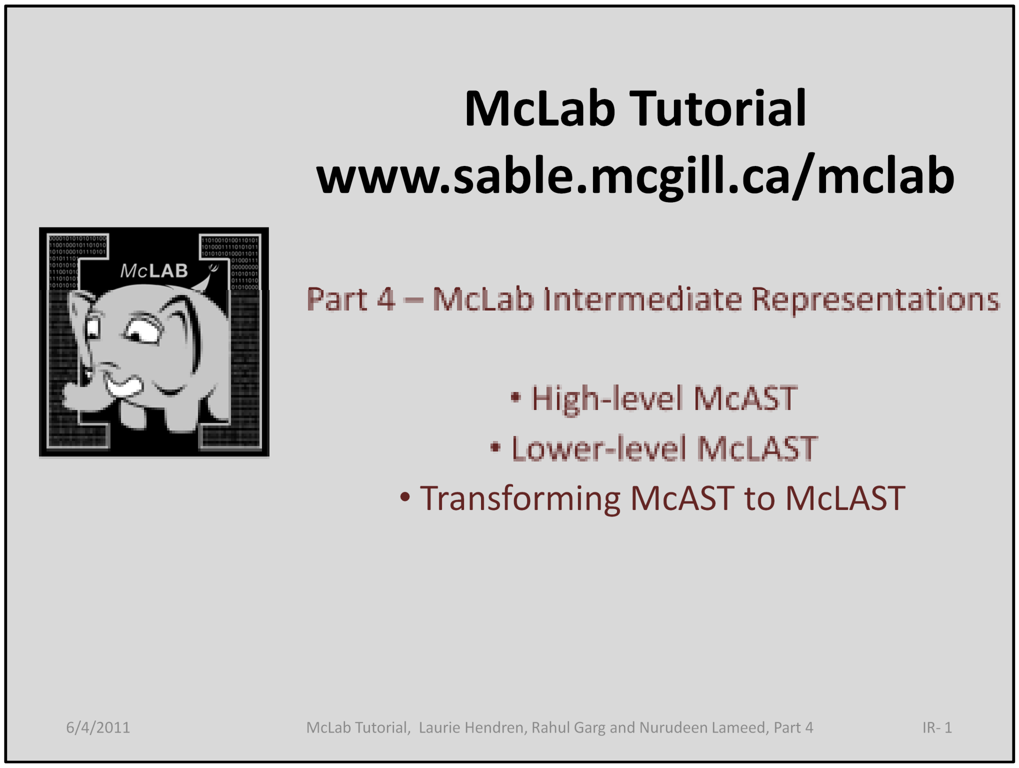


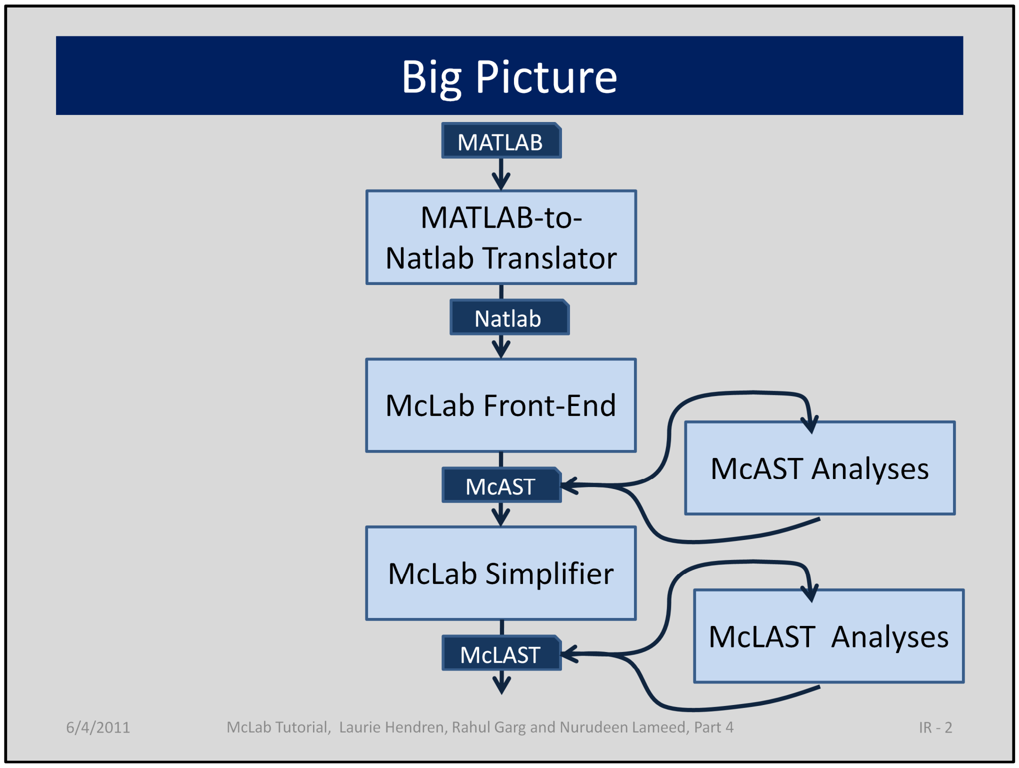
McLab Tutorial
www.sable.mcgill.ca/mclab

Part 4 – McLab Intermediate Representations

- High-level McAST
- Lower-level McLAST
- Transforming McAST to McLAST

6/4/2011 McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed, Part 4 IR- 1

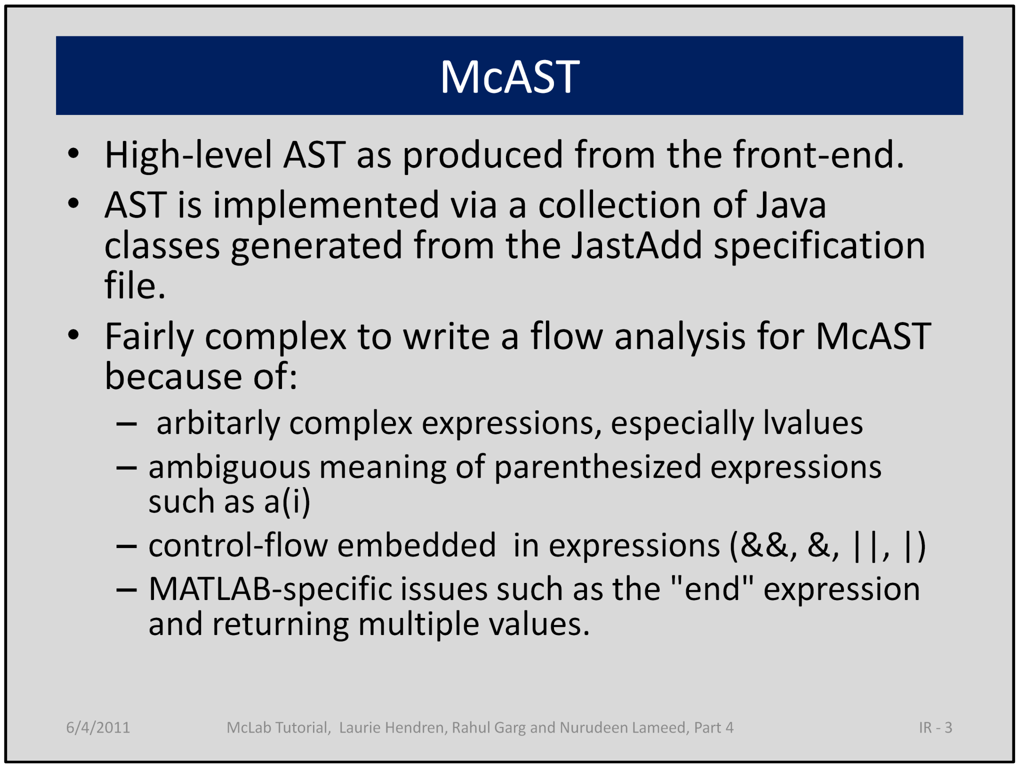
Now that we have explained the front-end, we need to explain our IRs. We will start with the high-level AST, McAST, which is produced by the front-end, then we will describe our lower-level IR, McLAST, which is still tree-based, but is simplified and more suitable for flow analysis. Then we will describe how we transform McAST into McLAST.



Big Picture

6/4/2011 McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed, Part 4 IR - 2

We have already shown you the top two light blue boxes in this figure. The front-end produces McAST. Now, we must do some analysis on this AST to determine the kind of each identifier (VAR, FN or ID), and then simplify the McAST yielding a lower level representation called McLAST. Various analyses can then be applied to McLAST. Both the McAST and McLAST analyses can be implemented using our flow analysis framework, which will be introduced in the next part of this tutorial.

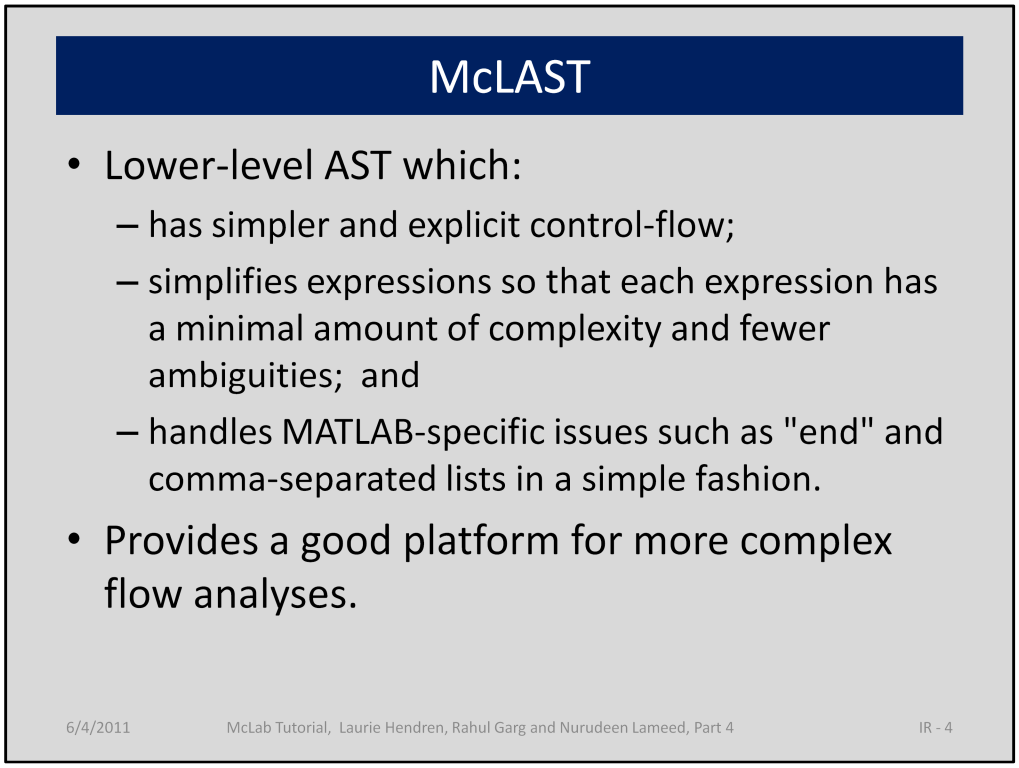


McAST

- High-level AST as produced from the front-end.
- AST is implemented via a collection of Java classes generated from the JastAdd specification file.
- Fairly complex to write a flow analysis for McAST because of:
  - arbitarly complex expressions, especially lvalues
  - ambiguous meaning of parenthesized expressions such as a(i)
  - control-flow embedded in expressions (&&, &, ||, |)
  - MATLAB-specific issues such as the "end" expression and returning multiple values.

6/4/2011 McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed, Part 4 IR - 3

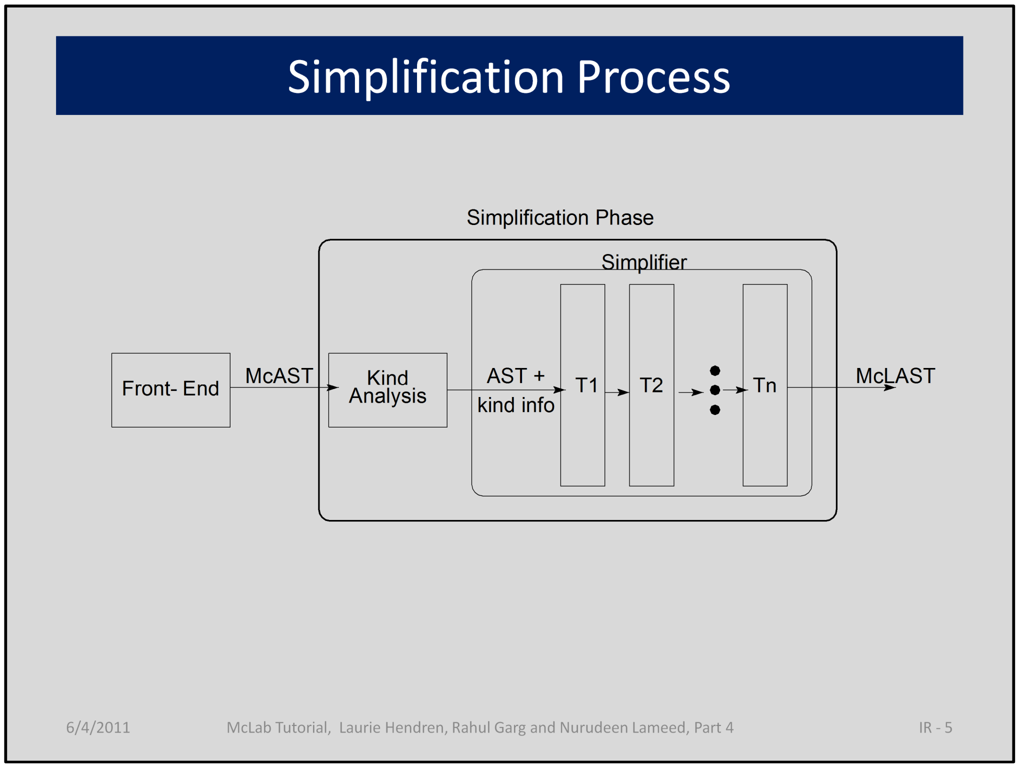
Best source of further documentation – Chapters 3 and 4 of Jesse Doherty's M.Sc. thesis.



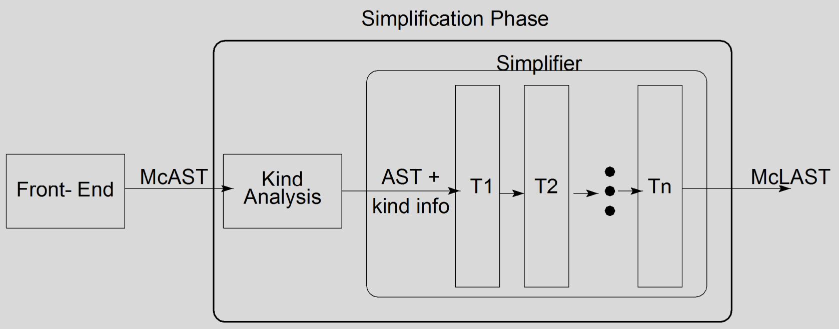
McLAST

- Lower-level AST which:
  - has simpler and explicit control-flow;
  - simplifies expressions so that each expression has a minimal amount of complexity and fewer ambiguities; and
  - handles MATLAB-specific issues such as "end" and comma-separated lists in a simple fashion.
- Provides a good platform for more complex flow analyses.

6/4/2011 McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed, Part 4 IR - 4



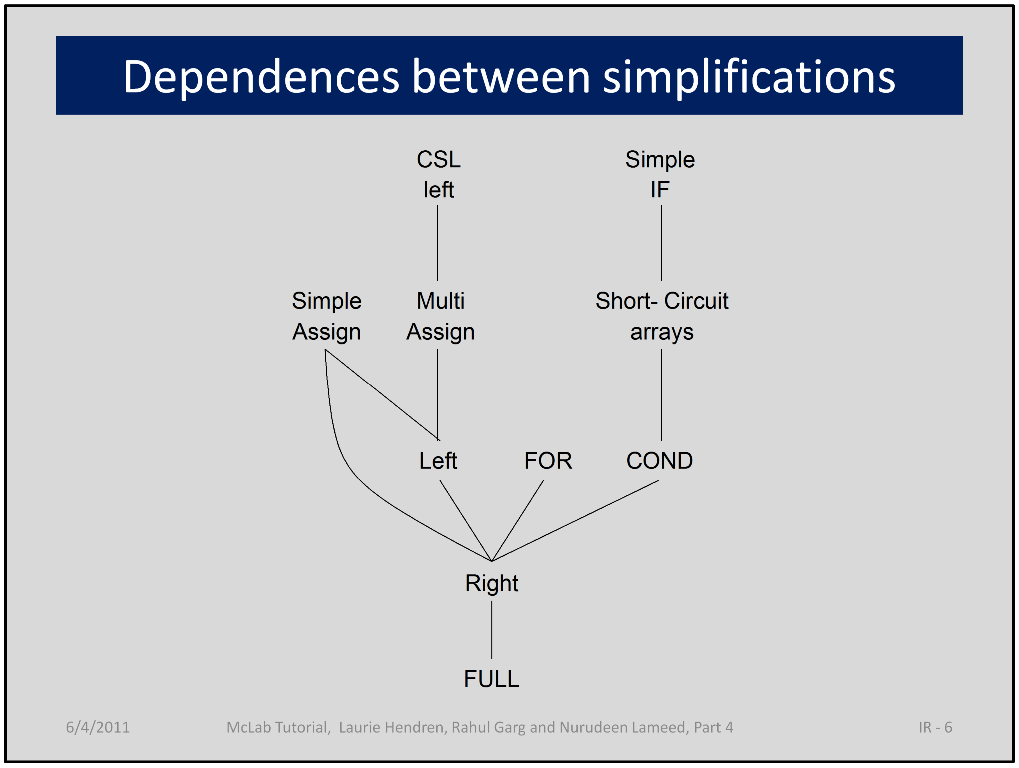
Simplification Process



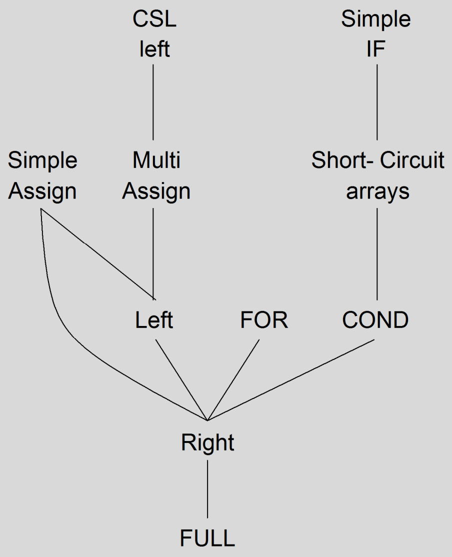
6/4/2011 McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed, Part 4 IR - 5

The simplificatin proceeds by getting the McAST from the front-end, and then applying the kind analysis to that AST, then given the AST and the kind analysis info a sequence of simplifying transformations are applied, finally yielding the lower-level McLAST.

The default behaviour is that all the simplifications are run, but there may be situations where a framework user only wants to apply some of the simplifications. However, some simplifications depend on others having already been performed, so how do we deal with this?

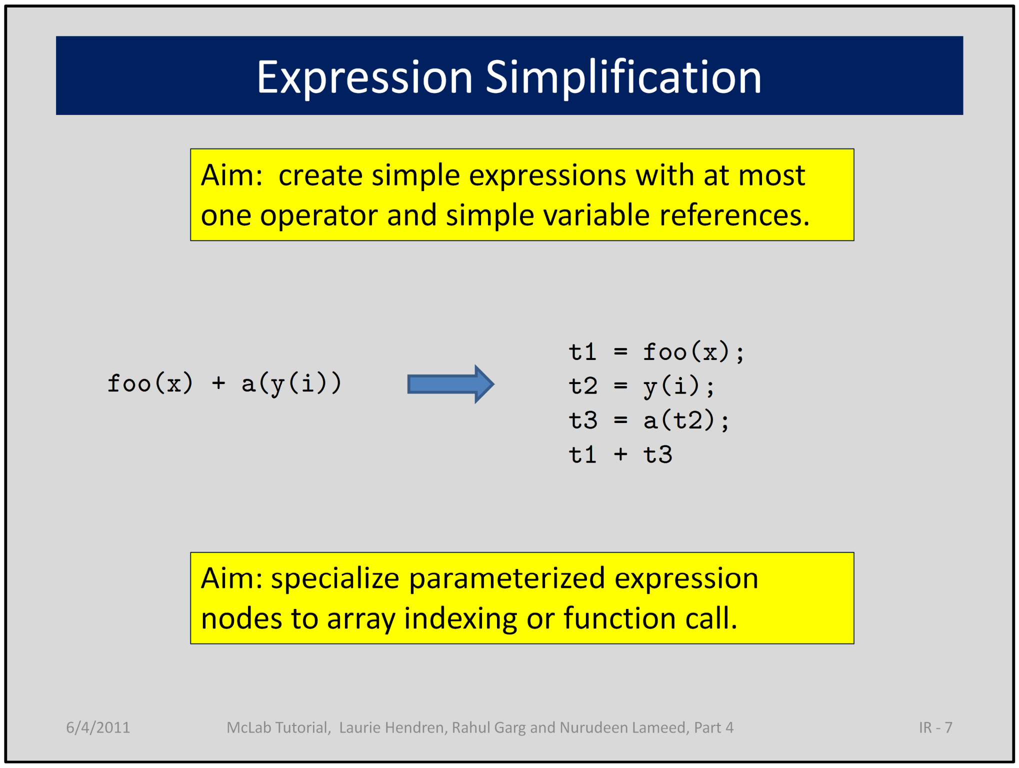


Dependences between simplifications



6/4/2011 McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed, Part 4 IR - 6

Each simplification specifies which other simplifications it depends on, giving us a DAG of dependences. Now, if only some simplifications are desired, the system will run only those simplifications, plus those that it depends on. If a user adds a new simplification to the framework, they must also specify which simplifications it depends on.



Expression Simplification

Aim: create simple expressions with at most one operator and simple variable references.

```
foo(x) + a(y(i))
```

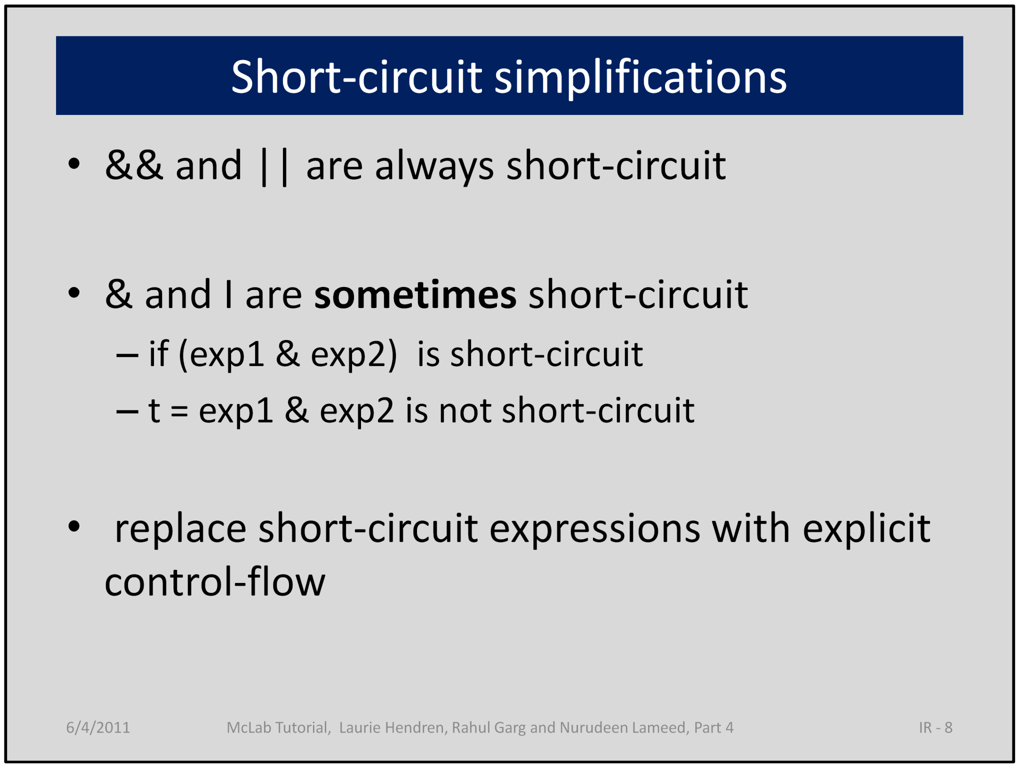
$\Rightarrow$

```
t1 = foo(x);
t2 = y(i);
t3 = a(t2);
t1 + t3
```

Aim: specialize parameterized expression nodes to array indexing or function call.

6/4/2011 McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed, Part 4 IR - 7

One of the main simplifications is to simplify expressions, so that each expression has at most one operation. In addition we need to resolve the meaning of expressions like a(i) which in the high-level AST are stored as parameterized expressions. In the simplification we can to encode these as either array indexing or function calls.

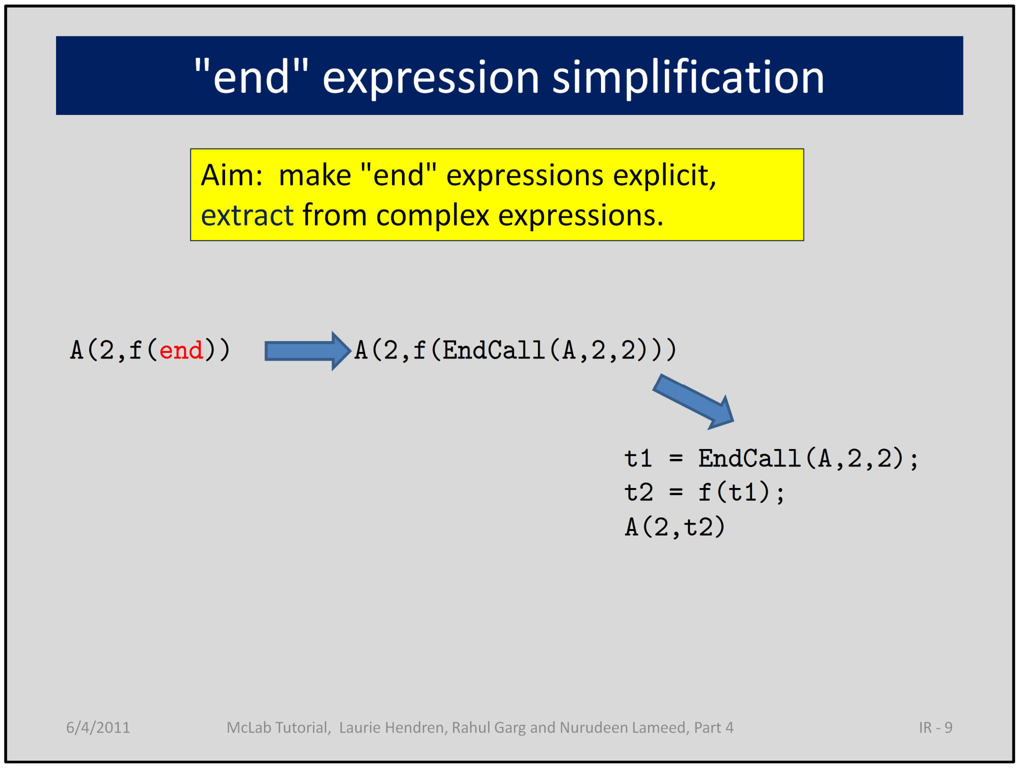


Short-circuit simplifications

- && and || are always short-circuit
- & and I are **sometimes** short-circuit
  - if (exp1 & exp2) is short-circuit
  - t = exp1 & exp2 is not short-circuit
- replace short-circuit expressions with explicit control-flow

6/4/2011 McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed, Part 4 IR - 8

In order to simplify program analysis, we want to remove any control-flow from expressions. The && and || operators are always short-circuit, and so do involve control flow. Thus these are always converted to equivalent nested conditional statements. However, in MATLAB the itemwise operators & and | are also sometimes short-circuit. If the appear in the condition of an if or while they are short-circuit, otherwise they are not. Thus, only the short-circuit occurances are simplified.



"end" expression simplification

Aim: make "end" expressions explicit, extract from complex expressions.

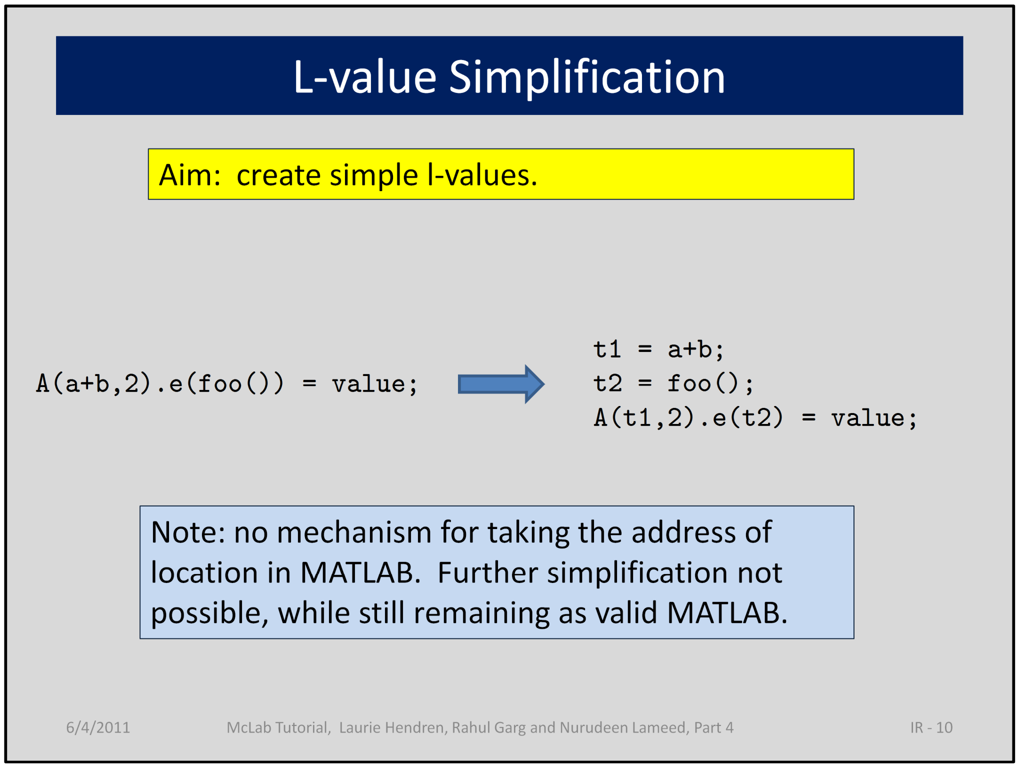
```
A(2,f(end))  →  A(2,f(EndCall(A,2,2)))
```

```
t1 = EndCall(A,2,2);
t2 = f(t1);
A(2,t2)
```

6/4/2011 McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed, Part 4 IR - 9

MATLAB includes and "end" expression which is used to capture the last index of the closest enclosing array. So for example,
A(2,f(end)) could mean two things. If f is a variable, then it means the last index of f. If f is a function and A is a variable, then it means the last index of the 2nd dimension of A, where A is being indexed using 2 dimensions. We use the kind analysis to determine the closest enclosing variable, and then convert the end expression into and explicit EndCall.

In this new expression, EndCall(A,2,2) is the explicit end. It is specifying that the end binds to
the array A interpreted as two dimensional where the end is used in the second dimension.



L-value Simplification

Aim: create simple l-values.

```
A(a+b,2).e(foo()) = value;
```

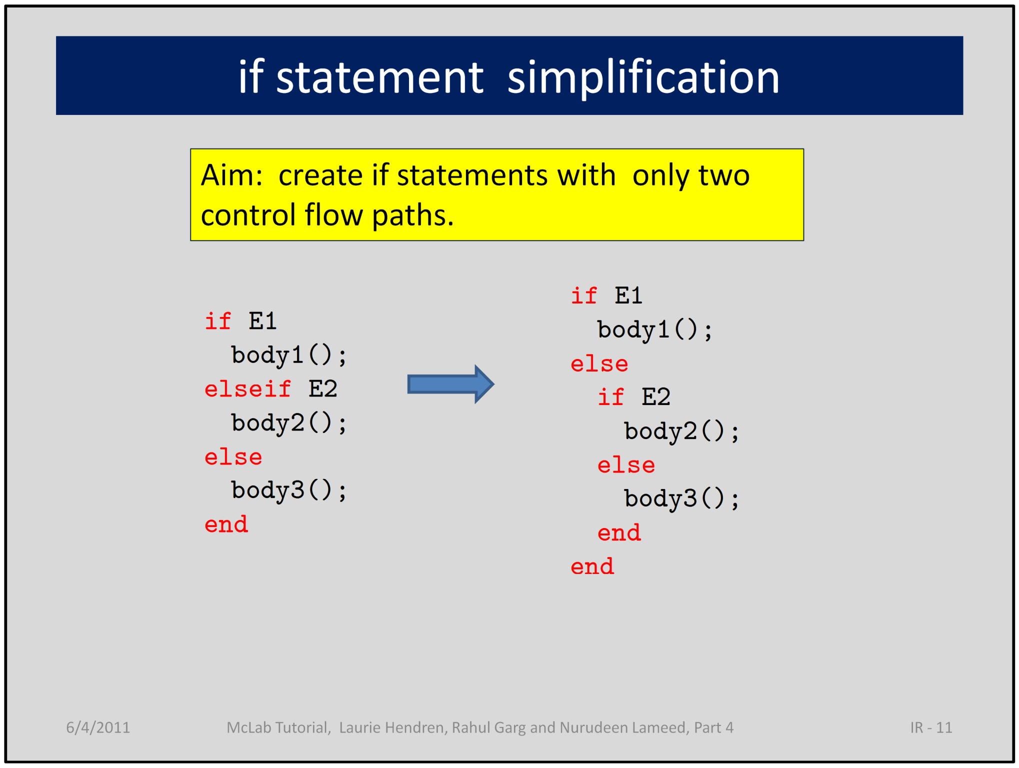
$\Rightarrow$

```
t1 = a+b;
t2 = foo();
A(t1,2).e(t2) = value;
```

Note: no mechanism for taking the address of location in MATLAB. Further simplification not possible, while still remaining as valid MATLAB.

6/4/2011 McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed, Part 4 IR - 10

Simplifying l-values is a bit tricky in MATLAB. We want to break down the computation of l-values as much as possible However, MATLAB has no way of taking the address of variables, so we can only simplify these expressions to a limited extent.



if statement simplification

Aim: create if statements with only two control flow paths.

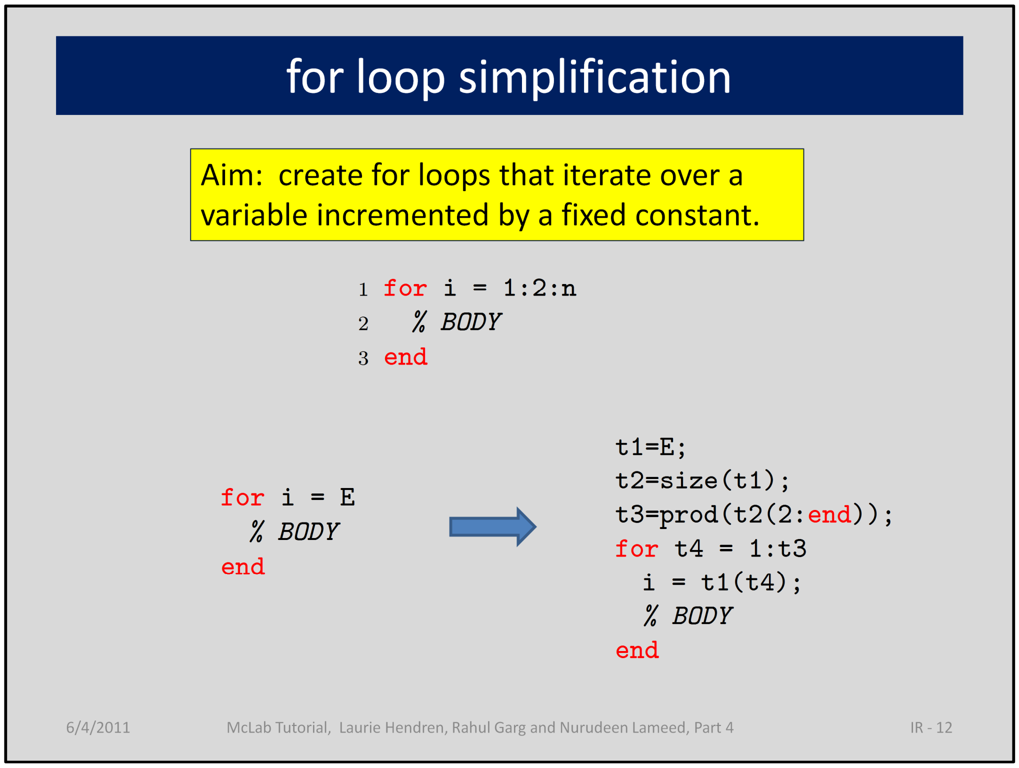
```
if E1
  body1();
elseif E2
  body2();
else
  body3();
end
```

$\Rightarrow$

```
if E1
  body1();
else
  if E2
    body2();
  else
    body3();
  end
end
```

6/4/2011 McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed, Part 4 IR - 11

MATLAB if statements have multiple possible elseif branches. To simplify subsequent flow analysis, we convert these into if-then-else which have at most two branches.



for loop simplification

Aim: create for loops that iterate over a variable incremented by a fixed constant.

```
for i = 1:2:n
  % BODY
end
```

```
for i = E
  % BODY
end
```

```
t1=E;
t2=size(t1);
t3=prod(t2(2:end));
for t4 = 1:t3
  i = t1(t4);
  % BODY
end
```

6/4/2011 McLab Tutorial, Laurie Hendren, Rahul Garg and Nurudeen Lameed, Part 4 IR - 12

The semantics of a for loop are that the rhs expression of the header is evaluated to form a vector, and then the body of the loop is executed is executed over each of the elements of this vector. If it is higher-dimensional array, then it loops through the columns. When the rhs expression is very simple, generated by the colon operator, then the values of i are quite simple to reason about and will behave like an ordinary induction variable. However, when the rhs is some arbitrary other expression E, then we convert the loop to something that does use a simple colon expression in the head of the loop.