Analýza systémů založená na modelech

2022/2023

Domácí úloha 3

Pavel Šesták (xsesta07)

Brno, 8. května 2023

Obsah

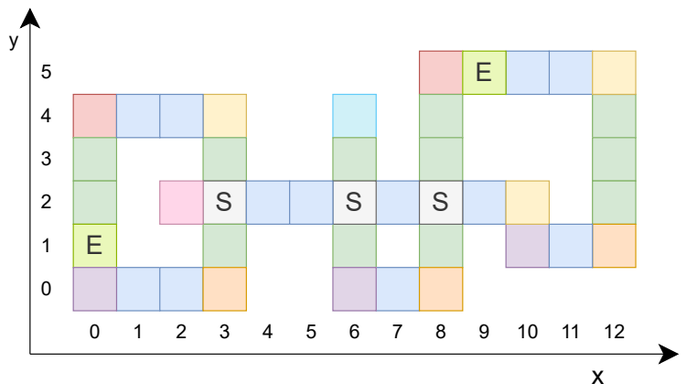
Seznam obrázků

Seznam tabulek

MBA 2022/2023
Projekt č. 3

1 Popis problému

Uvažujte robota, který se pohybuje bludištěm znázorněném na Obr. 1. Na začátku robot je náhodně umístěn do jedné z buněk označených symbolem S a snaží se co nejoptimálněji se dostat do jakéhokoliv výstupu z bludiště označeného symbolem E. Robot nikdy nezná svoji přesnou polohu v bludišti, ale spoléhá na své senzory, podle kterých umí rozlišit, zda je aktuálně schopen jet v jednom ze čtyř směrů: nahoru (*up*), doprava (*right*), dolů (*down*) nebo doleva ($left$). Z toho plyne, že v bludišti existují buňky se stejnou konfigurací stěn (označené stejnou barvou), které nejsou pro robota rozlišitelné, např. buňky $(1, 0)$ a $(2, 0)$ nebo buňky $(3, 0)$ a $(8, 0)$. Oproti tomu buňka $(2, 2)$ má unikátní konfiguraci stěn, takže je pro robota jednoznačně identifikovatelná.



Obrázek 1: Bludiště. Barvy označují skupiny nerozlišitelných buněk.

Pohyb robota probíhá ve dvou střídajících se fázích:

1. (`[steer] clk=1`) kontrolér robota nastaví požadovaný směr *dir* (*up*, *right*, *down* nebo $left$), kterým se robot má vydat.
2. (`[drive] clk=2`) robot se pokusí o jízdu ve směru *dir* a s pravděpodobností 90% uspěje; se zbývajícími 10% robot může uklouznout 'doleva' vůči směru jízdy, tedy
   - byl-li zvolen směr *up*, robot může uklouznout ve směru $left$
   - byl-li zvolen směr $left$, robot může uklouznout ve směru *down*, apod.

Pokud výsledný směr jízdy nelze uskutečnit (např. směr *right* v buňce $(0, 2)$), robot narazí do stěny a zůstane v původní buňce.

2 Kostra programu

Soubor `maze.prism` obsahuje připravenou kostru pro implementaci výše popsaného systému jako DTMC v nástroji PRISM. Kostra obsahuje:

- deklaraci konstant a formulí, pomocí kterých se kóduje stavový prostor bludiště; všimněte si zejména formulí $u/r/d/l$ popisujících aktuální stav senzorů, tedy zda lze v dané buňce uskutečnit jízdu ve směru $up/right/down/left$
- definici modulu `clk` zajišťujícího synchronizaci
- definici modulu `maze` popisujícího náhodné umístění robota do bludiště a také pohyb robota tímto bludištěm včetně náhodného uklouznutí; k volbě směru jízdy se používá proměnná *dir*
- definici modulu `controller` obsahujícího deklaraci proměnné *dir*
- definici reward struktury "`steps`"pro počítaní kroků robota.

3 Zadání

Uvažme následující dvě vlastnosti (specifikaci) chování robota:

1. Pravděpodobnost, že se robot někdy dostane ven z bludiště, je alespoň α (tuto vlastnost označme $\varphi(\alpha)$)
2. Očekávaný počet kroků potřebný k tomu, aby se robot dostal ven z bludiště, je nejvýše β (tuto vlastnost označme $\psi(\beta)$).

(1)

Do připravené kostry doplňte implementaci modulu `controller` pro nastavení proměnné *dir* používané ke řízení robota. Při implementaci kontroléru **nelze přistupovat k hodnotám proměnných** x **a** y – smíte použít pouze formule $u/r/d/l$ (stav senzorů), samostatně definované formule neobsahující proměnné x/y, případně hodnoty samostatně definovaných proměnných (paměť kontroléru, včetně *dir*). Každý příkaz v modulu `controller` musí být synchronizován návěštím `[steer]`. Pro inspiraci je v kostře ukázka dvou jednoduchých kontrolérů. Hodnocení Vašeho řešení závisí na tom, jak dobře Vámi navržený kontrolér splňuje vlastnosti $\varphi(\alpha)$ a $\psi(\beta)$: kontrolér splňující $\varphi(1)$ a $\psi(8.4)$ bude hodnocen maximálním počtem bodů (za předpokladu rozumně zpracované zprávy, viz níže).

(2)

Sepište krátkou (1-2 str.) zprávu obsahující popis Vámi navrženého kontroléru, postup při ověření vlastností φ a ψ, výsledky jednotlivých analýz včetně Vaší interpretace, případně také grafy získané v rámci experimentů. Kvalita zprávy bude zohledněna ve výsledném hodnocení projektu.

4 Odevzdání

Součástí odevzdání je soubor `maze.pdf` se zprávou a soubor `maze.prism` obsahující doplněnou kostru. Soubory odevzdávejte prostřednictvím E-learning systému VUT.

Deadline na odevzdání: 16. května 23:59.

Poznámky k implementaci

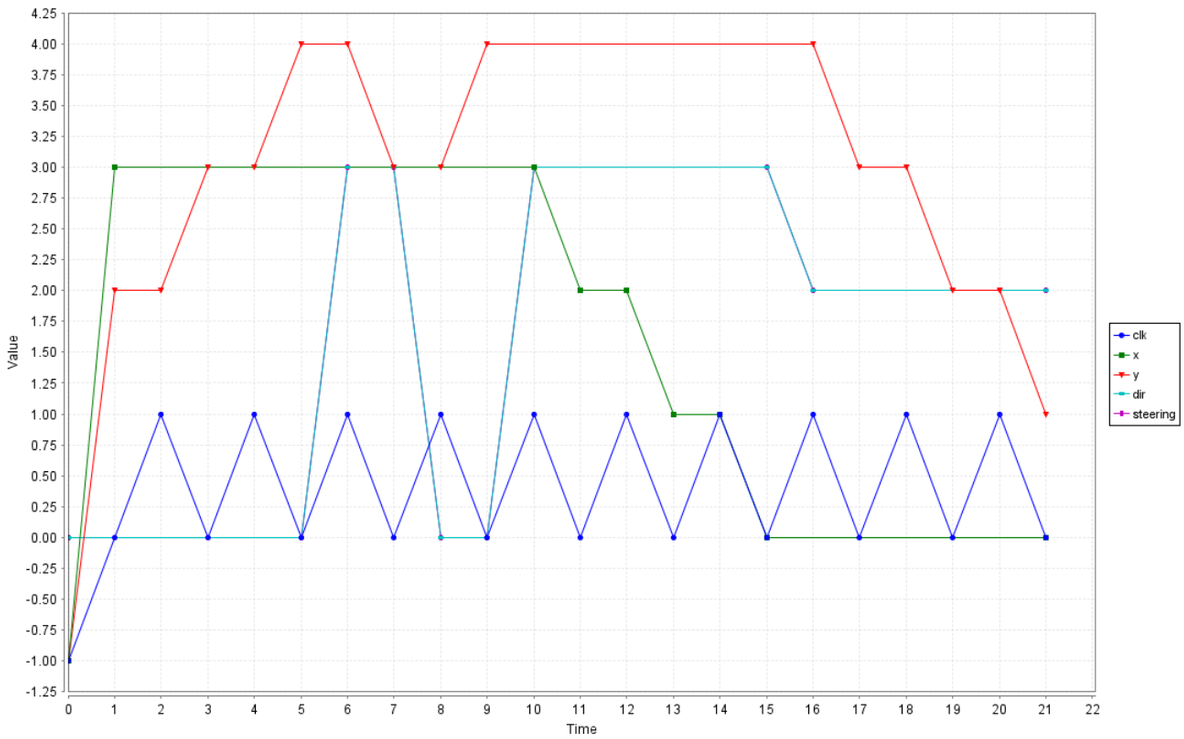
- Pro jednodušší ladění Vašeho kontroléru můžete vyzkoušet deterministické umístění robota do bludiště před začátkem běhu, vizte příkaz na řádku 60.
- Popsaný systém lze modelovat jako částečně pozorovatelný MDP (partially observable MDP, POMDP), o kterých se dozvíte na poslední přednášce. K vypracování projektu však tyto znalosti nejsou zapotřebí.

1 Popis kontroleru

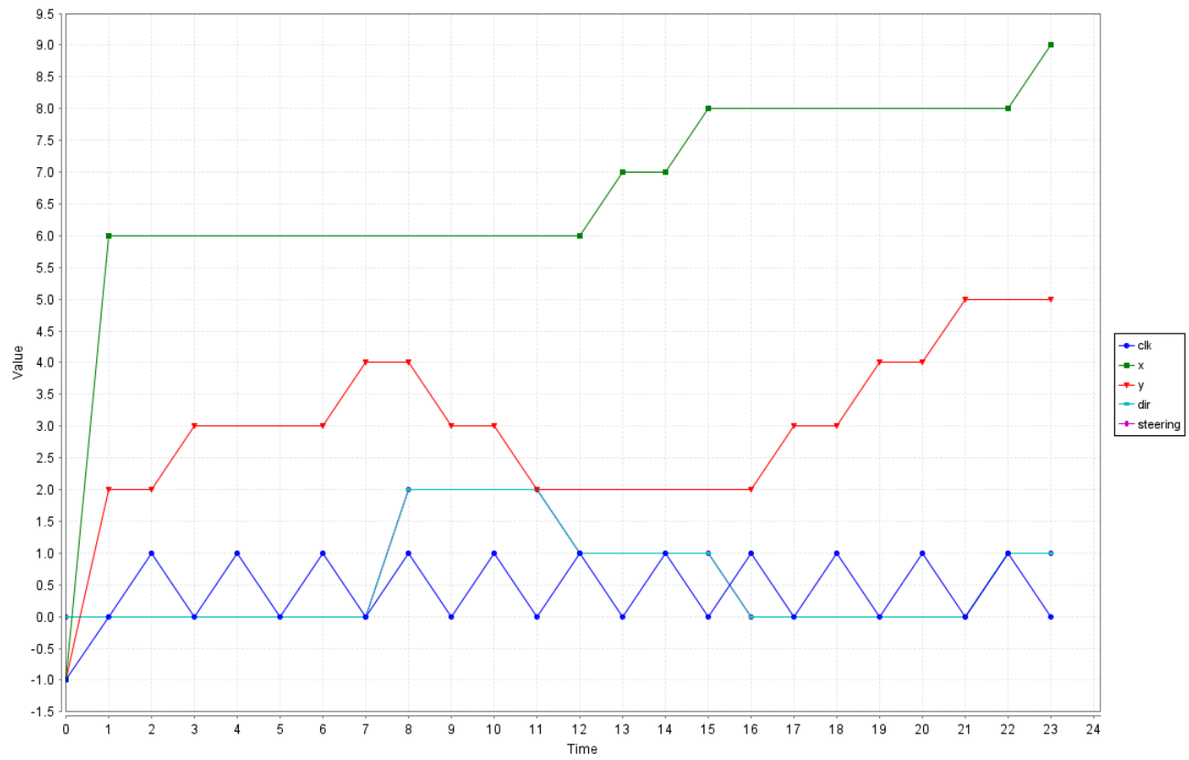
Při návrhu kontroleru, jsem začal u náhodné strategie, kde jsem postupně začal přidávat podmínky na možné tahy. Následně jsem začal tahy optimalizovat pro danou úlohu na základě měření. Pro ladění jsem zkoušel jednotlivé startovací lokace a oddělal jsem náhodné uklouznutí robota. Výsledkem je sada pravidel, která bere v potaz předchozí pohyb a snaží se dostat do cíle co nejrychleji.

2 Analýza kontroleru

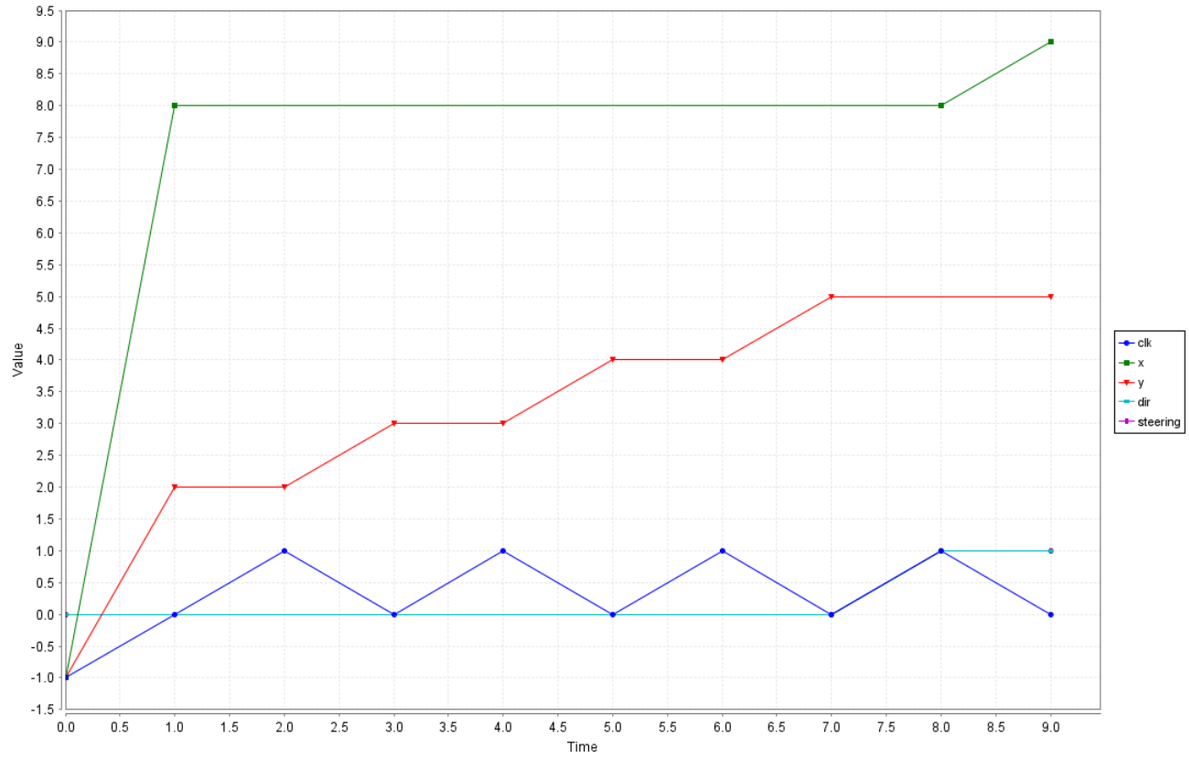
Pro analýzu kontroleru jsem ověřoval dvě vlastnosti, které jsou uvedeny v zadání projektu. První vlastnost pravděpodobnost, že se robot někdy dostane ven z bludiště byla modelována následující formulí: **P =? [F exit]**. Druhá vlastnost se vztahuje k očekávanému počtu kroků k tomu, aby se robot dostal ven z bludiště a byla modelována následující formulí: **R{"steps"}=? [F exit]**. Výsledky daného kontroleru jsou, že se robot dostane vždy z bludiště s průměrně 8.35 kroky. Největší problém dělá simulace z bodu x = 6 a y = 2, jelikož na začátku robot není schopen zjistit kde se v bludišti nachází. Nejlépe mi vycházelo jít ze startu na začátku nahoru, což u tohoto bodu vede bohužel do slepé uličky. Z tohoto důvodu, jakmile se robot nachází na rozcestí a přišel z vrchu tak jde doprava jinak jde nahoru.



Obrázek 1: Simulace z bodu x = 3 a y = 2



Obrázek 2: Simulace z bodu x = 6 a y = 2



Obrázek 3: Simulace z bodu x = 8 a y = 2