Generating Code for Control Structures

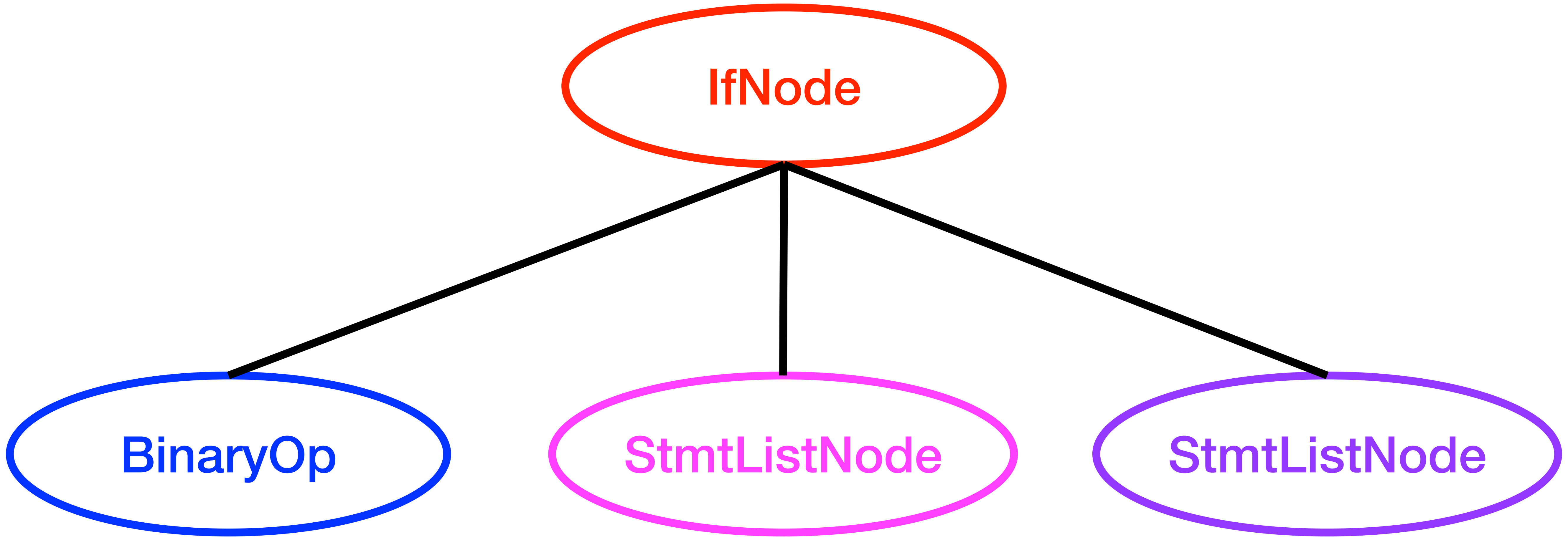
code generation

- Generating code for control structures works the same as generating code for statements and expressions: generate the code bottom-up
    - Generate code for the sub-components before “gluing” the code together to create code for overall control structure

- Two key challenges:
    - Generating **labels** for branch targets
    - Generating code for **conditionals**

if statements

```
if (<cond_expr>) {
 <stmt_list_1>
} else {
 <stmt_list_2>
}
```



if statements

```
if (<cond_expr>) {
 <stmt_list_1>
} else {
 <stmt_list_2>
}
```

→

```
<cond_expr>
b<!op> l_else
<stmt_list_1>
j l_end
l_else:
<stmt_list_2>
l_end:
```

if statements—problem 1

- Labels need to be unique
- Code generator needs to keep track of what labels have been used (similar to keeping track of which virtual registers have been used)
- Tip: give labels human-readable names (lab_end, not lab_029) to make it easier to debug

```
<cond_expr>
b<!op> l_else
<stmt_list_1>
j l_end
l_else:
<stmt_list_2>
l_end:
```

if statements—problem 2

- branch type depends on comparison operation, branch target depends on labels
- Two possible solutions:
  - Generate labels in code generator *prefix* (before stepping into conditional expression subtree) → be careful, because "valence" of branch can depend on how the conditional is used
  - Patch up code block for conditional when stitching the code blocks together → be careful, because branch type depends on the comparison operator

```
<cond_expr>
b<!op> l_else
<stmt_list_1>
j l_end
l_else:
<stmt_list_2>
l_end:
```