Online Replanning

Section 11.3.3

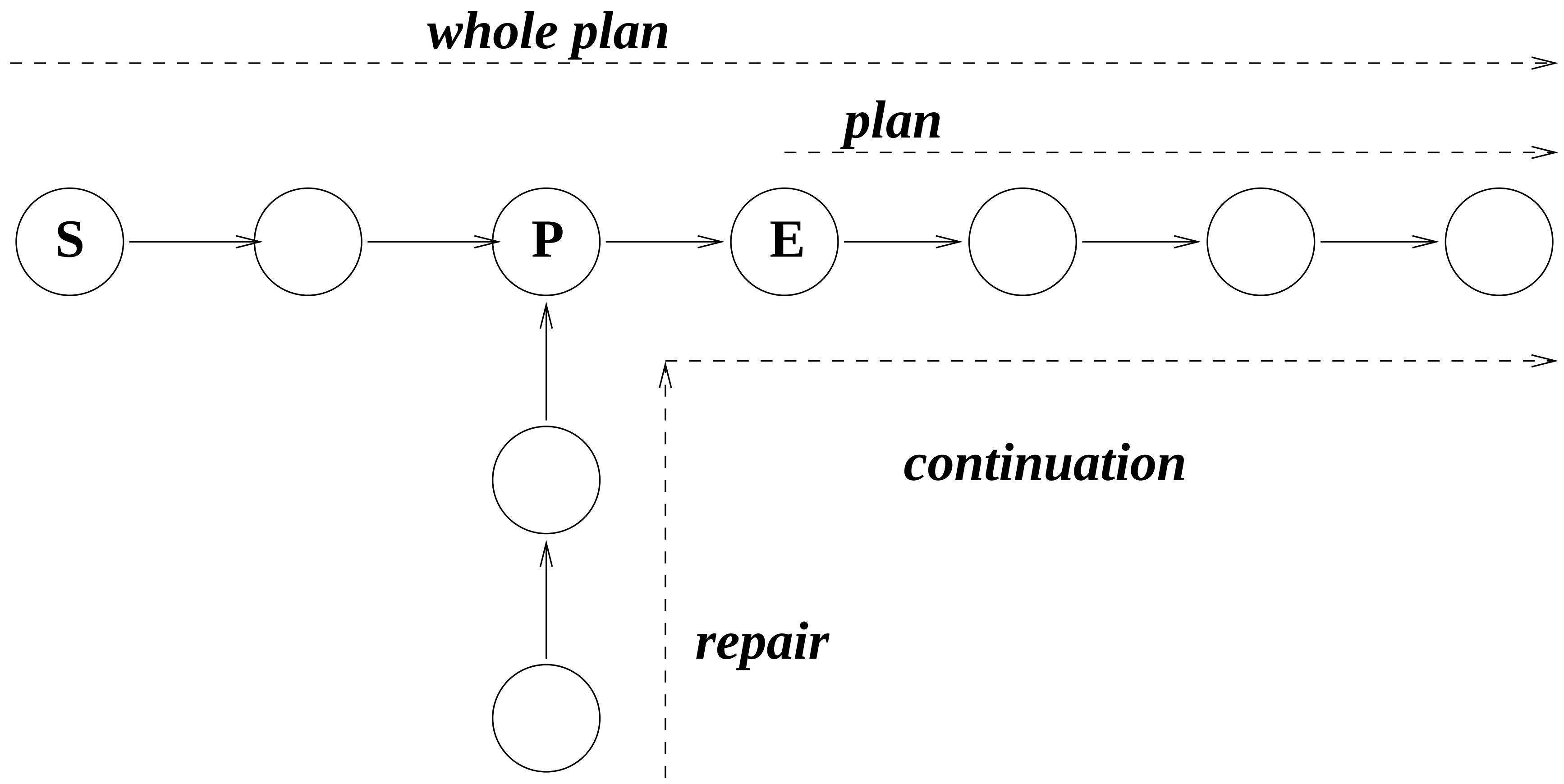
Outline

- Contingency planning vs. replanning
- Replanning agent algorithm
- Execution monitoring
- Continuous planning
- (Multiagent planning)

Contingency planning vs. replanning

- *Contingency planning*: prepare in advance. Useful when some conditions needed for the contingency plan can be gathered before execution.
- *Execution monitoring*: ignore contingencies during planning, then handle them as they arise. Useful when planning time is a concern: not everything can be planned for.
- Basic idea: handle execution time failures at execution time.

Repairing a plan



Chair and table example

Init(Color(Chair,Blue) ∧ Color Table(Green)
  ∧ ContainsColor(BC,Blue) ∧ PaintCan(BC)
  ∧ ContainsColor(RC,Red) ∧ PaintCan(RC))

Goal(Color(Chair,x) ∧ Color(Table,x))

Action(Paint(object,color),
  PRECOND: HavePaint(color)
  EFFECT: Color(object,color))

Action(Open(can),
  PRECOND: PaintCan(can) ∧ ContainsColor(can,color)
  EFFECT: HavePaint(color))

Chair and table example (cont'd)

Whole plan: [Start; Open(BC); Paint(Table,Blue); Finish]

What to do when

- it notices a missed green spot on the table just before finishing
- the agent plans to paint both red and it opens the can of red paint and finds there is only enough paint for the chair.

Algorithm

```
function REPLANNING AGENT(percept) returns an action
  static: KB, a knowledge base (includes action descriptions)
          plan, a plan, initially []
          whole-plan, a plan, initially []
          goal, a goal

  TELL (KB, MAKE-PERCEPT-SENTENCE (percept,t))
  current ← STATE-DESCRIPTION (KB, t)
  if plan = [] then
    whole-plan ← plan ← PLANNER(current, goal, KB)
  if PRECONDITIONS(FIRST (plan)) not currently true in KB then
    candidates ← SORT(whole-plan, ordered by distance to current)
    find state s in candidates such that
      failure ≠ repair ← PLANNER (current,s,KB)
    continuation ← the tail of whole-plan starting at s
    whole-plan ← plan ← APPEND(repair, continuation)
  return POP(plan)
```

What to monitor, what to ignore

- *Action monitoring:* Check the preconditions of the next action to execute
- *Plan monitoring:* Check the preconditions of all the actions to execute
- monitor a selected set based on priority
- Look for opportunities (*serendipity*)

Other important questions

- Which contingencies to plan for, which ones to leave until execution
- Should replanning be a plan step
- learning/modifying actions
- side note: “don’t touch” conditions

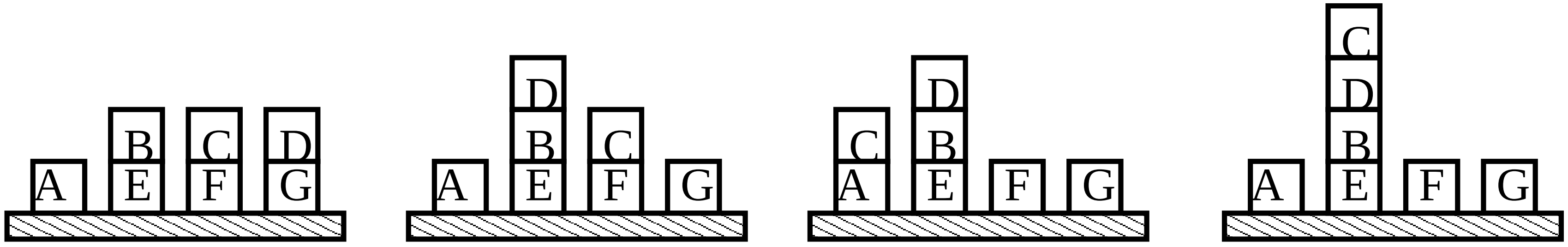
Fixing plan flaws continually

- *Missing goal*: adding new goals
- *Open precondition*: close using causal links (POP)
- *Causal conflict*: resolve threats (POP)
- *Unsupported link*: remove causal links supporting conditions that are no longer true
- *Redundant action*: remove actions that supply no causal links
- *Unexecuted action*: return an action that can be executed
- *Unnecessary historical goal*: if the current goal set has been achieved, remove them and allow for new goals

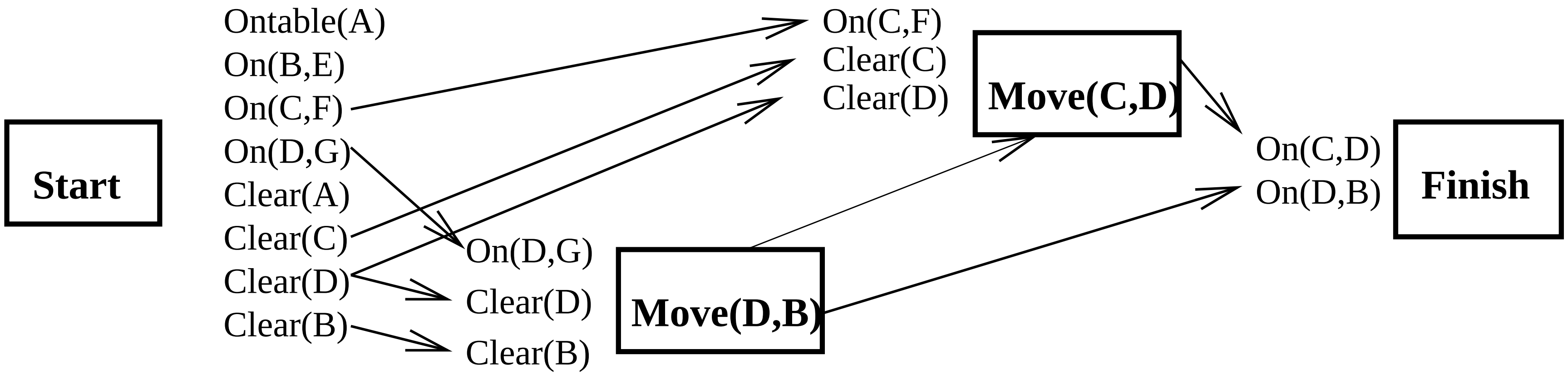
Continuous planning algorithm

```
function CONTINUOUS-POP-AGENT(percept) returns an action
    action ← NoOp (the default)
    EFFECTS[Start] = UPDATE(EFFECTS [Start], percept)
    REMOVE-FLAW(plan)     // possibly updating action
    return action
```

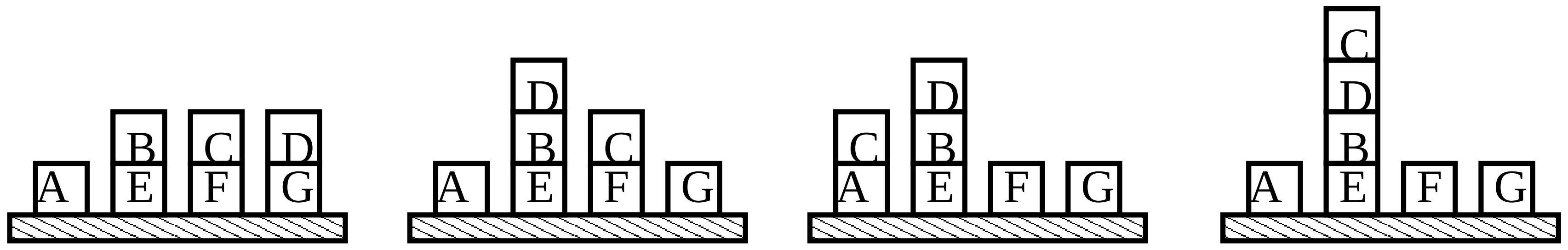
Example - start



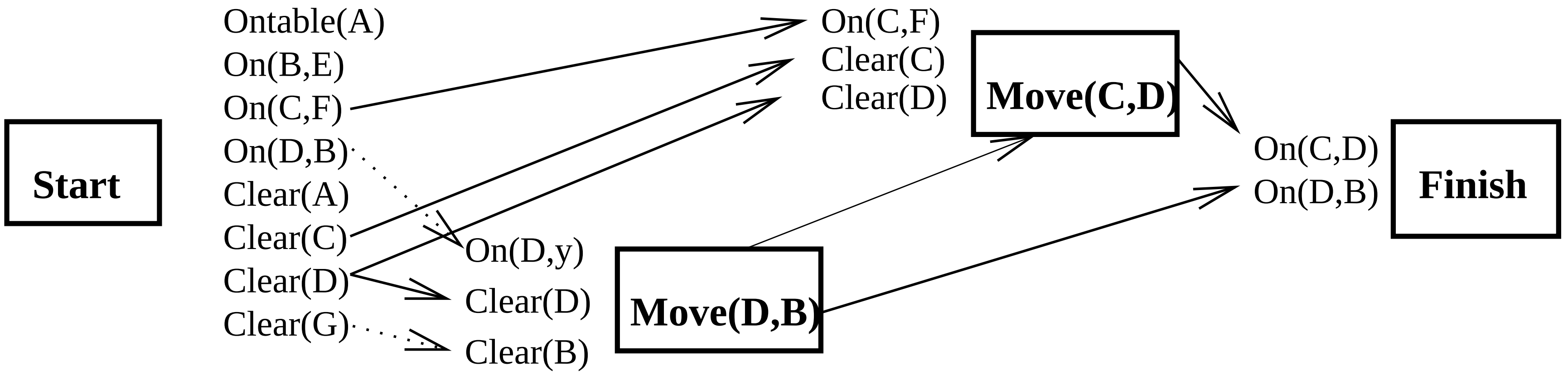
Start **Someone moved D** **dropped C** **tried again**



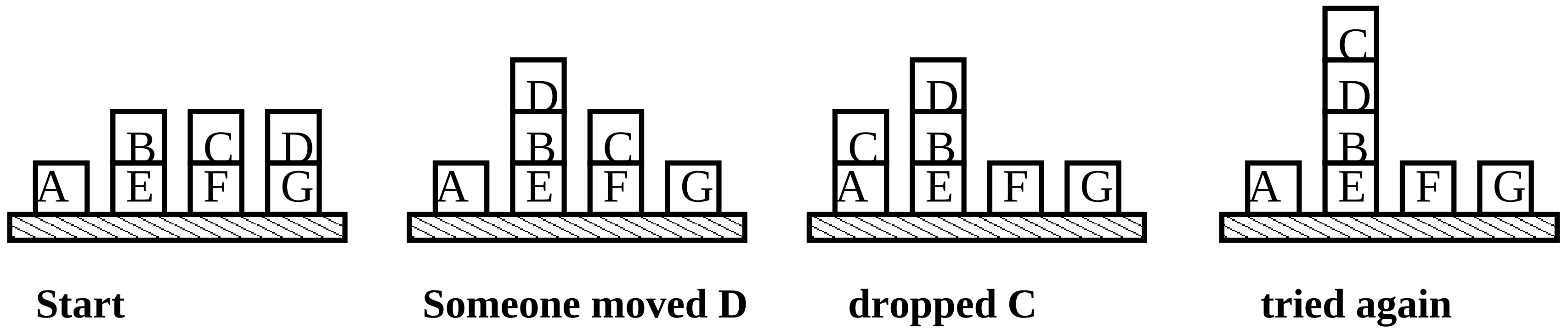
Example - after D is moved onto B

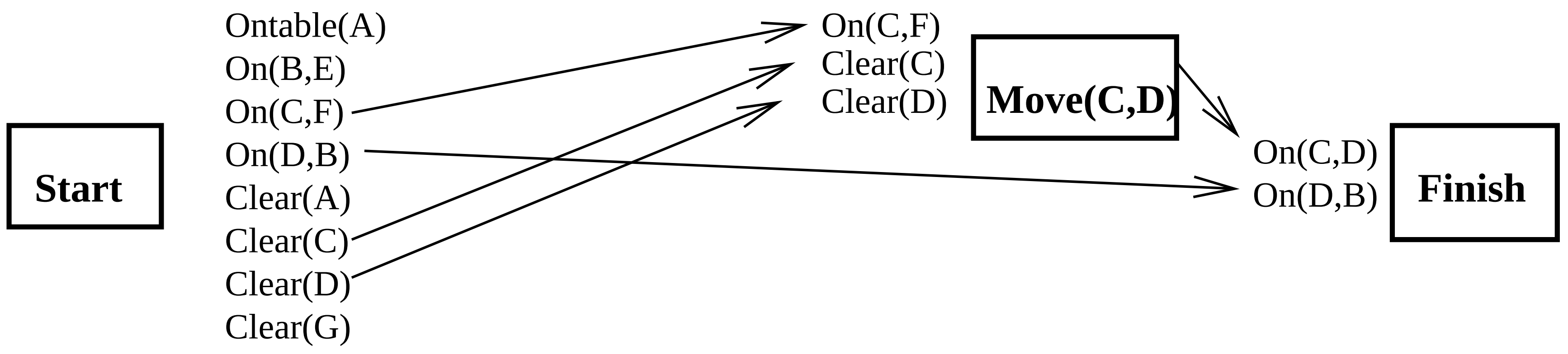


Start **Someone moved D** **dropped C** **tried again**

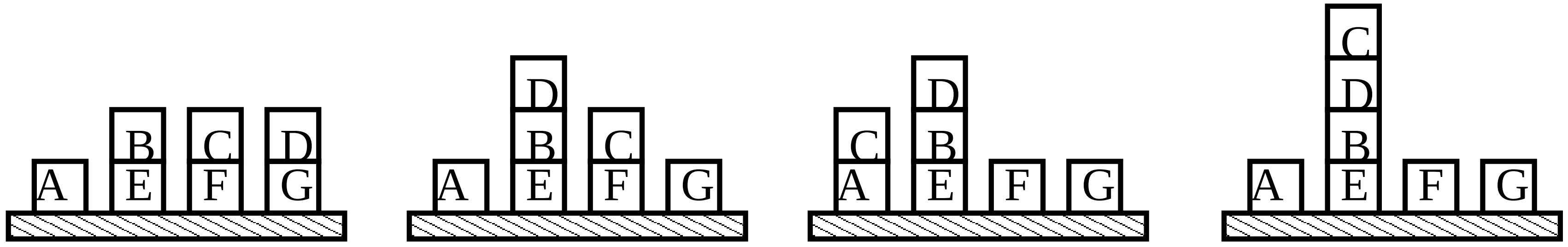


Example - Move(D,B) was redundant

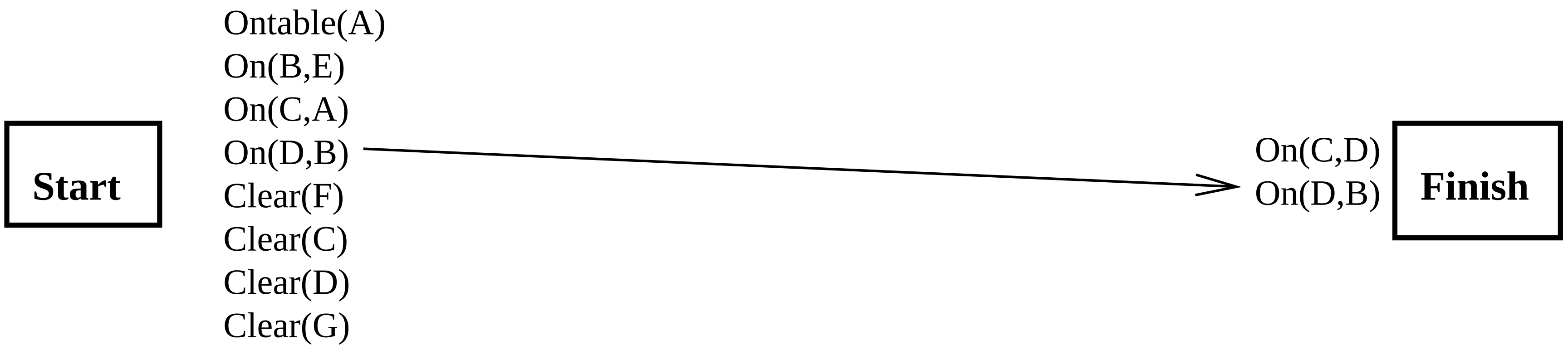




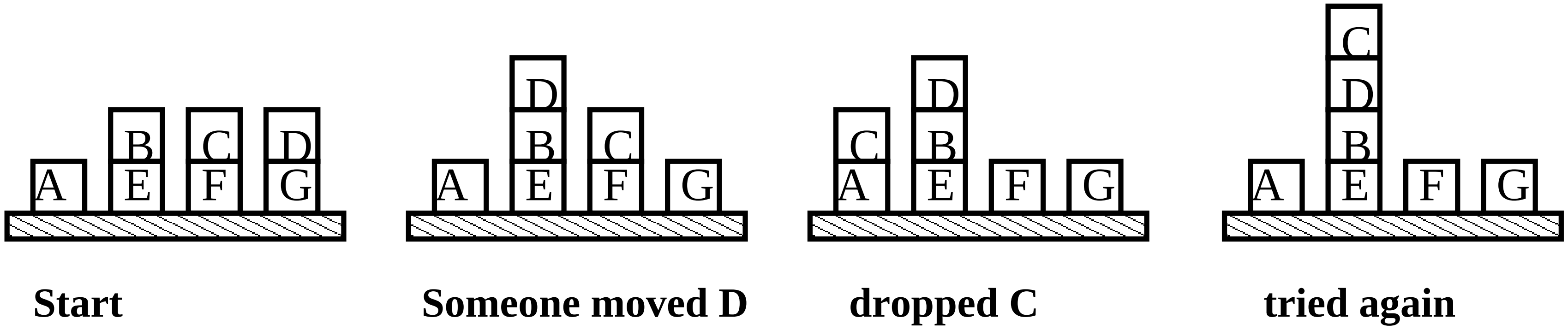
Example - Move(C,D) was executed

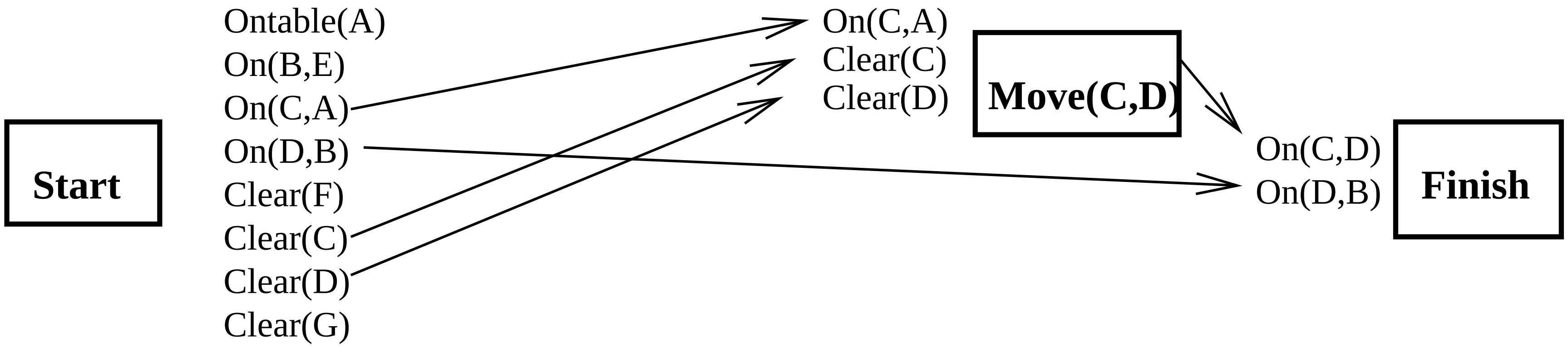


Start **Someone moved D** **dropped C** **tried again**

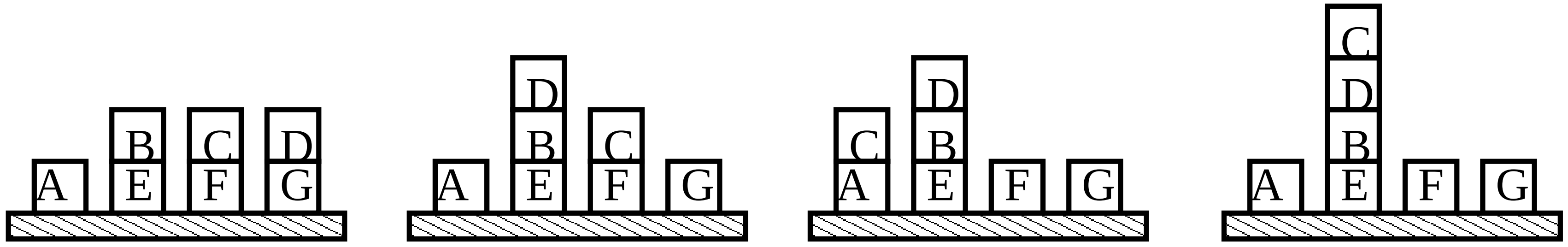


Example - put Move(C,D) back in

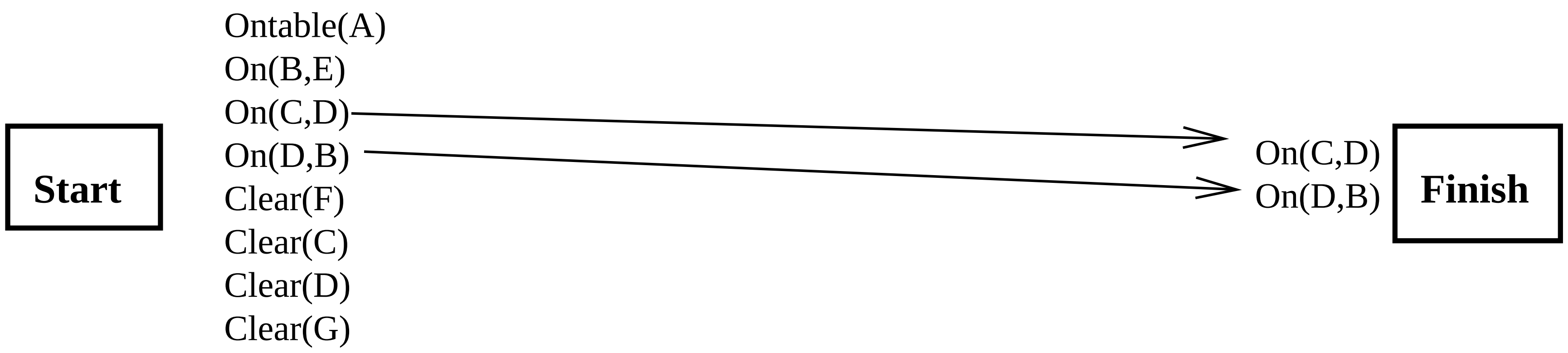




Example - plan complete



Start **Someone moved D** **dropped C** **tried again**



Multiagent planning

- *Cooperation:* Joint goals and plans
- *Multibody planning:* Synchronization, joint actions, concurrent actions
- *Coordination mechanisms:* convention, social laws, emergent behavior, communication, plan recognition, joint intention
- *Competition:* agents with conflicting utility functions