Landmarks Revisited

By Richter, Helmert, and Westphal

Presented by Nabuat Zaman Nahim

Spring 2024

CS 660: Special Topics in AI

Planning Algorithms

Previously

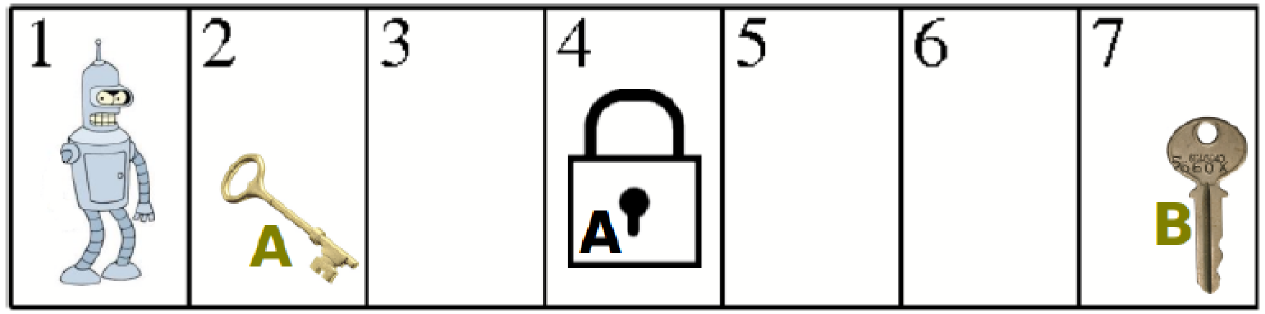
- Heuristic Planning
- HSP
- FF (Fast Forward)
- FD (Fast Downward)
- **Starting Today: other ways to build State-Space Planners starting with Landmarks**

What is a Landmark?

- A condition that must be True at some point for a solution to a problem
- Example: if we have a cargo and a plane, the cargo has to go into the plane
- Paper: finding landmarks and using it to break a bigger problem down into smaller problems

Landmarks in a Nutshell

Problem: Bring key B to position 1.



Landmarks:

- robot-at-2, robot-at-3, robot-at-4, robot-at-5, robot-at-6, robot-at-7.
- lock-open, have-key-A, have-key-B, . . .

→ A landmark is something that every plan for the task must satisfy at some point.

- Find landmarks in a pre-process to planning.
- Heuristic value(state) := number of yet un-achieved landmarks. ("Number of open items on the to-do list")

History

- Landmarks for propositional planning were introduced by Porteous, Sebastia and Hoffmann (2001)
- Studied as a method for problem decomposition [Hoffmann et al. (2003)]
- LM^{RPG}: extracts landmarks and their orderings from the relaxed planning graph of a planning task
- LM^{local} : local search procedure which searches iteratively for plans to the “nearest” landmarks, rather than searching for a plan to the goal

History

- Idea to generate heuristics based on landmarks was first conceived by [Zhu and Givan (2003)], never properly published and forgotten
- Basic idea was re-discovered by the authors of LAMA [Richter et al. (2008); Richter and Westphal (2010)] which subsequently won two IPCs

History

- Both the initial attempt and LAMA use non-admissible landmarks heuristics, basically counting the number of non-achieved fact landmarks
- LAMA uses additional methods based on domain transition graphs
- Introduced the idea to use both, a delete relaxation heuristic and a LM heuristic, in Fast Downward's dual-queue greedy best-first search framework

Today's paper

- Novel approach for using landmarks in planning by deriving a pseudo-heuristic and combining it with other heuristics
- Using landmark information improves success rates and solution qualities of a heuristic planner
- Additional landmarks and orderings can be found using information present in multi-valued state variable representations of planning tasks

Today's paper

- Their landmark extraction algorithm provides stronger guarantees of correctness for the generated landmark orderings
- Solves more planning tasks and delivers considerably better solutions (shorter plans)
- Compared to original LM^{local} and heuristics that don't use landmarks
- Unlike LM^{RPG}, only generates sound orderings

Example: Blocks World

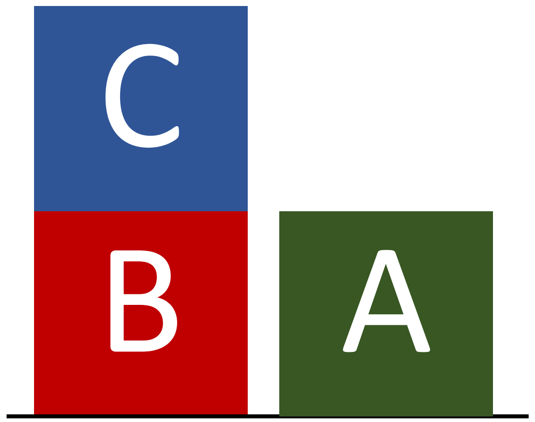
"Landmarks are propositions that must be true at some point in every solution plan for a given planning task."

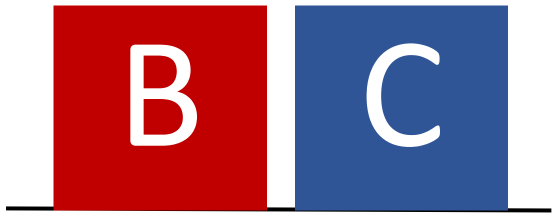
What are the landmarks here?

clear(B)

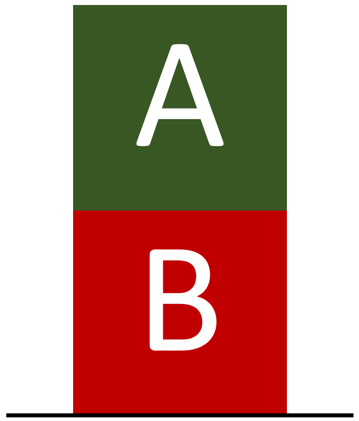
on(A, B)

(initial state)





(goal state)



Ordering:

clear(B) → on(A, B)

Landmarks as intermediary goals

- LM^{local} : breaks down planning task into subtasks
- Makes landmarks intermediary goals
- Builds a landmark graph: landmarks are vertices and orderings are arcs
- Base planner takes disjunctive goals and generates plan to achieve one of the landmarks
- Landmark and arcs removed and repeat
- When landmark graph is empty, base planner (FF used) generates plan to original goal

Algorithm

- Similar to LM^{RPG} with some differences
- Uses the possibly before criterion to ensure only sound orderings
- Instead of one-step lookahead of LM^{RPG}, uses more general approach to admit disjunctive landmarks (Porteous and Cresswell 2002)
- Like LM^{RPG}, created disjunctive sets of facts from the preconditions of first achievers of a landmark B such that a set contains one precondition fact from each first achiever of B

Algorithm

- Like LM^{RPG}, requires all facts come from same predicate symbol
- Discards any sets of size greater than 4 in order to limit the number of possible sets
- Each set A found this way is then recorded as a disjunctive landmark and ordered greedy-necessarily before B
- If B is a disjunctive landmark, then the first achievers of B are all operators which achieve one of the facts in B
- An additional cheap and easy way of extracting more landmarks is also presented by using domain transition graphs

Landmarks as a Pseudo-Heuristic

- Integrates the landmark information with other useful heuristics
- Estimates the goal distance of a state s by the number of landmarks l that still need to be achieved from s onwards
- $\hat{l} := n - m + k$
- n: total number of landmarks
- m: number of landmarks that are accepted
- k: number of accepted landmarks that are required again

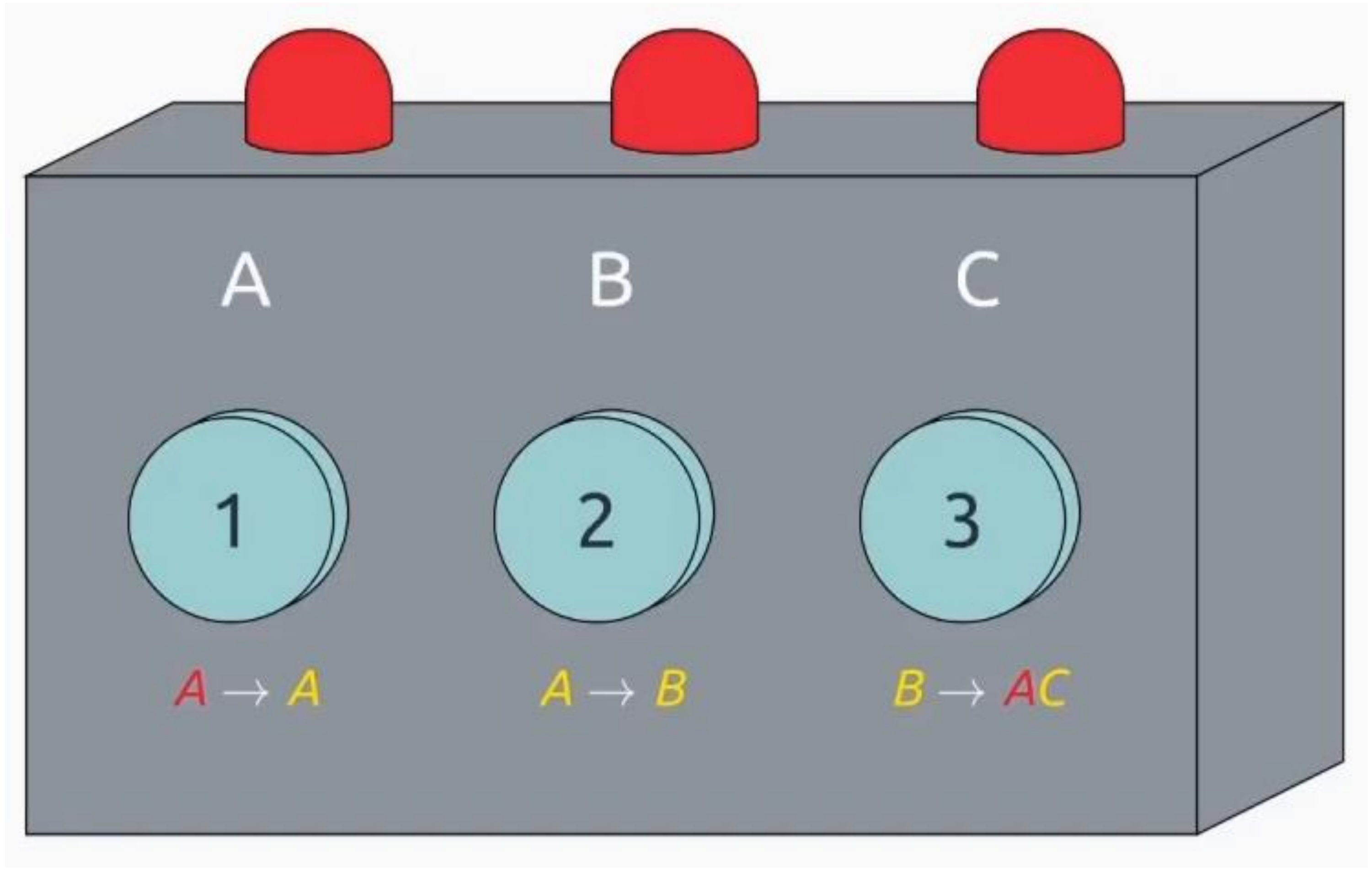
Landmarks as a Pseudo-Heuristic

- Landmark B is accepted in a state s if it is true in that state and all landmarks ordered before B are accepted in the predecessor state from which s was generated
- An accepted landmark remains accepted in all successor states
- An accepted landmark is required again if it is not true in s and it is the greedy necessary predecessor of some landmark which is not accepted

Landmarks as a Pseudo-Heuristic

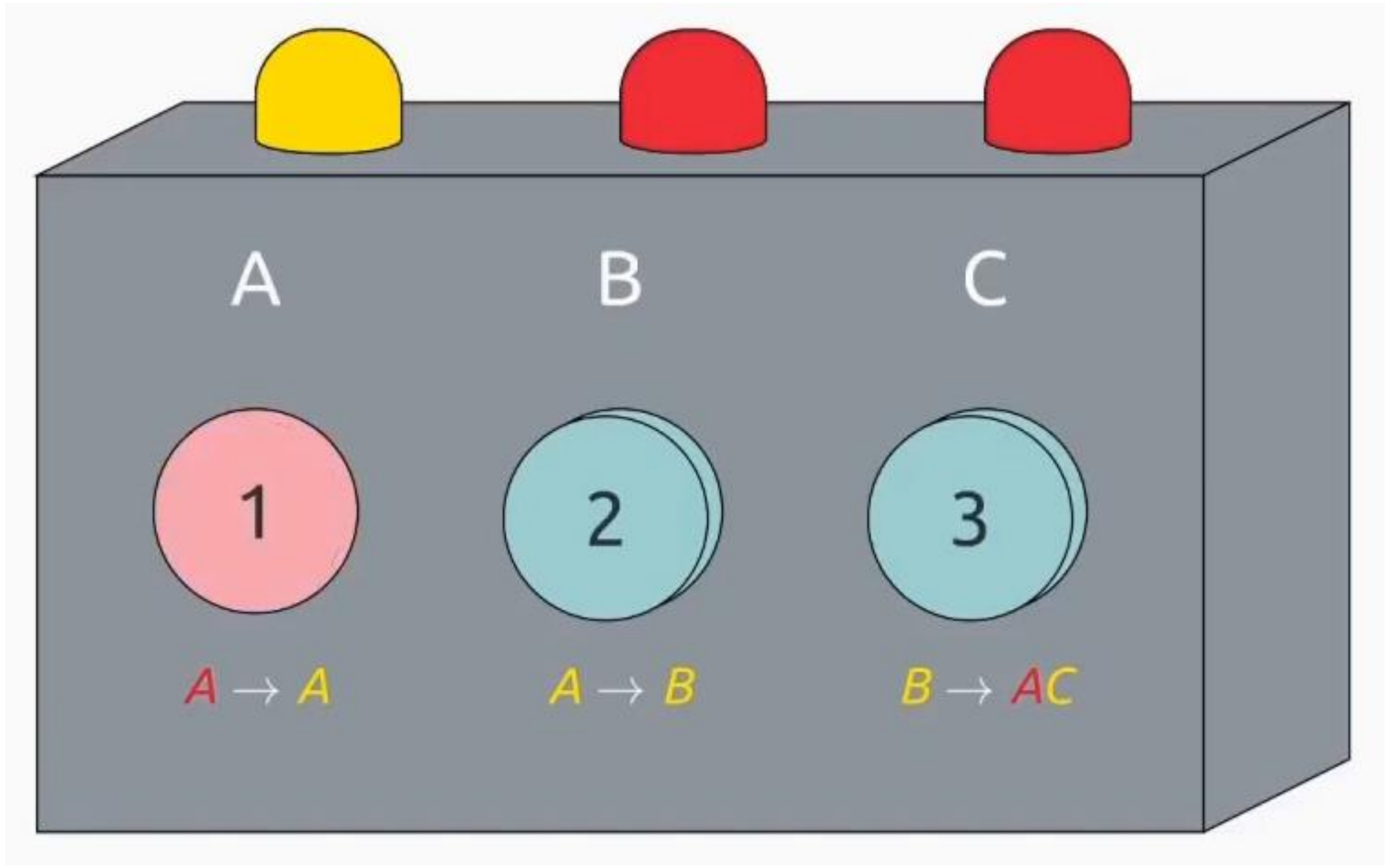
- Results can be improved more by combining landmark counting with other heuristics, and by using preferred operators (like helpful actions in FF)
- Operators that may be useful for improving heuristic value from a given state
- Operator is preferred in a state if applying it achieves an acceptable landmark in the next step, i.e., a landmark whose predecessors have already been accepted

Another Landmarks Example



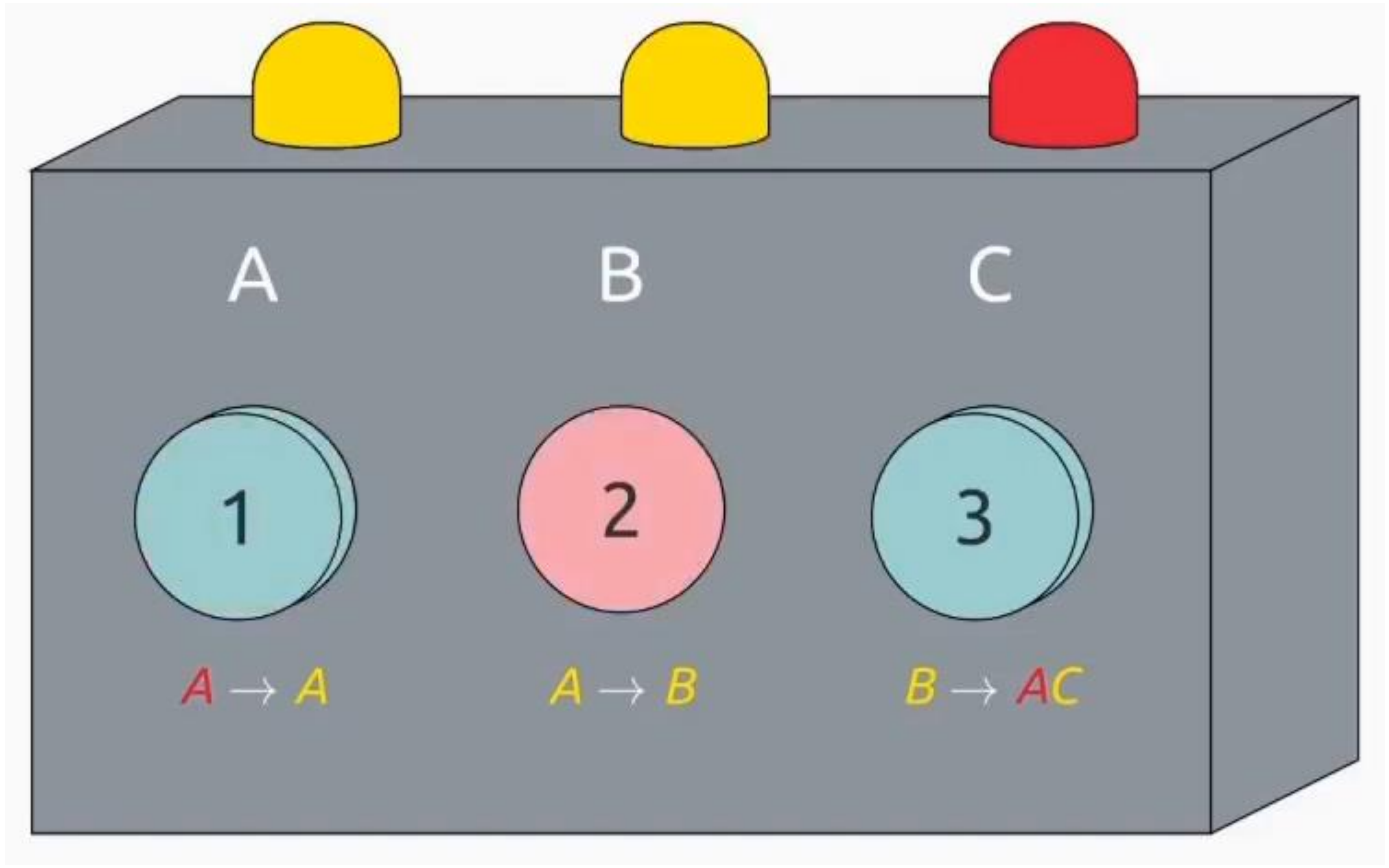
- Box with 3 lights on top: A, B, C
- 3 buttons: 1, 2, 3
- Initially all turned OFF
- Goal: Turn them all ON

Another Landmarks Example



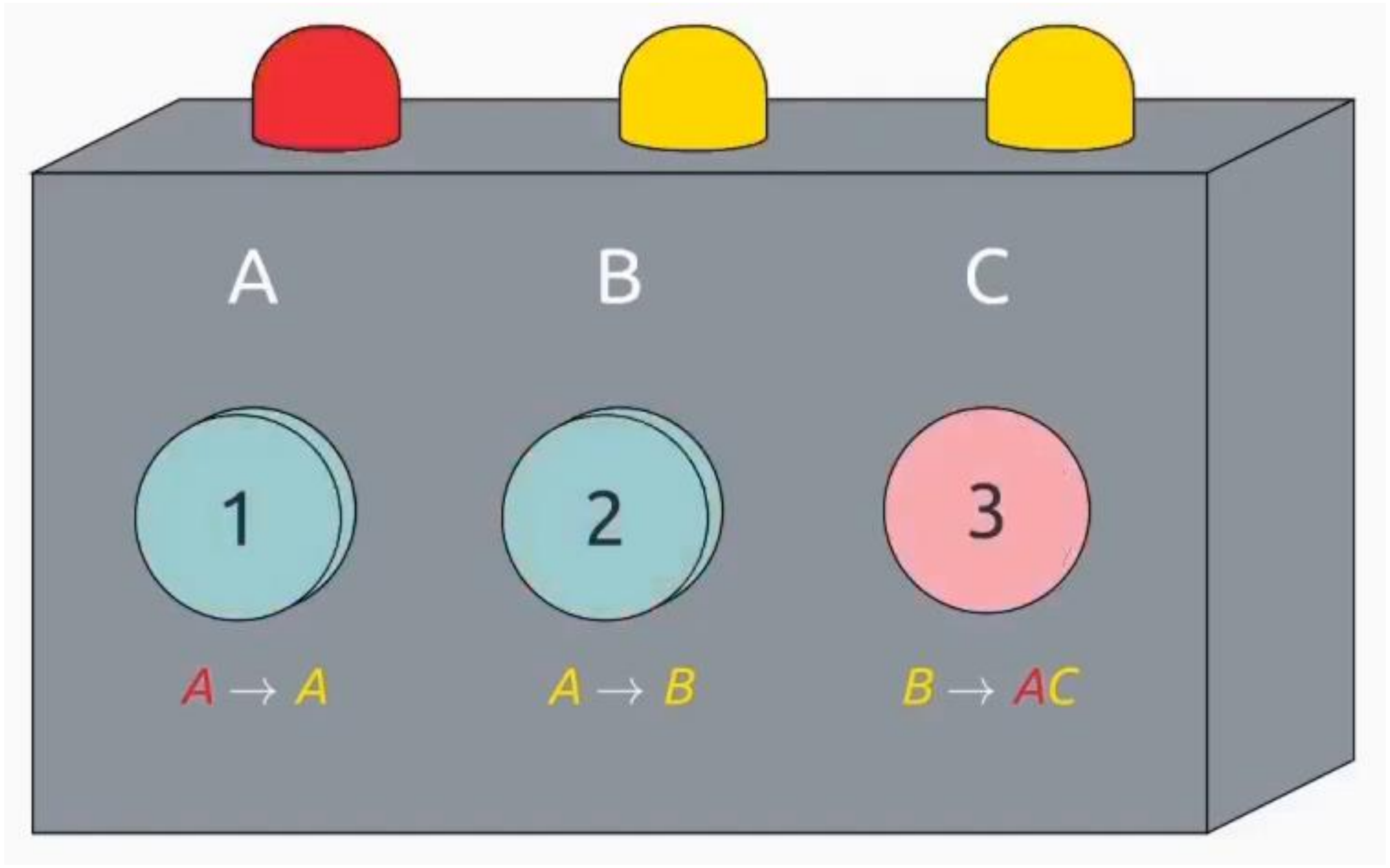
- Press button 1
- Turns ON light A

Another Landmarks Example



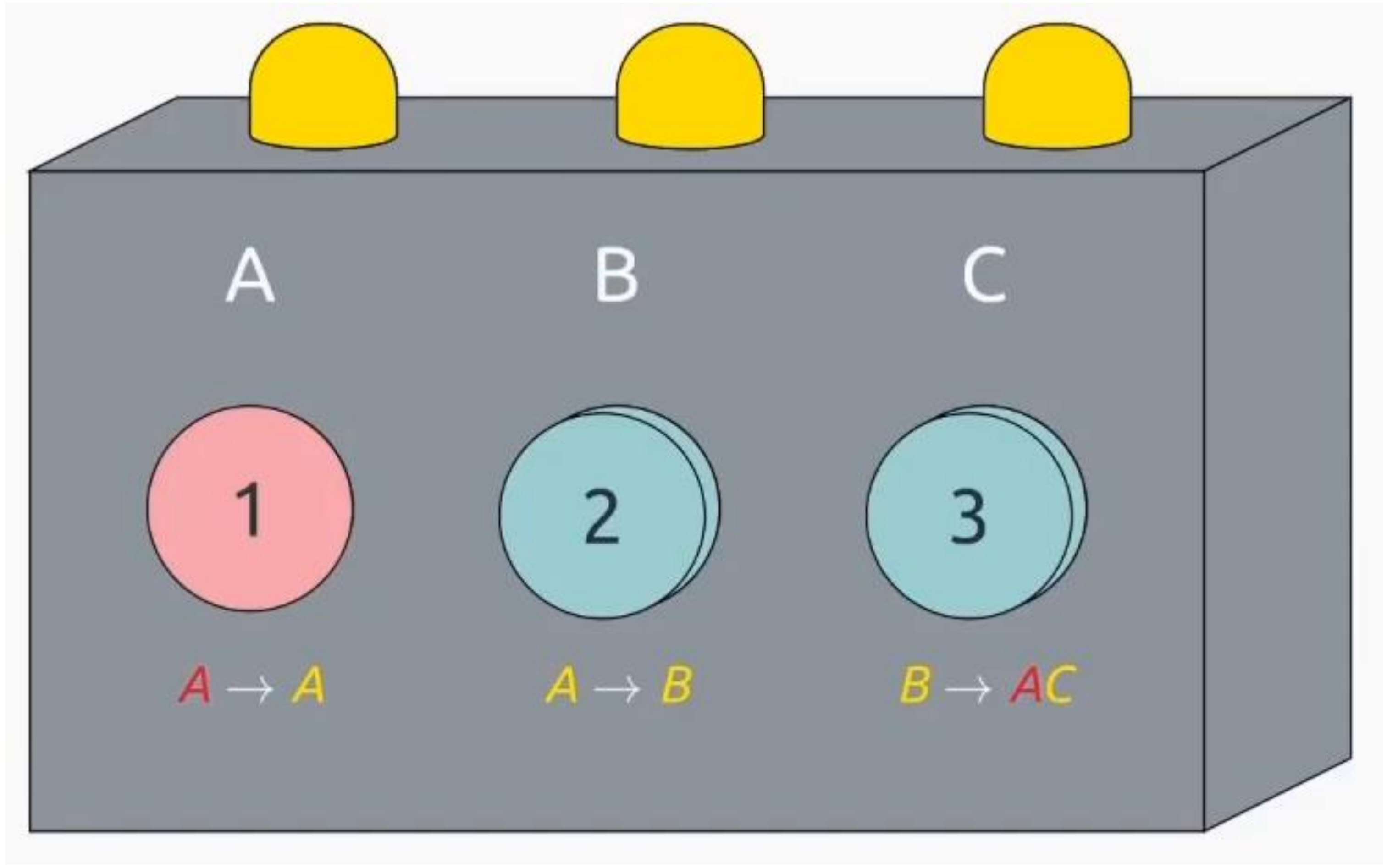
- Press button 2
- Turns ON light B

Another Landmarks Example



- Press button 3
- Turns ON light C
- Turn OFF light A

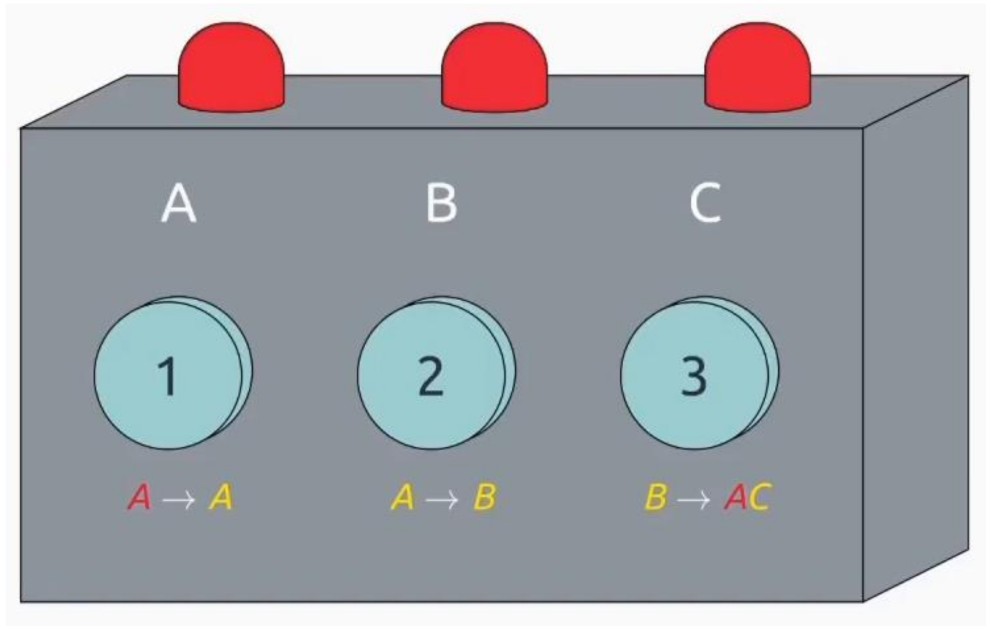
Another Landmarks Example



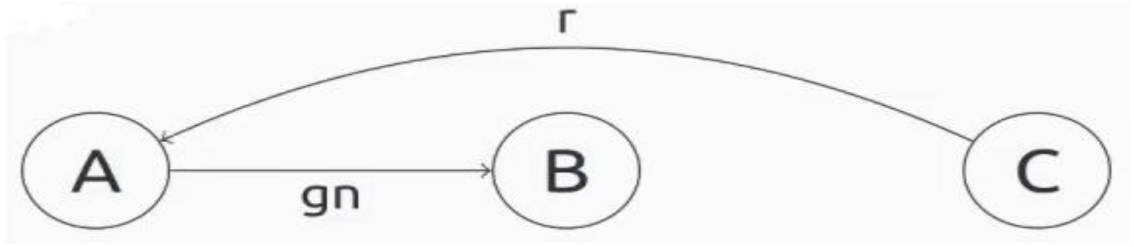
- Press button 1 again
- Turns ON light A again
- All lights ON

Landmarks

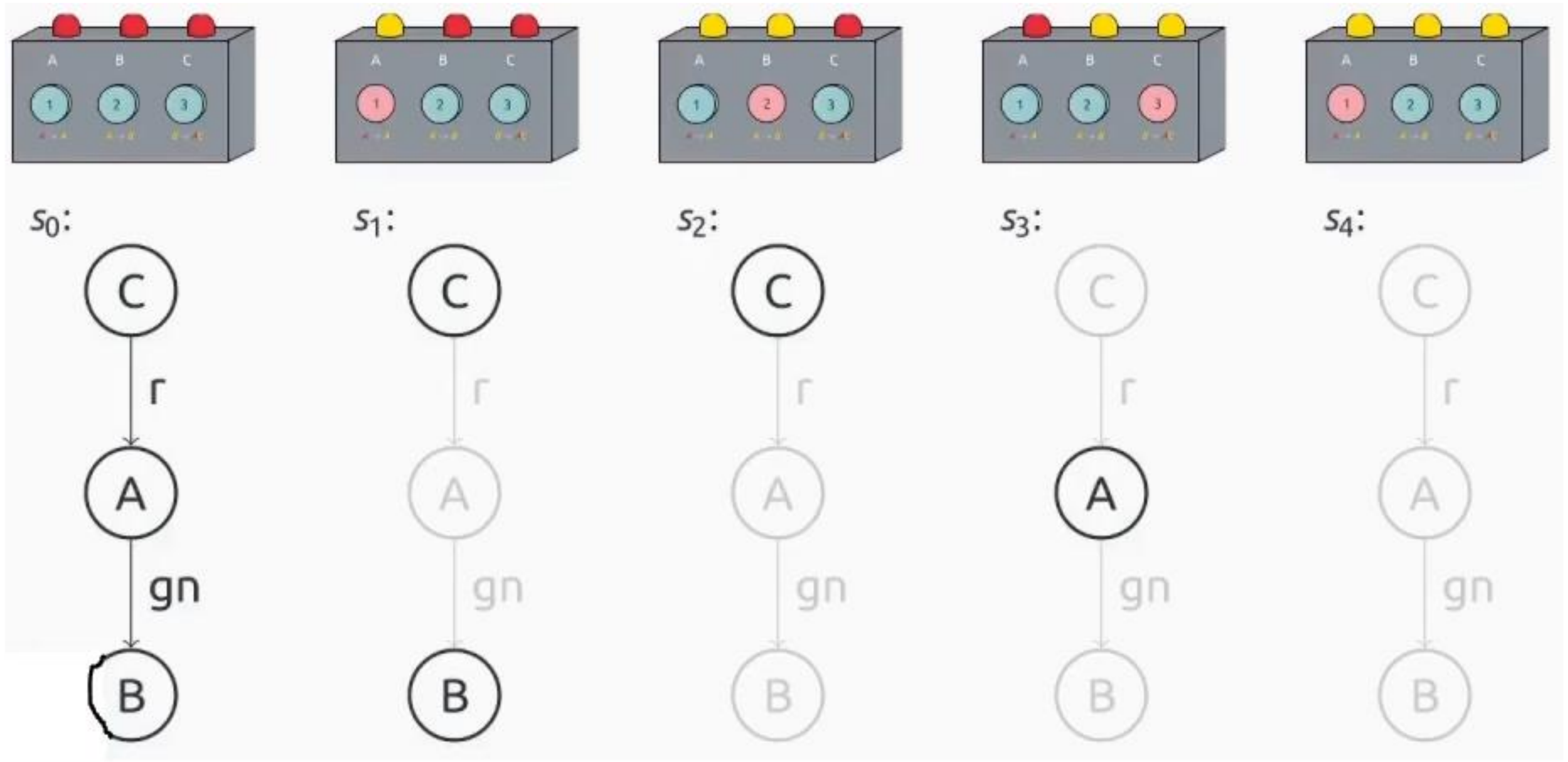
- Landmarks must occur in all plans
- A, B, and C must glow
- Landmark orderings: all plans follow specified order



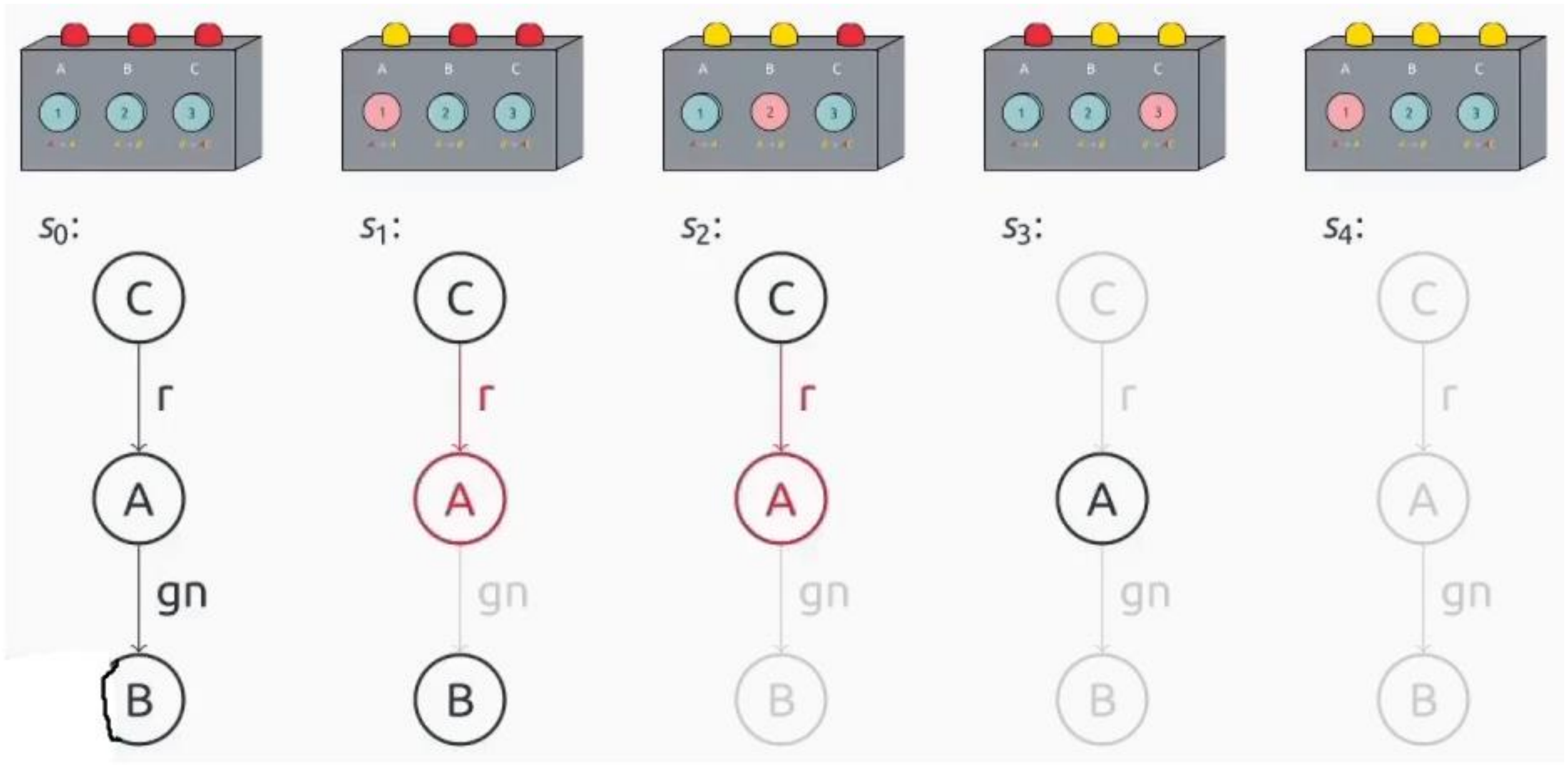
Landmark Graph



LM-A* (Karpas and Domshlak, 2009)



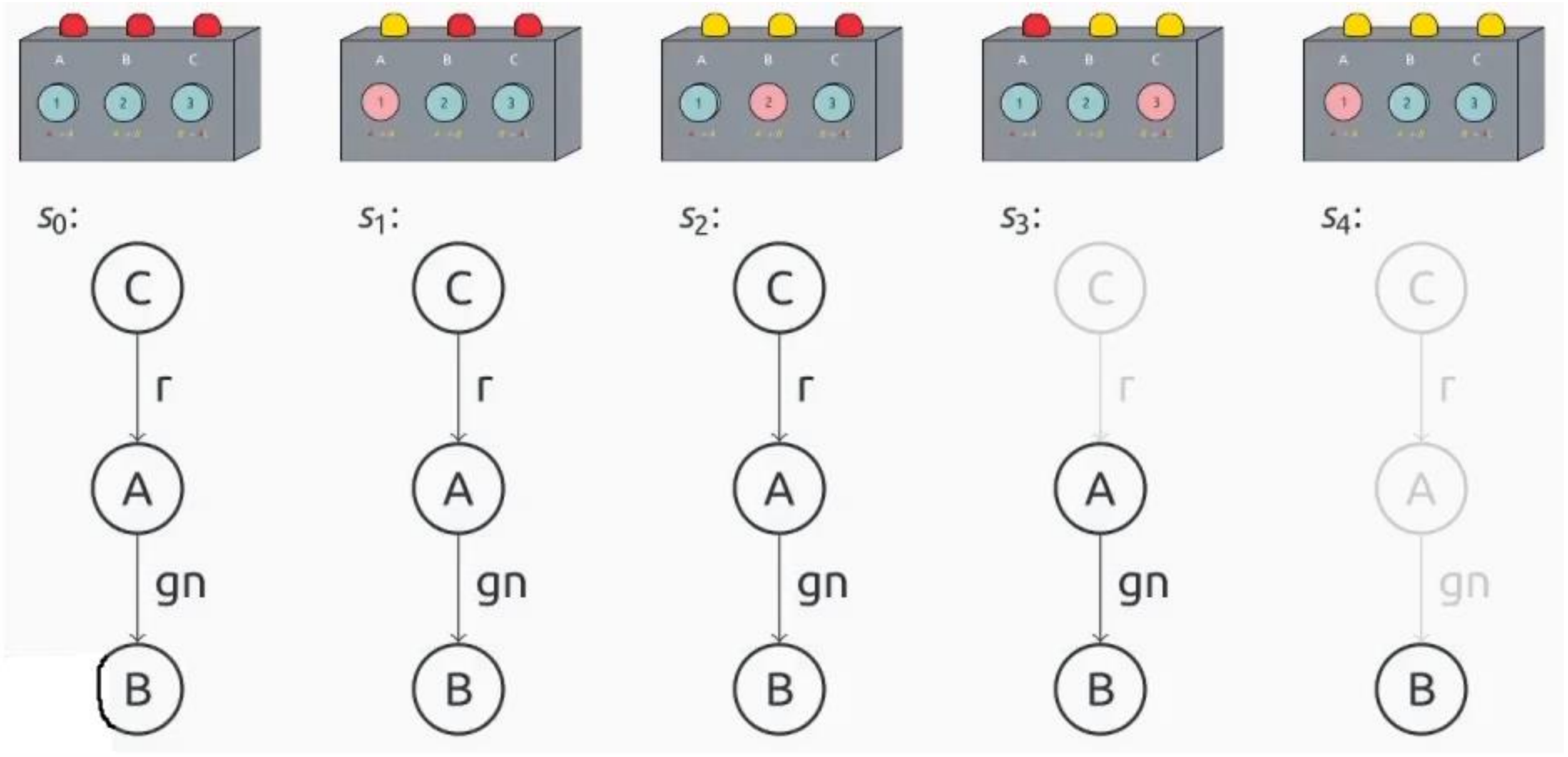
LM-A* (Karpas and Domshlak, 2009)



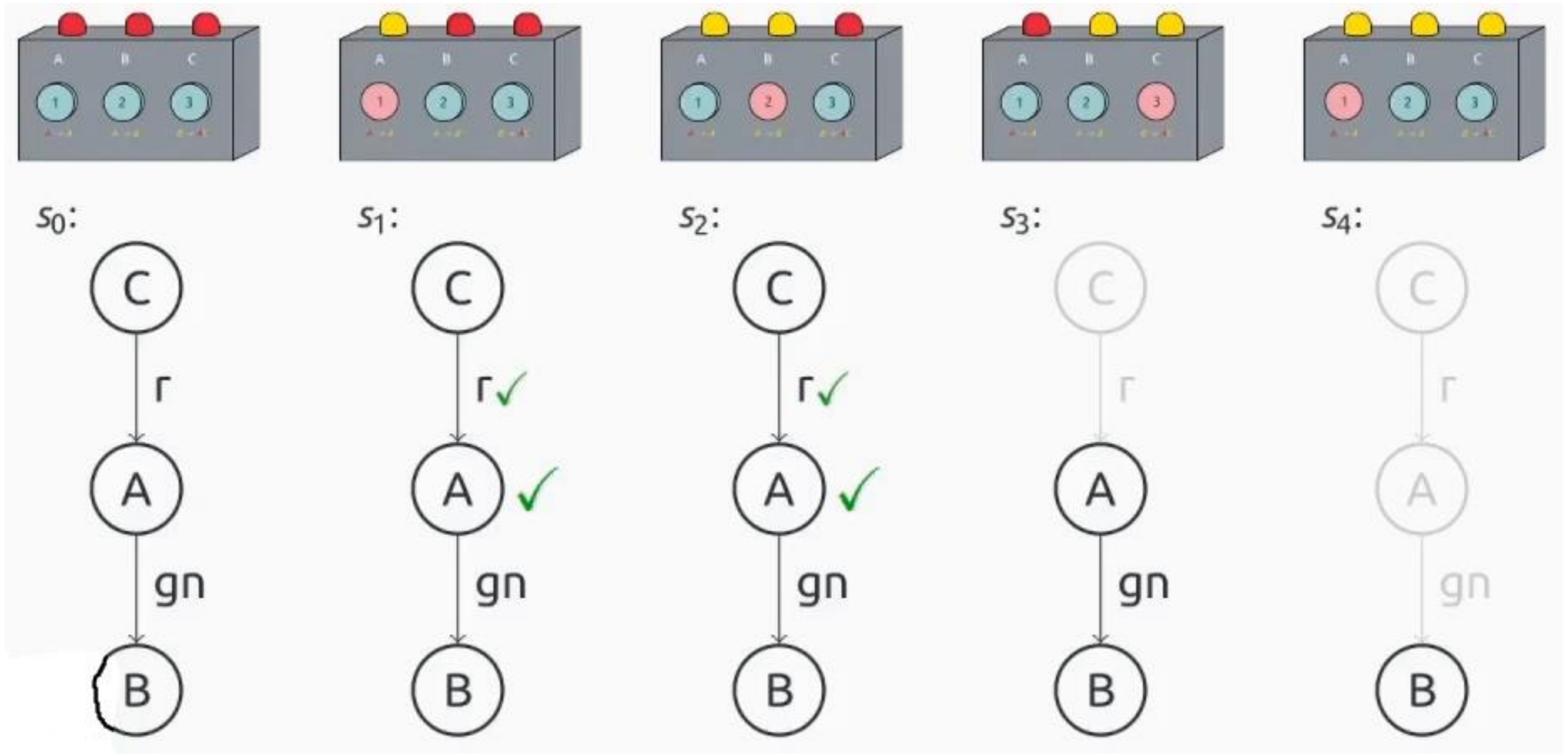
LM-A* & LAMA

- Both: Some landmarks are required again – goals if they do not hold and greedy-necessary preconditions if they do not hold
- LAMA: do not accept landmarks with unaccepted parents

LAMA (Richter and Westphal, 2010)



LAMA (Richter and Westphal, 2010)



Search Algorithm Used

- Framework: FD (Fast Downward) planner
- FD translates STRIPS tasks to SAS+ and uses best-first search to solve them
- Already contains functionality to combine various heuristics and use preferred operators
- When configured to use more than one heuristic and no preferred operators, the FD planner manages several queues for state expansion, one for each heuristic

Search Algorithm Used

- Any state that is evaluated during search is evaluated by all heuristics, and its successors are saved in each queue with the heuristic value computed by the heuristic of that queue
- When retrieving the state to evaluate next, FD alternates between the queues, thus giving equal importance to all heuristics
- If FD is configured to use preferred operators with one or more of the heuristics, it constructs an additional queue for each such heuristic

Search Algorithm Used

- When a state is evaluated and expanded, those successor states that are reached via a preferred operator are put into the preferred operator queues, in addition to being put into the regular queues
- States in the preferred operator queues thus are evaluated earlier on average
- In addition, FD can be configured to give even more impact to preferred operators by using those queues more often than the regular queues

Experiments

- First experiment: new method evaluated against 3 different base planners
- Landmarks pseudo-heuristic outperforms the other two alternatives (base and local)
- Even better results combining landmarks pseudo-heuristic with base heuristic: when using the FF or Causal Graph heuristic the results are significantly better than with the blind heuristic

Experiments

- Second experiment results: Finds more orderings
- Means better plan quality in terms of length of the solutions
- Using the landmarks pseudo-heuristic reduces plan lengths compared to the base planner for all three landmark generation methods
- Best result is achieved when using the landmarks generation procedure (heur-RHW) proposed in the paper

Conclusion

- A landmark (LM) is something that every plan must satisfy
- Landmark information can be used in a heuristic search framework to increase the number of problem instances solved and improve the quality of the solutions
- Can easily be combined with other heuristic information compared to earlier approach: FF and Causal Graph improved significantly with landmark information
- Next paper: Improvements to Fast Forward (Who Said We Need to Relax All Variables? by Katz, Hoffmann, and Domshlak)

Any Questions?

Thank you.

Have a good day!

References

1. Landmarks Revisited by Richter, Helmert, and Westphal
2. https://fai.cs.uni-saarland.de/hoffmann/papers/icaps13-award-slides.pdf
3. https://ai.dmi.unibas.ch/_files/teaching/hs21/po/slides/po-e02.pdf
4. https://cms.sic.saarland/planning19/dl/38/15._Landmark_Heuristics_post-handout.pdf
5. https://www.youtube.com/watch?v=rL4ja2x_nFQ

Extras

- 3 stages of LM^{RPG} algorithm by Hoffmann et al.
- First, potential landmarks and orderings are suggested by a fast candidate generation procedure: uses an approximation based on relaxed planning graphs
- Second, a filtering procedure evaluates a sufficient condition for landmarks on each candidate fact, removing those which fail the test (unsound landmark orderings may remain)
- Third, reasonable and obedient reasonable orderings between the landmarks are approximated

Extras

- Rather than building the relaxed planning graph using all operators and stopping when B first occurs, any operator is left out that would add B
- When the relaxed planning graph levels out, its last set of facts is an over-approximation of the set of facts that can be achieved before B in the planning task
- Denoted by pb(B) (for possibly before): any operator that achieves B and is applicable given pb(B) qualifies as being possibly applicable before B in the original task