Project: Sokoban Solver

Samuel Ješík

20.1.2025

Úvod

Sokoban je klasická logická hra, ktorej cieľom je posúvať krabice na určené skladovacie pozície v uzavretej mriežke. Hráč sa môže pohybovať hore, dole, doľava alebo doprava a môže tlačiť krabicu, ak je za ňou voľné miesto. Naopak, hráč nemôže krabice ťahať ani prechádzať cez steny. Táto hra je ideálnym príkladom plánovacieho problému v umelej inteligencii (AI), pretože vyžaduje strategické rozhodovanie a efektívne riešenie algoritmami.

Cieľom tohto projektu je vyriešiť Sokoban logické úlohy pomocou algoritmu Breadth-First Search (BFS), implementovaného v Prologu, a zobraziť riešenie prostredníctvom knižnice Pygame v jazyku Python. Projekt sa zameriava na formalizáciu Sokobanu ako plánovacieho problému, definovanie jeho pravidiel a mechaník, a poskytuje štruktúrovaný proces riešenia od vstupu (definícia mapy) po výstup (vizualizácia riešenia).

Pravidlá hry a definícia vstupu a výstupu

Sokoban sa hrá na mriežke, ktorá obsahuje nasledujúce elementy:

- **Hráč (S)** – Hráč sa môže pohybovať hore, dole, doľava alebo doprava, ak je cieľová pozícia prázdna alebo obsahuje cieľové pole.
- **Krabica (C)** – Krabica sa posúva v smere pohybu hráča, ak je za ňou voľné pole.
- **Cieľové pole (X)** – Miesto, kde má byť umiestnená krabica.
- **Stena (#)** – Neprekonateľná prekážka.
- **Sokoban v cieľovej pozícii (s)** – Hráč stojaci na cieľovom poli.
- **Krabica na cieľovom poli (c)** – Krabica už správne umiestnená na cieľovom poli.

Cieľom hry je dostať všetky krabice na cieľové polia s použitím minimálneho počtu krokov.

Vstup

Vstupom do riešenia je mapa Sokoban hádanky definovaná v textovom súbore. Každý znak predstavuje určitý element mriežky podľa predošlého bodu.

Príklad vstupnej mapy z textového súboru `maps/map1.txt`:

```
######
#s   #
#CCCX#
#X   #
######
```

Výstup

Výstup obsahuje nasledujúce informácie:

- **Sekvenciu krokov**: Zoznam akcií potrebných na vyriešenie mapy, pričom každá akcia zahŕňa smer pohybu hráča a prípadne potlačenie krabice.
- **Generované fakty**: Fakty uložené v súbore `facts.pl`, ktoré reprezentujú počiatočný stav, steny, cieľové polia, krabice a riešenie.
- **Animovaná vizualizácia**: Zobrazenie riešenia pomocou knižnice Pygame.

Príklad výstupu pre mapu `map1.txt`:
Sekvencia akcií:

```
Dole + potlačil krabičku
Hore
Vpravo
Vpravo
Vpravo
Dole
Dole
VĺavoVĺavo
Hore + potlačil krabičku
Vpravo + potlačil krabičku
Hore
VĺavoVĺavo + potlačil krabičku
```

Animovaná vizualizácia zobrazí jednotlivé kroky riešenia a na záver vypíše správu *"Mapa je vyriešená!"*.

Formalizácia Sokobanu ako plánovacieho problému

Sokoban bol implementovaný ako plánovací problém s jasne definovanými akciami, predpokladmi, efektmi, počiatočným a cieľovým stavom. Riešenie zahŕňa elimináciu problému s rámcom (*frame problem*) a využíva algoritmus BFS na hľadanie riešenia.

Akcie

Akcie predstavujú pohyby hráča v mriežke:

```
action(up,    0, -1).
action(down,  0,  1).
action(left, -1,  0).
action(right, 1,  0).
```

Predpoklady a efekty:

- **Pohyb bez tlačenia krabice:**
  - **Predpoklady:** Cieľová pozícia (`NewX, NewY`) nie je stenou ani krabicou.

  - **Efekty:** Aktualizuje sa len pozícia hráča.

- **Pohyb s tlačením krabice:**
  - **Predpoklady:** Pred hráčom je krabica a jej cieľová pozícia nie je stenou ani obsadená inou krabicou.
  - **Efekty:** Aktualizuje sa pozícia hráča a presunutie krabice.

Počiatočný a cieľový stav

Počiatočný stav: Obsahuje pozíciu hráča a krabíc:

```
initial_state(state(player(X, Y), Boxes)).
```

Cieľový stav: Všetky krabice sú na skladovacích miestach:

```
goal(state(_, Boxes)) :-
    forall(member(box(Bx, By), Boxes), storage(Bx, By)).
```

Frame problem

Efekty akcií ovplyvňujú iba zmenené časti stavu:

- Pohyb hráča mení iba jeho pozíciu.
- Pohyb krabice aktualizuje len konkrétnu krabicu.

Pozadie (Background Knowledge)

Riešenie Sokobanu vyžaduje definovanie pravidiel a predpokladov, ktoré popisujú svet hry:

- **Steny a skladovacie miesta:** Skladovacie miesta (`storage/2`) a steny (`wall/2`) sú pevne definované pre každú mapu.
- **Krabice a hráč:** Počiatočné pozície krabíc (`initial_box/2`) a hráča (`initial_player/2`) sa načítajú z mapového súboru.
- **Stavy:** Stav je reprezentovaný ako `state(player(X, Y), Boxes)`, kde `player(X, Y)` je pozícia hráča a `Boxes` obsahuje zoznam pozícií krabíc.
- **Pohyby:** Definované akcie (`action/3`) umožňujú hráčovi pohybovať sa hore, dole, vľavo a vpravo, pričom berú do úvahy obmedzenia ako steny a pozície krabíc.
- **Deadlocky:** Aby sa predišlo zbytočnému skúmaniu nevyriešiteľných stavov, implementovali sme detekciu mŕtvych uzlov (*deadlocks*). Tieto stavy vznikajú, ak je krabica zatlačená do pozície, z ktorej sa nemôže posunúť na cieľové miesto.

Implementácia

Riešenie Sokobanu bolo implementované kombináciou jazyka Prolog a Python. Logická časť riešenia, zahŕňajúca formálne pravidlá hry, definície stavov a akcií, ako aj algoritmus na hľadanie riešenia pomocou BFS, bola napísaná v Prologu. Python bol použitý na spracovanie vstupov, generovanie faktov pre Prolog, vizualizáciu riešenia pomocou knižnice Pygame a na interakciu s používateľom.

Kľúčové časti implementácie:

- **Prologové súbory:**
  - `actions.pl`: Obsahuje definície akcií (`up`, `down`, `left`, `right`) a implementáciu pohybov hráča (`move/3`). Zahŕňa aj podmienky a efekty spojené s tlačením krabíc.
  - `state.pl`: Definuje počiatočný stav (`initial_state/1`) a cieľový stav (`goal/1`), vrátane podmienok na úspešné umiestnenie krabíc na cieľové polia.
  - `utils.pl`: Obsahuje pomocné predikáty, napríklad na výpis riešenia (`print_solution/1`) a detekciu mŕtvych uzlov (`deadlock/1`).
  - `solver.pl`: Implementuje algoritmus BFS (`solve_bfs/0`), ktorý hľadá optimálne riešenie Sokoban mapy.
  - `facts.pl`: Dynamicky generovaný súbor obsahujúci fakty o aktuálnej mape, ako napríklad polohy stien, krabíc, cieľových polí a hráča.
- **Pythonové súbory:**
  - `main.py`: Hlavný spúšťací bod projektu. Umožňuje výber mapy, generovanie faktov pre Prolog, spustenie solvera, spracovanie výsledkov a vizualizáciu riešenia.
  - `prolog_executor.py`: Spúšťa Prologový solver, spracováva jeho výstup a generuje sekvenciu akcií riešenia.
  - `sokoban_parser.py`: Načítava mapy zo súborov a generuje fakty pre Prolog na základe definovaných symbolov (napr. #, C, X, S).
  - `sokoban_renderer.py`: Implementuje vizualizáciu riešenia pomocou knižnice Pygame, vrátane animácie pohybov hráča a krabíc.

Záver

Tento projekt bol zaujímavou skúsenosťou pri modelovaní a riešení Sokobanu ako plánovacieho problému. Kombinácia Prologu a Pythonu umožnila efektívne oddeliť logickú časť od vizuálnej prezentácie, čím sa dosiahlo prehľadné a funkčné riešenie.

Pri práci sme narazili na výzvy spojené s optimalizáciou riešenia pre väčšie a zložitejšie mapy, kde sa BFS ukázal ako menej efektívny. Napriek tomu projekt splnil svoj cieľ a demonštroval schopnosť riešiť logické úlohy.