Prohledávání v nejistých prostředích

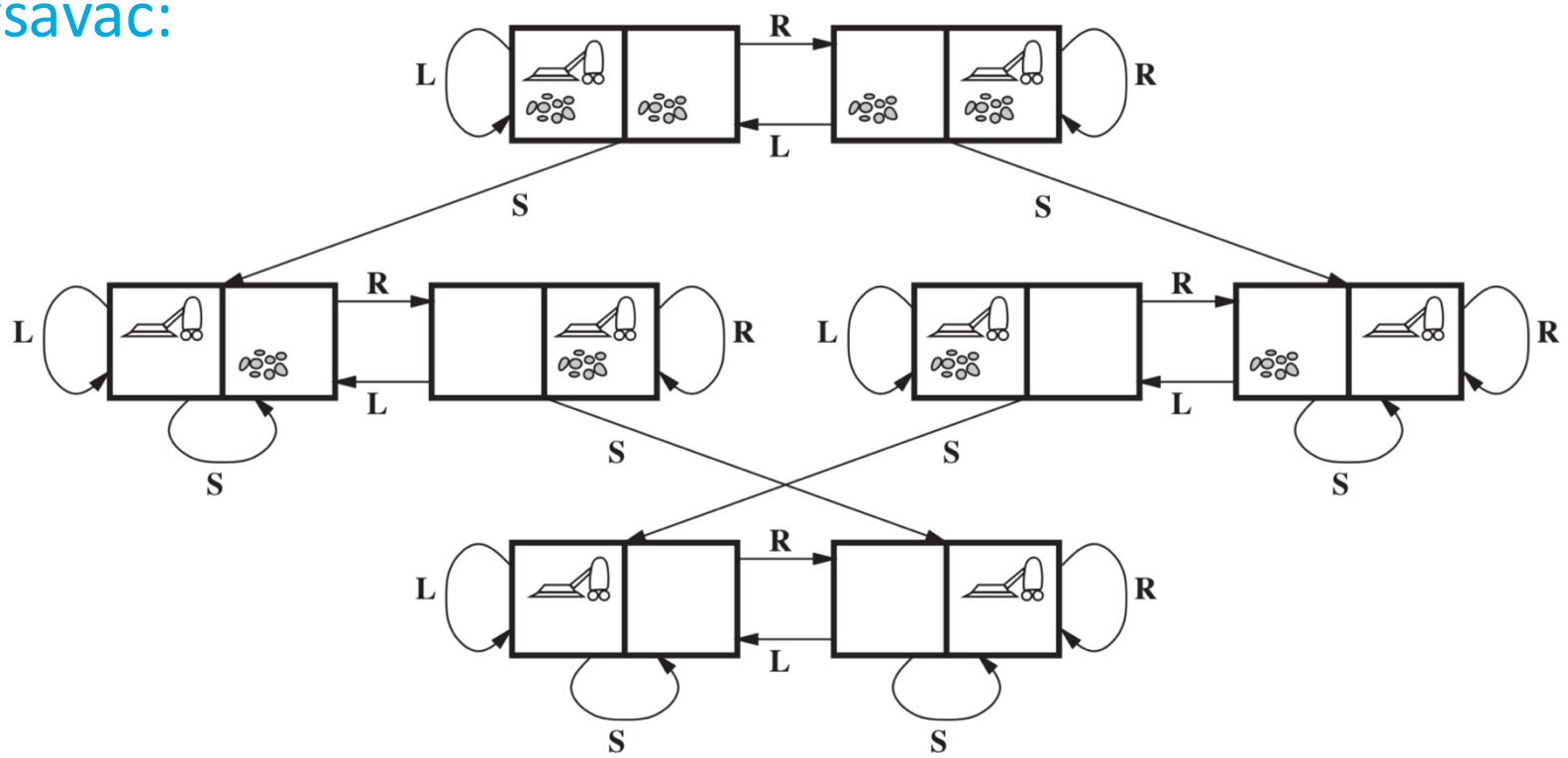
Martin Šůstek

isustek@fit.vutbr.cz

Prohledávání s nedeterministickými akcemi

- Problém nelze řešit jen jako pevně danou sekvenci akcí
  - → Potřeba **plánu/strategie** (contingency plan, strategy)
- Úprava formální definice problému
  - Přechodový model: $\{s_1, s_2, \ldots, s_n\} = RESULT(s_{old}, a)$
    - ← Nahrazuje $s_{new} = RESULT(s_{old}, a)$
  - Cena akce může záviset na novém stavu $COST(s, a, s')$
    - ← Nahrazuje $COST(s, a) \rightarrow \mathbb{R}_0^+$

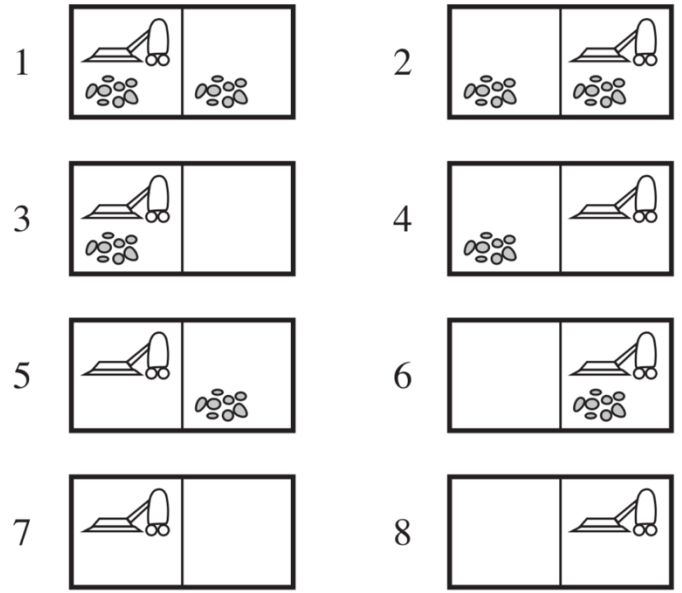
- Původní vysavač:

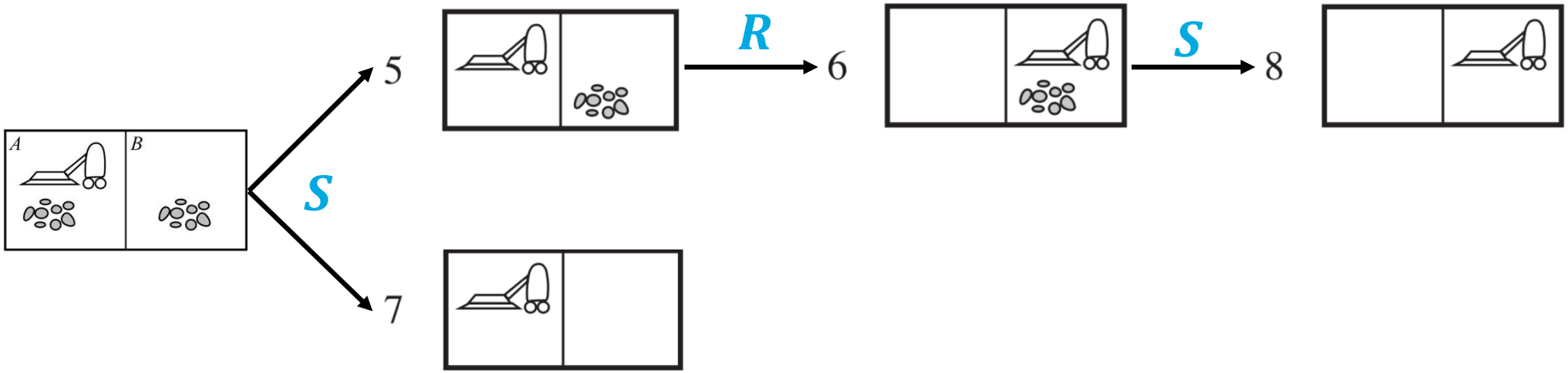


- Modifikovaný vysavač
  1. Akce ***S*** *vysát* (*suck*) na špinavém políčku někdy vysaje i vedlejší políčko
  2. Akce ***S*** *vysát* (*suck*) na čistém políčku někdy pole ušpiní

Příklad: Modifikovaný vysavač – příklad plánu

- Počáteční stav 1
- Plán: [*Suck*, **if** State = 5 **then** [*Right*, *Suck*] **else** [ ]]
- Potřebujeme vjemy → nový stav





AND-OR prohledávací stromy

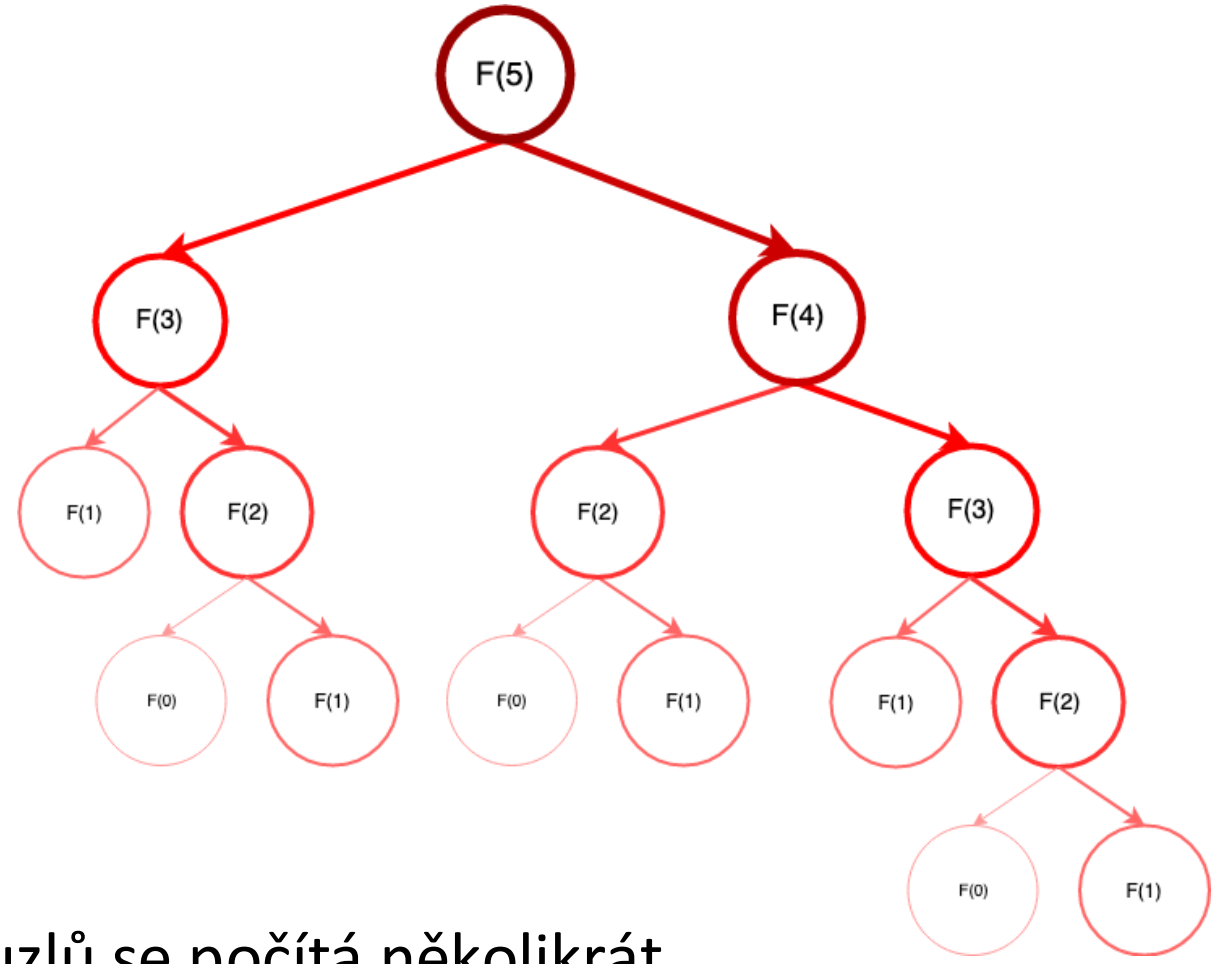
- OR – můžu si vybrat akci, kterou provedu
- AND – po provedení akce se mohu dostat do více stavů
  - Musím vyřešit všechny vzniklé podproblémy

- Jak souvisí AND/OR pojetí s případem s deterministickými akcemi?

- Dynamické programování – myšlenka uložení a znovupoužití výpočtu
  - Pokud po prozkoumání podstromu uzlu se stavem A zjistíme, jestli je řešitelný, narazíme-li znovu na uzel se stavem A, rovnou použijeme předchozí výsledek
  - Pokud narazíme na stav, který se již na cestě k tomuto uzlu nachází, pak dál neprohledáváme, abychom neprocházeli pořád ty stejné sekvence stavů (detekovali jsme **cyklus**) → algoritmus skončí v konečném stavovém prostoru

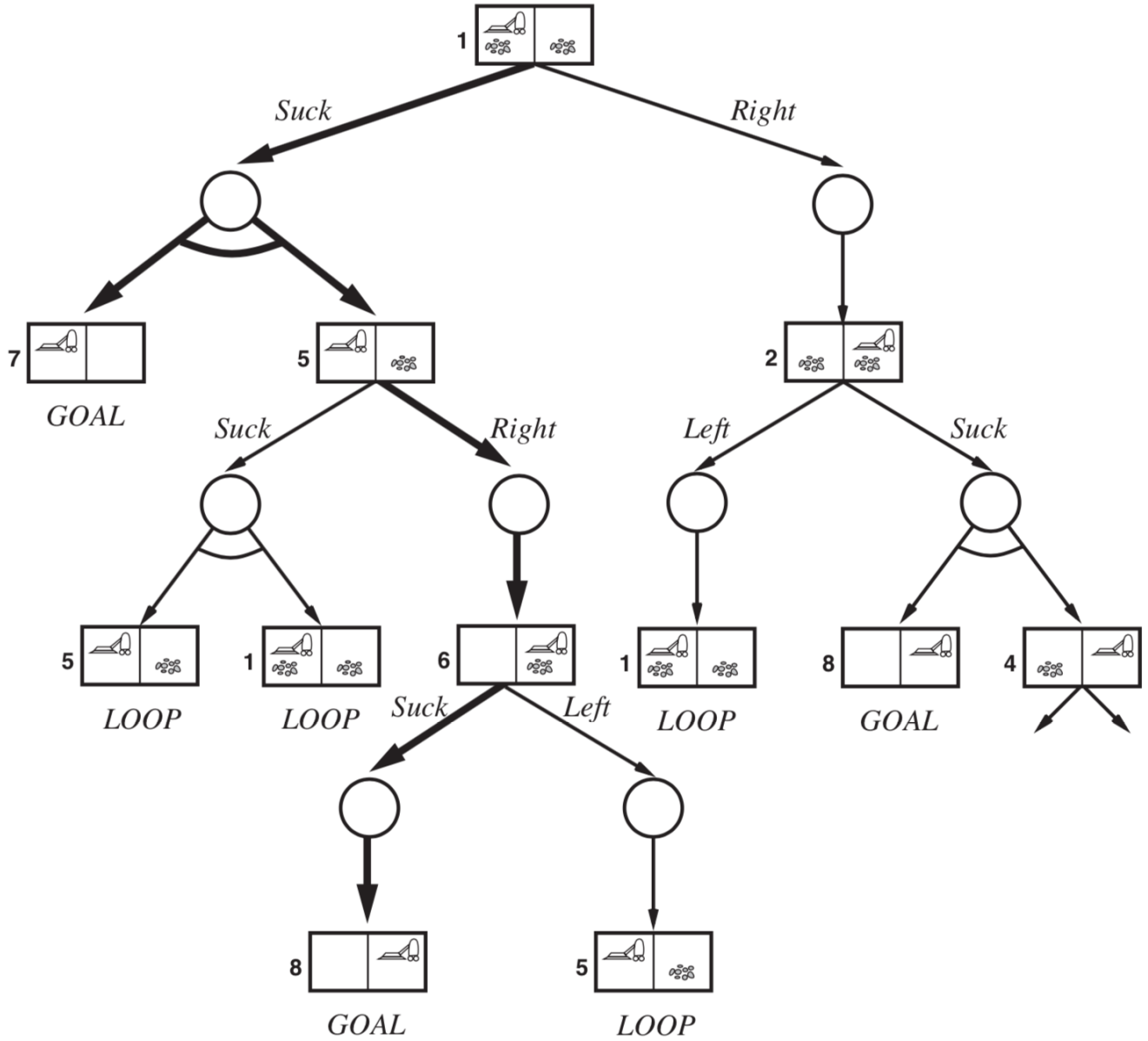
Příklad dynamického programování: Fibonacciho posloupnost

- Fibonacciho posloupnost:
  - $F(0) = 0$
  - $F(1) = 1$
  - $F(n) = F(n-2) + F(n-1)$



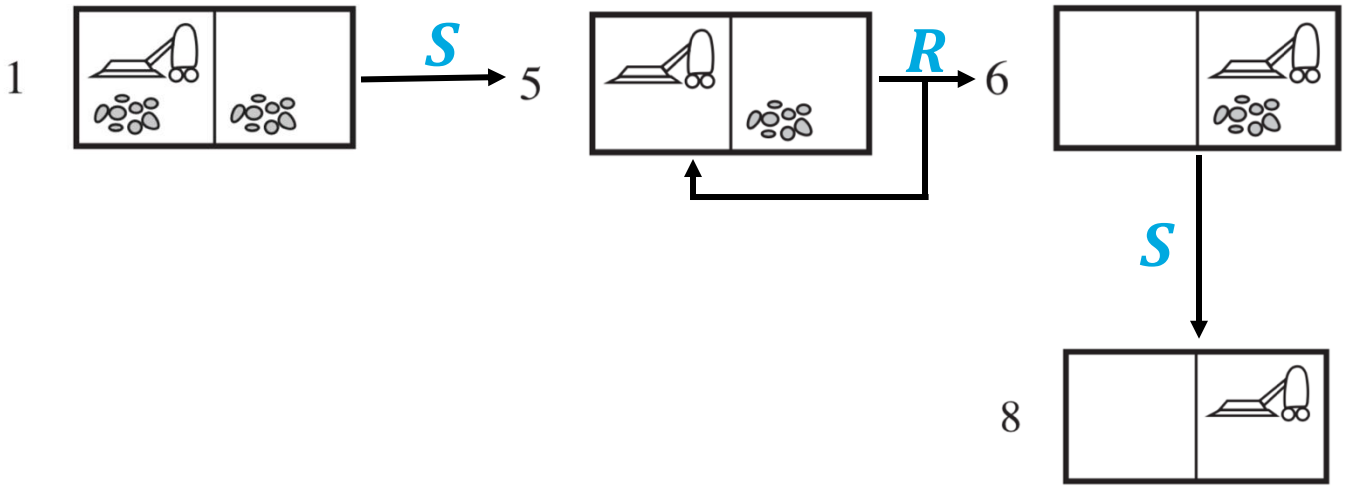
- Můžeme využít rekurzi, ale většina uzlů se počítá několikrát
  - → Spočítáme jednou a uchováme

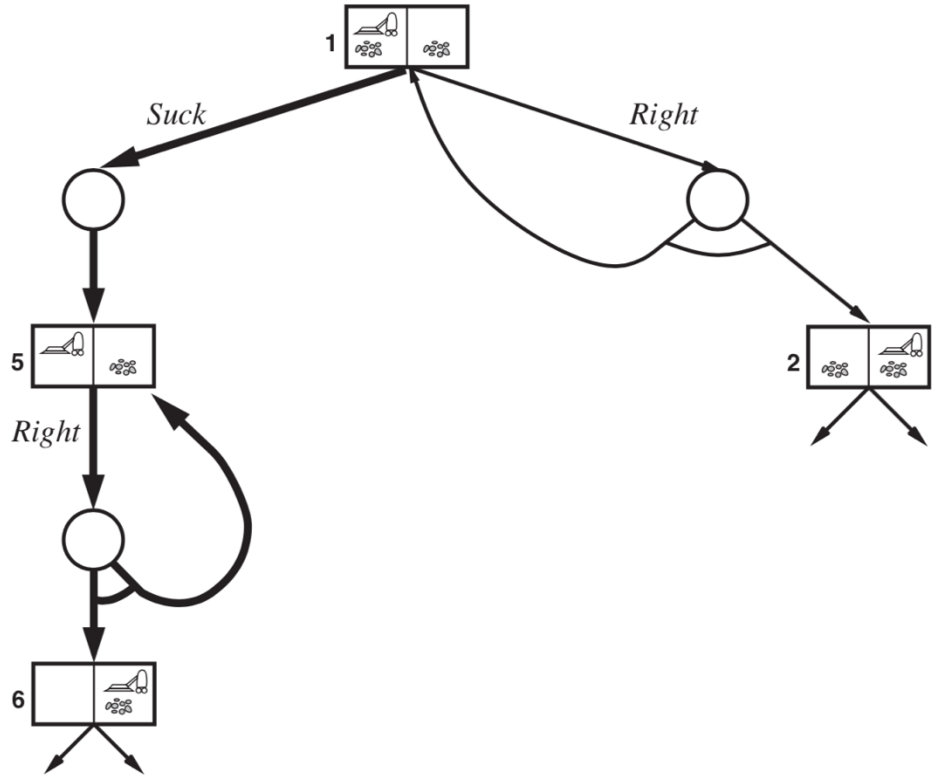
AND-OR prohledávací stromy: Příklad



Nedeterministické akce: Pohyb může selhat

- Akce pro pohyb (left $\boldsymbol{L}$, right $\boldsymbol{R}$) někdy selžou → vysavač zůstane na stejné pozici
- Abychom byli schopni nalézt řešení, musíme povolit cykly a předpokládat, že se pouhým opakováním dostaneme do alternativního stavu
  - Možný plán ze stavu 1: [*Suck*, L1 : Right, **if** State = 5 **then** L1 **else** Suck]





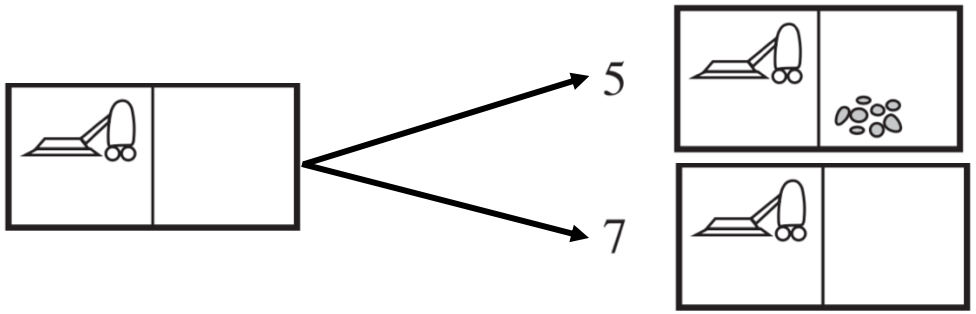
Nedeterministické akce: Komplikace

- Uvažujeme opakování akce, dokud nedosáhneme kýženého stavu
  - → Potenciálně nekonečný cyklus
- Příklad:
  - Otevření dveří pomocí čipové karty se povede s 90% pravděpodobností; nyní jsem zkusil 10x v řadě, ale bez úspěchu → jak řeší lidé?
  - Co když upadne kolečko od vysavače?
- Je třeba změna formulace typu prostředí:
  - Plně pozorovatelné nedeterministické → **částečně pozorovatelné a deterministické**

Částečně pozorovatelné prostředí

- Belief state – reflektuje agentovu důvěru o aktuálním stavu
  - V jakých všech stavech se agent může aktuálně nacházet

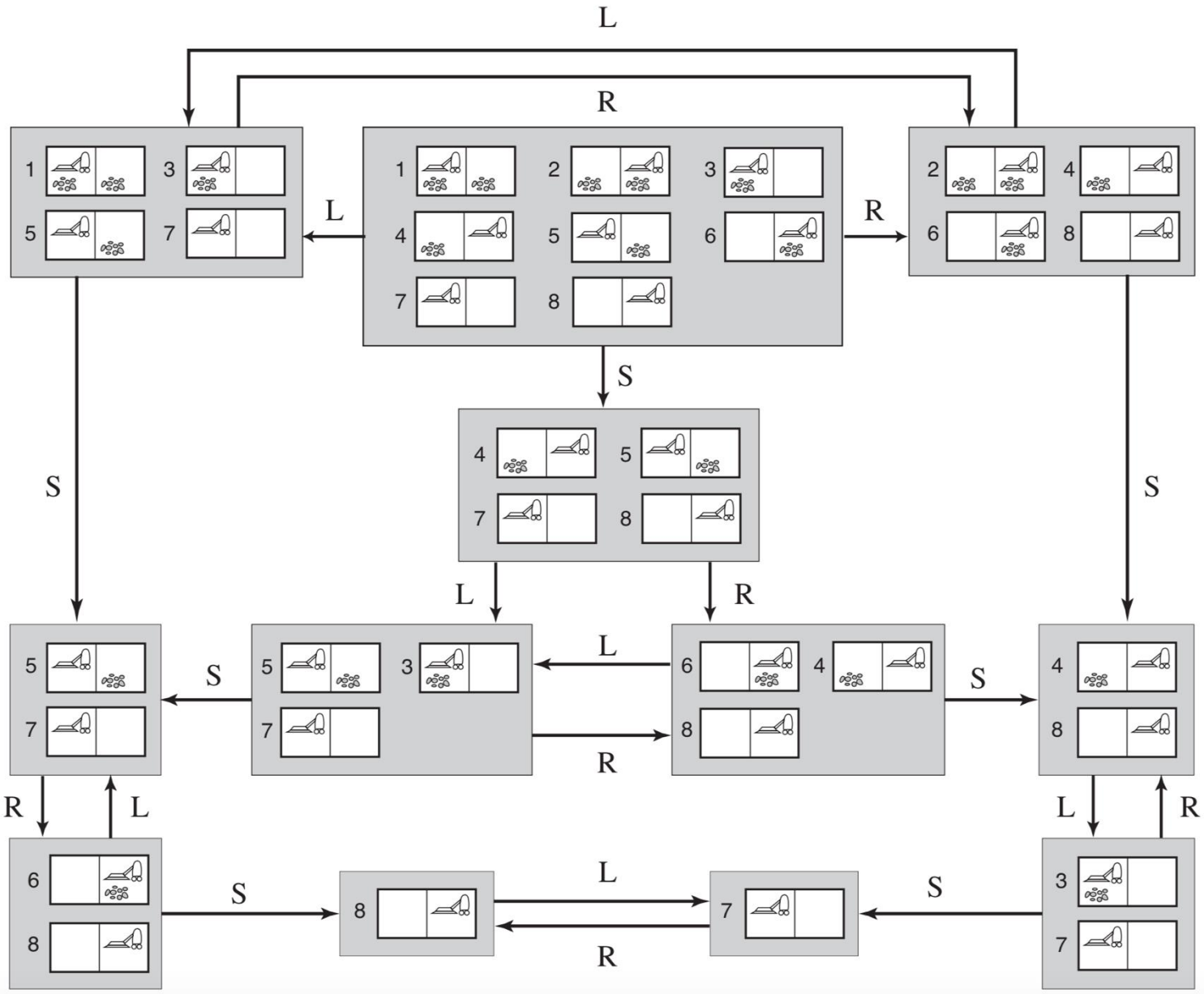
- Příklad: Lokální senzor
  - Vysavač dokáže detekovat nečistotu jen v aktuálním poli
    - Vysavač je vlevo a pole je čisté



Částečně pozorovatelné prostředí: Prohledávání bez senzorů

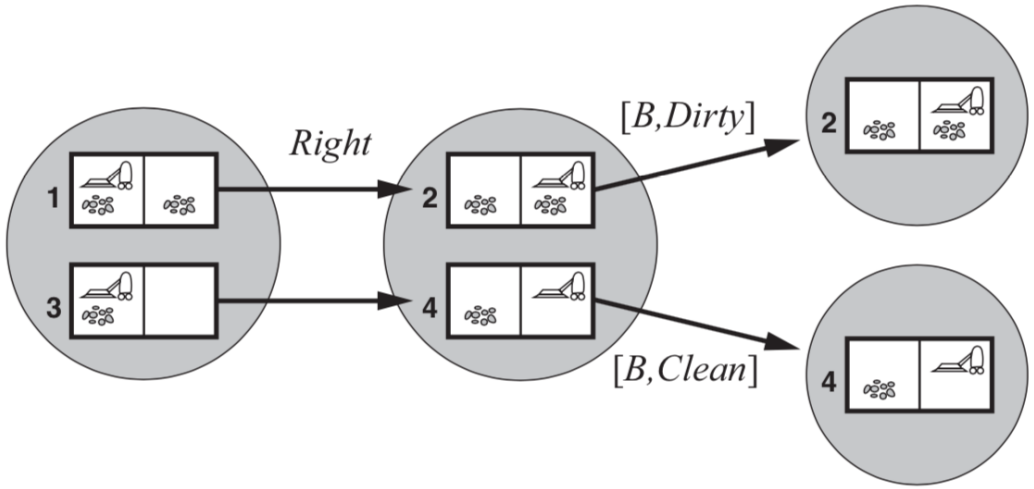
- Nemá senzory → nepozorovatelné prostředí
- Příklad: Širokospektrá antibiotika, krevní skupina 0 – racionalita?
- Pro N možných stavů je $\mathbf{2^N}$ možných **belief states**
  - Prakticky nebývá problém → velká část belief state je nedosažitelná
- Obtížná reprezentace **jednoho** belief state, příklad: 10 políček k vysávání
  - → 2^{10} kombinací špinavých a čistých polí
  - → Reprezentace výčtem stavů není možná
  - **Incremental belief-state search** – postupně hledáme řešení pro každý stav
    - Výhoda: pokud neexistuje řešení, zjistí se rychle

Prohledávání bez senzorů: Příklad

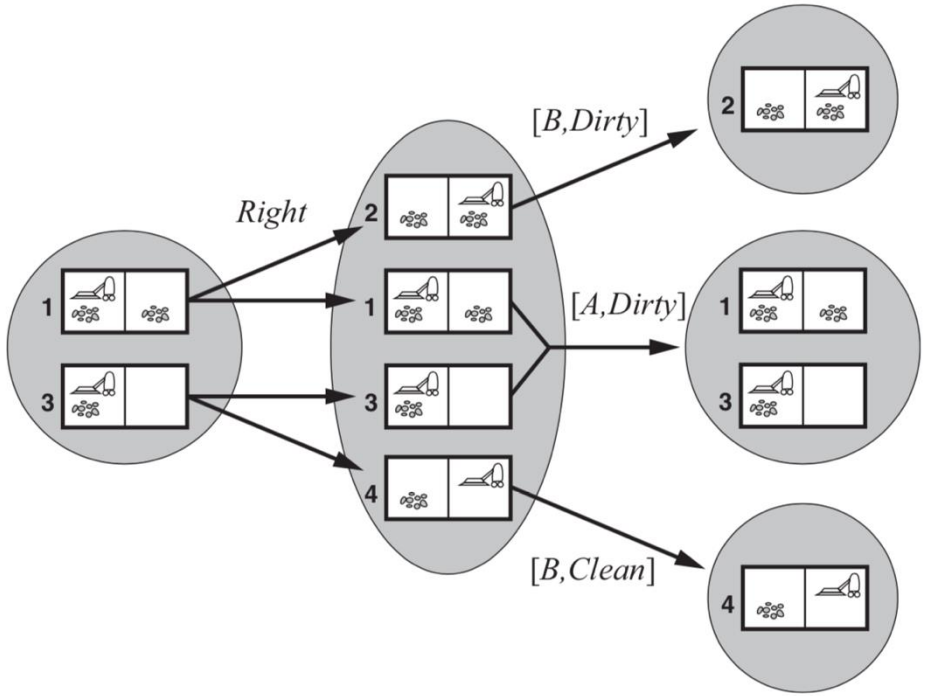


- Nedeterminismus v částečně pozorovatelných prostředích plyne z nemožnosti predikovat, co bude výsledkem akce (které vjemy agent obdrží)

Deterministické akce

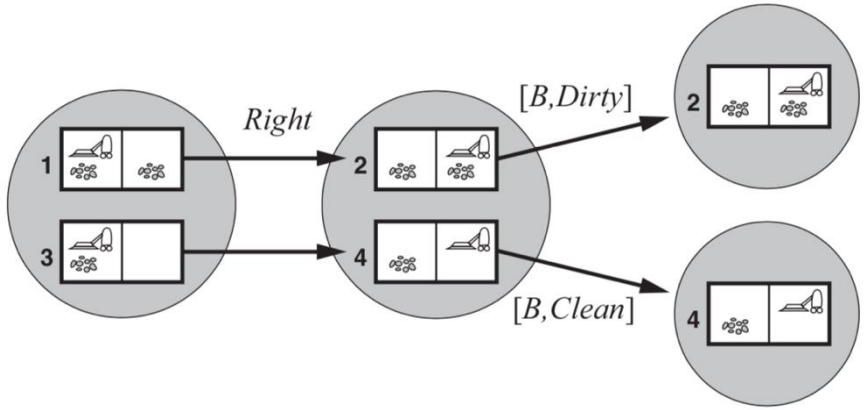


Pohyb někdy selže



Lokální senzor: Řešení

- Možný plán řešení pro deterministické akce:
  - [*Suck*, *Right*, **if** Bstate = {6} **then** *Suck* **else** [ ]]

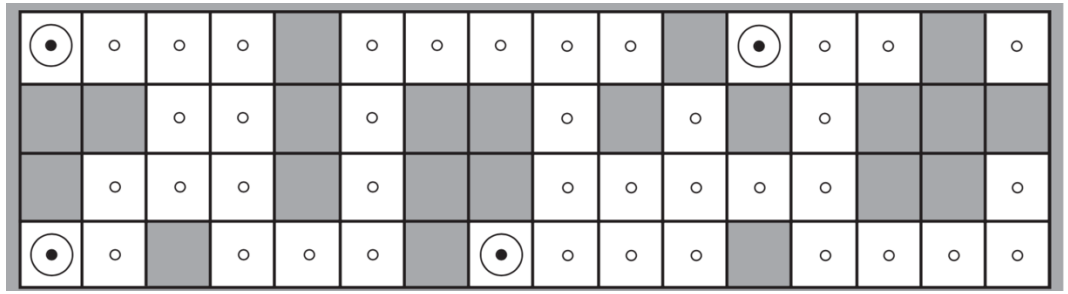


- Každý belief state (Bstate) je reprezentován atomicky – bez vnitřní struktury
  - Optimalizace: některé stavy jsou nadmnožiny/podmnožiny jiných
    - Detekce cyklů → neprozkoumáváme
    - Pokud známe řešení pro nadmnožinu stavů → lze aplikovat i pro podmnožinu

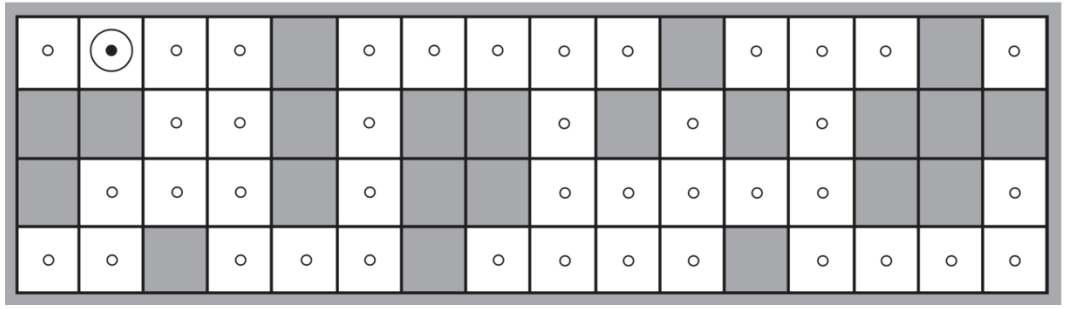
- **Udržování si believe state** je základem inteligentního agenta v částečně pozorovatelných prostředích – většina reálných prostředí

- Lokalizace – určení, kde se agent nachází
- Příklad:
  - Agent má mapu areálu, je na neznáme pozici, ale dokáže určit, jestli je ve směru každé světové strany překážka nebo volno

1. krok (NWS)

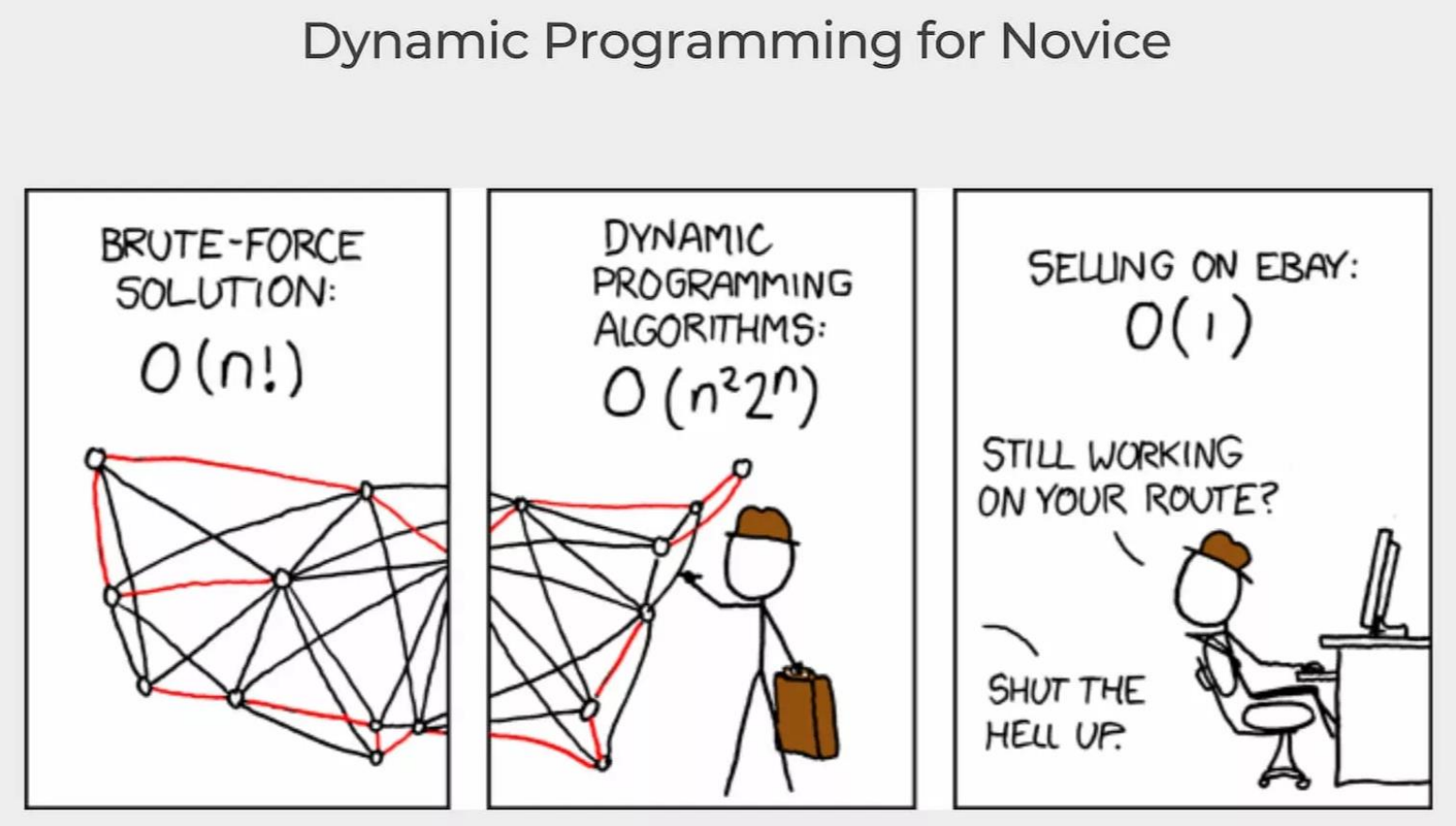


Akce: posuň se na východ



2. krok (NS)

Děkuji za pozornost!



- https://ksp.mff.cuni.cz/encyklopedie/zakladni-algoritmy/
- https://www.synebo.io/blog/the-anatomy-of-dynamic-programming-with-codes-and-memes/