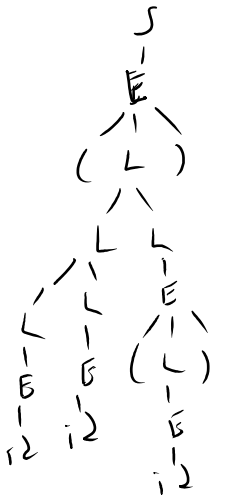
1)

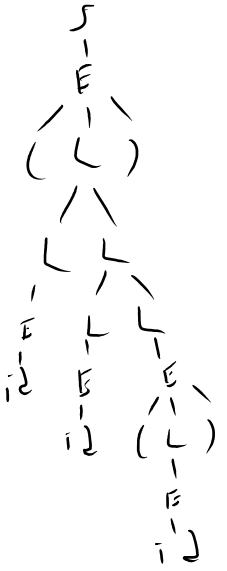
```
S -> E
E -> ( L ) | id
L -> L L | E
```

a) ( id id ( id ) )

Leg. 1: S S R3 R5 S R3 R5 R4 S S R3 R5 S R2 R5 R4 S R2 R1 A

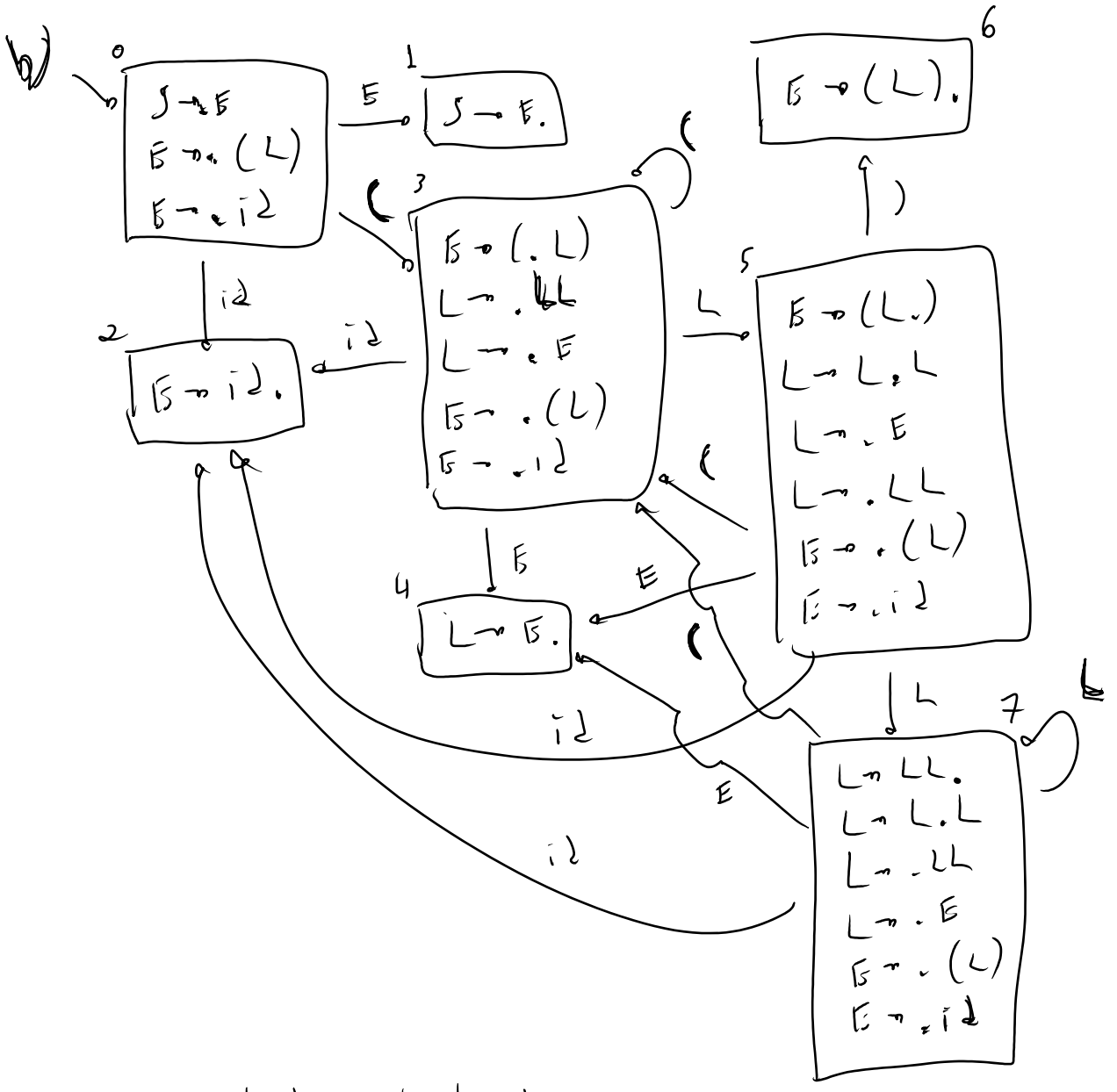


Leg. 2: S S R3 R5 S R3 R5 S S R3 R5 S R2 R5 R4 R4 S R2 R1 A



Cada sequência corresponde a uma árvore, logo existem duas árvores para a mesma entrada, logo a gramática é ambígua.

b)




| | id | ( | ) | eof | S | E | L |
|---|---|---|---|---|---|---|---|
| 0 | S2 | S3 | | | | 1 | |
| 1 | R1 | R1 | R1 | R1 | | | |
| 2 | R3 | R3 | R3 | R3 | | | |
| 3 | S2 | S3 | | | | 4 | 5 |
| 4 | R5 | R5 | R5 | R5 | | | |
| 5 | S2 | S3 | S6 | | | 4 | 7 |
| 6 | R2 | R2 | R2 | R2 | | | |
| 7 | R4 | R4 | R4 | R4 | | 4 | 7 |

Dois conflitos shift/reduce no estado 7, em id e (, resolvidos com reduce, poderiam ter sido resolvidos shift 2 e shift 3, respectivamente

2)

a)

```
class Throw implemets Cmd {
  String id;

  public <C, R> R accept(Visitor<C, R> vis, C ctx) {
    return vis.visit(this, ctx);
  }
}

class TryCatch implements Cmd {
  Cmd ctry;
  String tex;
  String nex;
  Cmd ccatch;

  public <C, R> R accept(Visitor<C, R> vis, C ctx) {
    return vis.visit(this, ctx);
  }
}
```

b)

```
public Set<String> visit(Throw no, SymbolTable<String> ctx) {
  Set<String> texs = new HashSet<>();
  texs.add(no.id);
  return texs;
}

public Set<String> visit(TryCatch no, SymbolTable<String> ctx) {
  Set<String> texs = new HashSet<>();
  texs.addAll(no.ctry.accept(this, ctx));
  texs.remove(no.tex);
  SymbolTable<String> ctxcatch = new SymbolTable<>(ctx);
  ctxcatch.put(no.nex, no.tex);
  texs.addAll(no.ccatch.accept(this, ctxcatch));
  return texs;
}
```