CIS 842: Specification and Verification of Reactive Systems

Lecture Specifications:

Progress Properties

Objectives

- To understand the essential difference between safety and liveness properties
- To understand the algorithm used to check for progress properties

Cyclic Behavior

- It is clear from looking at nearly all of our examples that systems can cycle indefinitely
  - e.g., dining philosophers
- This is a characteristic of reactive systems
- We will want to be able to characterize the fact that we expect the system to eventually perform some action

Safety Properties

Are fundamentally about not reaching certain undesirable states

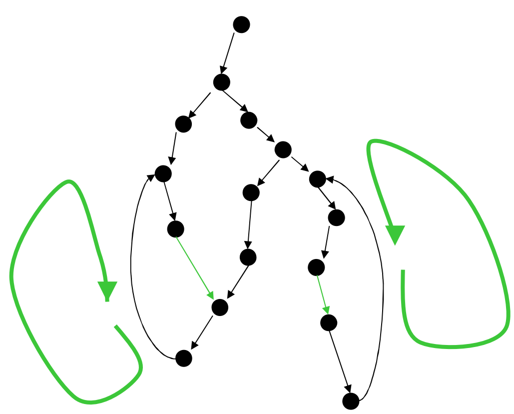
Progress …

- Intuitively means that the system eventually will *do* something
- From every state we should be able to make progress

Specifying Progress

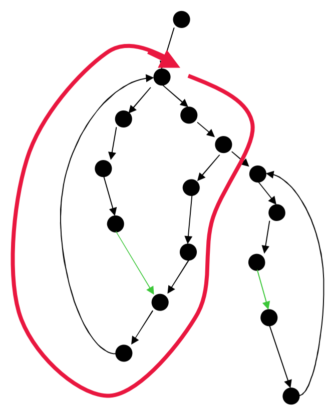
- A simple way to designate progress is to name labels of actions that should eventually be performed
- For example
  - {Philosopher1.eating, Philosopher2.eating}
- Property states
  - From all states in the system, eventually a (all) progress labeled action (s) will be executed

Progress



Progress Violations

- A cyclic behavior on which no progress label occurs



Checking for Progress

- Reachability works well for predecessors of progress actions
- Cycle detection works well for successors of progress actions
- Need to combine both checks in a single algorithm

Recall : DFS Algorithm

1 *seen* $:= \{s_0\}$
2 $pushStack(s_0)$
3 $DFS(s_0)$

$DFS(s)$
4 $workSet(s)$:= *enabled*(s)
5 while $workSet(s)$ is not empty
6 let $\alpha \in workSet(s)$
7 $workSet(s) := workSet(s) \setminus \{\alpha\}$
8 $s' := \alpha(s)$
9 if $s' \notin$ *seen* then
10 *seen* := *seen* $\cup \{s'\}$
11 $pushStack(s')$
12 $DFS(s')$
13 $popStack()$
end DFS

Nested DFS Algorithm

1 *seen* $:= \{s_0\}$
2 $pushStack(s_0)$
3 $DFS(s_0)$

Deleted stack maintaining statements for brevity

$DFS(s)$
4 $workSet(s)$:= *enabled*(s)
5 while $workSet(s)$ is not empty
6 let $\alpha \in workSet(s)$
7 $workSet(s) := workSet(s) \setminus \{\alpha\}$
8 $s' := \alpha(s)$
9 if $s' \notin$ *seen* then
10 *seen* := *seen* $\cup \{s'\}$
12 $DFS(s')$
13.1 if α is not progress then
13.2 $NDFS(s', s')$
end DFS

Nested DFS Algorithm

```
NDFS(s, seed)
workSet2(s) := enabled(s)
while workSet2(s) is not empty
    let α ∈ workSet2(s)
    if α is progress
      return
    workSet2(s) := workSet2(s) \ {α}
    s' := α(s)
    if s' =seed then
      Non-progress cycle detected
    if s' ∉ seen' then
        seen' := seen' ∪ {s'}
        NDFS(s',seed)
end DFS
```

milar to DFS (with
parate data structures)

For You To Do

- Take the dining philosophers example, with eating progress labels and apply the nested DFS algorithm to it
- Do you find an error?