
8    a = cse1 + 5.0;
9    cse1 = sin(cse2);
10   (cse2, b, ) = foo(x, x);
11   x = sin(cse3);
12   cse3 = cos(time);
13 end wfc_result;
```

⇒ The complexity of the algorithm is $O(n)$, where $n$ is the number of equations.

# *wrapFunctionCalls* in OpenModelica

## IV. **Create CSE-equations**

| **Array+** | | |
|---|---|---|
| CSE | Call | Int.list |
| $cse_1$ | $\sin(cse_2)$ | $\{\}$ |
| $(cse_2, b, \_)$ | $foo(x, x)$ | $\{\}$ |
| $x$ | $\sin(cse_3)$ | $\{\}$ |
| $cse_3$ | $\cos(time)$ | $\{\}$ |
| ~~$(\_, b, \_)$~~ | ~~$(cse_2, b, \_)$~~ | ~~$\{\}$~~ |

```
1  model wfc_result
2    function foo
3      [...]
4    end foo;
5    Real a, b, x;
6    Real cse1, cse2, cse3;
7  equation
8    a = cse1 + 5.0;
9    cse1 = sin(cse2);
10   (cse2, b, ) = foo(x, x);
11   x = sin(cse3);
12   cse3 = cos(time);
13 end wfc_result;
```

⇒ The complexity of the algorithm is $O(n)$, where $n$ is the number of equations.

# Verification

WaterIF97 of MSL 3.2.1 shows code motion:

```
Nonlinear loop.
Iteration variable: DER.medium.p
-----------------------------------------------
  DER.medium.h = DER.medium.u -
  (medium.p * DER.medium.d -
  der(medium.p) * medium.d)/(medium.d ^ 2.0)
-----------------------------------------------
Residual equation:
Modelica.Media.Water.IF97_Utilities.rho_ph_der
(medium.p, medium.h, Modelica.Media.Water.
IF97_Utilities.waterBaseProp_ph(medium.p,
medium.h, medium.phase, 0), DER.medium.p,
DER.medium.h) - DER.medium.d = 0.0
```

# Verification

WaterIF97 of MSL 3.2.1 shows code motion:

```
cse4 := Modelica.Media.Water.IF97_Utilities.
 waterBaseProp_ph(medium.p, medium.h,
 medium.phase, 0);
```

```
Nonlinear loop.
Iteration variable: DER.medium.p
-----------------------------------------
  DER.medium.h = DER.medium.u -
  (medium.p * DER.medium.d -
  der(medium.p) * medium.d)/(medium.d ^ 2.0)
-----------------------------------------
Residual equation:
Modelica.Media.Water.IF97_Utilities.rho_ph_der
(medium.p, medium.h, cse4, DER.medium.p,
DER.medium.h) - DER.medium.d = 0.0
```

# Verification

Improvement in simulation time:

| Models (**MSL 3.2.1**) | − WFC [s] | + WFC [s] | Imp. [%] |
|---|---|---|---|
| DrumBoiler | 4.52 | | |
| MomentumBalanceFittings | 4.01 | | |
| HeatExchangerSimulation | 36.31 | | |
| InverseParameterization | 78.81 | | |
| NonCircularPipes | 8.71 | | |
| R134a1 | 16.77 | | |
| DryAir2 | 21.85 | | |
| TestTwoPhaseStates | 2.38 | | |
| WaterIF97 | 1.90 | | |

The models are tested using Linux, 64 bit, on Intel(R) Xeon(R) CPU E5-2650 0 @2.00 GHz Processor.

# Verification

Improvement in simulation time:

| Models (**MSL 3.2.1**) | − WFC [s] | + WFC [s] | Imp. [%] |
|---|---|---|---|
| DrumBoiler | 4.52 | 1.73 | 62 |
| MomentumBalanceFittings | 4.01 | 1.63 | 60 |
| HeatExchangerSimulation | 36.31 | 12.89 | 65 |
| InverseParameterization | 78.81 | 41.03 | 48 |
| NonCircularPipes | 8.71 | 3.91 | 55 |
| R134a1 | 16.77 | 5.19 | 69 |
| DryAir2 | 21.85 | 5.17 | 76 |
| TestTwoPhaseStates | 2.38 | 0.93 | 61 |
| WaterIF97 | 1.90 | 0.90 | 53 |

The models are tested using Linux, 64 bit, on Intel(R) Xeon(R) CPU E5-2650 0 @2.00 GHz Processor.

# Verification

Improvement in simulation time:

| Models **(Library)** | − WFC [s] | + WFC [s] | Imp. [%] |
|---|---|---|---|
| **PlanarMechanics** | | | |
| KinematicLoop_ DynamicStateSelection | 19.26 | 6.12 | 68 |
| **PowerSystems** | | | |
| PowerWorld | 7.59 | 1.73 | 77 |
| **ThermoPower** | | | |
| TestMixerSlowFastSteam | 8.80 | 0.69 | 92 |
| TestWaterFlow1DFV_F | 45.73 | 15.82 | 65 |
| **ScalableTestSuite** | | | |
| SteamPipe_N_10 | 23.84 | 4.35 | 81 |
| SteamPipe_N_160 | 412.87 | 73.8 | 82 |

# Summary

- *wrapFunctionCalls* is based on *common subexpression elimination*.
- Algorithm is divided into four parts, whereby it traverses the equation system two times.

$\Rightarrow$ Encapsulation of expensive functions and code motion

$\Rightarrow$ Significant improvement in simulation time on some libraries

# Summary

- *wrapFunctionCalls* is based on *common subexpression elimination*.
- Algorithm is divided into four parts, whereby it traverses the equation system two times.

$\Rightarrow$ Encapsulation of expensive functions and code motion

$\Rightarrow$ Significant improvement in simulation time on some libraries

# Summary

- *wrapFunctionCalls* is based on *common subexpression elimination*.
- Algorithm is divided into four parts, whereby it traverses the equation system two times.

$\Rightarrow$ Encapsulation of expensive functions and code motion

$\Rightarrow$ Significant improvement in simulation time on some libraries

# Summary

- *wrapFunctionCalls* is based on *common subexpression elimination*.
- Algorithm is divided into four parts, whereby it traverses the equation system two times.

$\Rightarrow$ Encapsulation of expensive functions and code motion

$\Rightarrow$ Significant improvement in simulation time on some libraries

Thanks for your attention!