

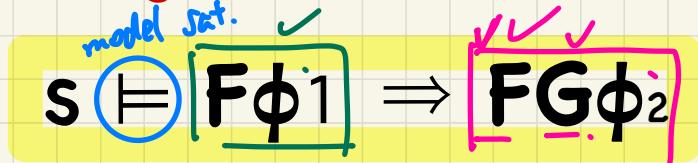
Lecture 12 - March 2

Model Checking

Path Satisfaction: Nested LTL Operators

$$F \phi_1 \Rightarrow FG \phi_2$$

Nesting "Global" and "Future" in LTL Formulas



Each path π starting with s is s.t. if eventually ϕ_1 holds on π , then ϕ_2 eventually holds on π continuously.

Q. Formulate the above nested pattern of LTL operators.

$$\begin{aligned} * \forall \pi \cdot \pi = S \rightarrow \dots \Rightarrow & \\ (\exists i_1 \cdot i_1 > | \wedge \pi^{i_1} \models \phi_1) & \\ (* \Rightarrow (\exists i_2 \cdot i_2 > | \wedge (\forall j \cdot j > i_2 \Rightarrow & \\ \pi^j \models \phi_2))) & \end{aligned}$$

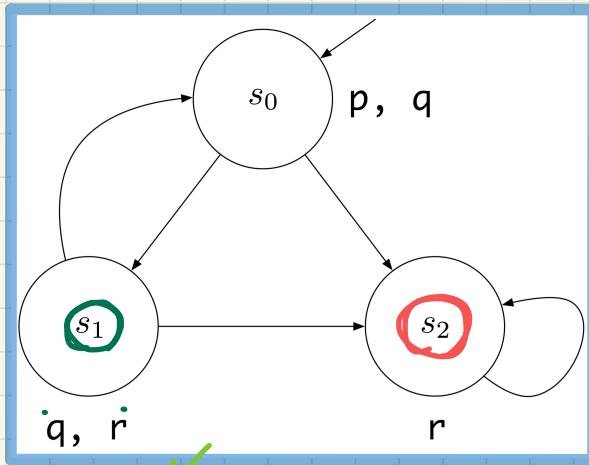
Q. How to prove the above nested pattern of LTL operators?

① Consider all path patterns ② a. $T \Rightarrow T$ b. $F \Rightarrow -$ c. $- \Rightarrow T$

Q. How to disprove the above nested pattern of LTL operators?

① Find a witness path $\Rightarrow T \Rightarrow F$.

Alt to *: s_2 satisfies $\neg q \wedge r$, then from s_2 ✓ is don't (4) $s_0 \rightarrow s_1 \rightarrow s_0 \rightarrow s_1 \rightarrow \dots \rightarrow s_2 \rightarrow \dots$
Path Satisfaction: Exercises (5.2) (5) $s_0 \rightarrow s_1 \rightarrow s_0 \rightarrow s_1 \rightarrow \dots \rightarrow s_0 \rightarrow s_2 \rightarrow \dots$



(2) $s_0 \rightarrow \boxed{s_2} \rightarrow s_2 \rightarrow \dots$

\hookrightarrow satisfies $\neg q \wedge r$

$\hookrightarrow F(\neg q \wedge r) \text{ is } \top$

Starting from s_2 , $G r$ is satisfied. $\top \Rightarrow \top = \top$

Exercise: What if we change the LHS to s_2 ?

$s \models \phi \Leftrightarrow \text{all } \pi \text{ starting at } s, \pi \models \phi$

(3) $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \dots$ (exercise)
 $s_0 \models F(\neg q \wedge r) \Rightarrow \underline{FG r}$ \top.

① $s_0 \rightarrow s_1 \rightarrow s_0 \rightarrow s_1 \rightarrow \dots$

\hookrightarrow no state in this path satisfies $\neg q \wedge r$

$\hookrightarrow F(\neg q \wedge r)$ is false

false $\Rightarrow P \equiv \boxed{\text{True}}$.

$s_0 \models F(\neg q \vee r) \Rightarrow \underline{FG r}$ \top.

Witness: \top $s_0 \rightarrow s_1 \rightarrow s_0 \rightarrow s_1 \rightarrow \dots$

\hookrightarrow satisfies $F(\neg q \vee r)$: s_1

\hookrightarrow violates: $FG r$: s_0
does not satisfy r.

$F G \phi$



$G E \phi$



Lab2 Solution: getAllSuffixes (V2: Tuple of Tuples)

```
----- MODULE getAllSuffixes_v2 -----
EXTENDS Integers, Sequences, TLC
CONSTANT input
ASSUME Len(input) > 0
(*
--algorithm getAllSuffixes_v2 {
    variable result = input, postfixSoFar = <<>>, i = Len(input) - 1;
    { outputs will be variables
        postfixSoFar := <<input[Len(input)]>>;
        result[Len(input)] := postfixSoFar;
        while (i > 0) {
            postfixSoFar := <<input[i]>> \o postfixSoFar;
            result[i] := postfixSoFar;
            i := i - 1;
        };
        assert \A j \in 1..Len(input): Len(result[j]) = Len(input) - j + 1;
        assert \A j \in 1..Len(result): \A k \in 1..Len(result[j]): result[j][k] = input[j - 1 + k];
    }
}
*)
```

input vars will be constants
↳ need to be instantiated for model checking.

input: [23, 46, 69]
result: len(input)

[[23, 46, 69],
 [46, 69],
 [69]]
I 3 1 3
3 2 2
3 3 1

IE

----- MODULE getAllSuffixes_v2 -----

EXTENDS Integers, Sequences, TLC

CONSTANT input

ASSUME Len(input) > 0

*

--algorithm getAllSuffixes_v2 {

 variable result = input, postfixSoFar = <>>, i = Len(input) - 1;

{

 postfixSoFar := <<input[Len(input)]>>;

 result[Len(input)] := postfixSoFar;

 while (i > 0) {

 postfixSoFar := <<input[i]>> \o postfixSoFar;

 result[i] := postfixSoFar;

 i := i - 1;

 };

 assert $\forall j \in 1..Len(input)$: Len(result[j]) = Len(input) - j + 1;

 assert $\forall j \in 1..Len(result)$: ($\forall k \in 1..Len(result[j])$: result[j][k] = input[j - 1 + k]);

}

*

(j)

1 K = 1, 2, 3

2 K = 1, 2

3 K = 1

(Len(result[j]))

input: [23, 46, 69]

result: use K to refer to an item in tuple.

[23, 46, 69],

I

2

3

[46, 69],

1

[69]

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

Lab2 Solution: getRightShifts

----- MODULE getRightShifts -----

```

EXTENDS Integers, Naturals, Sequences, TLC
CONSTANT input, n
ASSUME  $\wedge \text{Len}(\text{input}) > 0$ 
 $\wedge n \geq 0$ 

(*
--algorithm getRightShifts {
    variable result = input, nos =  $n \% \text{Len}(\text{input})$  (* number of shifts *), i = 1;
{
    while ( $i \leq \text{Len}(\text{input})$ ) {
        if  $((i + \text{Len}(\text{input})) - nos) \% \text{Len}(\text{input}) = 0$  {
            result[i] := input[Len(input)];
        } else {
            result[i] := input[((i + Len(input)) - nos) \% Len(input)];
        };
        i := i + 1;
    };
    /* version 1 of postcondition: for each index in the result, what is the corresponding index in the input?
    assert  $\forall j \in 1..Len(result)$ : IF  $((j + Len(result)) - (n \% Len(result))) \% Len(result) = 0$ 
        THEN result[j] = input[Len(input)]
        ELSE result[j] = input[((j + Len(input)) - (n \% Len(input))) \% Len(input)];

    /* version 2 of postcondition: for each index in the input, what is the corresponding index in the result?
    assert  $\forall j \in 1..Len(input)$ : IF  $(j + (n \% Len(input))) \% Len(input) = 0$ 
        THEN result[Len(input)] = input[j]
        ELSE result[((j + (n \% Len(input))) \% Len(input))] = input[j];
}
*)

```

**IF B
THEN P
ELSE Q**

shift to R by one pos.

input: [23, 46, 69]
result: [69, 23, 46]

input $\xrightarrow{+1}$ **output**

1	2	2
2	3	3
3	4	1

Assertion : explicit about the variables
that can be used.

`<< 1,2,3 >> \o << 4,5,6 >>`

`output1 \o << t >>`