**Short-Circuit Evaluation**

For correct semantics, we express &&,|| using ?: operator:

x && y === x ? y : false

x || y === x ? true : y

Resulting translation:

[[ x && y ]] =

[[ x ]]

ifeq nFalse

[[ y ]]

goto nAfter

nFalse: iconst\_0

nAfter:

Applying the rule above, we obtain

[[ (x && y) && z ]] =

[[ x ]]

ifeq nFalse1

[[ y ]]

goto nAfter1

nFalse1: iconst\_0

nAfter1: ifeq nFalse2

[[ z ]]

goto nAfter2

nFalse2: iconst\_0

nAfter2:

The above is correct

* example below is a little more compact

**Example**

class Test {

boolean and(boolean p, boolean q, boolean r) {

return p && q && r;

}

boolean or(boolean p, boolean q, boolean r) {

return p || q || r;

}

}

Translation:

boolean and(boolean, boolean, boolean);

Code:

0: iload\_1

1: ifeq 16

4: iload\_2

5: ifeq 16

8: iload\_3

9: ifeq 16

12: iconst\_1

13: goto 17

16: iconst\_0

17: ireturn

boolean or(boolean, boolean, boolean);

Code:

0: iload\_1

1: ifne 12

4: iload\_2

5: ifne 12

8: iload\_3

9: ifeq 16

12: iconst\_1

13: goto 17

16: iconst\_0

17: ireturn