1. Levels and goals

III. From Navigation to Analysis

Giacomo Bergami

Newcaslte University

Objectives

- Characterizing non-stochastic games.
- Characterizing agent planning via completion of intermediate goals.
- Exploiting rule-based planning techniques for checking whether a game is solvable or not.
- Exploiting theorem-proving based techniques for:
  1. debugging unsolvable games, and
  2. identifying different strategies for winning the game
- Characterizing game balance as an interplay of solvability and difficulty.
- Ranking strategies by difficulty.

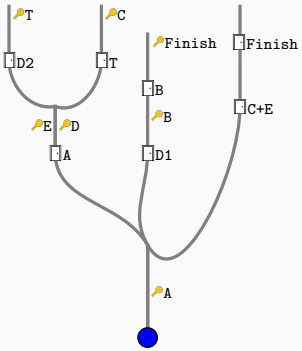
Procedural Generation: a pseudocode

Procedural Generation for locks-and-keys game often start with generating a map. The generation of such a map resembles the generation of a *planar* graph.

```
1: function GENERATE_DATA(#room, prob)
2:     for i ← 1 to #room do                                  ▷ Generating #room rooms.
3:         do r ← RANDOMDOOR() while INTERSECTS(r,rooms)
4:         rooms.add(r)
5:     for i ← 1 to #room do
6:         for j ← 1 to i − 1 do
7:             if random.prob() < prob then continue          ▷ Randomly discarding an edge.
8:             if not INTERSECTS((i ↔ j), corridors) then corridors.add(i ↔ j)
                                                              ▷ Making the level a planar graph
```

After this, we might follow any preferred way to generate locks and keys. But how to guarantee the game's solvability?

Solvability: Locks and Keys (1/2)



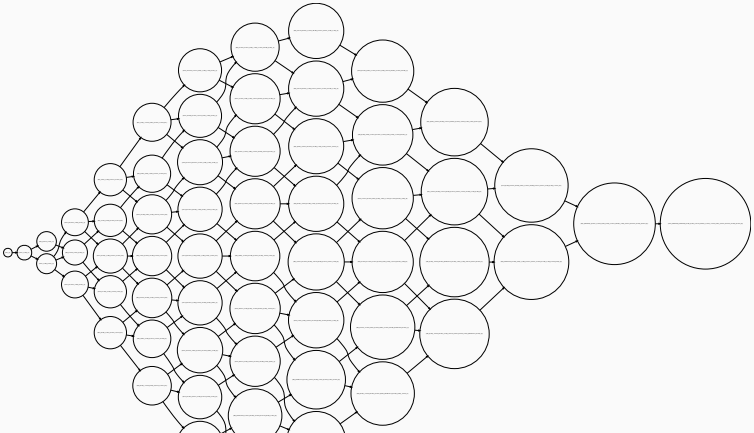
We can encode our level using predicate logic through Horn clauses:

1. Set all the resources accessible from the start as the initial state ($\{\mathrm{Key}_{\mathtt{A}}\}$).
2. Set the final door as the goal ($\mathrm{Door}_{\mathtt{Finish}}$).
3. If I have access to the key $\mathrm{Key}_{\mathtt{B}}$ and to all of the doors leading to $\mathrm{Door}_{\mathtt{B}}$, then i can open the door: e.g., $\mathrm{Key}_{\mathtt{B}} \wedge \mathrm{Door}_{\mathtt{D}} \Rightarrow \mathrm{Door}_{\mathtt{B}}$
4. If I have access to a door, then I have access to all the resources between it and the next obstacles: e.g., $\mathrm{Door}_{\mathtt{A}} \Rightarrow \mathrm{Key}_{\mathtt{E}}$, $\mathrm{Door}_{\mathtt{A}} \Rightarrow \mathrm{Key}_{\mathtt{D}}$

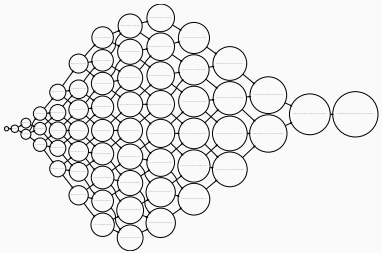
Solvability is generally independent from the player's skills and probabilistic chances. Let's see how we can exploit the navigation rules to reach `Finish`.

State Eplosion with State Machines

Given the first configuration for the Locks and Key generation, if we decide to represent each node as a possible state configuration reached by applying a set of rules, we obtain the following (`GenerateUnweightedPossibleStates<T>::generateUnweightedPossibleStates`):



State Eplosion with State Machines



- While traditional State Machine techniques require to generate all the possible states, logical rules might generate one single graph per strategy.
- Determining the costs over this graph is more challenging, as many other different combinations must be considered.
- In the next example, we will show how exploiting this representation for even a simplistic (but realistic) scenario will lead to an explosion to the number of the possible states.

Solution #1: "Forward" Rule-Based Planning for Solvability (1/2)

`ClosedWorldInference<T>::grounding`:

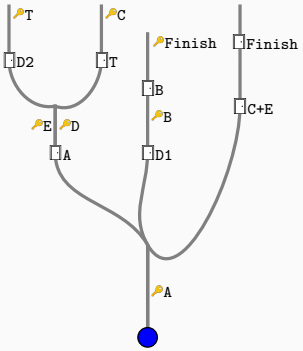
- Each node of the "graph" represents the set of collected resources and solved tasks at a given time in the game. We apply one single tratego, and therefore we might produce one single path!
- The previous approach proves whether the game is solvable by identifying a possible solution.
- We do not generate multiple possible strategies, rather than *we apply systematically each rule* until we can meet our goal.
- This same strategy could be adopted to enact rational reasoning to a NPC agent.
- The game cost is not necessarily the one with optimal cost (**Exercise**: *greedy heuristic over costs* for short-sighted strategies)!

Solution #1: "Forward" Rule-Based Planning for Solvability (2/2)

Given that we have a "simple logic" (we restrict to properties P Q, and $\bigwedge_i P_i \Rightarrow Q$, and **no** negations), we can use an algorithm to automate the boolean check (that can be optimized!):

1: **function** SOLVABILITY_TEST($Init, Goal, Rules$)
2: **if** $Goal \subseteq S$ **then return true**
3: **else** ▷ Grounding()
4: $S := Init; \quad Tmp := S$
5: **repeat**
6: $S = Tmp$
7: **for all** $H \Rightarrow T \in Rules$ **s.t.** $H \subseteq Tmp \wedge T \notin Tmp$ **do** ▷ For each rule applicable to the resources
8: $Tmp := Tmp \cup \{T\}$ ▷ Add the activated resources back to Tmp
9: **until** $Tmp = S$ ▷ Continue until I cannot further expand
10: **return** $Goal \subseteq Tmp$ ▷ We meet the goal if this is included in the expanded set of consequences.

Solvability: Locks and Keys (2/2)

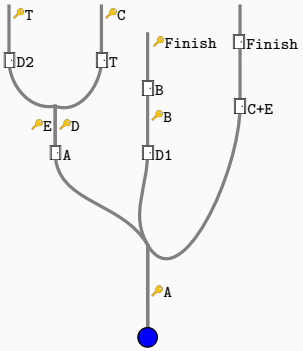


Rules (*Horn Clauses*):

1. $\mathrm{Key_A} \Rightarrow \mathrm{Door_A}$
2. $\mathrm{Door_A} \Rightarrow \mathrm{Key_E}$, $\mathrm{Door_A} \Rightarrow \mathrm{Key_D}$
3. $\mathrm{Door_A} \wedge \mathrm{Key_D} \Rightarrow \mathrm{Door_{D2}}$, $\mathrm{Key_D} \Rightarrow \mathrm{Door_{D1}}$
4. $\mathrm{Door_A} \wedge \mathrm{Key_T} \Rightarrow \mathrm{Door_T}$
5. $\mathrm{Door_{D2}} \Rightarrow \mathrm{Key_T}$
6. $\mathrm{Door_T} \Rightarrow \mathrm{Key_C}$
7. $\mathrm{Door_{D1}} \Rightarrow \mathrm{Key_B}$
8. $\mathrm{Door_{D1}} \wedge \mathrm{Key_B} \Rightarrow \mathrm{Door_B}$
9. $\mathrm{Door_B} \Rightarrow \mathrm{Key_{Finish}}$
10. $\mathrm{Key_C} \wedge \mathrm{Key_E} \Rightarrow \mathrm{Door_{C+E}}$
11. $\mathrm{Door_{C+E}} \wedge \mathrm{Key_{Finish}} \Rightarrow \mathrm{Door_{Finish}}$

Resources: $\mathrm{Key_A}$

Solvability: Locks and Keys (2/2)

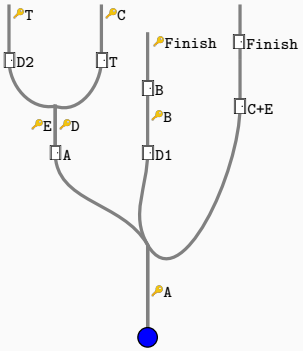


Rules (*Horn Clauses*):

1. $\mathrm{Key_A} \Rightarrow \mathrm{Door_A}$
2. $\mathrm{Door_A} \Rightarrow \mathrm{Key_E}$, $\mathrm{Door_A} \Rightarrow \mathrm{Key_D}$
3. $\mathrm{Door_A} \wedge \mathrm{Key_D} \Rightarrow \mathrm{Door_{D2}}$, $\mathrm{Key_D} \Rightarrow \mathrm{Door_{D1}}$
4. $\mathrm{Door_A} \wedge \mathrm{Key_T} \Rightarrow \mathrm{Door_T}$
5. $\mathrm{Door_{D2}} \Rightarrow \mathrm{Key_T}$
6. $\mathrm{Door_T} \Rightarrow \mathrm{Key_C}$
7. $\mathrm{Door_{D1}} \Rightarrow \mathrm{Key_B}$
8. $\mathrm{Door_{D1}} \wedge \mathrm{Key_B} \Rightarrow \mathrm{Door_B}$
9. $\mathrm{Door_B} \Rightarrow \mathrm{Key_{Finish}}$
10. $\mathrm{Key_C} \wedge \mathrm{Key_E} \Rightarrow \mathrm{Door_{C+E}}$
11. $\mathrm{Door_{C+E}} \wedge \mathrm{Key_{Finish}} \Rightarrow \mathrm{Door_{Finish}}$

Resources: $\mathrm{Key_A}$, $\mathrm{Door_A}$

Solvability: Locks and Keys (2/2)

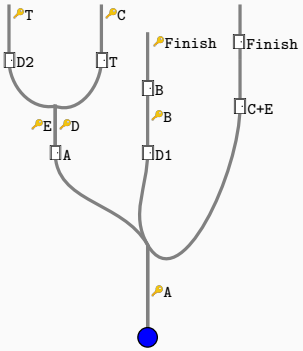


Rules (*Horn Clauses*):

1. $\text{Key}_{\texttt{A}} \Rightarrow \text{Door}_{\texttt{A}}$
2. $\text{Door}_{\texttt{A}} \Rightarrow \text{Key}_{\texttt{E}}$, $\text{Door}_{\texttt{A}} \Rightarrow \text{Key}_{\texttt{D}}$
3. $\text{Door}_{\texttt{A}} \wedge \text{Key}_{\texttt{D}} \Rightarrow \text{Door}_{\texttt{D2}}$, $\text{Key}_{\texttt{D}} \Rightarrow \text{Door}_{\texttt{D1}}$
4. $\text{Door}_{\texttt{A}} \wedge \text{Key}_{\texttt{T}} \Rightarrow \text{Door}_{\texttt{T}}$
5. $\text{Door}_{\texttt{D2}} \Rightarrow \text{Key}_{\texttt{T}}$
6. $\text{Door}_{\texttt{T}} \Rightarrow \text{Key}_{\texttt{C}}$
7. $\text{Door}_{\texttt{D1}} \Rightarrow \text{Key}_{\texttt{B}}$
8. $\text{Door}_{\texttt{D1}} \wedge \text{Key}_{\texttt{B}} \Rightarrow \text{Door}_{\texttt{B}}$
9. $\text{Door}_{\texttt{B}} \Rightarrow \text{Key}_{\texttt{Finish}}$
10. $\text{Key}_{\texttt{C}} \wedge \text{Key}_{\texttt{E}} \Rightarrow \text{Door}_{\texttt{C+E}}$
11. $\text{Door}_{\texttt{C+E}} \wedge \text{Key}_{\texttt{Finish}} \Rightarrow \text{Door}_{\texttt{Finish}}$

Resources: $\text{Key}_{\texttt{A}}$, $\text{Door}_{\texttt{A}}$, $\text{Key}_{\texttt{D}}$

Solvability: Locks and Keys (2/2)

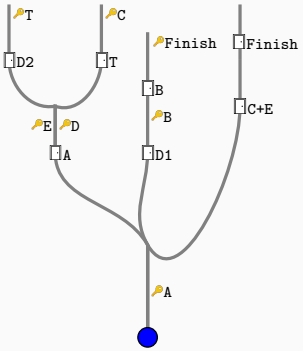


Rules (*Horn Clauses*):

1. $\text{Key}_{\text{A}} \Rightarrow \text{Door}_{\text{A}}$
2. $\text{Door}_{\text{A}} \Rightarrow \text{Key}_{\text{E}}$, $\text{Door}_{\text{A}} \Rightarrow \text{Key}_{\text{D}}$
3. $\text{Door}_{\text{A}} \wedge \text{Key}_{\text{D}} \Rightarrow \text{Door}_{\text{D2}}$, $\text{Key}_{\text{D}} \Rightarrow \text{Door}_{\text{D1}}$
4. $\text{Door}_{\text{A}} \wedge \text{Key}_{\text{T}} \Rightarrow \text{Door}_{\text{T}}$
5. $\text{Door}_{\text{D2}} \Rightarrow \text{Key}_{\text{T}}$
6. $\text{Door}_{\text{T}} \Rightarrow \text{Key}_{\text{C}}$
7. $\text{Door}_{\text{D1}} \Rightarrow \text{Key}_{\text{B}}$
8. $\text{Door}_{\text{D1}} \wedge \text{Key}_{\text{B}} \Rightarrow \text{Door}_{\text{B}}$
9. $\text{Door}_{\text{B}} \Rightarrow \text{Key}_{\text{Finish}}$
10. $\text{Key}_{\text{C}} \wedge \text{Key}_{\text{E}} \Rightarrow \text{Door}_{\text{C+E}}$
11. $\text{Door}_{\text{C+E}} \wedge \text{Key}_{\text{Finish}} \Rightarrow \text{Door}_{\text{Finish}}$

Resources: Key_{A}, Door_{A}, Key_{D}, Key_{E}

Solvability: Locks and Keys (2/2)

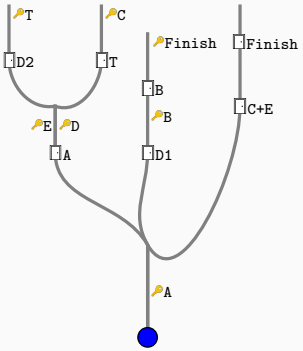


Rules (*Horn Clauses*):

1. $\text{Key}_{\texttt{A}} \Rightarrow \text{Door}_{\texttt{A}}$
2. $\text{Door}_{\texttt{A}} \Rightarrow \text{Key}_{\texttt{E}}$, $\text{Door}_{\texttt{A}} \Rightarrow \text{Key}_{\texttt{D}}$
3. $\text{Door}_{\texttt{A}} \wedge \text{Key}_{\texttt{D}} \Rightarrow \text{Door}_{\texttt{D2}}$, $\text{Key}_{\texttt{D}} \Rightarrow \text{Door}_{\texttt{D1}}$
4. $\text{Door}_{\texttt{A}} \wedge \text{Key}_{\texttt{T}} \Rightarrow \text{Door}_{\texttt{T}}$
5. $\text{Door}_{\texttt{D2}} \Rightarrow \text{Key}_{\texttt{T}}$
6. $\text{Door}_{\texttt{T}} \Rightarrow \text{Key}_{\texttt{C}}$
7. $\text{Door}_{\texttt{D1}} \Rightarrow \text{Key}_{\texttt{B}}$
8. $\text{Door}_{\texttt{D1}} \wedge \text{Key}_{\texttt{B}} \Rightarrow \text{Door}_{\texttt{B}}$
9. $\text{Door}_{\texttt{B}} \Rightarrow \text{Key}_{\texttt{Finish}}$
10. $\text{Key}_{\texttt{C}} \wedge \text{Key}_{\texttt{E}} \Rightarrow \text{Door}_{\texttt{C+E}}$
11. $\text{Door}_{\texttt{C+E}} \wedge \text{Key}_{\texttt{Finish}} \Rightarrow \text{Door}_{\texttt{Finish}}$

Resources: $\text{Key}_{\texttt{A}}$, $\text{Door}_{\texttt{A}}$, $\text{Key}_{\texttt{D}}$, $\text{Key}_{\texttt{E}}$, $\text{Door}_{\texttt{D1}}$

Solvability: Locks and Keys (2/2)

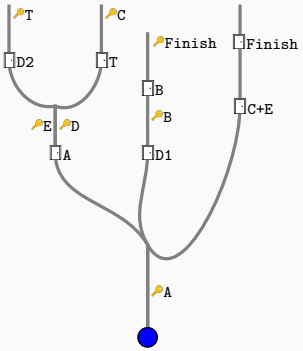


Rules (*Horn Clauses*):

1. $\text{Key}_{\text{A}} \Rightarrow \text{Door}_{\text{A}}$
2. $\text{Door}_{\text{A}} \Rightarrow \text{Key}_{\text{E}}$, $\text{Door}_{\text{A}} \Rightarrow \text{Key}_{\text{D}}$
3. $\text{Door}_{\text{A}} \wedge \text{Key}_{\text{D}} \Rightarrow \text{Door}_{\text{D2}}$, $\text{Key}_{\text{D}} \Rightarrow \text{Door}_{\text{D1}}$
4. $\text{Door}_{\text{A}} \wedge \text{Key}_{\text{T}} \Rightarrow \text{Door}_{\text{T}}$
5. $\text{Door}_{\text{D2}} \Rightarrow \text{Key}_{\text{T}}$
6. $\text{Door}_{\text{T}} \Rightarrow \text{Key}_{\text{C}}$
7. $\text{Door}_{\text{D1}} \Rightarrow \text{Key}_{\text{B}}$
8. $\text{Door}_{\text{D1}} \wedge \text{Key}_{\text{B}} \Rightarrow \text{Door}_{\text{B}}$
9. $\text{Door}_{\text{B}} \Rightarrow \text{Key}_{\text{Finish}}$
10. $\text{Key}_{\text{C}} \wedge \text{Key}_{\text{E}} \Rightarrow \text{Door}_{\text{C+E}}$
11. $\text{Door}_{\text{C+E}} \wedge \text{Key}_{\text{Finish}} \Rightarrow \text{Door}_{\text{Finish}}$

Resources: Key_{A}, Door_{A}, Key_{D}, Key_{E}, Door_{D1}, Door_{D2}

Solvability: Locks and Keys (2/2)

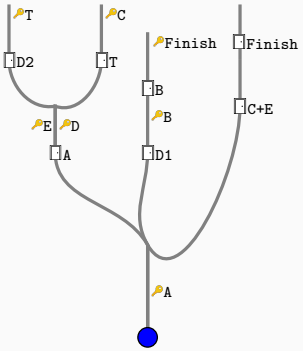


Rules (*Horn Clauses*):

1. $\text{Key}_{\texttt{A}} \Rightarrow \text{Door}_{\texttt{A}}$
2. $\text{Door}_{\texttt{A}} \Rightarrow \text{Key}_{\texttt{E}}$, $\text{Door}_{\texttt{A}} \Rightarrow \text{Key}_{\texttt{D}}$
3. $\text{Door}_{\texttt{A}} \wedge \text{Key}_{\texttt{D}} \Rightarrow \text{Door}_{\texttt{D2}}$, $\text{Key}_{\texttt{D}} \Rightarrow \text{Door}_{\texttt{D1}}$
4. $\text{Door}_{\texttt{A}} \wedge \text{Key}_{\texttt{T}} \Rightarrow \text{Door}_{\texttt{T}}$
5. $\text{Door}_{\texttt{D2}} \Rightarrow \text{Key}_{\texttt{T}}$
6. $\text{Door}_{\texttt{T}} \Rightarrow \text{Key}_{\texttt{C}}$
7. $\text{Door}_{\texttt{D1}} \Rightarrow \text{Key}_{\texttt{B}}$
8. $\text{Door}_{\texttt{D1}} \wedge \text{Key}_{\texttt{B}} \Rightarrow \text{Door}_{\texttt{B}}$
9. $\text{Door}_{\texttt{B}} \Rightarrow \text{Key}_{\texttt{Finish}}$
10. $\text{Key}_{\texttt{C}} \wedge \text{Key}_{\texttt{E}} \Rightarrow \text{Door}_{\texttt{C+E}}$
11. $\text{Door}_{\texttt{C+E}} \wedge \text{Key}_{\texttt{Finish}} \Rightarrow \text{Door}_{\texttt{Finish}}$

Resources: $\text{Key}_{\texttt{A}}$, $\text{Door}_{\texttt{A}}$, $\text{Key}_{\texttt{D}}$, $\text{Key}_{\texttt{E}}$, $\text{Door}_{\texttt{D1}}$, $\text{Door}_{\texttt{D2}}$, $\text{Key}_{\texttt{T}}$

Solvability: Locks and Keys (2/2)

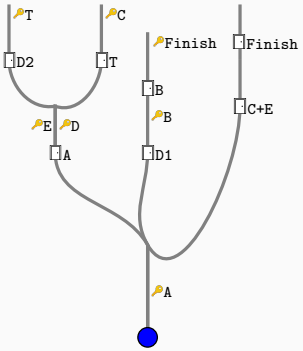


Rules (*Horn Clauses*):

1. $\text{Key}_{\texttt{A}} \Rightarrow \text{Door}_{\texttt{A}}$
2. $\text{Door}_{\texttt{A}} \Rightarrow \text{Key}_{\texttt{E}}$, $\text{Door}_{\texttt{A}} \Rightarrow \text{Key}_{\texttt{D}}$
3. $\text{Door}_{\texttt{A}} \wedge \text{Key}_{\texttt{D}} \Rightarrow \text{Door}_{\texttt{D2}}$, $\text{Key}_{\texttt{D}} \Rightarrow \text{Door}_{\texttt{D1}}$
4. $\text{Door}_{\texttt{A}} \wedge \text{Key}_{\texttt{T}} \Rightarrow \text{Door}_{\texttt{T}}$
5. $\text{Door}_{\texttt{D2}} \Rightarrow \text{Key}_{\texttt{T}}$
6. $\text{Door}_{\texttt{T}} \Rightarrow \text{Key}_{\texttt{C}}$
7. $\text{Door}_{\texttt{D1}} \Rightarrow \text{Key}_{\texttt{B}}$
8. $\text{Door}_{\texttt{D1}} \wedge \text{Key}_{\texttt{B}} \Rightarrow \text{Door}_{\texttt{B}}$
9. $\text{Door}_{\texttt{B}} \Rightarrow \text{Key}_{\texttt{Finish}}$
10. $\text{Key}_{\texttt{C}} \wedge \text{Key}_{\texttt{E}} \Rightarrow \text{Door}_{\texttt{C+E}}$
11. $\text{Door}_{\texttt{C+E}} \wedge \text{Key}_{\texttt{Finish}} \Rightarrow \text{Door}_{\texttt{Finish}}$

Resources: $\text{Key}_{\texttt{A}}$, $\text{Door}_{\texttt{A}}$, $\text{Key}_{\texttt{D}}$, $\text{Key}_{\texttt{E}}$, $\text{Door}_{\texttt{D1}}$, $\text{Door}_{\texttt{D2}}$, $\text{Key}_{\texttt{T}}$, $\text{Key}_{\texttt{B}}$

Solvability: Locks and Keys (2/2)

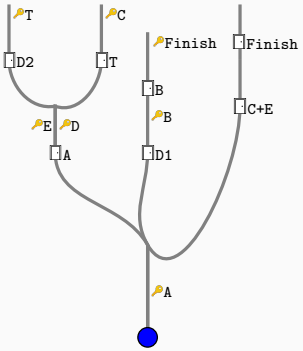


Rules (*Horn Clauses*):

1. $\mathrm{Key_A} \Rightarrow \mathrm{Door_A}$
2. $\mathrm{Door_A} \Rightarrow \mathrm{Key_E}$, $\mathrm{Door_A} \Rightarrow \mathrm{Key_D}$
3. $\mathrm{Door_A} \wedge \mathrm{Key_D} \Rightarrow \mathrm{Door_{D2}}$, $\mathrm{Key_D} \Rightarrow \mathrm{Door_{D1}}$
4. $\mathrm{Door_A} \wedge \mathrm{Key_T} \Rightarrow \mathrm{Door_T}$
5. $\mathrm{Door_{D2}} \Rightarrow \mathrm{Key_T}$
6. $\mathrm{Door_T} \Rightarrow \mathrm{Key_C}$
7. $\mathrm{Door_{D1}} \Rightarrow \mathrm{Key_B}$
8. $\mathrm{Door_{D1}} \wedge \mathrm{Key_B} \Rightarrow \mathrm{Door_B}$
9. $\mathrm{Door_B} \Rightarrow \mathrm{Key_{Finish}}$
10. $\mathrm{Key_C} \wedge \mathrm{Key_E} \Rightarrow \mathrm{Door_{C+E}}$
11. $\mathrm{Door_{C+E}} \wedge \mathrm{Key_{Finish}} \Rightarrow \mathrm{Door_{Finish}}$

Resources: $\mathrm{Key_A}$, $\mathrm{Door_A}$, $\mathrm{Key_D}$, $\mathrm{Key_E}$, $\mathrm{Door_{D1}}$, $\mathrm{Door_{D2}}$, $\mathrm{Key_T}$, $\mathrm{Key_B}$, $\mathrm{Door_T}$

Solvability: Locks and Keys (2/2)

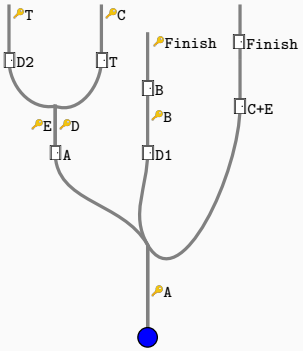


Rules (*Horn Clauses*):

1. $\text{Key}_{\text{A}} \Rightarrow \text{Door}_{\text{A}}$
2. $\text{Door}_{\text{A}} \Rightarrow \text{Key}_{\text{E}}$, $\text{Door}_{\text{A}} \Rightarrow \text{Key}_{\text{D}}$
3. $\text{Door}_{\text{A}} \wedge \text{Key}_{\text{D}} \Rightarrow \text{Door}_{\text{D2}}$, $\text{Key}_{\text{D}} \Rightarrow \text{Door}_{\text{D1}}$
4. $\text{Door}_{\text{A}} \wedge \text{Key}_{\text{T}} \Rightarrow \text{Door}_{\text{T}}$
5. $\text{Door}_{\text{D2}} \Rightarrow \text{Key}_{\text{T}}$
6. $\text{Door}_{\text{T}} \Rightarrow \text{Key}_{\text{C}}$
7. $\text{Door}_{\text{D1}} \Rightarrow \text{Key}_{\text{B}}$
8. $\text{Door}_{\text{D1}} \wedge \text{Key}_{\text{B}} \Rightarrow \text{Door}_{\text{B}}$
9. $\text{Door}_{\text{B}} \Rightarrow \text{Key}_{\text{Finish}}$
10. $\text{Key}_{\text{C}} \wedge \text{Key}_{\text{E}} \Rightarrow \text{Door}_{\text{C+E}}$
11. $\text{Door}_{\text{C+E}} \wedge \text{Key}_{\text{Finish}} \Rightarrow \text{Door}_{\text{Finish}}$

Resources: Key_{A}, Door_{A}, Key_{D}, Key_{E}, Door_{D1}, Door_{D2}, Key_{T}, Key_{B}, Door_{T}, Door_{B}

Solvability: Locks and Keys (2/2)

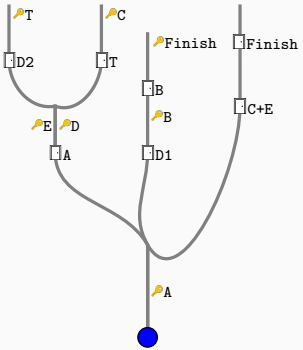


Rules (*Horn Clauses*):

1. $\text{Key}_\text{A} \Rightarrow \text{Door}_\text{A}$
2. $\text{Door}_\text{A} \Rightarrow \text{Key}_\text{E}$, $\text{Door}_\text{A} \Rightarrow \text{Key}_\text{D}$
3. $\text{Door}_\text{A} \wedge \text{Key}_\text{D} \Rightarrow \text{Door}_\text{D2}$, $\text{Key}_\text{D} \Rightarrow \text{Door}_\text{D1}$
4. $\text{Door}_\text{A} \wedge \text{Key}_\text{T} \Rightarrow \text{Door}_\text{T}$
5. $\text{Door}_\text{D2} \Rightarrow \text{Key}_\text{T}$
6. $\text{Door}_\text{T} \Rightarrow \text{Key}_\text{C}$
7. $\text{Door}_\text{D1} \Rightarrow \text{Key}_\text{B}$
8. $\text{Door}_\text{D1} \wedge \text{Key}_\text{B} \Rightarrow \text{Door}_\text{B}$
9. $\text{Door}_\text{B} \Rightarrow \text{Key}_\text{Finish}$
10. $\text{Key}_\text{C} \wedge \text{Key}_\text{E} \Rightarrow \text{Door}_\text{C+E}$
11. $\text{Door}_\text{C+E} \wedge \text{Key}_\text{Finish} \Rightarrow \text{Door}_\text{Finish}$

Resources: Key_A, Door_A, Key_D, Key_E, Door_D1, Door_D2, Key_T, Key_B, Door_T, Door_B, Key_C

Solvability: Locks and Keys (2/2)

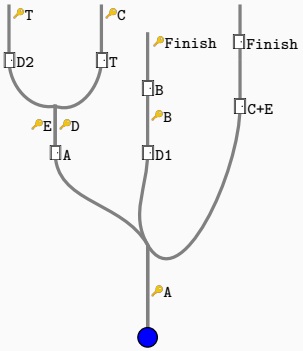


Rules (*Horn Clauses*):

1. $\text{Key}_{\text{A}} \Rightarrow \text{Door}_{\text{A}}$
2. $\text{Door}_{\text{A}} \Rightarrow \text{Key}_{\text{E}}$, $\text{Door}_{\text{A}} \Rightarrow \text{Key}_{\text{D}}$
3. $\text{Door}_{\text{A}} \wedge \text{Key}_{\text{D}} \Rightarrow \text{Door}_{\text{D2}}$, $\text{Key}_{\text{D}} \Rightarrow \text{Door}_{\text{D1}}$
4. $\text{Door}_{\text{A}} \wedge \text{Key}_{\text{T}} \Rightarrow \text{Door}_{\text{T}}$
5. $\text{Door}_{\text{D2}} \Rightarrow \text{Key}_{\text{T}}$
6. $\text{Door}_{\text{T}} \Rightarrow \text{Key}_{\text{C}}$
7. $\text{Door}_{\text{D1}} \Rightarrow \text{Key}_{\text{B}}$
8. $\text{Door}_{\text{D1}} \wedge \text{Key}_{\text{B}} \Rightarrow \text{Door}_{\text{B}}$
9. $\text{Door}_{\text{B}} \Rightarrow \text{Key}_{\text{Finish}}$
10. $\text{Key}_{\text{C}} \wedge \text{Key}_{\text{E}} \Rightarrow \text{Door}_{\text{C+E}}$
11. $\text{Door}_{\text{C+E}} \wedge \text{Key}_{\text{Finish}} \Rightarrow \text{Door}_{\text{Finish}}$

Resources: Key_{A}, Door_{A}, Key_{D}, Key_{E}, Door_{D1}, Door_{D2}, Key_{T}, Key_{B}, Door_{T}, Door_{B}, Key_{C}, $\text{Door}_{\text{C+E}}$

Solvability: Locks and Keys (2/2)

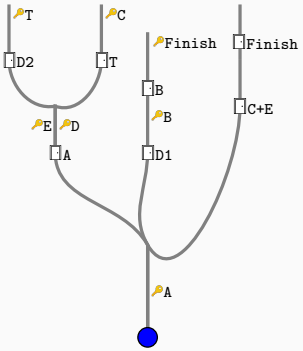


Rules (*Horn Clauses*):

1. $\text{Key}_{\texttt{A}} \Rightarrow \text{Door}_{\texttt{A}}$
2. $\text{Door}_{\texttt{A}} \Rightarrow \text{Key}_{\texttt{E}}$, $\text{Door}_{\texttt{A}} \Rightarrow \text{Key}_{\texttt{D}}$
3. $\text{Door}_{\texttt{A}} \wedge \text{Key}_{\texttt{D}} \Rightarrow \text{Door}_{\texttt{D2}}$, $\text{Key}_{\texttt{D}} \Rightarrow \text{Door}_{\texttt{D1}}$
4. $\text{Door}_{\texttt{A}} \wedge \text{Key}_{\texttt{T}} \Rightarrow \text{Door}_{\texttt{T}}$
5. $\text{Door}_{\texttt{D2}} \Rightarrow \text{Key}_{\texttt{T}}$
6. $\text{Door}_{\texttt{T}} \Rightarrow \text{Key}_{\texttt{C}}$
7. $\text{Door}_{\texttt{D1}} \Rightarrow \text{Key}_{\texttt{B}}$
8. $\text{Door}_{\texttt{D1}} \wedge \text{Key}_{\texttt{B}} \Rightarrow \text{Door}_{\texttt{B}}$
9. $\text{Door}_{\texttt{B}} \Rightarrow \text{Key}_{\texttt{Finish}}$
10. $\text{Key}_{\texttt{C}} \wedge \text{Key}_{\texttt{E}} \Rightarrow \text{Door}_{\texttt{C+E}}$
11. $\text{Door}_{\texttt{C+E}} \wedge \text{Key}_{\texttt{Finish}} \Rightarrow \text{Door}_{\texttt{Finish}}$

Resources: $\text{Key}_{\texttt{A}}$, $\text{Door}_{\texttt{A}}$, $\text{Key}_{\texttt{D}}$, $\text{Key}_{\texttt{E}}$, $\text{Door}_{\texttt{D1}}$, $\text{Door}_{\texttt{D2}}$, $\text{Key}_{\texttt{T}}$, $\text{Key}_{\texttt{B}}$, $\text{Door}_{\texttt{T}}$, $\text{Door}_{\texttt{B}}$, $\text{Key}_{\texttt{C}}$, $\text{Door}_{\texttt{C+E}}$, $\text{Key}_{\texttt{Finish}}$

Solvability: Locks and Keys (2/2)

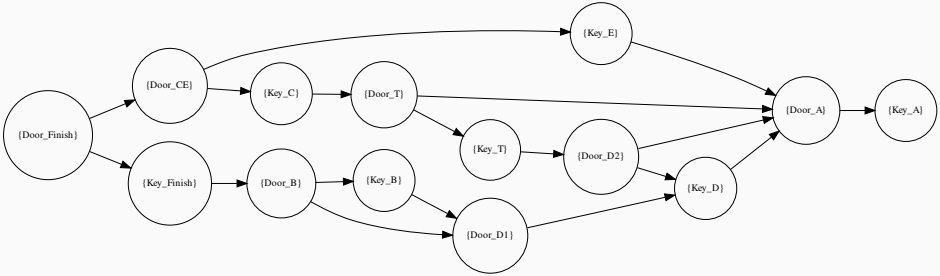


Rules (*Horn Clauses*):

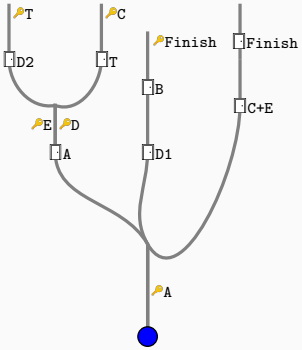
1. $\text{Key}_{\texttt{A}} \Rightarrow \text{Door}_{\texttt{A}}$
2. $\text{Door}_{\texttt{A}} \Rightarrow \text{Key}_{\texttt{E}}$, $\text{Door}_{\texttt{A}} \Rightarrow \text{Key}_{\texttt{D}}$
3. $\text{Door}_{\texttt{A}} \wedge \text{Key}_{\texttt{D}} \Rightarrow \text{Door}_{\texttt{D2}}$, $\text{Key}_{\texttt{D}} \Rightarrow \text{Door}_{\texttt{D1}}$
4. $\text{Door}_{\texttt{A}} \wedge \text{Key}_{\texttt{T}} \Rightarrow \text{Door}_{\texttt{T}}$
5. $\text{Door}_{\texttt{D2}} \Rightarrow \text{Key}_{\texttt{T}}$
6. $\text{Door}_{\texttt{T}} \Rightarrow \text{Key}_{\texttt{C}}$
7. $\text{Door}_{\texttt{D1}} \Rightarrow \text{Key}_{\texttt{B}}$
8. $\text{Door}_{\texttt{D1}} \wedge \text{Key}_{\texttt{B}} \Rightarrow \text{Door}_{\texttt{B}}$
9. $\text{Door}_{\texttt{B}} \Rightarrow \text{Key}_{\texttt{Finish}}$
10. $\text{Key}_{\texttt{C}} \wedge \text{Key}_{\texttt{E}} \Rightarrow \text{Door}_{\texttt{C+E}}$
11. $\text{Door}_{\texttt{C+E}} \wedge \text{Key}_{\texttt{Finish}} \Rightarrow \text{Door}_{\texttt{Finish}}$

Resources: $\text{Key}_{\texttt{A}}$, $\text{Door}_{\texttt{A}}$, $\text{Key}_{\texttt{D}}$, $\text{Key}_{\texttt{E}}$, $\text{Door}_{\texttt{D1}}$, $\text{Door}_{\texttt{D2}}$, $\text{Key}_{\texttt{T}}$, $\text{Key}_{\texttt{B}}$, $\text{Door}_{\texttt{T}}$, $\text{Door}_{\texttt{B}}$, $\text{Key}_{\texttt{C}}$, $\text{Door}_{\texttt{C+E}}$, $\text{Key}_{\texttt{Finish}}$, $\text{Door}_{\texttt{Finish}}$

Solution #2: "Backward" Rule-Based Planning: Stalemates and Difficulty



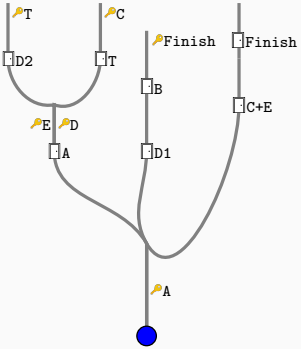
- **Each state does** not **represent** a possible gameplay state as in GOAP, rather than **a resource/subgoal to be obtained/solved**.
- Nodes reachable in one step determine tasks/resources that *should be done/obtained simultaneously*.
- One single graph is generated *for each possible solution* independently from the order of the actions; each solution is different only if different strategies are used to reach a same (possibly intermediate) goal.
- If not all the $Init$ preconditions or resources/goals are used to solve the game, *some initial resources might be discarded*, as irrelevant for solving the final riddle with the given strategy.

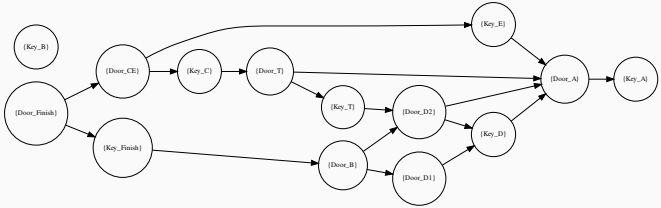


Is this game level solvable? Why?

Stalemates: Missing Resources

The previous algorithm will generate a graph with one isolated node (the source of the error).





Some debugging errors are returned in `WeightedMultiGraph.errors`:

`Error: Cannot apply rule for:` $\mathrm{Key_B}$

`Error: Cannot generate precondition` $\mathrm{Key_B}$ `for` $\mathrm{Door_{D1}} \wedge \mathrm{Key_B} \Rightarrow \mathrm{Door_B}$

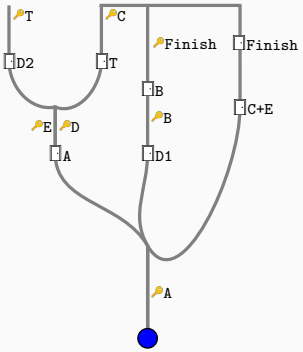
Solution #2: “Backward” Rule-Based Planning: Solvability

`GenerateBacktrackStates<T>::generateGraphs`:

- Is it possible to detect the reasons that bring to a stalemate?
  1. We need to navigate backwards from the goal,
  2. try to apply all the rules backwards (from the result towards the prerequisites),
  3. and detect at which step we halt the navigation towards the initial configuration.
- Each node of the graph provides only one resource and task at a time. The path represent dependencies among the tasks, that might be visited to enact a strategy!
- By doing so, please observe that we might generate multiple graphs, where each graph represents a possible strategy.

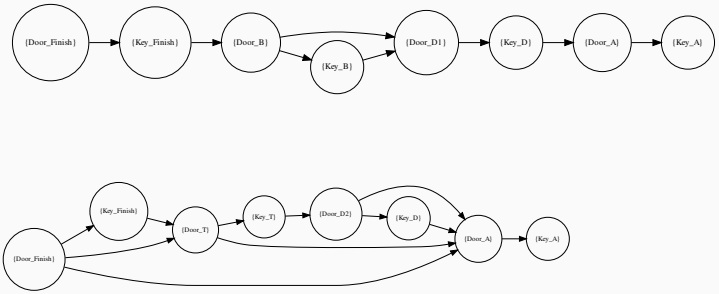
Multiple Solutions: Example (1/2)

Let us further change the game level so to allow multiple possible ways to solve the game: what is the difference?



1. $\text{Key}_{\text{A}} \Rightarrow \text{Door}_{\text{A}}$
2. $\text{Door}_{\text{A}} \Rightarrow \text{Key}_{\text{E}}$, $\text{Door}_{\text{A}} \Rightarrow \text{Key}_{\text{D}}$
3. $\text{Door}_{\text{A}} \wedge \text{Key}_{\text{D}} \Rightarrow \text{Door}_{\text{D2}}$, $\text{Key}_{\text{D}} \Rightarrow \text{Door}_{\text{D1}}$
4. $\text{Door}_{\text{A}} \wedge \text{Key}_{\text{T}} \Rightarrow \text{Door}_{\text{T}}$
5. $\text{Door}_{\text{D2}} \Rightarrow \text{Key}_{\text{T}}$
6. $\text{Door}_{\text{T}} \Rightarrow \text{Key}_{\text{C}}$, $\text{Door}_{\text{T}} \Rightarrow \text{Key}_{\text{Finish}}$
7. $\text{Door}_{\text{D1}} \Rightarrow \text{Key}_{\text{B}}$
8. $\text{Door}_{\text{D1}} \wedge \text{Key}_{\text{B}} \Rightarrow \text{Door}_{\text{B}}$
9. $\text{Door}_{\text{B}} \Rightarrow \text{Key}_{\text{Finish}}$, $\text{Door}_{\text{B}} \Rightarrow \text{Key}_{\text{C}}$
10. $\text{Key}_{\text{C}} \wedge \text{Key}_{\text{E}} \Rightarrow \text{Door}_{\text{C+E}}$
11. $\text{Door}_{\text{C+E}} \wedge \text{Key}_{\text{Finish}} \Rightarrow \text{Door}_{\text{Finish}}$,
    $\text{Door}_{\text{A}} \wedge \text{Door}_{\text{T}} \wedge \text{Key}_{\text{Finish}} \Rightarrow \text{Door}_{\text{Finish}}$,
    $\text{Door}_{\text{D1}} \wedge \text{Door}_{\text{B}} \wedge \text{Key}_{\text{Finish}} \Rightarrow \text{Door}_{\text{Finish}}$

Multiple Solutions: Example (2/2)



1. The algorithm might generate nodes that are irrelevant, as they are not directly reachable from the goal (those could be pruned):
2. as a result, the algorithm might generate duplicate solutions, but those are easily detectable.
3. Given that these are "dependency graphs", we might exploit a topological sort for getting the best order to perform the actions to to reach the goal.

Checking whether a current Dungeon Level is solvable

For each possible path between start and final room (`ProceduralDungeon::isFeasable()`):

- Collect all of the keys before the first door appearing in the path as an "initial state".
- Track all the doors met while scanning the path in a list ℓ:
- When a new door is met d_i associated to a key k_i, create a new rule $k_i \wedge \bigwedge_{d_j \in \ell} d_j \Rightarrow d_i$, as well as the requirement that, for opening the door, you need its key: $k_i \Rightarrow d_i$.
- For each key κ met between the last door $\delta \in \ell$ and the current d_i, generate a new rule $\delta \Rightarrow \kappa$

For each backward graph having no computation errors:

- Sort the vertices in inverse topological order (`graph.TopologicalSort()`)
- Return the associated strategy by collecting the node labels.

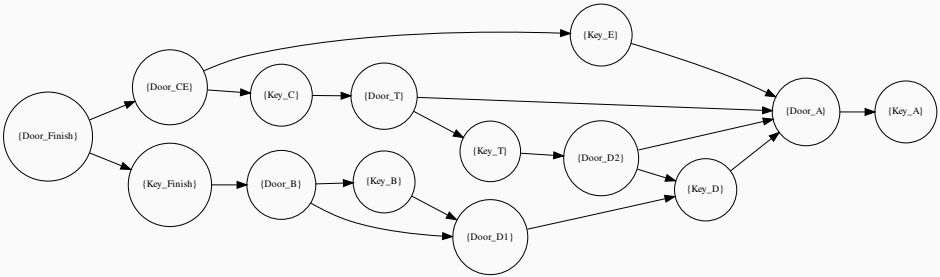
The game is unsolvable if there are no graphs or if the graphs have errors or empty strategies.

Game balance

Game balance is the study of the mathematical properties of the game so to assess its *fairness* and *engagingness*. Some of the key points for Game balance are:

1. Solvability
   - By analyzing the game mechanics, we need to assess whether we might reach the end of the game or not (e.g., winning).
   - No stalemates should arise, in which nobody can win or lose.
   - We also want to avoid *dominant strategies* that always lead to winning the game.
2. Difficulty
   - Difficulty should be proportional to the player's skills
   - Difficulty should increase with the progress with the game

Difficulty in Non-Stochastic Games (1/2)



- We might also generate graphs with costs/probabilities associated to both nodes and edges: `GenerateWeightedPossibleStates<T>`.
- We use all the graphs with no errors for handling difficulty within a game level. Suppose that each rule is associated to a cost coming from:
  1. the [*min*,*max*] cost of reaching a door or a resource from the previous position.
  2. eventually, the cost s for collecting the resource or opening the door.
  3. Some resources might be available only from a given XP level.
- This cost, namely c_e, is then associated to each edge e.

Difficulty in Non-Stochastic Games (2/2)

If the algorithm returns one single graph, the user has only one choice (up to performing actions in a different order), and therefore we can only assess its hardness.

The maximum c_e of an edge e gives the maximum cost M that is expected to be solved (e.g.) by the maximum level of expertise X:

- Given the current user level of expertise $\chi \leq X$,
- Remove all the unavailable resources for level χ ($\Rightarrow$ rule update)
- Remove edges having minimum cost $min + s$ greater than M/χ
- Remove the isolated nodes

If there are no paths connecting a $Init$ resource to a $Goal$ resource/goal, the resulting game might be hard (if not impossible) to solve for an user of XP χ, and such scenario might be discarded.

The suitable solutions for χ are the remaining graphs.

Conclusions

Solvability:

- by representing a game with a set of rules, we might detect if the game is solvable and,
- if not, we might get where the bug is while generating the game.

Difficulty:

- In non-stochastic games, we might exploit solvability techniques to generate and rank possible alternatives to solve the game.

Further Algorithms

Appendix: Simple Theorem Proving for Stalemate detection

```
1:  function GenerateBacktrackStates(Init, Goal, Rules)
2:    global 𝒢;   𝒢 := ∅;   G := 𝒢.new()                ▷ 𝒢 is a vector of graphs, initially containing one single empty graph, G
3:    for all s ∈ Goal do DFSGenerateBacktrackStates(G, s, Init, Rules)   ▷ Decompose the Goal into its constituents s
4:    return 𝒢
5:  function DFSGenerateBacktrackStates(G = (V, E), s, Init, Rules)
6:    global 𝒢                                          ▷ Vector of graphs, containing the results
7:    if s ∈ V then return s                            ▷ Skip a vertex if already visited
8:    else
9:      if s ∈ Init then accepting(s) := true           ▷ We reach an initial state: we can stop the iteration.
10:     else
11:       count := 0                                    ▷ Count how many rules satisfy the current sub-goal s
12:       for all H ⇒ s ∈ Rules do
13:         count++;                                    ▷ As the rule terminates with s, it might satisfy the rule if all the pre-conditions in H are met.
14:         if count = 1 then G' := G else  G' := 𝒢.new_copy(G)   ▷ Generate distinct graphs for each rule application
15:         for all P ∈ H do                            ▷ Check if each precondition P in H is satisfied by recursively running the algorithm
16:           s' := DFSGenerateBacktrackStates(G', P, Init, Rules)
17:           E' := E' ∪ {s --H⇒s--> s'}
18:           if s' = ε then                            ▷ If the recursive call fails at P ∼ s', then the error...
19:             G'.error += "Cannot generate precondition P for H ⇒ s"   ▷ ... propagates back at s (getting the source of the error).
20:       if count = 0 then                             ▷ If no rule satisfies the current sub-goal, then raise an error
21:         G'.error += "Cannot apply rule for: s"
22:         return ε                                    ▷ The impossibility of finding a suitable rule is marked by returning an empty state.
23:       else return s
```