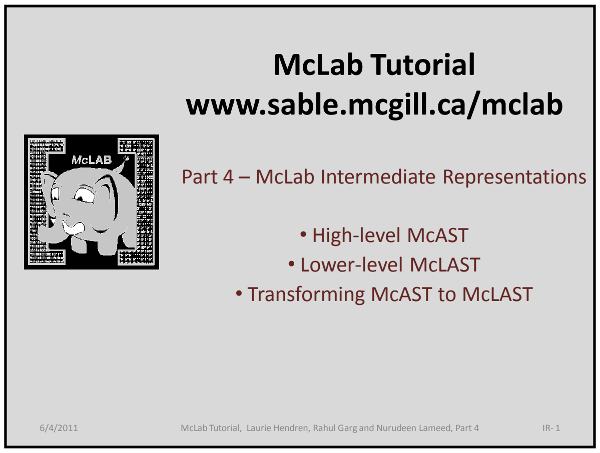
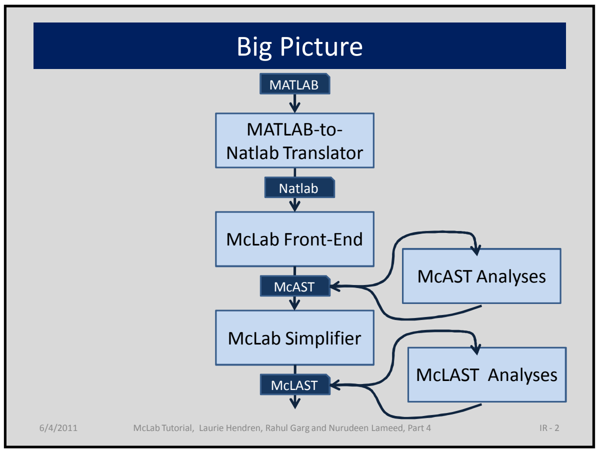


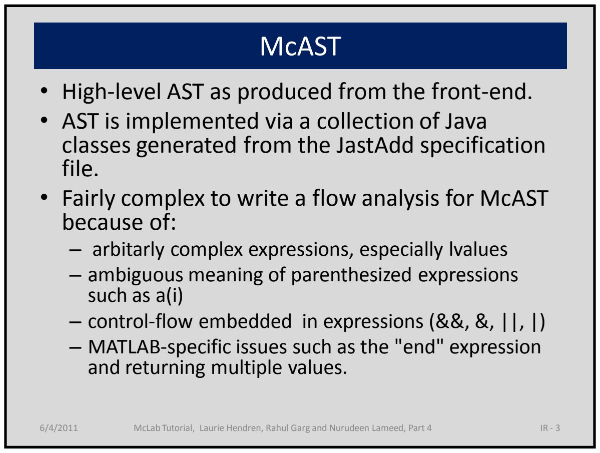
McLab Tutorial
www.sable.mcgill.ca/mclab

Part 4 – McLab Intermediate Representations

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

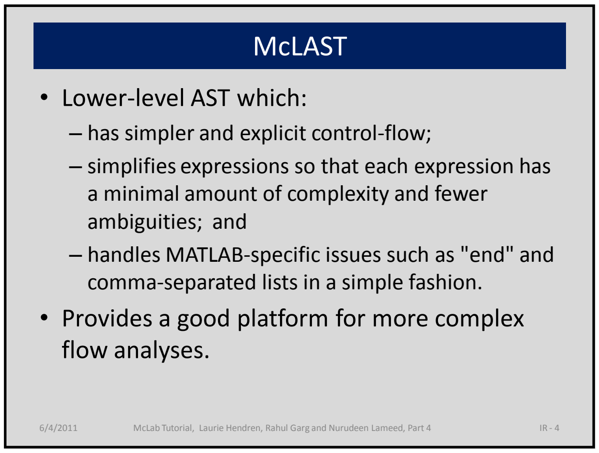


Big Picture



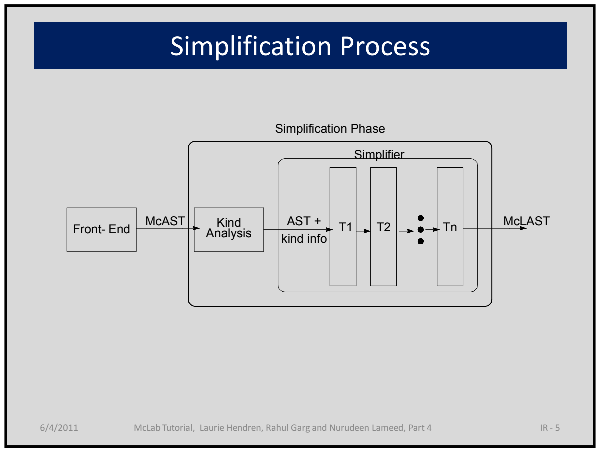
McAST

- 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:
  - arbitaly 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.

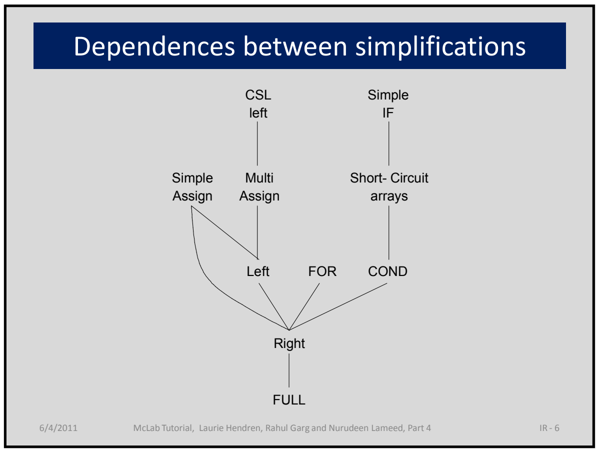


McLAST

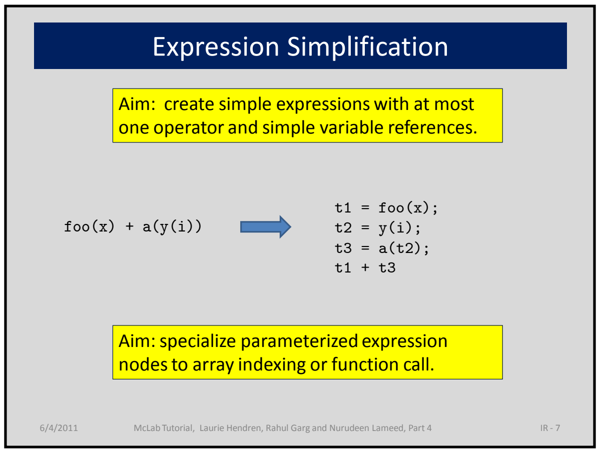
- Lower-level AST which:
  - has simpler and explicit control-flow;
  - simplifies expressions so that each expression has a minimal amount of complexity and fewer ambiguities; and
  - handles MATLAB-specific issues such as "end" and comma-separated lists in a simple fashion.
- Provides a good platform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



Simplification Process



Dependences between simplifications



Expression Simplification

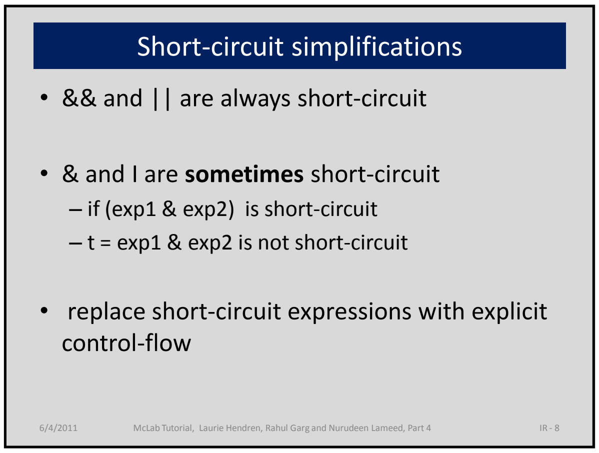
Aim: create simple expressions with at most one operator and simple variable references.

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

$\Rightarrow$

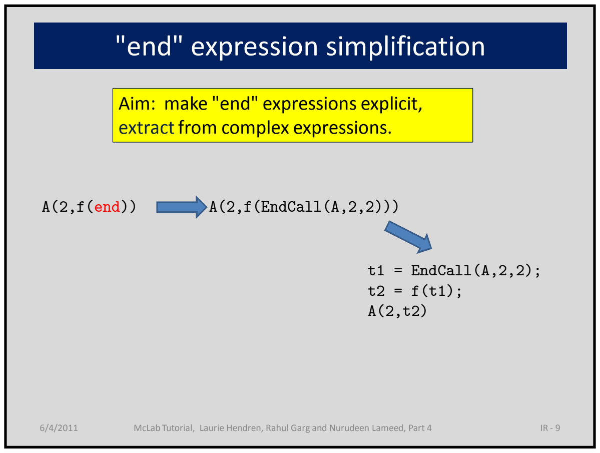
```
t1 = foo(x);
t2 = y(i);
t3 = a(t2);
t1 + t3
```

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



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



"end" expression simplification

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

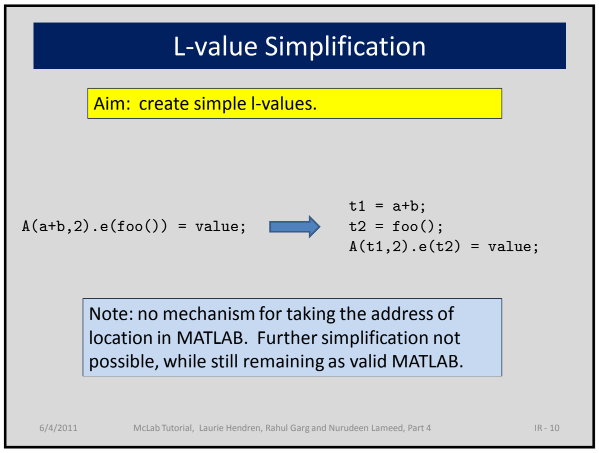
```
A(2,f(end))
```

$\Rightarrow$

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

$\Rightarrow$

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



L-value Simplification

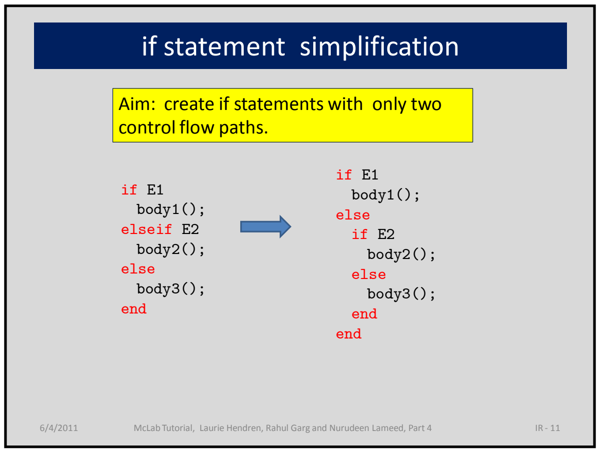
Aim: create simple l-values.

```
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.



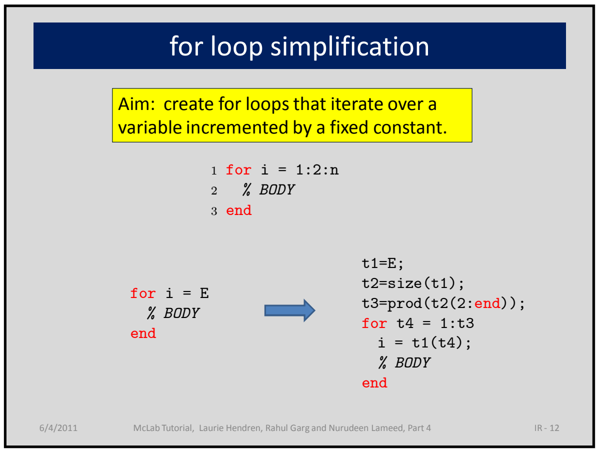
if statement simplification

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
```



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
```

$\Rightarrow$

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