VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

INDUKTIVNÍ SYNTÉZA KONEČNĚ STAVOVÝCH KONTROLÉRŮ PRO DECENTRALIZOVANÉ POMDP

INDUCTIVE SYNTHESIS OF FINITE STATE CONTROLLERS FOR DECENTRALIZED POMDPS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. VOJTĚCH HRANIČKA**
AUTHOR

VEDOUCÍ PRÁCE **doc. RNDr. MILAN ČEŠKA, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání diplomové práce

156910

Ústav:	Ústav inteligentních systémů (UITS)
Student:	**Hranička Vojtěch, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Matematické metody
Název:	**Induktivní syntéza konečně stavových kontrolérů pro decentralizované POMDP**
Kategorie:	Formální verifikace
Akademický rok:	2023/24

Zadání:

1. Nastudujte existující metody pro syntézu kontrolérů v částečně pozorovatelných Markovských rozhodovacích procesech (Partially Observable Markov Decision Processes -- POMDPs) a jejich rozšíření na decentralizované multiagentních systémů.
2. Rozšiřte metody induktivní syntézy pro POMDP, aby podporovaly decentralizované POMDP
3. Implementujte tyto rozšíření v rámci nástroje PAYNT.
4. S využitím vhodných případových studií, proveďte detailní experimentální vyhodnocení implementovaných metod včetně porovnání s existujícími metodami pro řešení POMDP.

Literatura:

- Kochenderfer, M.J., Wheeler, T.A., and Wray K.H, Algorithms for Decision Making, MIT Press 2021.
- Oliehoek, Frans A. "Decentralized POMDPs." *Reinforcement Learning*. Springer, Berlin, Heidelberg, 2012. 471-503.
- Andriushchenko, R., Češka, M., Junges, S., and Katoen, J.P. Inductive synthesis of finite-state controllers for POMDPs. In UAI'22. Proceedings of Machine Learning Research.
- Andriushchenko, R., Češka, M., Junges, S., Katoen, J.P. and Stupinský, Š. PAYNT: A Tool for Inductive Synthesis of Probabilistic Programs. In *CAV 2021*. Springer.
- Kumar, A., Mostafa, H., and Zilberstein, S. Dual formulations for optimizing Dec-POMDP controllers. In ICAPS'16. AAAI.
- You, Y., Thomas, V., Colas, F., and Buffet, O. Solving infinite-horizon Dec-POMDPs using Finite State Controllers within JESP. In ICTAI'21. IEEE.

Při obhajobě semestrální části projektu je požadováno:
Body 1, 2 a částečně bod 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Češka Milan, doc. RNDr., Ph.D.**
Konzultant:	Andriushchenko Roman, Ing.
Vedoucí ústavu:	Hanáček Petr, doc. Dr. Ing.
Datum zadání:	1.11.2023
Termín pro odevzdání:	17.5.2024
Datum schválení:	6.11.2023

Abstrakt

Markovovy rozhodovací procesy s částečným pozorováním (POMDP) jsou významným stochastickým modelem pro sekvenční rozhodování s nejistotou. Decentralizované POMDP (Dec-POMDP) tento model rozšiřují o možnost práce s více agenty. Tato práce se zaměřuje na rozšíření metody induktivní syntézy kontrolérů pro POMDP, tak aby podporovala práci s Dec-POMDP. Hlavním cílem syntézy je nalézt takové kontroléry pro každého z agentů, aby jejich společné chování nejlépe splňovalo požadované specifikace. V této práci se zaměřuji na strategie, které jsou reprezentovány pomocí konečně stavových kontrolérů (FSC). Experimentální výsledky ukazují, že použití této metody pro návrh kontrolérů dosahuje srovnatelných výsledků se state-of-the-art přístupy. Navíc tento přístup jako první umožňuje práci v nekonečném horizontu bez použití discount faktoru. Díky tomu je tato metoda vhodnější pro řešení problémů, kde je důležité rozhodnutí provedeno až v pozdějším horizontu.

Abstract

Markov decision processes with partial observation (POMDP) is an important stochastic model for sequential decision making with uncertainty. Decentralized POMDPs (Dec-POMDPs) extend this model to handle multiple agents. This work focuses on extending the inductive controller synthesis method for POMDPs to support work with Dec-POMDPs. The main goal of the synthesis is to find controllers for each agent such that their joint behavior best satisfies the desired specifications. In this paper, I focus on strategies that are represented using finite state controllers (FSC). Experimental results show that using this method for controller design achieves comparable results to state-of-the-art approaches. Moreover, this approach is the first to allow working in infinite horizon without using a discount factor. This makes this method more suitable for solving problems where an important decision is made at a later point in time.

Klíčová slova

Markovovy modely, automatizovaná syntéza, decentralizace, pravděpodobnostní modely, model checking, částečné pozorování, formální metody

Keywords

Markov models, automated synthesis, decentralization, probabilistic models, model checking, partial observability, formal methods

Citace

HRANIČKA, Vojtěch. *Induktivní syntéza konečně stavových kontrolérů pro decentralizované POMDP*. Brno, 2024. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Milan Češka, Ph.D.

Induktivní syntéza konečně stavových kontrolérů pro decentralizované POMDP

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením a RNDr. Milana Češky, Ph.D. Další informace mi poskytl Roman Andriushchenko. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Vojtěch Hranička
15. května 2024

Poděkování

Chtěl bych poděkovat vedoucímu práce Milanovi Češkovi za odborné rady a vedení práce. Dále bych chtěl poděkovat Romanovi Andriushchenkovi za ochotné zodpovídání mých dotazů ohledně nástroje PAYNT.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Mnoho problémů reálného světa zahrnuje spolupráci více agentů bez možnosti vzájemné komunikace. Mezi tyto problémy patří například koordinace planetárních roverů [7], sledování cíle pomocí skupiny senzorů [28] a nebo optimalizace propustnosti v bezdrátových sítích [32]. Všechny tyto problémy vyžadují vytvoření strategií pro jednotlivé agenty tak, aby jejich společné chování vedlo k dosažení určitého cíle. Současně se v rámci těchto problémů pracuje s nejistotou a náhodností. Ty jsou zde obsaženy pomocí náhodného chování a stavové nejistoty.

Markovovy rozhodovací procesy (MDP) [35] pracují s faktorem náhodnosti a jsou jedním z nejpoužívanějších stochastických modelů. MDP jsou ale plně pozorovatelné a z tohoto důvodu neřeší problém stavové neurčitosti. Markovovy rozhodovací procesy s částečným pozorováním [18] jsou rozšířením MDP které umožňuje přidat do modelů faktor nejistoty. Agent zde nemá úplné informace o tom, ve kterém stavu se nachází a musí se rozhodovat pouze na základě jeho pozorování. Decentralizované markovovy rozhodovací procesy s částečným pozorováním (Dec-POMDP) [8] reprezentují sekvenční rozhodovací problémy pro více agentů. Cílem řešení Dec-POMDP problémů je nalézt taková pravidla chování pro všechny agenty, aby jejich decentralizované provedení optimalizovalo jejich chování vůči požadované specifikaci. Každý z agentů se může při rozhodování o dalších akcích spoléhat pouze na svou vlastní historii. Tedy zejména na sekvenci jeho provedených akcí a pozorování.

Hlavním důvodem obtížnosti řešení Dec-POMDP problémů je fakt, že stav daného systému se vyvíjí v závislosti na chování všech agentů. Optimální chování každého agenta je tedy závislé na možné historii chování všech agentů a také na jejich budoucích strategiích. Současně ale akce provedené každým agentem musí být založeny pouze na jeho individuální historii. Z toho vyplývá, že kontrolér pro každého z agentů musí při rozhodování v každém stavu pracovat s tím, že neví které akce zvolili ostatní agenti. Z tohoto důvodu se naproti řešení POMDP problémů výrazně zvyšuje neurčitost toho, ve kterém ze stavů se daný systém nachází. Například belief-based metody [9], které jsou jedním z nejmodernějších přístupů pro analýzu POMDP z tohoto důvodu nejsou příliš efektivní pro práci s Dec-POMDP. Tyto metody jsou založené na vytvoření belief MDP, což je MDP zahrnující pravděpodobnostní rozdělení stavů se stejným pozorováním. Tento model je pak možné analyzovat jako klasické MDP. Z výše zmíněného důvodu tyto metody při práci s Dec-POMDP narážejí na problém nedostatku kompaktnosti *belief state* reprezentace.

Řešení takovýchto problémů v konečném horizontu je NEXP složité [8] a to i pouze pro 2 agenty. Existuje mnoho přístupů k řešení Dec-POMDP problému v konečném horizontu [13, 31, 23]. Tyto přístupy většinou hledají optimální kontrolér ve formě rozhodovacího stromu. Tyto metody jsou často založeny na transformaci Dec-POMDP problému na deterministický

problém nalezení nejkratší cesty jako $Multi - AgentA^*$. Ty dokáží získat dobré výsledky, naráží ale na problém exponenciálního růstu rozhodovacího stromu. To způsobuje vysokou složitost výpočtu pro při zvýšení stavů nebo počtu akcí. Existují ale i přístupy například $RS - MAA^*$ [26] které dopady tohoto problému omezují pomocí kombinace $Small - step - MAA^*$ využívající speciální rozhodovací strom převádějící exponenciální růst uzlů z šířky do výšky daného stromu a využití přípustné rekurzivní heuristiky.

V této práci se zaměřuji na problém Dec-POMDP v nekonečném horizontu. Tento problém je nerozhodnutelný již pro problémy s jedním agentem, tedy pro POMDP [26]. Existují ale metody, které pro daný problém aproximují řešení. Výstupem těchto metod bývá skupina konečných kontrolérů (FSC) [2], kde každé FSC reprezentuje strategii jednoho z agentů. Jedná se o malé konečné automaty, které v sobě mají zakódované strategie pro Dec-POMDP. Použití FSC navíc umožňuje řešit problém Dec-POMDP v nekonečném horizontu. Mimo jiné také využití strategií na základě FSC vyžaduje pouze jednoduchou *look-up* tabulku. Možnost takto efektivního využití je žádoucí v mnoha odvětvích, například u mobilních zařízení s omezenou kapacitou baterie [15] nebo u bezdrátových zařízení [32]. Navíc FSC poskytují výrazně lepší sémantické informace o dané strategii než *point-based* metody což umožňuje intuitivnější pochopení daných strategií lidmi [13].

První skupinou metod tohoto typu jsou přístupy založené na optimalizaci FSC všech agentů současně. Příklady takovýchto přístupů jsou například využití duálního smíšeného celočíselného lineárního programování (MILP) [21] nebo použití expectation maximization algoritmu [22] k optimalizaci periodických FSC [33]. Druhým způsobem je optimalizace jednotlivých FSC iterativně. Příkladem může být využití JESP (*Joint Equilibrium-Based Search for Policies*) [27]. Strategie se zde hledá pro každého agenta zvlášť, přičemž se znají aktuální strategie ostatních agentů. Díky fixaci strategií ostatních agentů při optimalizaci strategie jednoho agenta lze daný problém řešit jako rozšířený POMDP. Iterací optimalizace strategií pro každého z agentů se dochází až do stavu, kde není možné žádnou strategii zlepšit. Převedením Dec-POMDP problému na problém jednoho agenta se výrazně snižuje složitost výpočtu, současně ale tato metoda může nalézt pouze lokální optima. Všechna state-of-the-art řešení Dec-POMDP problému pracují s discount faktorem. To znamená, že akce provedené dříve mají větší význam něž akce provedené později. To umožňuje efektivnější práci, jelikož se v těchto metodách ví že po určitém počtu kroků nemají zvolené akce téměř žádný vliv na vlastnosti dané strategie. Z tohoto důvodu nejsou vhodné pro modely kdy je důležité rozhodnutí provedeno v pozdější fázi daného problému.

V této práci se zabývám řešením Dec-POMDP problémů pomocí induktivní syntézy skupiny FSC. Ta hledá vhodné FSC pomocí iterativní analýzy stále větších rodin FSC. Konkrétně jsem navrhl rozšíření induktivní syntézy POMDP [5], tak aby podporovala také řešení Dec-POMDP. Klíčovými objekty modifikace dané metody provedené v rámci této práce jsou úprava Quotient MDP a obarvení Quotient MDP, tak aby bylo možné pracovat i s Dec-POMDP. Quotient MDP je MDP nad-aproximující chování daného Dec-POMDP při použití FSC. Obarvení Quotient MDP je proces umožňující hledání konzistentních strategií v získaném MDP. Konzistence strategie v tomto případě zajišťuje pro dané kombinace pozorování a stavu paměti stejný výběr akce a to bez ovlivnění stavu ostatních agentů. Konkrétně ověřuje schopnost FSC reprezentovat danou strategii a současně nezávislost rozhodnutí agenta na historii ostatních agentů. Díky tomu je možné získané strategie aplikovat decentralizovaně pro každého z agentů. Toto rozšíření implementuji v rámci nástroje PAYNT [6]. Tato metoda jako jediná dokáže pracovat s modely v nekonečném horizontu bez discount faktorizace. Při porovnání se state-of-the-art přístupy bylo nutné přizpůsobit modely tak aby reprezentovaly chování s discount faktorem. Navzdory skutečnosti, že ostatní

přístupy jsou navrženy přímo pro práci s discount faktorem, nově navržená metoda dosahovala srovnatelných výsledků u problémů, kde lze kvalitní strategii reprezentovat pomocí malých kontrolérů. Dále v rámci práce byly navrženy úpravy původní metody, tak aby lépe pracovala s vlastnostmi Dec-POMDP a umožnila nalezení větších kontrolérů. Tyto úpravy neměly významný vliv na kvalitu nalezených strategií.

Struktura práce

Kapitola 2 obsahuje teorii potřebnou pro pochopení této práce. Kapitola 3 popisuje existující přístupy pro řešení Dec-POMDP problémů. Současně je zde popsán přístup pro řešení POMDP, ze kterého tato práce vychází. Kapitola 4 popisuje rozšíření metody induktivní syntézy pro POMDP, aby podporovala Dec-POMDP. V kapitole 5 je popsáno experimentální vyhodnocení implementovaného rozšíření dané metody. Závěrečná kapitola 6 obsahuje souhrn této práce a potenciální směry pro budoucí rozšíření.

Kapitola 2

Základní definice

Tato kapitola představuje teorii potřebnou k pochopení následujících částí této práce. Nejdříve se zabývá popisem stochastických modelu jako jsou Markovovy řetězce a Markovovy rozhodovací procesy. Poté jsou zde popsány modely POMDP, které umožňují přidání faktoru nejistoty. V této sekci se také zabývám různými způsoby reprezentace strategií pro POMDP. Následně se věnuji Dec-POMDP, které jsou rozšířeny o možnost modelování chování více decentralizovaných agentů současně. Závěr této kapitoly se věnuje specifikaci vlastností Markovových řetězců. Základní teorii a definice k první části 2.1 jsem čerpal z [11, 35, 12]. Teorii pro sekce 2.2 a 2.3 jsem čerpal z [18, 29, 38]. Pro sekci 2.5 popisující specifikaci MC jsem čerpal z [25, 36].

2.1 Markovovy řetězce a Markovovy rozhodovací procesy

V této práci se pracuje s popisem stochastických procesů. Nejzákladnějším typem stochastických modelů ze kterých budeme následně vycházet jsou Markovovy řetězce (MC). Existují dva druhy Markovových řetězců. Diskrétní Markovovy řetězce, které pracují s diskrétním časem a druhým typem jsou Markovovy řetězce se spojitým časem. V této práci se věnuji pouze modelům pracujícím s diskrétním časem a tedy i nadále se budeme zabývat pouze diskrétními Markovovými řetězci.

Definice 2.1.1 ***(Pravděpodobnostní rozložení).*** *Pravděpodobnostní rozložení nad konečnou množinou A je funkce $\mu : A \to [0, 1]$, kde platí: $\sum_{a \in A} \mu(a) = 1$. Nechť $Distr(A)$ značí množinu pravděpodobnostních rozložení nad A. Nechť $supp(\mu) = \{s \in S | \mu(s) > 0\}$.*

Definice 2.1.2 ***(DTMC).*** *Diskrétní Markovův řetězec D je trojice (S, s_{init}, P), kde S značí množinu všech stavů daného řetězce, s_{init} je počátečním stavem řetězce, pro který platí $s_{init} \in S$ a $P = S \times S \to [0, 1]$ značí množinu pravidel pro přechod mezi stavy. Zápis $P(s, s')$ popisuje pravděpodobnost přechodu ze stavu s do stavu s'. Pro každé $s \in S$ platí $\sum_{s' \in S} P(s, s') = 1$. Množinu všech rozdělení pravděpodobnosti nad množinou S budeme dále značit $Distr(S)$.*

Cestou π DTMC se nazývá posloupnost stavů DTMC, pro které platí $P(s, s') > 0$. Pravděpodobnost cesty π se vypočítá pomocí součinu pravděpodobností všech přechodů mezi stavy dané cesty. Tedy podle vzorce$P(\pi) = \prod_{i>0} P(s_{i-1}, s_i)$.

Definice 2.1.3 ***(Markovova vlastnost).*** *Nechť $\mathcal{D}$ je DTMC. Nechť $X(k) \in S$ je náhodnou proměnou popisující současný stav $\mathcal{D}$ v diskrétním čase $k \leq 0$. Markovova vlastnost je*

definována jako $\mathbb{P}(X(k) = s_k | X(k-1) = s_{k-1}, ..., X(0) = s_0) = \mathbb{P}(X(k) = s_k | X(k-1) = sk-1)$.

Markovova vlastnost zaručuje, že budoucí chování po každém stavu je nezávislé na předchůdcích tohoto stavu. Tedy že historie procesu před vstupem do daného stavu neovlivní následující chování pro tento stav.

Markovovy rozhodovací procesy jsou dalším rozšířením Markovových řetězců umožňující nedeterministické volby následující akce. Na rozdíl od DTMC, které každému stavu přiřazují konkrétní pravděpodobnostní rozložení přechodů do následujících stavů, Markovovy rozhodovací procesy umožňují každému stavu nedeterministickou volbu z více pravděpodobnostních rozložení pro dané přechody.

Definice 2.1.4 ***(MDP).*** *Markovův rozhodovací proces je uspořádaná čtveřice* $M = (S, s_{init}, Act, \mathcal{P})$*, kde* S *značí množinu všech stavů daného řetězce,* s_{init} *je počátečním stavem řetězce, pro který platí* $s_{init} \in S$*,* Act *je konečná množina akcí a* $\mathcal{P} : S \times Act \nrightarrow Distr(S)$*. Pro všechny možné akce stavu* $s \in S$*, jsou označeny* $Act(s)$*. Platí že* $Act(s) = \{a \in Act | \mathcal{P}(s, a) \neq \perp\}$*.*

Nadále rozložení pravděpodobnosti přechodů do ostatních stavů při použití akce $a \in Act$ ve stavu $s \in S$ budeme značit $\mathcal{P}(s)(a)$. Obdobně pravděpodobnost přechodu do konkrétního stavu $s' \in S$ budeme označovat $\mathcal{P}(s)(a)(s')$ nebo zkráceně $\mathcal{P}(s, a, s')$. Můžeme si všimnout, že DTMC je pouze speciálním typem MDP, kde pro každý stav $s \in S$ platí $|Act(s)| = 1$. V takovém případě každému stavu náleží pouze jedna akce a tedy je i zbaven nedeterministické volby. Cesta π MDP je neprázdná sekvence stavů a akcí $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n$, kde platí $s_i \in S, a_i \in Act(s_i)$ *a* $\forall i \in N_0 | \mathcal{P}(s_i, a_i, s_{i+1}) > 0$. Nechť $Paths^M(s)_{fin}$ je množina všech konečných cest ze stavu s a $Paths^M_{inf}(s)$ je množina všech nekonečných cest z daného stavu pro MDP M. Dále $Paths^M(s)$ označuje průnik těchto množin tedy $Paths^M(s) = Paths^M(s)_{inf} \bigcup Paths^M(s)_{fin}$. Nechť je cesta $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n$, poslední stav cesty π dále budeme značit $last(\pi)$. Pravděpodobnost průchodu určité cesty π lze vypočítat pomocí $P(\pi) = \prod_{i>0} \mathcal{P}(s_{i-1}, a, s_i)$.

Pro vyřešení problému nedeterminismu v MDP se používá strategie (sheduler). Strategie σ je funkce určující při příchodu do každého stavu akci $a \in Act(s)$, která bude použita. Díky tomu se aplikací strategie na MDP zbavujeme nedeterminismu. Strategie mohou brát v úvahu i historiji průchodu daného MDP.

Definice 2.1.5 ***(Strategie).*** *Strategie MDP* $M = (S, s_0, Sct, \mathcal{P})$ *je funkcí* $\sigma : Paths^M_{fin} \to Act$ *ve které platí* $\sigma(\pi) \in Act(last(\pi))$ *pro všechny* $\pi \in Paths^M_{fin}$*. Bezpaměťovou strategií nazýváme strategie, pro které platí* $last(\pi) = last(\pi') \implies \sigma(\pi) = \sigma(\pi')$ *pro všechny* $\pi, \pi' \in Paths^M_{fin}$*. Pro bezpaměťové strategie se dále bude používat značení* $\sigma(last(\pi))$ *namísto značení* $\sigma(\pi)$*.*

Strategie μ je deterministická pokud pro všechny cesty $\pi \in Paths^M$ platí $|supp(\mu(\pi))| = 1$. To znamená že strategie μ pro každý stav $last(\pi)$ pouze jednu akci namísto pravděpodobnostního rozložení více akcí. V této práci se dále budu zabívat pouze deterministickými strategiemi.

Definice 2.1.6 ***(Indukce DTMC strategií).*** *Pro MDP* $M = (S, s_{init}, Act, \mathcal{P})$ *je strategie* $\sigma \in \sum^M$ *indukující DTMC* $M^\sigma = (Paths^M_{fin}, s_{init}, \mathbf{P}^\sigma)$*, právě když* $\mathbf{P}^\sigma(\pi, \pi \xrightarrow{\sigma(\pi)} s') = \mathcal{P}(last(\pi), \sigma(\pi), s')$*. Pro ostatní případy platí* $\mathbf{P}^\sigma(\cdot, \cdot) = 0$*.*

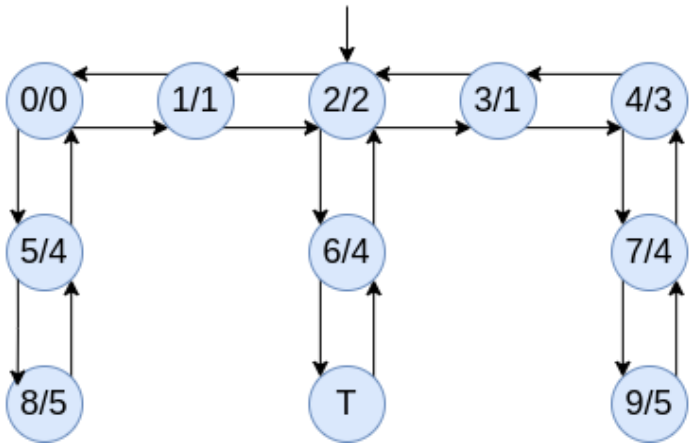
2.2 MDP s částečným pozorováním

Markovovy rozhodovací procesy s částečným pozorováním (POMDP) jsou dalším rozšířením MDP. POMDP umožňuje přidat do modelů faktor nejistoty. Agent zde nemá úplné informace o tom, ve kterém stavu se nachází, a musí se rozhodovat pouze na základě svých pozorování.

Definice 2.2.1 ***(POMDP).*** *POMDP je trojice $\mathcal{M} = (M, Z, O)$ kde $M = (S, s_0, Act, P)$ je základním MDP pro $\mathcal{M}$. Z je konečná množina pozorování a O je funkcí $O : S \to Z$. Pokud je vždy pouze jeden stav $s \in S$ pro který platí $O(s) = z$ pro $z \in Z$ pak je množina Z nazvána triviální.*

Pokud dva stavy mají přiřazené stejné pozorovaní, tak pro ně také platí, že mají umožněné použít stejné akce. Tedy platí $\forall s, s' \in S : O(s) = O(s') \Rightarrow Act(s) = Act(s')$. Funkci pozorování lze také aplikovat i na cesty. Aplikací funkce pozorování O na cestu $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n \in Paths^M(s)_{fin}$ bychom získali cestu $O(\pi) = O(s_0) \xrightarrow{a_0} O(s_1) \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} O(s_n)$. Pokud pro dvě cesty π_1 a π_2 platí $O(\pi_1) = O(\pi_2)$, tak jsou ekvivalentní vůči pozorování a z pohledu *decision makingu* jsou nerozeznatelné. Jako příklad POMDP zde uvedu zjednodušenou verzi maze problému [16] převzatou z[5].

Příklad 2.2.2 *Mějme POMDP $\mathcal{M}$ kde $S = \{s_0, ..., s_9, s_\mathcal{T}\}$, $Act = \{u, d, l, r\}$ a $Z = \{z_0, ..., z_5\}$ kde agent prochází modelem a snaží se dosáhnout stavu $s_\mathcal{T}$. Počáteční stav je stav s_2 s pozorováním z_2. Tento POMDP je popsán na obrázku 2.1, kde jsou popsány P a O. Každý stav s_x je zde označen x/y, kde x značí index daného stavu a y značí index daného pozorování. Tedy platí že $O(s_x) = z_y$. Akce příslušící daným přechodům mezi stavy jsou určeny směrem daného přechodu. Tedy například $\leftarrow$ přísluší akce l a $\rightarrow$ přísluší akce r. Úspěšnost každého přechodu má pravděpodobnost 0.9. Zbylá pravděpodobnost 0.1 se rovnoměrně rozdělí mezi všechny zbývající směry, kterými se agent v daném stavu může pohybovat. Tedy pokud by agent ve stavu s_2 provedl akci d, tak se s pravděpodobností 0.9 přesune do stavu s_6 a s pravděpodobností 0.05 se přesune do stavů s_1 a s_3.*



Obrázek 2.1: Graf POMDP pro zjednodušení maze problém

Definice 2.2.3 ***(Strategie založená na pozorování).*** *Strategie založená na pozorování ρ pro POMDP $\mathcal{M}$ je strategií pro základní MDP M takové že platí:*

$\forall \omega, \omega' \in Paths^M_{fin} | \rho(\omega) = \rho(\omega')$ *kde* $O(\omega) = O(\omega')$. $\sum^{\mathcal{M}}$ *je množina všech strategií založených na pozorování pro POMDP* $\mathcal{M}$.

Strategie založené na pozorování určují pro každý stav $last(\pi)$ akci, která se má v daný moment použít. O tom jaká akce má být vybrána rozhodují na základě daného pozorování stavu a historie použití akcí v dané cestě. Aplikováním strategie založené na pozorování ρ na POMDP $\mathcal{M}$ se získá indukovaný DTMC M^ρ.

Strategie založené na pozorování lze reprezentovat pomocí Konečně stavových automatů. Jedná se o kompaktní způsob reprezentace založený na Mealyho automatech, který je navíc jednoduchý na použití.

Definice 2.2.4 ***(FSC).*** *Konečně stavový kontrolér (FSC) pro POMDP* $\mathcal{M}$ *je čtveřicí* $\mathcal{F} = (N, n_0, \gamma, \delta)$, *kde* N *je konečnou množinou stavů paměti,* $n_0 \in N$ *je počátečním stavem paměti,* γ *je funkcí* $\gamma : N \times Z \to Act$ *přiřazující akce na základě stavu paměti a pozorování. A* δ *je funkce* $\delta : N \times Z \to N$ *určující následující stav paměti na základě aktuálního stavu paměti a pozorování. Na základě velikosti paměti* $|N| = k$ *nazýváme dané FSC konkrétně k-FSC. Strategie odvozené od FSC* $\mathcal{F}$ *budeme dále označovat* $\rho_{\mathcal{F}}$.

Definice 2.2.5 ***(Rodina FSC).*** *Rodina k-FSC je trojice* $\mathcal{F}_k = (N, n_0, K)$, *kde N konečnou množinou stavů paměti,* n_0 *je počátečním stavem paměti a* $K = N \times Z$ *konečnou množinou parametrů v doméně* $V_{n,z} \subseteq Act \times N$.

Dosazením určitých hodnot parametrům K a tedy určením akcí $\gamma(n, z)$ a následujících stavů paměti $\delta(n, z)$ pro jednotlivé přechody mezi stavy lze z rodiny FSC získat jeden k-FSC. Každá rodina FSC tedy popisuje množinu FSC definovanou kombinací různých dosazení daných parametrů. Pro označení takovéto množiny k-FSC budeme používat značení $\mathcal{F}_k$. Z POMDP $\mathcal{M}$ a rodiny $\mathcal{F}_k$ lze indukovat rodinu Markovových řetězců $\mathcal{M}^{\mathcal{F}_k} = \{\mathcal{M}^F | F \in \mathcal{F}_k\}$.

Druhým používaným způsobem specifikace strategií pro POMDP je použití *belief-based* metod. Tyto metody využívají *belief*, což je pravděpodobností distribuce nad stavy která určuje konkrétní pravděpodobnost, že se agent nachází v určitém stavu. Tento *belief* je určený na základě historie pozorování a historie použití akcí. Nechť $\mathcal{B} = \cup_{z \in Z} Distr(s_z)$ je množina všech možných belief stavů POMDP $\mathcal{M} = (M, Z, O)$, kde množina $s_z = \{s \in S | O(s) = z\}$ značí množinu všech stavů daného POMDP které mají stejné pozorování $z \in Z$.

Dále $\mathbf{P}(s, \alpha, z) = \sum_{s' \in S} [O(s') = z] * \mathbf{P}(s, \alpha, s')$ značí pravděpodobnost přesunu do stavu s pozorováním z ze stavu s pomocí akce α. $\mathbf{P}(\mathbf{b}, \alpha, z) = \sum_{s \in S} * \mathbf{P}(s, \alpha, z)$ značí pravděpodobnost pozorování z po provedeni akce α v belief stavu $\mathbf{b}$. Následně definujme belief stav získaný provedením akce α v belief stavu $\mathbf{b}$ za podmínky získaného pozorování z:

$$\langle \mathbf{b} | \alpha, z \rangle (s') = \frac{[O(s') = z] * \sum_{s \in S} \mathbf{b}(s) * \mathbf{P}(s, \alpha, s')}{\mathbf{P}(\mathbf{b}, \alpha, z)}$$

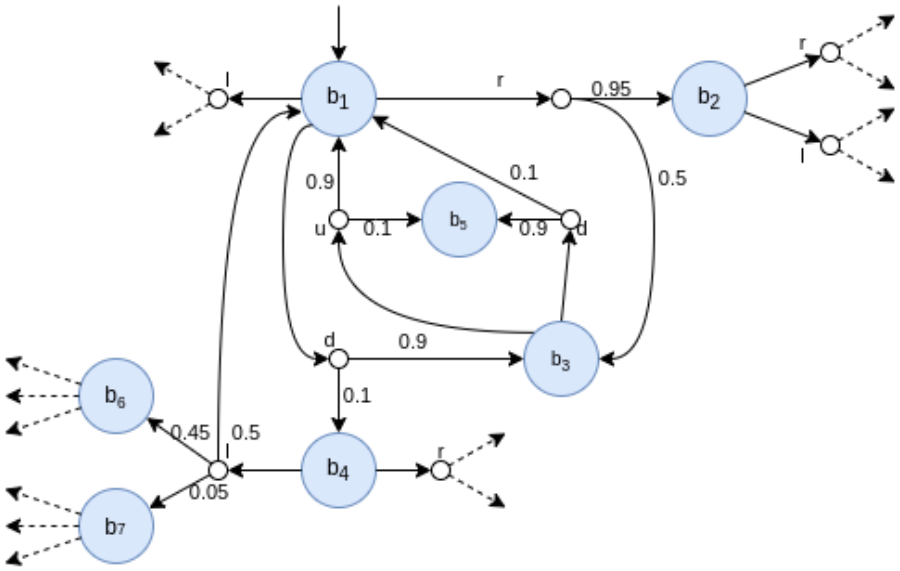
Belief based metody konkrétně využívají tvorbu *belief* MDP. Jedná se o plně pozorovatelné MDP tvořeno *belief* stavy. Toto MDP slouží k popisu chování daného POMDP. Každý ze stavů *belief* MDP definuje pro každý ze stavů POMDP s daným pozorováním pravděpodobnost, že se POMDP nachází právě v daném stavu.

Definice 2.2.6 ***(Belief MDP).*** *Belief MDP pro POMDP* $\mathcal{M} = (M, Z, O)$ *je MDP* $bel(\mathcal{M}) = (\mathcal{B}, Act, \boldsymbol{P}^{\mathcal{B}}, \boldsymbol{b}_I)$, *kde* $\mathcal{B}$ *je množina belief které reprezentují stavy daného MDP,* $\boldsymbol{b}_I$ *je počáteční belief stav* $\boldsymbol{b}_I = \{s_I \to 1\}$ *a přechodová funkce* $\boldsymbol{P}^{\mathcal{B}}$:

$$\boldsymbol{P}^{\mathcal{B}}(\boldsymbol{b}, \alpha, \boldsymbol{b'}) = \begin{cases} \boldsymbol{P}(\boldsymbol{b}, \alpha, O(\boldsymbol{b'})) & pokud\ \boldsymbol{b'} = \langle \boldsymbol{b} | \alpha, O(\boldsymbol{b'}) \rangle \\ 0 & jinak \end{cases}$$

Toto belief MDP popisuje chování daného POMDP a lze jej použít pro získání strategie pro dané POMDP. Takto získaná strategie lze získat pouze pomocí standardních metod pro MDP a tedy se při jejím získávání dále nemusí pracovat s částečným pozorováním.

Příklad 2.2.7 *Obrázek 2.2 představuje část belief MDP problému maze z příkladu 2.2.2. Začíná se v belief stavu b_1 reprezentujícím To že daný POMDP je v počátečním stavu s_2. Po provedení akce d s pravděpodobností 0,9 agent získá pozorování, že se nachází ve stavu s_6 a s pravděpodobností 0,1 získá pozorování, že se nachází ve stavech s_1, nebo s_3. V tomto případě se belief vypočte pomocí normalizace pravděpodobnosti přechodu. Belief b_4 určuje že POMDP je ve stavu s_1 s pravděpodobností $\frac{0,05}{0,1} = \frac{1}{2}$ a ve stavu s_3 obdobně s pravděpodobností $\frac{0,05}{0,1} = \frac{1}{2}$. Pokud by v tomto belief stavu b_4 byla provedena akce l, tak se přejde do stavu s_2 s pravděpodobností $b_4(s_1) * P(s_1, l, s_0) + b_4(s_3) * P(s_3, l, s_0) = \frac{1}{2} * 0,1 + \frac{1}{2} * 0,9 = 0,5$.*

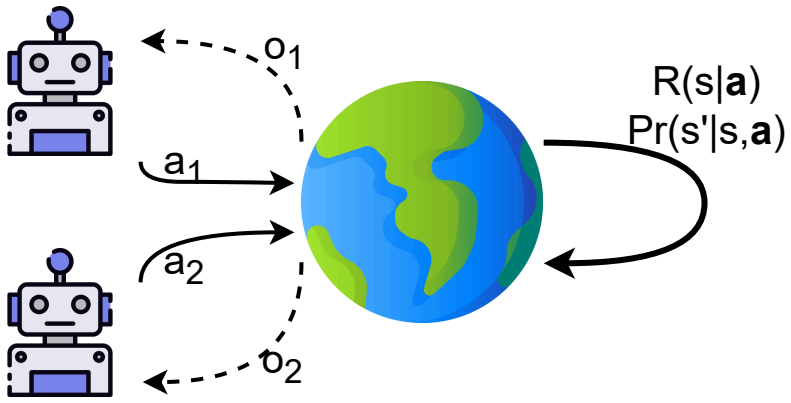


Obrázek 2.2: Část belief MDP pro problém *maze* popsaný v příkladu 2.2.2. Belief stavy odpovídají $b_1 : \{s_2 \to 1\}$, $b_2 : \{s_3 \to \frac{18}{19}, s_1 \to \frac{1}{19}\}$, $b_3 : \{s_6 \to 1\}$, $b_4 : \{s_3 \to \frac{1}{2}, s_1 \to \frac{1}{2}\}$, $b_5 : \{s_T \to 1\}$, $b_6 : \{s_0 \to 1\}$, $b_7 : \{s_4 \to 1\}$

2.3 Decentralizované MDP s částečným pozorováním

Decentralizované markovovy rozhodovací procesy s částečným pozorováním (dále pouze Dec-POMDP) jsou dalším rozšířením MDP. Podobně jako POMDP umožňují do modelů přidání faktoru nejistoty pomocí pozorování. Dec-POMDP ale navíc umožňují pracovat s více agenty současně. Jedná se tedy o modelování spolupráce více agentů se stejným společným cílem. Každý z agentů provádí svou akci nezávisle na ostatních, na základě

svého pozorování. Na základě stavu daného Dec-POMDP a akce všech agentů provede následující přechod. Vztah tohoto decentralizovaného chování jednotlivých agentů s Dec-POMDP modelem je ilustrován na obrázku 2.3.



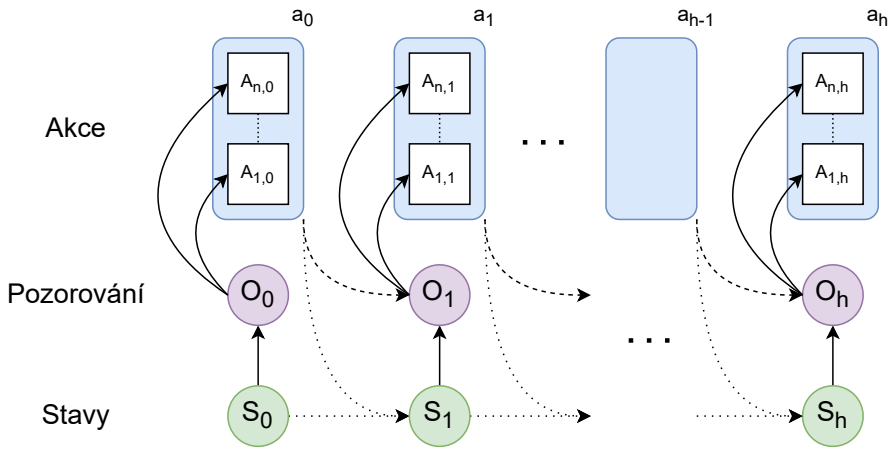
Obrázek 2.3: Ilustrace provedení akcí v Dec-POMDP. Každý z agentů vidí pouze své dílčí pozorování o_1 nebo o_2. Na základě těchto pozorování provádí dílčí akci a_1 nebo a_2. Tyto akce společně tvoří sdruženou akci $\mathbf{a}$ na základě které je model převeden do následujícího stavu. Tento obrázek byl adaptován z [30]. Použité obrázky jsou převzaty z *www.freepik.com*

Definice 2.3.1 ***(Dec-POMDP).*** *Decentralizovaný markovovův rozhodovací proces s částečným pozorováním (Dec-POMDP) je osmice* $(\mathcal{D}, S, \boldsymbol{A}, \mathcal{T}, R, \boldsymbol{O}, O, I)$*, kde* $\mathcal{D} = \{1, ..., n\}$ *je množina n agentů, S je konečná množina stavů s prostředí daného Dec-POMDP,* $\boldsymbol{A}$ *je konečná množina společných akcí,* $\mathcal{T}$ *je funkce pravděpodobnostích přechodů, R je reward funkce,* $\boldsymbol{O}$ *je konečná množina společných pozorování, O je funkce pravděpodobnosti pozorování a* $I \in \mathcal{P}(\mathcal{S})$ *je počáteční stav distribuce ve fázi* $t = 0$*.*

Dec-POMDP rozšiřuje POMDP s jedním agentem o možnost společných akcí a společných pozorování. Množina $\mathbf{A} = \times_{i \in \mathcal{D}} A^i$ je množina společných akcí, kde A^i je množina akcí dostupných pro agenta i. Tyto akce mohou být pro různé agenty rozdílné. Při každém kroku se zvolí jedna společná akce $\mathbf{a} = \langle a^1, ..., a^n \rangle$. Funkce $\mathcal{T}$ určuje jak volba dané společné akce ovlivní prostředí daného Dec-POMDP. Tato funkce specifikuje pravděpodobnost přechodů do následujících stavů při použití konkrétních společných akcí $Pr(s'|s, \mathbf{a})$. V Dec-POMDP agenti sledují pouze své akce a tedy neví které akce provádí ostatní agenti.

Obdobně jako u množiny společných akcí, tak i $\mathbf{O} = \times_{i \in \mathcal{D}} O^i$ je množina společných pozorování, kde O^i je množina pozorování agenta i. Při každém kroku se zvolí jedno společné pozorování $\mathbf{o} = \langle o^1, ..., o^n \rangle$, ze kterého každý z agentů získá informaci pouze o svém pozorování o^i. Funkce pravděpodobnosti pozorování O určuje pravděpodobnost pozorování na základě zvolené společné akce a stavu prostředí daného Dec-POMDP.

Každý z agentů se o volbě své následující akce rozhoduje pouze na základě svého pozorování a není tedy umožněna žádná další komunikace mezi agenty. To ale neznamená, že by Dec-POMDP nemohlo modelovat prostředí, které zahrnují komunikaci mezi agenty. Jejich komunikace ale musí být modelována pomocí příslušných stavů a akcí již v daném Dec-POMDP. Například pokud jeden z agentů může provést akci *"pošli zprávu"* zatím co druhý agent zvolí akci *"přijmi zprávu"*, tak agenti mají možnost komunikovat pomocí rozhraní stavů a použití jejich akcí. V Dec-POMDP tedy není přímá sémantika pro komunikaci



Obrázek 2.4: Ilustrace chování Dec-POMDP systému. V každé fázi je model je určitém stavu. Z tohoto stavu je odvozeno společné pozorování z kterého každý z agentů vidí pouze své pozorování. Na základě těchto pozorování každý z agentů vybere konkrétní akci kterou chce vykonat. Tyto akce dohromady zformují společnou akci která určí přechod do následujícího stavu. Tento obrázek byl adaptován z [30]

mezi agenty. Zato se tato komunikace dá implicitně popsat pomocí kombinace akcí, stavů a pozorování.

Příklad 2.3.2 *Jako příklad Dec-POMDP si zde uvedeme model Dec-Tiger [27], který je často používaným benchmarkem pro práci s Dec-POMDP. Tento model simuluje dva agenty, kde každý z nich stojí před dvěma dveřmi: "levé"a "pravé". Za jedněmi z těchto dveří je hladový tygr a za druhými je poklad. Agenti však neví, za kterýma z těchto dveří je tygr a za kterýma je poklad. Tedy máme dva stavy* $S = \{SL, SP\}$*, které symbolizují zda se tygr nachází nalevo (SL), nebo napravo (SP). Každý z agentů se sám individuálně rozhoduje, zda otevře některé z těchto dvou dveří. Druhou možností, kterou může agent vykonat, je poslouchat zda tygra neuslyší za dveřmi. Množiny možných akcí pro oba agenty jsou totožné a tedy jsou:* $A^1 = A^2 = \{"OtevriLeve", OtevriPrave", "Poslouchej"\}$*. Pokud některý z agentů otevře dveře, tak je stav modelu opět resetován na stav SL nebo SP. Pravděpodobnost nastavení obou stavů je stejná. Tyto pravidla změn stavů jsou popsány v přechodové funkci* $\mathcal{T}$ *která je konkrétně popsána v tabulce 2.1.*

Akce/přechod	SL → SL	SL → SP	SP → SP	SP → SL
$\langle OtevriPrave, * \rangle$	0.5	0.5	0.5	0.5
$\langle OtevriLeve, * \rangle$	0.5	0.5	0.5	0.5
$\langle *, OtevriLeve \rangle$	0.5	0.5	0.5	0.5
$\langle *, OtevriPrave \rangle$	0.5	0.5	0.5	0.5
$\langle Poslouchej, Poslouchej \rangle$	1.0	0.0	1.0	0.0

Tabulka 2.1: Přechodová funkce $\mathcal{T}$ pro Dec-Tiger.

Pokud ale oba agenti zvolí akci "Poslouchej", tak stav modelu zůstane nezměněn. Po provedení akce každý agent získá novou hodnotu pozorování. Funkce pravděpodobnosti pozorování O_1 *a* O_2 *jsou pro oba agenty totožné a jsou popsány v tabulce 2.2.*

Akce	**Stav**	HL	HP
$\langle Poslouchej, Poslouchej \rangle$	SL	0.85	0.15
$\langle Poslouchej, Poslouchej \rangle$	SP	0.15	0.85
$\langle OtevriPrave, * \rangle$	SL	0.5	0.5
$\langle OtevriLeve, * \rangle$	SP	0.5	0.5
$\langle *, OtevriPrave \rangle$	SL	0.5	0.5
$\langle *, OtevriLeve \rangle$	SP	0.5	0.5

Tabulka 2.2: funkce pravděpodobnosti pozorování O pro Dec-Tiger.

Tyto funkce vždy vrací buďto hodnotu HL a nebo HP. Tyto hodnoty jsou vraceny s určitou pravděpodobností závislou na společné akci, která byla provedena, a na stavu, ve kterém se model právě nachází. Například pokud pokud oba agenti zvolí akci "Poslouchej", a model se právě nachází ve stavu SL, tak každý z agentů dostane s pravděpodobností 0.85 *pozorování HL a s pravděpodobností* 0.15 *pozorování HP. Hodnoty rewardů získaných při jednotlivých akcích jsou dány reward funkcí R a detailně jsou popsány v tabulce 2.3.*

Akce	SL	SP
$\langle OtevriPrave, OtevriPrave \rangle$	+20	-50
$\langle OtevriLeve, OtevriLeve \rangle$	-50	+20
$\langle OtevriPrave, OtevriLeve \rangle$	-100	-100
$\langle OtevriLeve, OtevriPrave \rangle$	-100	-100
$\langle Poslouchej, Poslouchej \rangle$	-2	-2
$\langle Poslouchej, OtevriPrave \rangle$	+9	-101
$\langle OtevriPrave, Poslouchej \rangle$	+9	-101
$\langle Poslouchej, OtevriLeve \rangle$	-101	+9
$\langle OtevriLeve, Poslouchej \rangle$	-101	+9

Tabulka 2.3: Reward funkce R pro Dec-Tiger.

Můžeme si všimnout, že pokud alespoň jeden z agentů otevře dveře, za kterými je tygr, tak jsou oba agenti potrestáni zápornou hodnotou rewardu. A to ať už dveře otevřou společně, nebo je otevře pouze jeden z agentů. Obdobně malou zápornou hodnotu rewardu získají, pokud oba agenti zvolí akci "Poslouchej". Naopak kladnou hodnotu rewardu získají, pokud otevřou dveře s pokladem. Nejvýhodnější je situace, kdy oba agenti společně otevřou dveře, za kterými je poklad. Je pro ně tedy nejvýhodnější pro zvýšení pravděpodobnosti zisku a snížení pravděpodobnosti ztráty chovat se stejně. Protože ale agenti nesdílejí navzájem informace o pozorování, tak nejsou schopni zajistit identické chování a musí se spoléhat na odhad chování druhého agenta.

V případě MDP agent používá strategii mapující stavy ke konkrétním akcím. V takovém případě agent nemusí vůbec pracovat s historií, jelikož platí Markovovo pravidlo. V případě POMDP agentovi již nestačí k rozhodování pouze aktuální stav, ale je možné vypočítat belief-state shrnující historii a tím pádem také získat Markovovův signál. V Dec-POMDP má každý z agentů přístup pouze k vlastní historii dílčích akcí a vlastním pozorování a tak zde není možné získat informace o společné historii akcí a pozorování stejně jak je tomu u POMDP. Je to z důvodu, že přechodová funkce a funkce pozorování jsou závislé na sdružených akcích a sdružených pozorování. Z toho důvodu je pro rozhodování v Dec-

POMDP používat množinu strategií, kde je každému agentovi přiřazena jedna strategie. Díky tomu se agenti mohou rozhodovat decentralizovaně a pouze na základě svých vlastních pozorování a své vlastní historie. V této práci budu pracovat s n-ticemi FSC, kde je každému agentovi přiřazeno právě jedno FSC. FSC agenta i budu dále označovat FSC^i a množinu stavů paměti agenta i budu značit N^i. V rámci tuto n-tici FSC označuji jako sdružené FSC.

2.4 Dec-POMDP s deterministickým pozorováním

V předchozí sekci jsem definoval Dec-POMDP použitím pravděpodobnostního popisu pozorování. Konkrétní sdružené pozorování se v každém stavu získalo až pomocí pravděpodobnostní funkce O. Tato definice je používána ve většině publikací na dané téma a pracují s ní ostatní metody pro řešení Dec-POMDP problémů, se kterými budu následně porovnávat výsledky své implementace. Tento způsob popisu Dec-POMDP však není vhodný pro popis řešení v následující kapitole. Proto zde definuji Dec-POMDP znova a to pomocí deterministického popisu pozorování. Ten automaticky každému stavu přiřazuje společné pozorování. Vytváří tak více stavů, u kterých však deterministicky určuje jejich pozorování a není tak potřeba jej pravděpodobnostně určovat. Díky tomu je možné lépe z daným modelem pracovat v kontextu induktivní syntézy zejména pak při vytváření Quotient MDP. Ve standardní definici Dec-POMDP s pravděpodobnostním popisem pozorování totiž pozorování každého stavu získáme pomocí pravděpodobnostní funkce O a tedy není pro každý čas pevně určeno. V tomto novém popisu Dec-POMDP s deterministickým pozorováním je pro každý stav pevně dané právě jedno pozorování. To umožňuje s daným Dec-POMDP pracovat obdobně jako s POMDP a díky tomu je možné rozšířit existující metody pro syntézu FSC pro POMDP tak aby bylo možné pracovat i s Dec-POMDP modely. Mimo jiné je díky této reprezentaci možné Dec-POMDP lépe reprezentovat v grafech a intuitivněji popisovat.

Definice 2.4.1 ***(Dec-POMDP s deterministickým pozorováním).***
Nechť $D = (\mathcal{D}, S', \boldsymbol{A'}, \mathcal{T}', R', \boldsymbol{O'}, O', I)$ *je Dec-POMDP. Tento Dec-POMDP lze také popsat pomocí deterministického popisu pozorování jako šestici* $(\mathcal{D}, S, \boldsymbol{A}, \mathcal{T}, R, s_0)$*, kde* S *je konečná množina stavů daného Dec-POMDP, pro kterou platí* $S = \{S' \times \boldsymbol{O} \cup s_0\}$*. Sdružené pozorování stavu s budeme dále značit jako* $\boldsymbol{O}(s)$*. Dílčí pozorování agenta i ve stavu s budeme dále značit* $\boldsymbol{O}^i(s)$*.* $\boldsymbol{A} = \boldsymbol{A'} \cup a_0$ *je konečná množina společných akcí,* $\mathcal{T}$ *je funkce pravděpodobnostních přechodů po použití konkrétní akcí,kde platí* $\mathcal{T}((s', o)|(s, o), \boldsymbol{a}) = Pr'(s'|s, \boldsymbol{a}) * O(o|s')$ *a* $\mathcal{T}((s, o)|s_0, a_0) = I(s) * O(o|s)$*.*

Příklad 2.4.2 *Mějme Dec-POMDP s pravděpodobnostním pozorováním se třemi stavy* $S' = \{s_0, s_1, s_2\}$*, dvěmi sdruženými pozorováními* $\boldsymbol{O'} = \{z_0, z_1\}$ *a čtyřmi sdruženými akcemi* $\boldsymbol{A'} = \{\langle a_{00}, a_{10}\rangle, \langle a_{01}, a_{10}\rangle, \langle a_{00}, a_{11}\rangle, \langle a_{01}, a_{11}\rangle\}$ *První z agentů má tedy dvě akce* a_{00} *a* a_{01} *a druhý agent má akce* a_{10} *a* a_{11}*. Přechodová funkce* $\mathcal{T}$ *je popsána v tabulce 2.4. Pro zjednodušení tabulky jsou zde vynechány řádky pro přechody* $s_0 \to s_0$*,* $s_1 \to s_0$*,*$s_1 \to s_2$*,* $s_2 \to s_0$ *a* $s_2 \to s_1$*. Pravděpodobnost těchto přechodů pro všechny kombinace sdružených akcí je 0.0. Tabulka popisující funkci pravděpodobnosti pozorování* $\boldsymbol{O}$ *je popsána na obrázku 2.5.*

Při tvorbě Dec-POMDP s deterministickým pozorováním odpovídajícímu výše popsanému Dec-POMDP s pravděpodobnostním pozorováním nejdříve vytvoříme stavy, pro každou kombinaci stavů původního Dec-POMDP a sdružených pozorování. V tomto případě tedy vznikne sedm stavů $S = \{\langle s_0\rangle, \langle s_0, z_0\rangle, \langle s_0, z_1\rangle, \langle s_1, z_0\rangle, \langle s_1, z_1\rangle, \langle s_2, z_0\rangle, \langle s_2, z_1\rangle\}$*. Dále vytvoříme akci* a_0 *pro nově vytvořený stav* s_0*. Tato akce slouží k přechodu do počátečního stavu původního Dec-POMDP.*

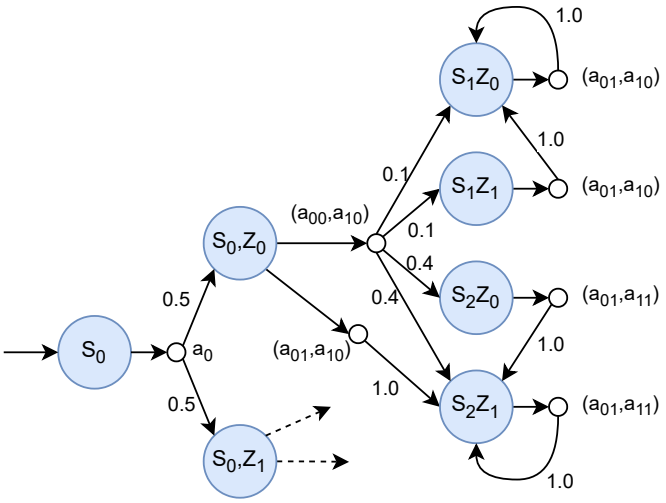
Akce/přechod	$s_0 \to s_1$	$s_0 \to s_2$	$s_1 \to s_1$	$s_2 \to s_2$
$\langle a_{00}, a_{10} \rangle$	0.2	0.8	0.0	0.0
$\langle a_{00}, a_{11} \rangle$	0.0	1.0	0.0	0.0
$\langle a_{01}, a_{10} \rangle$	0.0	0.0	1.0	0.0
$\langle a_{01}, a_{11} \rangle$	0.0	0.0	0.0	1.0

Tabulka 2.4: Přechodová funkce $\mathcal{T}$ pro Dec-POMDP příkladu 2.4.2

Akce	**Stav**	z_0	z_1
$\langle a_{00}, a_{10} \rangle$	s_0	0.5	0.5
$\langle a_{00}, a_{11} \rangle$	s_0	1.0	0.0
$\langle a_{01}, a_{10} \rangle$	s_1	1.0	0.0
$\langle a_{01}, a_{11} \rangle$	s_2	0.0	1.0

Tabulka 2.5: funkce pravděpodobnosti pozorování O pro pro Dec-POMDP příkladu 2.4.2.

Dále se vytvoří akce pro ostatní stavy. Konkrétně si předvedeme výpočet pravděpodobnosti přechodu do stavů $\langle s_1, z_0 \rangle$ *a* $\langle s_2, z_1 \rangle$ *ze stavu* $\langle s_0, z_0 \rangle$ *při použití akce* $\langle a_{00}, a_{10} \rangle$*. Pro stav* $\langle s_1, z_0 \rangle$ *se tato pravděpodobnost získá vynásobením pravděpodobnosti přechodu* $s_0 \to s_1$ *za použití akce* $\langle a_{00}, a_{10} \rangle$ *s pravděpodobností pozorování ve stavu* s_1 *po použití dané akce. Tedy v tomto případě je tato pravděpodobnost* $0,2 * 0,5 = 0,1$*. Obdobně pak pro stav* $\langle s_2, z_1 \rangle$ *a akci* $\langle a_{00}, a_{10} \rangle$ *je tato pravděpodobnost přechodu* $0,8 * 0,5 = 0,4$*. Stejným způsobem se následně vypočítají pravděpodobnosti přechodů pro ostatní kombinace stavů a akcí. Graf vytvořeného Dec-POMDP s deterministickým pozorováním je na obrázku 2.5.*



Obrázek 2.5: Graf Dec-POMDP s deterministickým pozorováním. Pro zjednodušení grafu je zde vynechán stav $\langle s_0, z_1 \rangle$. Akce tohoto stavu odpovídají akcím stavu $\langle s_0, z_0 \rangle$

2.5 Specifikace vlastností DTMC

Pro syntézu konečných kontrolérů Dec-POMDP, kterou se zabývám v následujících kapitolách, je nutné umět správně specifikovat vlastnosti DTMC indukovaných pomocí daných

kontrolérů, které hledáme. V této práci se zabývám prací v nekonečném horizontu a tak se i zde budu zabývat pouze vlastnostmi pracujícími pouze v nekonečném horizontu. Konkrétně se v rámci práce zabývám pouze vlastností pravděpodobnosti dosažení cílového stavu a očekávané hodnoty rewardu. Mějme DTMC $D = ((S, s_{init}, P)$ a množinu cílových stavů $T \subseteq S$. $\mathbb{P}^D[s \models \Diamond T]$ značí pravděpodobnost dosažení množiny stavů T ze stavu $s \in S$. Obdobně $ExpRew^{\mathcal{D}}(s_0 \models \Diamond T)$ značí očekávanou hodnotu rewardu získanou při přechodu do množiny stavů $T \subseteq S$ ze stavu $s \in S$. Tedy průměrnou hodnotu součtu rewardů cesty ze stavu s do některého ze stavů z množiny T. Pokud platí $\mathbb{P}^D(s_0 \models \Diamond T) < 1$, tak je hodnota $ExpRew^{\mathcal{M}}(s_0 \models \Diamond T)$ rovna ∞.

Při práci v konečném horizontu se pracuje s vlastnostmi omezenými počtem akcí po jejichž provedení chování modelu již danou vlastnost neovlivňuje. Další alternativou je uvažování o vlastnostech v kontextu nekonečného horizontu, ale s klesajícím významem provedených akcí. Rychlost klesání důležitosti akce s časem se určuje pomocí s discount faktoru, který je v rozmezí 0 až 1. Tento discount faktor zaručuje že akce provedené na začátku mají větší vliv na vlastnost než akce provedené později. Pokud je hodnota discount faktoru blízká 0, je největší vliv kladen na akce provedené zpočátku. Naopak čím bližší je hodnota 1, tím větší důraz je kladen na akce provedené v budoucnosti. Při použití discount faktoru akce provedené po určitém čase mají tak malý vliv na vlastnosti daného DTMC, že jejich volba téměř ztrácí význam.

Kapitola 3

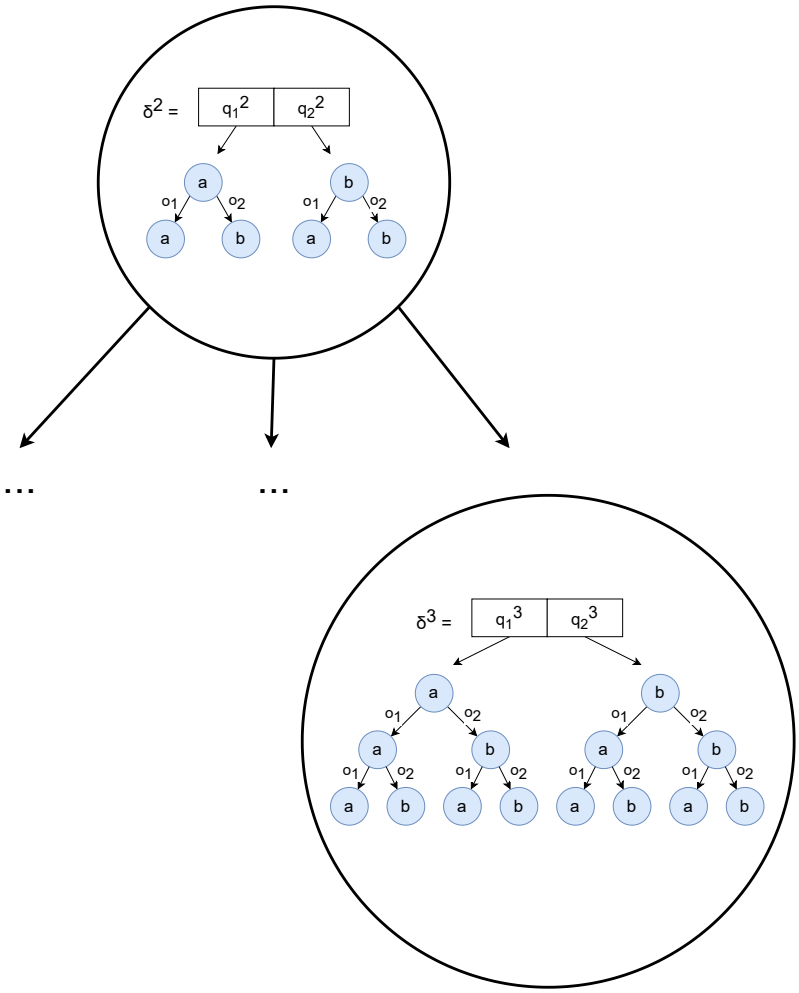
Existující přístupy

V rámci této kapitoly se věnuji existujícím přístupům k řešení Dec-POMDP problémů. Nejdříve se věnuji přístupům řešícím Dec-POMDP problémy v konečném horizontu. Tedy přístupy které hledají pouze strategie pro omezený počet kroků. Konkrétně se zde nejvíce věnuji metodě MMA* a její verzi RS MMA*. Poté se věnuji přístupu řešícímu Dec-POMDP problém v nekonečném horizontu. Nejdříve zde řeším dané metody obecně a následně se více věnují přístupům JESP a metodám využívajícím k reprezentaci strategií rekurzivní FSC. V závěru této kapitoly se věnuji induktivní syntéze kontrolérů pro POMDP. Z této metody vychází přístup k řešení Dec-POMDP problémů který je obsahem této práce.

3.1 Přístupy pracující v konečném horizontu

Jedním z přístupů řešení Dec-POMDP problémů je najít řešení pouze v konečném horizontu. V takovém případě strategie popisuje chování pouze pro určitý počet provedení akcí každého agenta. Pro POMDP s je možné řešení v konečném horizontu reprezentovat pomocí rozhodovacího stromu, kde uzly reprezentují akce a hrany reprezentují získané pozorování. Obdobným způsobem lze také reprezentovat Dec-POMDP [29]. To lze reprezentovat pomocí vektoru rozhodovacích stromů. Každý z těchto rozhodovacích stromů náleží jednomu z agentů a jejich akce jsou prováděny synchronně. Tento vektor dále budeme nazývat vektorem strategií. Hledání strategií pomocí vektoru strategií lze provést pomocí postupné konstrukce množiny strategií v horizontu $t + 1$ z jeho nadřazené strategie v horizontu t. Tímto způsobem vzniká strom vektorů strategií. Nad tímto stromem vektoru strategií lze provádět vyhledávání optimální strategie pomocí metody A*. Ukázka části takového rozhodovacího stromu vektorů strategií je na obrázku 3.1. Jedná se o strategie pro Dec-POMDP problému s dvěma agenty. Každý z agentů má pouze dvě pozorování o_1 a o_2 a dvě akce a a b. Na daném obrázku je jeden vektor strategií v horizontu 2 a jeden z něj odvozený vektor strategií horizontu 3.

V rámci MMA* je prováděno A* prohledávání nad jednotlivými vektory strategií φ^t. Pro specifikaci horizontu h je vypočtena hodnota heuristické funkce $\mathcal{V}(\varphi^t)$ pomocí hodnot $\mathcal{V}^{0\ldots t-1}(\varphi^t)$ a heuristické funkce H pro zbývajících $h-t$ fází. V rámci algoritmu je udržována prioritní fronta P. Zde jsou jednotlivé vektory strategií seřazeny na základě jejich hodnoty $\mathcal{V}(\varphi^t)$. Během každé iterace je z fronty P vybrán první vektor strategií a ten je následně expandován φ^t. Tím se získají nové vektory strategií φ^{t+1}, které jsou následně vloženy do fronty P. Prohledávání se ukončí při vyprázdnění fronty P. Existuje více přístupů využívajících různé heuristické funkce H. Jednou z možností je analyzovat danou část Dec-POMDP



Obrázek 3.1: Část prohledávacího stromu vektoru strategií metody MAA* pro Dec-POMDP problém s dvěma agenty. Idea obrázku převzata z [37].

jako centralizovaný MDP s úplným pozorováním. Další možností je například analýza dané části Dec-POMDP jako POMDP. V tomto případě se již pracuje s částečným pozorováním, ale stále pouze se společnými akcemi. Tímto způsobem se pracuje s danou částí Dec-POMDP jako centralizovanou a z tohoto důvodu získané řešení nad-aproximovávají reálné hodnoty rewardu pro dané Dec-POMDP. Dalším možným přístupem je využití rekurzivního MMA*. V tomto přístupu se heuristická funkce vypočte podle $H^{h-t} = MAA^{*H-t}$. To vede k rekurzivnímu průchodu danými vektory strategií. Tento přístup ostatní převyšuje kvalitou heuristické funkce, ale je velice výpočetně náročný.

Problémem metody MMA* je dvojitě exponenciální růst počtu vektorů strategií vzhledem k hodnotě t. Tento problém řeší například metoda RS MAA* [19]. Ta využívá kombinaci rekurzivního MAA* s *small step* přístupem. V tomto přístupu se strom vektorů strategií expanduje postupně po dílčích pozorováních a akcích každého z agentů. Pro porovnání v klasickém MMA* přístupu má každý uzel v horizontu t $|A_*|^{n|O_*|^t}$ následovníků, kde $|A_*|$ a $|O_*|$ značí největší lokální množinu akcí a pozorování. Ve *small step* MMA* se dané expanze rozdělí do $n|O_*|^t$ kroků, kde má každý uzel $|A_*|$ následovníků. V tomto přístupu

tedy počet uzlů roste pouze konstantní rychlostí, ale délka stromu vektorů strategií roste exponenciálně.

3.2 Přístupy pracující v nekonečném horizontu

Tato sekce se věnuje přístupům pracujícím v nekonečném horizontu. Většina metod pro řešení Dec-POMDP problémů v nekonečném horizontu reprezentuje strategie pomocí FSC. Existující metody pracující v nekonečném horizontu pracují se specifikacemi s discount faktorem. Tyto metody dávají větší vliv akcím provedeným dříve než akcím provedeným později. Tyto metody tedy pracují v nekonečném horizontu, současně ale pracují s informací, že akce provedené po určitém počtu kroků již mají na výsledek zanedbatelný vliv. Jednou z variant hledání řešení Dec-POMDP problémů je optimalizovat dané parametry přímo pomocí nelineárního programování [3]. Dalším možným přístupem je hledat optimální nastavení parametrů pro zisk maximální hodnoty očekávaného rewardu pomocí metody *Expectation-Maximization* [24, 34].

3.2.1 JESP

Joint Equilibrium based Search for Policies (JESP) je metoda, ve které se hledá n-tice takových dílčích strategií, kde pro každého agenta i je strategie π^i nejlepší možnou reakcí na chování ostatních agentů π^{-i}. Jedná se tedy o hledání lokálního optima, neboli konkrétně o hledání Nashova equilibria. Metoda JESP je založena na procesu *alternating maximization*. Tento proces je založen na výpočtu strategie π^i agenta i, tak aby se maximalizovala společná hodnota očekávaného rewardu. Ostatní strategie při tom zůstávají nezměněné. Poté je k optimalizaci svého chování pro maximalizaci společného rewardu zvolen následující agent. Tento proces je opakován až do chvíle, kdy je společná strategie nashovým equilibriem. Toto získané řešení je lokálním optimem. Tento proces tedy svým chováním odpovídá metodě *hill-climbing*. Na následujícím příkladu je ukázáno, že lokální optimum může v některých případech dosahovat velice špatného výsledku.

Příklad 3.2.1 *Mějme Dec-POMDP DecTyger z příkladu 2.3.2. Představme si situaci kdy by na začátku metody JESP byla strategie prvního agenta provést akci OtevriPrave. V tomto případě je správným rozhodnutím druhého agenta také provést tuto akci a otevřít pravé dveře. Při druhé iteraci nemá první agent opět žádnou lepší volbu na provedení akce. Výsledná společná strategie by tedy byla v prvním kroku společně provést akce OtevriPrave. Tato strategie, ale očividně není nejvhodnější strategií pro maximalizaci očekávaného rewardu.*

Metoda JESP využívá dynamické programování k výpočtu nejlepší odpovědi agenta i na sdruženou strategii ostatních agentů. Zachování identické strategie ostatních agentů po celou dobu optimalizace strategie agenta i umožňuje vnímat problém jako rozšířený POMDP. V tomto rozšířeném POMDP každý ze stavů reprezentuje stav Dec-POMDP a historii pozorování všech ostatních agentů v daném stavu. Díky fixnímu chování ostatních agentů splňuje každý z nově vytvořených stavů markovovu vlastnost. Toto rozšířené POMDP lze vytvořit pouze ze sdružené strategie ostatních agentů a původního Dec-POMDP.

3.2.2 Periodické FSC

Dalším přístupem jak reprezentovat strategii pro Dec-POMDP je využití periodického FSC [33]. To se skládá z několika vrstev, které jsou navzájem propojeny. Přechody první

vrstvy vedou vždy do vrstvy druhé, z druhé vrstvy se přechází do třetí atd. Z poslední vrstvy přechody vedou opět do vrstvy první. Šířka každé vrstvy odpovídá paměti FSC. Pokud by periodické FSC mělo pouze jednu vrstvu, tak by odpovídalo klasickému FSC. Akce a přechodové funkce jednotlivých vrstev periodického FSC se liší. Metody využívající periodické FSC nejdříve naleznou optimální strategii pro specifikaci v konečném horizontu. Poté vytvoří takové periodické FSC, kde první průchod tímto FSC odpovídá použití dané strategie v konečném horizontu. Pro jednoduché problémy již takové FSC dosahuje kvalitních výsledků. Poté je dále možné s toto periodické FSC dále optimalizovat, tak aby dosahovalo lepších výsledků i pro složitější problémy.

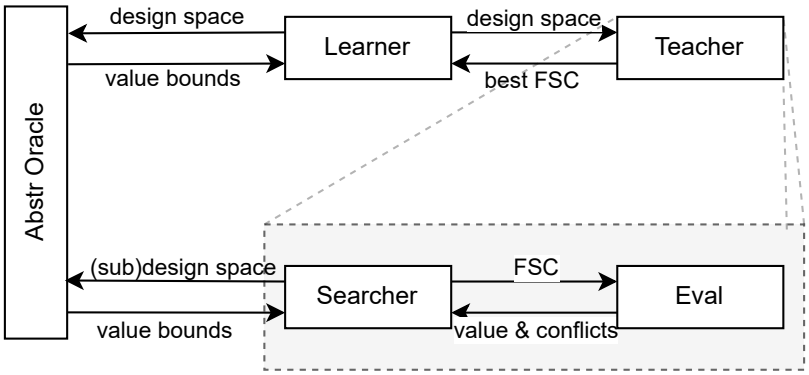
3.3 Induktivní syntéza POMDP

V této sekci se zabývám induktivním přístupem k syntéze konečných kontrolérů pro POMDP [5]. Ta se zabývá automatickým návrhem FSC pro daný POMDP tak, aby indukovaný DTMC splňoval požadované specifikace v nekonečném horizontu. Tato metoda se skládá ze dvou částí: vnitřní část a vnější část. Ve vnější části se nachází *learner*. Ten vytváří konečnou množinu FSC ze kterých se následně bude vybírat finální FSC. Tato množina FSC se nazývá *design space*. Druhou částí vnější části je *teacher*. Ten na vstupu získává od *learneru* daný *design space*. Jeho cílem je najít v daném *design space* potenciálně nejlepší FSC a vrátit ho *learneru*. Ten dané FSC buďto přijme nebo pozmění *design space* a znova jej odešle *teacheru*. Schéma tohoto procesu je popsáno na obrázku 3.2. Čím menší daný *design space* je, tím rychleji a lépe *teacher* nalezne dané FSC. Z toho důvodu jsou ve vnější části nejdříve vybírány malé *design space* a podle výsledků *teacheru* nad danými *design space* jsou rozšiřovány. K práci s FSC *learner* používá MC získané indukcí daných FSC a POMDP a také abstrakci rodiny MC získané indukcí daného *design space*. Použití abstrakce budu více popisovat dále.

Vnitřní část se zabývá nalezením nejlepšího FSC, který se nachází v daném *design space*. K prohledání daného *design space* existuje několik přístupů jako např. naivní prozkoumání všech FSC z daného *design space* (metoda one-by-one), využití smíšeného celočíselného lineárního programování (MILP) [3], nebo použití metody *branch-and-bound* [14]. V rámci této práce se zabývám využitím další vnořené induktivní syntézní smyčky. Z daného *design space* se vybere jedno FSC které se ohodnotí na základě požadované specifikace. Pokud dané FSC nesplňuje specifikované požadavky, jsou analyzovány kritické části dané strategie, které způsobují nevyhovění dané specifikace. Ty jsou dále použity k prořezání daného *design space*.

Ve vnitřní i vnější fázi se využívá *abstact oracle*. To obsahuje nadaproximaci daného *design space*. Tato nadaproximace *design space* lze analyzovat obdobně jako běžné FSC. Dále v práci je tato nadaproximace nazívána Quotient MDP. Díky analýze *abstact oracle* lze zjistit maximální *ub* a minimální hodnoty *lb*, kterých může dosáhnout FSC obsažené v daném *design space*. Tyto informace jsou využívány ve vnější i vnitřní části induktivní syntézy.

Mějme konečnou rodinu FSC $\mathcal{F}_k$ obsahující k-FSC, kde stavy paměti jsou $N = \{n_0, n_1 ... n_{k-1}\}$. Dále mějme indukovanou rodinu MC $\mathcal{M}^{\mathcal{F}_k} = \{\mathcal{M}_{\mathcal{F}} | F \in \mathcal{F}_k\}$, kde stavy každého MC jsou dvojicí $(s, n) \in S \times N$. Díky vytvoření Quotient MDP $\mathcal{Q}^{\mathcal{F}}$ rodiny $\mathcal{M}^{\mathcal{F}_k}$ můžeme získat obecné informace o dané rodině a uvažovat o ní jako o celku. Quotient MDP rodiny $\mathcal{M}^{\mathcal{F}_k}$ rozumíme takové MDP $A^{\mathcal{F}}$, kde je množina stavů rovna $S \times N$. Současně pokud některé z MC $M \in \mathcal{M}^{\mathcal{F}_k}$ umožňuje ve stavu $(s, n) \in S \times N$ provedení akce α, tak



Obrázek 3.2: Struktura vnořené induktivní smyčky. Na vstupu jsou POMDP a specifikace. Výstupem je FSC splňující dané specifikace. Idea obrázku převzata z [6].

je také v $\mathcal{Q}^{\mathcal{F}}$ tato akce umožněna ve stavu (s, n) a to se stejným efektem. MDP $\mathcal{Q}^{\mathcal{F}}$ tedy nadaproximovává chování celé rodiny $\mathcal{M}^{\mathcal{F}_k}$.

Definice 3.3.1 ***(Quotient MDP pro POMDP).*** *MDP* $\mathcal{Q}^{\mathcal{F}} = (S \times N, (s_0, n_0), Act^{\mathcal{F}}, P^{\mathcal{F}})$ *je Quotient MDP rodiny DTMC* $\mathcal{M}^{\mathcal{F}}$*, kde* $Act^{\mathcal{F}} = Act \times N$ *a* $P^{\mathcal{F}}((s', n')|(s, n), (a, n')) = P(s'|s, a)$ *pokud* $(a, n') \in V_{(n,O(s))}$*, jinak* $P^{\mathcal{F}}((s', n') = 0$*.*

Deterministická bezpaměťová strategie π pro Quotient MDP $\mathcal{Q}^{\mathcal{F}}$ je definována funkcí $\pi : S \times N \to Act^{\mathcal{F}}$. Tato strategie je konzistentní pokud $O(s) = O(s')$ implikuje to že $\pi((s, n)) = \pi((s', n))$ pro všechny $s, s' \in S$ a $n \in N$. Strategie π je naopak nekonzistentní pro parametr $(n, z) \in K$ pokud $\exists s, s' \in S : O(s) = O(s') \wedge \pi((s, n)) \neq \pi((s', n))$. Strategie je nekonzistentní v pozorování $z \in Z$ pokud je nekonzistentní v parametru $(n, z) \in K$ pro některé $n \in N$.

Informace získané z abstrakce MDP lze pro induktivní sytézu FSC využít následovně. Mějme podmínku $\varphi = P_{\geq\lambda}[\Diamond\ T]$. Díky Quotient MDP $\mathcal{Q}^{\mathcal{F}}$ můžeme získat strategii π^* která maximalizuje dosažitelnost stavu T. Pravděpodobnost dosažení stavu T při použití strategie π^* budeme dále označovat Pr^{π^*}. Pokud platí $Pr^{\pi^*} < \lambda$ tak víme že žádné FSC $F \in \mathcal{F}$ nesplňuje danou podmínku a mohou být zahozeny. V opačném případě se kontroluje konzistenčnost dané strategie π^*. Pokud je konzistentní, tak reprezentuje FSC $F \in \mathcal{F}$ které splňuje danou podmínku. V opačném případě se daná rodina FSC rozdělí a získané části se opět analyzují stejným způsobem.

Strategie dělení daných rodin FSC je založena na nalezení nekonzistence v optimální strategii π^* pro specifikované požadavky. Nejdříve se naleznou nekonzistentní parametry $(n, z) \in K$ pro danou strategii π^*. Poté se určí vliv těchto parametrů na základě rozptylů hodnot $ub((n, s))$ pro $O(s) = z$ váhované očekávanou pravděpodobností dosažení daného stavu za použití dané strategie π^*. Na základě získaných hodnot se určí nejvýznamnější parametr $(n, z) \in K$. Nechť $V_{(n,z)}$ je doménou parametru (n, z) a maximalizující strategie π^* zvolila parametry $v_1, ..., v_i$. V takovém případě se doména $V_{(n,z)}$ rozdělí na $\{v_1\}, ..., \{v_i\}$ a $V_{(n,z)} \backslash \{v_1, ..., v_i\}$. Díky této strategii dělení rodin FSC se získé i+1 nových MDP které jsou zbaveny nekonzistence na parametru (n, z).

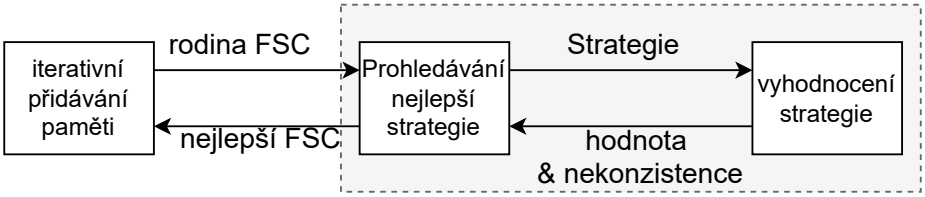
Kapitola 4

Induktivní syntéza pro Dec-POMDP

V rámci předchozí kapitoly jsem se věnoval existujícímu řešení induktivní syntézy konečně stavových kontrolérů pro POMDP. V této kapitole popisuji rozšíření tohoto přístupu tak, aby šel aplikovat i pro decentralizované POMDP. Klíčovým objektem mého přístupu je tvorba Quotient MDP a následně jeho obarvení. Jedná se o části syntézy, díky jejichž změně je možné aplikovat metody navržené pro POMDP i pro Dec-POMDP. Nejdříve se v rámci této kapitoly popisuje schéma induktivní syntézy pro Dec-POMDP. Následně je zde popsána tvorba Quotient MDP a jeho discount transformace. Poté je zde popsán pojem konzistence pro Quotient MDP a také popis různých druhů nekonzistence pro Dec-POMDP. Následně se věnuji popisu procesu obarvení Quotient MDP. V závěru kapitoly je popsána nová strategie pracující s rozdílem charakteristiky různých druhů nekonzistencí strategií.

4.1 Schéma induktivní syntézy pro Dec-POMDP

Tato sekce popisuje schéma induktivní syntézy Dec-POMDP, se kterým se dále pracuje v rámci této práce. Toto schéma je inspirováno strukturou syntézy kontrolérů pro POMDP probrané v sekci 2.2. V rámci práce s Dec-POMDP je schéma zjednodušeno. Toto schéma induktivní smyčky je popsáno na obrázku 4.1. Prohledávané rodiny sdružených FSC se zde získávají iterativním zvyšováním paměti daných k-FSC. Na počátku se prohledává rodina sdružených 1-FSC. Poté se postupně zvyšuje velikost paměti. Pro každou z těchto rodin FSC se následně vytvoří Quotient MDP. Každá z těchto iterací prohledávání má v sobě druhou induktivní smyčku. V té se postupně pro daný Quotient MDP hledá strategie π^*, která maximalizuje požadované specifikace. Tato strategie se dále používá k dělení dané rodiny FSC identicky jako u syntézy FSC pro POMDP. Rozdílem oproti práci s POMDP je zde určení konzistence dané strategie. Ta zde neurčuje pouze to, zda lze danou strategii popsat pomocí určité velikosti FSC, ale současně i zaručuje decentralizované chování agentů a jejich nezávislost na informacích o historii pozorování ostatních agentů. K určování nekonzistence dané strategie slouží obarvení Quotient MDP. Detailnější popis rozdílů konzistence u POMDP a Dec-POMDP je v sekci 4.4.



Obrázek 4.1: Schéma induktivní syntézy Dec-POMDP

4.2 Tvorba Quotient MDP

Quotient MDP pro Dec-POMDP a n-tici rodin FSC je MDP nadaproximující chování daného Dec-POMDP modelu při použití FSC z dané n-tice rodin FSC. Převod Dec-POMDP na Quotient MDP umožňuje aplikaci induktivní syntézy. Díky Quotient MDP je možné hledat optimální strategie Dec-POMDP pomocí metod pro MDP. Současně díky Quotient MDP lze získat nad-aproximaci vlastností daného Dec-POMDP a rodiny FSC.

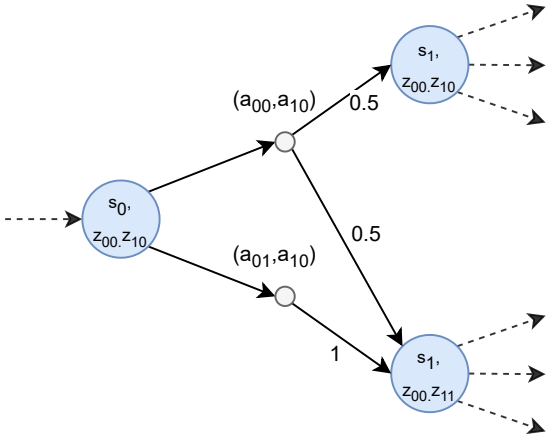
Tvorby Quotient MDP pro Dec-POMDP a n-tici rodin FSC je dosaženo vytvořením kopie každého stavu pro každou z kombinací nastavení pamětí FSC všech z agentů. Celkem se tedy pro každý stav vytvoří $\mathcal{K}^n$ kopií daného stavu, kde n je počet agentů daného Dec-POMDP, $\mathcal{K} = |N_0| * |N_1| * \cdots * |N_n|$ a N_n značí N agenta n. Obdobně je i pro každý z těchto vytvořených stavů vytvořeno $\mathcal{K}^n$ odpovídajících nových akcí. Daný Quotient MDP má tedy celkem $\mathcal{K}^{2*n}$ akcí.

Definice 4.2.1 ***(Quotient MDP).*** *Mějme Dec-POMDP $D = (\mathcal{D}, S, \boldsymbol{A}, \mathcal{T}, R, s_0)$ a n-tici rodin FSC $\mathcal{F}_{k,n} = (N_n, n_0, K)$ pro každého z n agentů $\mathcal{D}$. Quotient MDP $\mathcal{Q}$ daného Dec-POMDP a n-tici rodin FSC je MDP $\mathcal{Q} = (S', s'_{init}, Act, \mathcal{P}')$, pro který platí $S' = (S \times N)$, $Act(s') = (\boldsymbol{A}(s) \times N)$, kde $N = \{N_0 \times \ldots \times N_n\}$, $s'_{init} = s_0$ a $\mathcal{P}' : S' \times Act \nrightarrow Distr(S')$. Pro pravděpodobnosti přechodů při použití konkrétních akcí platí $\mathcal{P}'((s, M), (a, N), (s', N) = \mathcal{T}(s)(a)(s')$, kde $s, s' \in S, a \in A$ a $M, N \in \{N_0 \times \ldots \times N_n\}$.*

Stavy Quotient MDP představují situace, kdy se model Dec-POMDP nachází v konkrétním stavu a současně se FSC každého z agentů nachází v určitém stavu paměti. Při syntéze kontrolérů se tak může uvažovat nad stavem paměti FSC všech agentů určitém stavu. Obdobně každá z nově vytvořených akcí v sobě zahrnuje informaci o nastavení paměti všech agentů v následujícím stavu. Zvolením této akce se tedy nejen rozhodne o tom, které akce budou zahrnuty do výsledné strategie, ale také o tom do kterého stavu paměti budou nastaveny FSC všech agentů v následujícím stavu. Díky tomuto rozšíření je možné vytvářet kontroléry pracující s pamětí. Současně Quotient MDP nad-aproximovává chování Dec-POMDP a lze z něj získat množiny hodnot *ub* a *lb* které pro každý ze stavů $s \in S$ určují maximální a minimální možnou pravděpodobnost dosažení cílového stavu z daného stavu s.

Příklad 4.2.2 *Mějme Dec-POMDP, které je popsané grafem na obrázku 4.2. Pro zjednodušení příkladu pracujeme pouze s částí Dec-POMDP modelu. Též se pro zjednodušení pracuje s modelem, kde jeden z agentů má pouze jedno pozorování a jednu akci. Druhý z agentů má dvě pozorování a dvě akce. Také si můžeme všimnout, že daný Dec-POMDP model je již převedený na Dec-POMDP s deterministickým pozorováním. Každý ze stavů tohoto modelu tedy reprezentuje určitou kombinaci stavu Dec-POMDP a pozorování každého z agentů. Pozorování jednotlivých agentů budeme dále značit pomocí z_{xy}, kde x značí*

zda se jedná o agenta 0 nebo o agenta 1 a y značí hodnotu daného pozorování. Obdobně akce budeme dále značit a_{xy}, kde x značí zda se jedná o agenta 0 nebo o agenta 1 a y značí hodnotu dané akce. Tento model se skládá ze tří stavů, kde nadále budeme pracovat pouze s akcemi prvního stavu. Ten reprezentuje kombinaci stavu Dec-POMDP s_0 a pozorování agentů z_{00} a z_{10}. Zbylé dva stavy reprezentují stav Dec-POMDP s_1 a kombinace pozorování agentů z_{00} a z_{10} u prvního ze stavů a u druhého z_{00} a z_{11}.



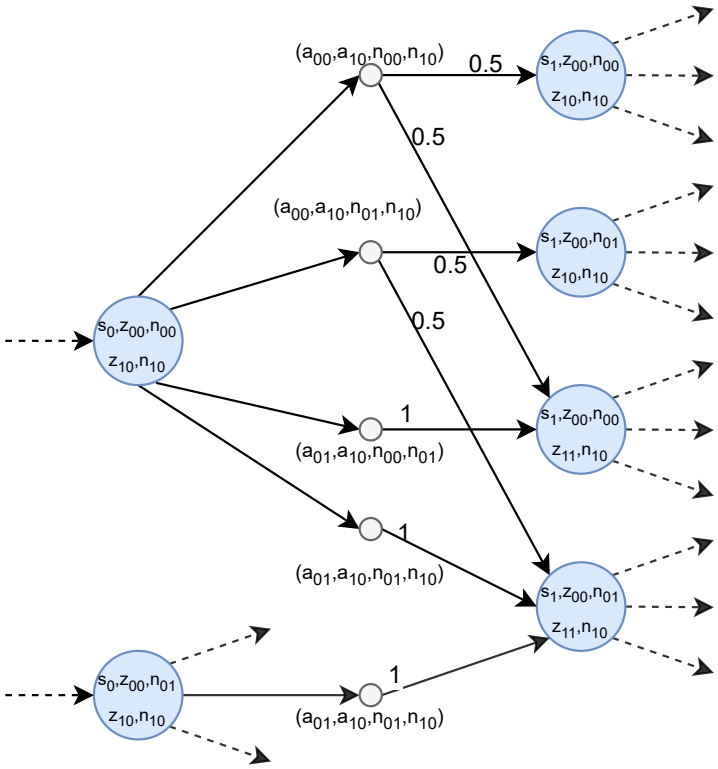
Obrázek 4.2: Graf Dec-POMDP k příkladu 4.2.2

Dále mějme n-tici rodin FSC obsahující rodinu 2-FSC pro prvního agenta a rodinu 1-FSC pro druhého agenta. Stavy paměti těchto FSC budeme dále značit pomocí n_{xy}, kde x značí zda se jedná o agenta 0 nebo o agenta 1, y značí hodnotu daného stavu paměti. Množina stavů rodiny FSC prvního agenta je tedy $\{n_{00}, n_{01}\}$ a pro druhého agenta $\{n_{10}\}$.

Quotient MDP pro daný Dec-POMDP a n-tici FSC je popsán na obrázku 4.3. V něm jsou pro každý stav původního Dec-POMDP vytvořeny dva stavy. První z nich vždy reprezentuje kombinaci stavů paměti FSC jednotlivých agentů n_{00} a n_{10}. Druhý reprezentuje stavy paměti n_{01} a n_{10}. Obdobně je tomu tak i u akcí, u kterých se také vytvoří pro každou akci původního Dec-POMDP dvě akce a to pro každé možné nastavení stavů pamětí následujícího stavů. Tyto akce se vytvoří pro každý z nově vytvořených stavů odpovídající původnímu stavu, kterému tyto akce náleželi v původním Dec-POMDP. Je důležité si uvědomit, že nově získané akce již nejsou společnými akcemi skládajícími se z akcí každého z agentů. Jsou to pouze akce MDP reprezentující kombinaci těchto akcí spolu s kombinací nastavení stavů paměti. Na obrázku 4.3, jsou kvůli zjednodušení zobrazeny všechny akce pouze pro stav $\langle s_0, z_{00}, n_{00}, z_{10}, n_{10}\rangle$. Stav $\langle s_0, z_{00}, n_{01}, z_{10}, n_{10}\rangle$ má akce totožné. Celkem je tedy z původních tří stavů a dvou akcí vytvořeno 6 stavů a 8 akcí.

4.3 Discount transformace Dec-POMDP

Metoda induktivní syntézy kontrolérů pro Dec-POMDP kterou popisuji v rámci této práce pracuje v nekonečném horizontu. Většina existujících metod pro řešení Dec-POMDP problémů, které pracují v nekonečném horizontu, ale pracuje s discount faktorem o velikosti $\gamma < 1$. Pro porovnání výsledků s ostatními metodami je tedy nutné vytvořit Quotient MDP tak aby vlastnosti tohoto nového Quotient MDP odpovídaly specifikacím s odpovídajícím discount faktorem.



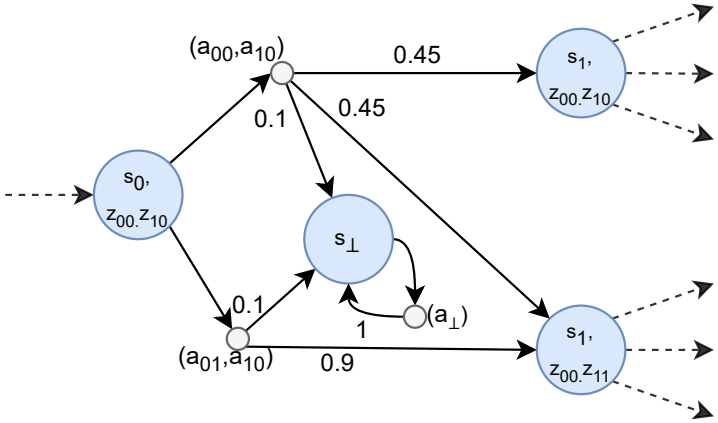
Obrázek 4.3: Graf Quotient MDP vytvořeného pro Dec-POMDP popsané na obrázku 4.2 a rodinu 2-FSC pro prvního agenta a 1-FSC pro agenta druhého. Graf zobrazuje všechny akce pouze pro stav $\langle s_0, z_{00}, n_{00}, z_{10}, n_{10}\rangle$. Stav $\langle s_0, z_{00}, n_{01}, z_{10}, n_{10}\rangle$ má akce totožné.

Tvorba Quotient MDP, tak aby simuloval vlastnosti s discount faktorem, probíhá nejdříve pomocí discount transformace původního Dec-POMDP s deterministickým pozorováním a následného vytvoření Quotient MDP klasickým způsobem popsaným sekci 4.2. Discount transformace Dec-POMDP zahrnuje vytvoření nového stavu $s_\perp$. Tento stav reprezentuje *discount sink* stav. Tento stav vždy slouží jako cílový stav a nelze z něj přejít nazpět do žádného z ostatních stavů. Dále se upraví všechny akce tak, aby po provedení každé akce byla určitá pravděpodobnost přechodu do stavu $s_\perp$. Tato pravděpodobnost je nastavena na základě velikosti discount faktoru γ.

Definice 4.3.1 ***(Discount transformace Dec-POMDP).*** *Mějme Dec-POMDP s deterministiským pozorováním $D = (\mathcal{D}, S, \boldsymbol{A}, \mathcal{T}, R, s_0)$. Provedením discount transformace daného Dec-POMDP D s hodnotou discount faktoru γ vzniká Dec-POMDP $D' = (\mathcal{D}, S', \boldsymbol{A}', \mathcal{T}', R', s_0)$, kde $S' = S \cup \{s_\perp\}$, $\boldsymbol{A}' = \boldsymbol{A} \cup \{a_\perp\}$, $\mathcal{T}'(s'|s, \boldsymbol{a}) = \mathcal{T}(s'|s, \boldsymbol{a}) * \gamma$ pro každé $s, s' \in S$ a $\boldsymbol{a} \in \boldsymbol{A}$. Také platí $\mathcal{T}'(s_\perp|s, \boldsymbol{a}) = 1 - \gamma$ pro všechna $s, s' \in S$ a $\boldsymbol{a} \in \boldsymbol{A}$ a $\mathcal{T}'(s_\perp|s_\perp, a_\perp) = 1$.*

Příklad 4.3.2 *Mějme Dec-POMDP s deterministickým pozorováním z příkladu 4.2.2. Toto Dec-POMDP je popsáno na obrázku 4.2. Z důvodu zjednodušení opět pracujeme pouze s částí daného Dec-POMDP. V tomto příkladu si ukážeme discount transformaci tohoto Dec-POMDP pro hodnotu discount faktoru $\gamma = 0,9$. V rámci discount transformace nejdříve vytvoříme nový stav $s_\perp$. Tento stav má pouze jednu akci $a_\perp$ po jejímž provedení se přechází nazpět do stavu $s_\perp$ s pravděpodobností 1. Následně si ukážeme transformaci akcí ze*

stavu $\langle s_0, z_{0,0}, z_{1,0}\rangle$*. Nejdříve provedeme transformaci akce* $\langle a_{0,0}, a_{1,0}\rangle$*. V původním Dec-POMDP se po provedení této akce přechází s pravděpodobností 0,5 do stavů* $\langle s_1, z_{0,0}, z_{1,0}\rangle$ *a* $\langle s_1, z_{0,0}, z_{1,1}\rangle$*. V Dec-POMDP po provedení discount transormace jsou tyto přechody vynásobeny hodnotou discount faktoru* γ*. Po provedení této akce je tedy pravděpodobnost přechodu do daných stavů rovna* $0,5 * 0,9 = 0,45$*. Pravděpodobnost přechodu do stavu* $s_\perp$ *je rovna* $1 - \gamma$*, tedy* $1 - 0,9 = 0,1$*. Obdobně po provedení akc* $\langle a_{0,1}, a_{1,0}\rangle$ *je pravděpodobnost přechodu do stavu* $\langle s_1, z_{0,0}, z_{1,1}\rangle$ *rovna* $1 * 0.9$ *a pravděpodobnost přechodu do stavu* $s_\perp$ *má hodnotu* $1 - 0,9 = 0,1$*. Výsledný Dec-POMDP po discount transformaci je popsán grafem na obrázku 4.4.*



Obrázek 4.4: Graf Dec-POMDP z příkladu 4.2.2 po discout transformaci s discount faktorem $\gamma = 0,9$.

4.4 Konzistence Quotient MDP

V předchozí sekci jsem popsal Quotient MDP, který nadaproximovává chování Dec-POMDP při použití množiny k-FSC. Z toho plyne, že je v něm provést i sekvence akcí, které by v původním Dec-POMDP za použití daných k-FSC nebylo možné provést. Dříve než popíši obarvení Quotient MDP které umožňuje tomuto nekonzistentnímu chování zamezit zde popíši typy konzistence Quotient MDP pro Dec-POMDP a jejich rozdílnost vůči konzistenci Quotient MDP pro POMDP.

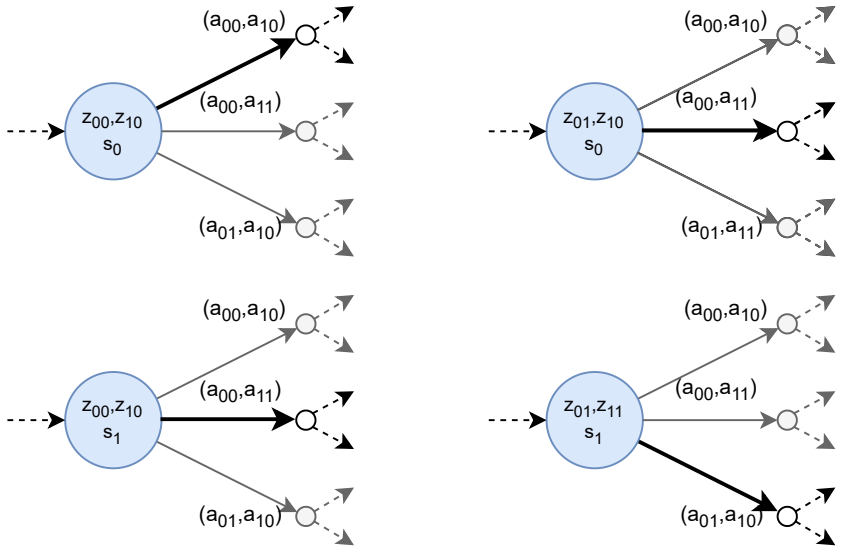
Konzistentním chování vůči pozorování c Dec-POMDP rozumíme, že $\mathbf{O}(s) = \mathbf{O}(s')$ implikuje $\pi(s, n) = \pi(s', n)$ pro $s, s' \in \mathbf{S}$ a $n \in N$. Tedy platí že se každý z agentů pro určité pozorování a stav paměti FSC rozhodne vždy stejně. Tento popis konzistence je obdobný jako u POMDP a budeme jej nadále označovat jako nekonzistenci prvního typu.

Problémem Dec-POMDP je skutečnost, že ověření nekonzistence prvního typu ke korektnímu výsledku nestačí, tak jak tomu je u POMDP. Dec-POMDP totiž modeluje decentralizované chování více agentů. Tito agenti se musí rozhodovat pouze na základě svých vlastních pozorování a historie svých akcí. V tomto případě, kdy využíváme množinu FSC, kde každý z agentů má vlastní FSC, se tedy smí rozhodovat pouze na základě svých dílčích pozorování a stavu paměti svého vlastního FSC. V případě kontroly pouze konzistence prvního typu by konzistence pozorování závisela pouze na sdruženém pozorování. Agenti v Dec-POMDP ale vidí pouze své vlastní dílčí pozorování a tak je potřeba dodržovat konzistentní chování vůči němu. Konzistence každého agenta tedy závisí pouze na jeho vlastním

dílčím pozorování v kombinaci se stavem paměti jeho FSC. Konzistence prvního typu tedy nezaručuje možnost definování dané strategie pomocí daného sdruženého FSC. Z toho důvodu je potřeba definovat konzistenci druhého typu.

Definice 4.4.1 ***(Konzistence druhého typu).*** *Pro strategii π platí konzistence druhého typu právě tehdy, pokud platí že $\boldsymbol{O}^i(s) = \boldsymbol{O}^i(s')$ implikuje $\pi^i(s,n) = \pi^i(s',n)$, pro $s, s' \in \boldsymbol{S}$, $i \in \mathcal{D}$, $n \in N^i$ a $\boldsymbol{O}^i$ značí dílčí pozorování agenta i pro pozorování $\boldsymbol{O}$. Strategie π^i je nekonzistentní vůči parametru rodiny FSC^i $(n,z) \in K$ pokud $\exists s, s' \in S : \boldsymbol{O}^i(s) = \boldsymbol{O}^i(s') = z \wedge \pi^i((s,n)) \neq \pi^i(s',n))$ pro $s, s' \in \boldsymbol{S}$, $i \in \mathcal{D}$ a $n \in N^i$. Nekonzistentní vůči pozorování $z \in Z$ je strategie tehdy, pokud je nekonzistentní pro parametr $(n,z) \in K$ pro některé $n \in N^i$.*

Příklad 4.4.2 *Mějme Dec-POMDP kde $s_0, s_1 \in S$. První z agentů má svá dílčí pozorování z_{00} a z_{01} a druhý agent má dílčí pozorování z_{10} a z_{11}. Společné akce jsou $\langle a_{00}, a_{10}\rangle, \langle a_{00}, a_{11}\rangle$, $\langle a_{01}, a_{10}\rangle, \langle a_{01}, a_{11}\rangle \in \boldsymbol{A}$. První část názvu sdružených akcí vždy reprezentuje dílčí akci prvního agenta a obdobně druhá část reprezentuje dílčí akci druhého agenta. První agent má tedy akce a_{00} a a_{01}. Druhý agent má dílčí akce a_{10} a a_{11}. Pro účely tohoto příkladu nejsou podstatné ostatní komponenty daného Dec-POMDP. Z tohoto důvodu je zde pro zjednodušení neubudu uvádět. Na obrázku 4.5 jsou zobrazeny čtyři stavy Quotient MDP daného Dec-POMDP pro n-tici 1-FSC. Jelikož se jedná o Quotient MDP s 1-FSC, tak každý z vytvořených stavů reprezentuje stejné nastavení paměti FSC obou agentů. Z tohoto důvodu s nastavení paměti FSC již nebudeme pracovat. Také jsou na obrázku pro zjednodušení uvedeny pouze jednotlivé stavy bez kontextu zbylého MDP, jelikož nám tyto informaci pro demonstraci nekonzistencí postačují. Na obrázku jsou tučně označeny sdružené akce vybrané konkrétní strategií reprezentované určitými FSC z daných rodin. Dále tedy budeme uvažovat nad konzistencí daných stavů vůči sobě za předpokladu zvolení dané strategie.*



Obrázek 4.5: Graf částí Quotient MDP vytvořeného pro Dec-POMDP za účelem popisu různých druhů konzistence

Nejdříve se věnujme dvojicí stavů, které jsou navzájem konzistentní a to konzistencí prvního i druhého typu. Jedná se o stavy $\langle s_0, z_{00}, z_{10}\rangle$ a $\langle s_1, z_{01}, z_{11}\rangle$. V těchto stavech je

zvolena jiná akce. A to akce $\langle a_{00}, a_{10}\rangle$ v dříve jmenovaném stavu a akce $\langle a_{01}, a_{11}\rangle$ ve stavu druhém. Dokonce i každý z agentů v těchto stavech provádí rozdílné dílčí akce. Jedná se ale o stavy jejichž pozorování jsou rozdílná. Tyto stavy tedy mezi sebou splňují nekonzistenci prvního typu. Každý z agentů zde má také rozdílné dílčí pozorování a tedy se jedná o stavy s konzistencí druhého typu.

Dále se podívejme na stavy $\langle s_0, z_{00}, z_{10}\rangle$ a $\langle s_1, z_{01}, z_{10}\rangle$. Tyto stavy také mají rozdílné sdružené pozorování. Z toho vyplývá, že zde platí konzistence prvního typu a to za zvolení libovolné sdružené akce. Oba tyto stavy symbolizují stejné dílčí pozorování druhého agenta. Jelikož je v těchto stavech zvolené rozdílná dílčí akce druhého agenta, je mezi těmito stavy nekonzistence druhého typu.

Na závěr zde uvedu příklad nekonzistence prvního i druhého typu. Jedná se o stavy $\langle s_0, z_{00}, z_{10}\rangle$ a $\langle s_1, z_{00}, z_{10}\rangle$. Tyto stavy mojí totožné sdružené pozorování, ale jejich zvolené sdružené akce jsou rozdílné. Z toho důvodu je mezi těmito stavy nekonzistence prvního typu. Jelikož nekonzistence druhého typu je pouze podmnožinou nekonzistence prvního typu, tak je mezi těmito stavy také nekonzistence druhého typu.

Všimněme si, že při uvažování o konzistenci daných stavů se vůbec neuvažovalo nad stavy daného Dec-POMDP ale pouze nad pozorováními a stavy paměti FSC. Je to z důvodu, že konzistence daných stavů se rozhoduje pouze na základě pozorování těchto stavů, stavů paměti a zvolených akcí.

4.5 Obarvení Quotient MDP

V předchozích sekcích byla vysvětlena tvorba Quotient MDP pro Dec-POMDP a n-tici rodin FSC. Poté byly popsány dva typy nekonzistence strategie pro daný Quotient MDP. V této sekci se důkladněji věnuji způsobu ověření, zda konkrétní strategie obsahuje nekonzistence prvního nebo druhého typu. K tomuto ověření ve své práci využívám tzv. obarvování (coloring) Quotient MDP. Během tohoto obarvování se dále popsaným způsobem označí veškeré akce daného Quotient MDP a tyto označení lze nadále používat k zjištění, zda konkrétní strategie obsahuje nekonzistence.

Základní částí obarvování Quotient MDP pro Dec-POMDP a n-tice rodin FSC jsou tzv. díry. Jedná se o návěští vlastností daného MDP. Pro Quotient MDP je vždy pro každého z agentů $d \in \mathcal{D}$ vytvořeno $2 * |\mathbf{O}^d| * |N^d|$ děr. Pro každého z agentů tedy existují dvě díry pro každou z kombinací jeho dílčích pozorování a stavů paměti jeho FSC. Nastavení první díry pro danou kombinaci dílčích pozorování a stavů paměti vždy reprezentuje volbu akce pro danou strategii, která bude provedena pokud se agent nalezne ve stavu s tímto pozorováním a jeho FSC bude v odpovídajícím stavu paměti. Tento typ děr budu dále označovat jako díry akcí. Nastavení druhé díry obdobně značí stav paměti, do kterého přejde FSC daného agenta po provedení dané akce v tomto pozorování a stavu paměti FSC. Tento typ děr dále označuji jako díry paměti. Každá z děr má svůj název ve tvaru $\langle d, z, n, t\rangle$, kde $d \in \mathcal{D}$ značí agenta, kterému tato akce přísluší, $z \in \mathbf{O}^d$ značí dílčí pozorování daného agenta, $n \in N^d$ značí stav paměti FSC daného agenta a parametr $t \in \{A, N\}$ je nastaven na A v případě kdy se jedná o díru akce a na N v případě díry paměti. V první fázi obarvování Quotient MDP se tedy vytvoří tyto díry. Pro každou z těchto děr se určí všechny možnosti jejího nastavení. Pro díry akcí to značí všechny akce, které daný agent v tomto pozorování může vykonat a pro díry pamětí to značí všechny stavy paměti do kterých FSC daného agenta při použití dané akce může přejít.

Po vytvoření těchto děr následuje fáze obarvení akcí. V této fázi je každé konkrétní akci daného Quotient MDP přiřazeno $2 * |\mathcal{D}|$ děr. Jedna díra akce a jedna díra paměti pro

každého z agentů. Dále se u každé z těchto děr určí, které nastavení odpovídá dané akci. Tedy která z možností nastavení dané díry musí být nastavena při zahrnutí dané akce do výsledné strategie. Nastavení děr akcí pro každou z konkrétních akcí se pro každého agenta určí na základě dílčí akce daného agenta, která je součástí této sdružené akce. Přiřazení nastavení děr paměti se určí na základě stavů, do kterých se po provedení akce přechází. To je umožněno díky tomu, že stavy do kterých se přechází po provedení určité akce reprezentují vždy jeden stejný stav paměti FSC pro každého agenta. Podle tohoto stavu se nastaví přiřazení volby dané díry paměti pro každého z agentů. Každá z děr může být využita pro označení více akcí.

Při vytvoření libovolné strategie pro dané obarvené Quotient MDP je možné ověřit konzistenci následujícím způsobem. Postupně se projdou všechny akce, které jsou součástí dané strategie. Pro každou z těchto akcí se u děr, které jim náleží, nastaví volby, které odpovídají daným akcím. Pokud je některé z děr přiřazeno více rozdílných nastavení, značí to nekonzistenci dané strategie. Tuto strategii tedy nelze reprezentovat pomocí sdruženého FSC pro jehož velikost bylo vytvořeno dané Quotient MDP.

Příklad 4.5.1 *Mějme Quotient MDP z příkladu 4.2.2 popsaný na obrázku 4.3. Nejdříve ukáži tvorbu děr akcí a děr pamětí pro stav* $\langle s_0, z_{00}, z_{10}, n_{00}, n_{10}\rangle$. *Jelikož se jedná o Quotient MDP pro Dec-POMD se dvěma agenty, budou zde vytvořeny celkem dvě díry akcí a dvě díry paměti. Pro prvního agenta* d_1 *se vytvoří díra akce* $\langle d_1, z_{00}, n_{00}, A\rangle$. *Parametr* d_1 *zde reprezentuje daného agenta, pro kterého je tato díra akce vytvořena.* z_{00} *značí dílčí pozorování daného agenta v tomto stavu. Obdobně parametr* n_{00} *značí stav paměti FSC daného agenta v téže stavu. Parametr A značí, že se jedná o díru akce. Tato díra akce může nabývat hodnot* a_{00} *a* a_{01} *a to z důvodu, že daný agent v tomto stavu může provést právě tyto dílčí akce. Obdobně se pro druhého agenta* d_2 *vytvoří díra akce* $\langle d_2, z_{10}, n_{10}, A\rangle$. *Tato díra akce může nabývat pouze hodnoty* a_{10}. *Pouze jedna možná hodnota nastavení je zde z důvodu existence pouze jedné akce, kterou daný agent může v tomto pozorování provést. Dále se pro agenta* d_1 *vytvoří díra paměti* $\langle d_1, z_{00}, n_{00}, N\rangle$ *s možnostmi nastavení* n_{00} *a* n_{01}. *Tyto hodnoty zde značí stavy paměti FSC daného agenta ve stavech, do kterých se po provedení této akce přechází. Tedy v tomto případě se v případě* n_{00} *jedná o stavy* $\langle s_1, z_{00}, z_{10}, n_{00}, n_{10}\rangle$ *a* $\langle s_1, z_{00}, z_{11}, n_{00}, n_{10}\rangle$. *Obdobně pro parametr* n_{01} *jde o stavy* $\langle s_1, z_{00}, z_{10}, n_{01}, n_{10}\rangle$ *a* $\langle s_1, z_{00}, z_{11}, n_{01}, n_{10}\rangle$. *Pro agenta* d_2 *se vytvoří díra paměti* $\langle d_2, z_{00}, n_{00}, N\rangle$ *opět pouze s jednou možností nastavení* n_{10} *z důvodu, že všechny stavy, do kterých lze z daného stavu přejít, mají stav paměti FSC daného agenta* n_{10}.

Dále se pro každou volbu akce v tomto stavu přiřadí konkrétní volby pro dané díry. Mějme např. akci $\langle a_{00}, a_{10}, n_{00}, n_{10}\rangle$. *Této akci přísluší volba parametru* a_{00} *pro díru* $\langle d_1, z_{00}, n_{00}, A\rangle$ *z důvodu, že je právě tato dílčí akce daného agenta obsažena v této sdružené akci. V díře paměti* $\langle d_1, z_{00}, n_{00}, N\rangle$ *pro tuto akci náleží hodnota* n_{00}, *jelikož je tento stav paměti FSC daného agenta obsažen ve stavech do kterých se po provedení této akce přechází. Obdobně jsou pro díry náležící agentovi* d_2 *přiřazeny* a_{10} *pro díru akce* $\langle d_2, z_{10}, n_{10}, A\rangle$ *a* n_{10} *pro díru paměti* $\langle d_2, z_{00}, n_{00}, N\rangle$.

Tímto způsobem jsou vytvořeny díry pro všechny ostatní kombinace pozorování a stavů paměti FSC ve stavech daného Quotient MDP. Můžeme si všimnout, že stavům $\langle s_0, z_{00}, z_{10}, n_{00}, n_{10}\rangle$ *a* $\langle s_0, z_{00}, z_{10}, n_{01}, n_{10}\rangle$ *náleží stejné díry pro agenta* d_2. *Díry náležící agentovi* d_1 *jsou ale rozdílné. Pokud by tedy zvolená strategie obsahovala např. akce* $\langle a_{00}, a_{10}, n_{00}, n_{10}\rangle$ *a* $\langle a_{01}, a_{10}, n_{00}, n_{10}\rangle$ *v těchto stavech, tak by tato strategie byla stále konzistentní. Tyto akce mají pro agenta* d_1 *odlišné dílčí akce a současně je v těchto stavech stejné dílčí pozorování tohoto agenta. Jelikož je v těchto stavech nastaven jiný stav paměti*

FSC daného agenta, tak tato strategie je popsatelná daným FSC a je tedy i konzistentní. Pokud by se tyto akce lišily i v dílčích akcích druhého agenta, tak by se jednalo o nekonzistenci druhého typu z důvodu rovnosti dílčích pozorování í stavů paměti těchto stavů.

4.6 Rozšíření preferující určitý typ nekonzistence

V rámci sekce 3.3 byla popsána metoda induktivní syntézy FSC pro POMDP. Tato metoda je založena na dělení rodin FSC na základě nekonzistencí v optimální strategii Quotient MDP. Tento proces dělení bude dále označován splitting Quotient MDP. Tato metoda je navržena pro induktivní syntézu FSC pro POMDP. Z tohoto důvodu vůbec nebere v potaz rozdíl mezi nekonzistencemi prvního a druhého typu. Za účelem zvýšení efektivity dané metody jsem se rozhodl implementovat rozšíření dané metody, umožňující preferenci výběru nekonzistentních akcí podle kterých je prováděn splitting Quotient MDP, na základě různého typu nekonzistence. Za tímto rozhodnutím byla myšlenka, že by preference dělení na základě nekonzistencí prvního typu mohla zefektivnit hledání optimálního řešení. Nekonzistence prvního typu značí nemožnost reprezentace dané strategie pomocí aktuální velikosti FSC stejně jako u POMDP. Naproti tomu nekonzistence druhého typu značí potřebu centralizace rozhodování k možnosti chování se podle dané strategie. Z tohoto důvodu byla hypotéza, že prioritizace dělení abstrakce rodiny sdružených FSC na základě nekonzistencí prvního typu více napodobí chování metody pro POMDP a tedy bude efektivnější. Dále následuje popis úpravy této metody.

V rámci tohoto rozšíření splittingu Quotient MDP se pro optimální strategii π^* vyhledají všechny nekonzistence tak jak tomu je u původní verze této metody. Pro každou z těchto nekonzistencí se vypočte skóre. Pro výpočet skóre se v každém stavu vyberou akce, které mezi sebou mají nekonzistenci prvního typu. Pro každou z těchto akcí se získá hodnota $ub((n, s))$ po provedení dané akce. Popis získání hodnoty $ub((n, s))$ je již popsán v 3.3. Následně se vypočte rozdíl maximální a minimální získané hodnoty. Tímto způsobem se získá hodnota reprezentující maximální možný vliv výběru různých akcí dané nekonzistence pro tento konkrétní stav. Ta se dále vynásobí pravděpodobností dosažení daného stavu při použití dané strategie π^*. Tento krok je proveden z důvodu přidání většího vlivu stavům, které jsou s větší pravděpodobností dosažitelná a tedy více ovlivní vlastnosti výsledné strategie. Pro každou z děr jejichž volby jsou v nekonzistenci prvního typu pro danou strategii se získá průměr těchto váhovaných hodnot. Ten nazýváme skóre dané nekonzistence. Toto skóre ukazuje jak moc výběr různých akcí dané díry ovlivní chování daného Dec-POMDP. Pokud zde není žádná akce s nekonzistencí prvního typu, je tato metoda provedena i se zahrnutím nekonzistencí druhého typu. Dělení abstrakce MDP je dále provedeno podle díry s největším skóre tak jak je popsáno v sekci 3.3. Obdobné rozšíření lze provést i pro preferenci nekonzistencí druhého typu.

Kapitola 5

Experimentální vyhodnocení

V této kapitole se věnuji experimentálnímu vyhodnocení metod navržených v kapitole 4. Nejdříve se tato kapitola zabývá podmínkami experimentálního vyhodnocení. Poté se probírá postup ověřování korektnosti implementace, nad kterou byly prováděny experimenty. Poté následuje porovnání induktivní syntézy kontrolérů pro Dec-POMDP se state-of-the-art metodami. Dále je popsáno vyhodnocení efektivity rozšíření dané implementace o preferenci dělení určitého typu nekonzistence při dělení Quotient MDP. Na závěr se věnuji demonstraci dané metody bez použití discount transformace.

Před provedením experimentů je nutné položit hlavní otázky, na které bychom rádi zjistili odpovědi prostřednictvím těchto experimentů, a to jsou:

1. Vrací implementace v rámci nástroje PAYNT korektní výsledky?
2. Jakých výsledků dosahuje daná metoda v porovnání se state-of-the-art metodami?
3. Jak ovlivní kvalitu a rychlost získání výsleků modifikace strategie splittingu Quotient MDP, tak aby prioritizoval různé druhy nekonzistence?
4. Je tato metoda vhodná i pro syntézu kontrolérů pro specifikace bez discount faktoru?

5.1 Podmínky experimentálního vyhodnocení

Experimenty jsou zaměřeny na rozšíření existujících metod induktivní syntézy FSC pro POMDP tak, aby podporovala práci s Dec-POMDP. Tato metoda byla implementována v rámci nástroje *PAYNT* [6], který využívá nástroj *STORM* [17]. Konkrétně tvorba Quotient MDP byla implementována v rámci nástroje *STORM*. Ta také zahrnuje možnost discount transformace. Obarvování Quotient MDP bylo implemetováno v rámci nástroje *PAYNT*. V rámci nástroje *PAYNT* jsem také implementoval rozšíření splittingu Quotient MDP, které umožňuje prioritizovat různé druhy nekonzistencí. Toto rozšíření je popsáno v sekci 4.6. V rámci experimentů byla použita metoda využívající iterativní zvyšovaní paměti prohledávaných FSC. Tato metoda je popsána v sekci 4.1. Implementace všech těchto rozšíření byla inspirována již existující implementací pro POMDP problémy. Všechny experimenty probíhaly na počítači s procesorem Intel Core i7-8550U, RAM velikosti 8GB a s operačním systémem Linux Ubuntu 20.04. Experimenty pro získání hodnot k induktivní syntéze byly provedeny nástrojem *PAYNT*.

Modely Pro porovnávání se state-of-the-art metodami byly zvoleny modely, které jsou standardními benchmarky pro práci s Dec-POMDP. Byly vybrány z literatury pro možnost porovnání s ostatními přístupy. Konkrétně se jedná o modely:

- *DecTiger* je model, ve kterém se dva agenti na základě svých pozorování rozhodují o tom které dveře otevřou. [27] Tomuto modelu jsem se více věnoval v příkladu 2.3.2. Model DecTiger má 2 stavy Dec-POMD a každý z agentů má 2 pozorování a 2 akce. Po transformaci na Dec-POMDP s deterministickým pozorováním má 12 stavů a 92 akcí.
- *BoxPushing* [20] je model, ve kterém dva agenti musí spolupracovat aby přesunuli objekt do cílového stavu. Tento model má 100 stavů Dec-POMDP a každý z agentů má 5 vlastních pozorování a 4 různé akce. Po transformaci na Dec-POMDP s deterministickým pozorováním má 103 stavů a 1618 akcí.
- *Recycling* [1] je model popisující úklid plechovek v kancelářské budově pomocí dvou decentralizovaných robotů. Tito roboti si mohou samostatně sbírat malé plechovky, dobíjet si vlastní baterie nebo společně uklízet velké plechovky. Tento model má 4 stavy Dec-POMDP a každý z agentů má 2 dílčí pozorování a 3 akce. Po transformaci na Dec-POMDP s deterministickým pozorováním má 7 stavů a 47 akcí.
- *Grid3x3* [4] modeluje dva agenty, kteří se snaží setkat na poli o velikosti 3x3. Každý z agentů má k dispozici akce k pohybu doprava, doleva, nahoru a dolů. Dále má k dispozici akci k zůstání na stejném místě. Při každém pohybu je vždy malá pravděpodobnost nedosažení cíleného stavu ale některého z vedlejších stavů. Tento model má 81 stavů Dec-POMDP a každý z agentů má 9 dílčích pozorování a 5 akcí. Po transformaci na Dec-POMDP s deterministickým pozorováním má 84 stavů a 2052 akcí.

	|S|	|Z|	|A|	deterministic |S|	deterministic |A|
DecTiger	2	2	2	12	92
Recycling	4	2	3	7	47
Grid3x3	81	9	5	84	2052
BoxPushing	100	5	4	103	1618
Circle	9	1	2	18	69
Grid8x8	4096	2	4	9224	590096

Tabulka 5.1: Tabulka shrnující informace o modelech použitých k experimentálnímu vyhodnocení. |S| značí počet stavů Dec-POMDP daného modelu. Sloupec |Z| obsahuje počet dílčích pozorování každého z agentů a |A| značí počet dílčích akcí každého z agentů. Sloupce deterministic |S| s deterministic |A| značí počety stavů a akcí daného Dec-POMDP po transformaci na Dec-POMPD s deterministickým pozorováním.

Informace o daných modelech jsou shrnuty v tabulce 5.1. Z důvodu umožnění porovnávání s ostatními nástroji pro syntézu strategií Dec-POMDP byla pro každý z modelů provedena discount transformace s discount faktorem $\gamma = 0.9$. Pro práci bez discount faktorizace byly použity modely *Circle* a *Grid8x8* [13]. Tyto modely jsou lépe popsány v sekci 5.5.

5.2 Ověření korektnosti výsledků

Po dokončení implementace rozšíření umožňující induktivní syntézu FSC pro Dec-POMDP do nástroje PAYNT bylo nejdříve potřeba ověřit korektnost získaných řešení a zda výsledné hodnoty odpovídají nalezeným strategiím. Srovnání výsledků s ostatními přístupy není možné, jelikož jsou získané strategie reprezentovány různými způsoby a současně dosahují různé kvality. Porovnání získaných hodnot s výsledky ostatních přístupů tedy není vhodnou metodou pro ověření korektnosti programu. Nejdříve byla daná metoda ověřena na velice jednoduchých příkladech, u kterých bylo možné získat optimální řešení pouze pomocí vlastního výpočtu na papíře. Přidáním paměti do FSC a prací s komplexnějšími modely Dec-POMDP se ale velmi rychle zvyšuje složitost daných řešení. Z toho důvodu byl tento způsob ověření korektnosti řešení značně nedostačující. Z tohoto důvodu bylo zapotřebí ověřit korektnost dané implementace ještě dalším způsobem.

Toto ověření proběhlo srovnáním výsledků s výsledky induktivní syntézy kontrolérů pro POMDP. Jelikož daná implementace je součástí nástroje PAYNT, který zvládá induktivní syntézu FSC pro POMDP, tak je možnost nastavit stejné podmínky pro hledání řešení těchto FSC. Pro hledání FSC pro POMDP je také k dispozici spoustu modelů ve tvaru cassandra, se kterými se pracuje také při hledání řešení pro Dec-POMDP modely. Pro ověření korektnosti implementace byly tedy vytvořeny Dec-POMDP modely, které reprezentují stejné prostředí jako již existující POMDP modely. Pro tyto sobě odpovídající modely byla spuštěna induktivní syntéza a kontrolovalo se, zda jsou získaná řešení identické.

Dec-POMDP s odpovídajícím chování určitému POMDP byly tvořeny následovně. Pro dané POMDP bylo vytvořeno Dec-POMDP s dvěma agenty a identickými stavy jako mělo původní POMDP. První agent přesně simuluje chování daného POMDP. Jeho dílčí pozorovaní odpovídají pozorováním POMDP. Obdobně i jeho dílčí akce jsou totožné s akcemi původního POMDP. Druhý agent má vždy pouze jedno dílčí pozorování a jednu dílčí akci. V každém stavu tedy tento agent vždy zvolí totožnou akci. Jeho chování z tohoto důvodu nemá vliv na vlastnosti daného modelu. Díky faktu, že každý z agentů vidí pouze své vlastní pozorování a stav paměti FSC, tak chování prvního agenta není ovlivněno ani přidáním paměti do FSC druhého agenta. Z toho důvodu se tyto modely Dec-POMDP a POMDP chovají totožně i při práci s k-FSC kde $k > 1$. Z tohoto důvodu je možné porovnáním výsledků pro tyto Dec-POMDP a POMDP možné ověřit korektnost řešení dané implementace.

Experimenty v této sekci byly provedeny na POMDP modelech převzatých z [10]. Konkrétně byly použity modely *1d*,*mini-hall2* a *tiger95*. Tyto modely byly následně převedeny na Dec-POMDP s odpovídajícím chováním. Pro tyto modely byly následně hledány optimální strategie pomocí nástroje PAYNT. V rámci všech experimentů byla maximalizována očekávaná hodnota rewardu. Vždy byla fixně zvolena velikost FSC a pro ni byly prohledány všechny řešení. Výsledky jsou zobrazeny v tabulce 5.2. Z provedených experimentů je patrné, že nalezená řešení pomocí nové metody pro Dec-POMDP jsou identické s výsledky získanými pomocí již existující metody pro POMDP. Časy prohledání rodin FSC jsou ale výrazně delší. To je zapříčiněno rozšířením rodiny FSC o paměť druhého agenta a tedy výraznému zvětšení prohledávaných FSC. Současně přidání faktoru paměti druhého agenta výrazně zvětšuje použité Quotient MDP.

POMDP						
	1d		**mini-hall2**		**tiger95**	
	čas	rew	čas	rew	čas	rew
1-FSC	0.0 s	0.953599	0.02 s	2.558137	0.02 s	-19.992882
2-FSC	0.04 s	1.123356	43.13 s	2.686917	0.42 s	-19.992882
3-FSC	1.05 s	1.260094	-	-	20.6 s	-10.213642

Dec-POMDP						
	1d		**mini-hall2**		**tiger95**	
	čas	rew	čas	rew	čas	rew
1-FSC	0.0 s	0.953599	0.03 s	2.558137	0.01 s	-19.992882
2-FSC	0.08 s	1.123356	150.14 s	2.686917	1.63 s	-19.992882
3-FSC	87.1 s	1.260094	-	-	1409 s	-10.213642

Tabulka 5.2: Tabulky srovnávající výsledky induktivní syntézy pro POMDP a odpovídající Dec-POMDP modely. Pro strategie reprezentované pomocí různých velikostí FSC jsou zde uvedeny očekávané hodnoty rewardu nalezených výsledků. Dále jsou zde uvedeny časy nalezení potřebné k prohledání daných rodin FSC. Pole označená - značí, že se pro danou velikost FSC do dvou hodin nestihlo prohledat všechna řešení.

5.3 Srovnání se state-of-the-art metodami

V této části se zabývám porovnáním indukční syntézy konečně stavových kontrolérů pro Dec-POMDP s ostatními state-of-the-art přístupy. Tyto přístupy pro řešení Dec-POMDP problémů v nekonečném horizontu umí pracovat pouze s discount faktorem $\gamma < 1$. Induktivní přístup zvládne řešit Dec-POMDP problémy i s discout faktorem $\gamma = 1$. V této sekci se ale budeme zabývat porovnáváním s ostatními nástroji a tedy je nutné pro porovnání nejdříve provést discount transformaci daných modelů. Pro všechny experimenty v této kapitole byl nastaven discount faktor $\gamma = 0.9$.

Výsledky porovnávání jsou popsány v tabulkách 5.3. Každá z tabulek ukazuje porovnání výsledků syntézy strategií pro určitý model pomocí různých metod. Pro každou z metod je zde uveden čas, kdy byl nalezen výsledek a také hodnota očekávaného rewardu nalezené strategie. U metod reprezentujících strategii pomocí sdruženého FSC je zde uvedena i velikost výsledného kontroléru. Cílem všech experimentů bylo nalézt strategii maximalizující hodnotu získaného rewardu. Do porovnávání jsou mimo nástroj PAYNT zahrnuty metody MealyNLP [3] využívající k nalezení optimální strategie nelineární programování, FB-HSVI (*feature-based heuristic search value iteration*) [13] a Peri [33], využívající periodické FSC které jsou více popsané v 3.2.2. Také jsou v tabulce hodnoty pro metody IJ(M-D) a IJ(M-S) [39]. Jedná se o iterativní JESP metody 3.2.1, kde M-D a M-S jsou různé druhy inicializace prohledávání. Hodnoty velikosti výsledných FSC, času nalezení výsledku a hodnoty jeho očekávaného rewardu pro ostatní metody jsou převzaty z [39, 3]. Tyto hodnoty jsou tedy získány z experimentů na jiném hardware, než experimenty provedené v rámci této práce. To by však mělo ovlivnit pouze čas potřebný k získání daných hodnot. Kvalita získaných výsledku by tím neměla být výrazně ovlivněna.

Všechny tyto experimenty hledaly sdružené FSC maximalizující očekávanou hodnotu rewardu. Pro model recycling byla pomocí nástroje PAYNT nalezena nejlepší strategie již po třech sekundách. Tato strategie je reprezentována pomocí 2-FSC a její hodnota očekávaného rewardu je 31,911. Během zbylých dvou hodin byly prohledávány řešení reprezentovány

Recycling			
	|N|	čas	ExpRew
FB-HSVI	-	2.6 s	31.929
Peri	6×30	77 s	31.84
IJ(M-D)	$4, 4$	0 s	25.65
IJ(M-S)	$8, 8$	0 s	26.57
MealyNLP	-	0 s	31.928
PAYNT	$2, 2$	3.0 s	31.911

Boxpushing			
	|N|	čas	ExpRew
FB-HSVI	-	1715.1 s	224.43
Peri	15×30	5675 s	148.65
IJ(M-D)	$274, 342$	1436 s	203.41
IJ(M-S)	$250, 408$	963 s	220.25
MealyNLP	-	774 s	143
PAYNT	$2, 2$	1940 s	224.629

Dectiger			
	|N|	čas	ExpRew
FB-HSVI	-	153.7 s	13.448
Peri	10×30	220 s	13.45
IJ(M-D)	$6, 6$	201 s	13.44
IJ(M-S)	$6, 6$	213 s	13.44
MealyNLP	-	29 s	-1.49
PAYNT	$3, 3$	996 s	-18.31

Grid3x3			
	|N|	čas	ExpRew
FB-HSVI	-	67 s	5.802
Peri	-	9714 s	4.64
IJ(M-D)	8&10	2 s	5.81
IJ(M-S)	9&17	9 s	5.81
MealyNLP	-	-	-
PAYNT	$1, 1$	6.33 s	5.81903

Tabulka 5.3: Tabulky výsledků porovnávání induktivní syntézy v rámci programu PAYNT s ostatními state-of-the-art metodami. První sloupec značí metodu, která byla použita k získání daných výsledků. Hodnoty ve sloupci |N| značí velikost finálního FSC pro každého z agentů. Pro metodu *Peri* značí první hodnota velikost FSC a druhá počet vrstev. Tyto velikosti jsou identické pro oba agenty. Hodnoty ve třetím sloupci značí čas, který trvalo získat nejlepší výsledek získaný pomocí dané metody a hodnoty čtvrtého sloupce značí nejlepší výsledek získaný pomocí dané metody. Všechny experimenty maximalizovali očekávanou hodnotu rewardu získaného řešení.

pomocí sdružených 3-FSC a 4-FSC. Žádná z těchto strategií ale svou hodnotou rewardu nepřekonala danou hodnotu. U modelu Grid3x3 bylo nejlepší řešení nalezeno v čase 6,33 sekundy již pomocí reprezentace sdruženého 1-FSC. Vzhledem k velikosti daného modelu se během dvou hodin běhu programu nepodařilo dostat ani k hledání řešení reprezentovaných pomocí 3-FSC. Pro model Boxpushing bylo optimální řešení reprezentování pomocí sdruženého 2-FSC. Nalézt toto řešení trvalo přibližně půl hodiny. Během běhu programu se podařilo dojít pouze k prohledávání strategií reprezentovaných pomocí sdružených 3-FSC. Pro model DecTiger se nejlepší řešení nalezlo po 16ti minutách a je reprezentované pomocí sdruženého 3-FSC. Hodnota očekávaného rewardu tohoto řešení je pouze -18,31, což je mnohem méně než u ostatních metod. Takto nedostatečné výsledky u tohoto modelu přisuzuji zejména nedostatečnému množství paměti FSC v určitých stavech pro kontrolér umožňující reprezentaci kvalitnější strategie. Naopak u ostatních modelů dokázala metoda induktivní syntézy nalézt podobně kvalitní výsledky jako ostatní state-of-the-art metody.

Těchto výsledků metoda dosahuje i se strategiemi, které jsou reprezentovány pomocí k-FSC s malou velikostí paměti. Největší velikost k-FSC pro které se za dvě hodiny stačilo dojít k prohledávání strategií je 4-FSC. Ke větším velikostem paměti FSC nebylo dosaženo zejména z důvodu exponenciálního růstu stavů Quotient MDP při zvyšování paměti FSC. Musíme si uvědomit, že každý z agentů má vlastní FSC a velikost paměti značí velikost paměti každého dílčího FSC. Pro vytvoření dvojice kompatibilních FSC se musí pracovat s každou kombinací pamětí jednotlivých FSC. Z tohoto důvodu složitost nalezení strategie

pro dva agenty Dec-POMDP s velikostí $k = 4$ přibližně odpovídá složitosti nalezení strategie jednoho POMDP s velikostí $k = 16$. Z tohoto důvodu nebyla možnost prozkoumat komplexnější strategie.

Z tohoto porovnávání vyplývá, že použití induktivní syntézy FSC pro Dec-POMDP dokáže kvalitou získaných strategií konkurovat state-of-the-art přístupům pouze pokud k reprezentaci daných strategií stačí malé kontroléry. Jde to vidět například u problému *Boxpushing*, kde PAYNT dosahuje nejlepšího výsledku. K tomuto výsledku stačí pouze kontrolér o velikosti 2-FSC. Oproti tomu kontroléry metody JESP IJ(M-S) nejsou schopny nalézt takto kvalitní výsledek pomocí kontrolérů velikosti 250 pro prvního z agentů a 408 pro druhého agenta. Naopak u problémů, kde je k zapotřebí větších kontrolérů tato metoda není schopná nalézt vhodnou strategii. Z důvodu vysoké náročnosti nalezení strategií reprezentovaných většími kontroléry. Příkladem tomu je problém *DecTiger*, kde PAYNT dosahuje výrazně horších výsledků než ostatní metody. **Z výsledků vyplývá, že induktivní syntéza dokáže vytvořit pouze malé kontroléry. Přesto tyto malé kontroléry dokáží dosahovat obdobně kvalitních výsledků jako state-of-the-art-metody.** Díky iterativnímu zvyšování velikostí FSC je induktivní syntéza schopna nalézt ty nejmenší kontroléry s danými vlastnostmi.

V porovnání s induktivní syntézou kontrolérů pro POMDP je syntéza kontrolérů pro Dec-POMDP schopna pracovat s výrazně menšími kontroléry. To je způsobeno závislostí velikosti Quotient MDP na velikosti kontrolérů každého agenta. Velikost Quotient MDP roste oproti velikosti Quotient MDP pro POMDP exponenciálně rychleji a to způsobuje náročnost práce s většími kontroléry. V následující kapitole se věnuji pokusu o zrychlení vyhledávání optimálního řešení pomocí úpravy heuristiky pro splitting Quotient MDP.

5.4 Experimenty s preferencí určitého typu nekonzistence

Tato sekce se věnuje vyhodnocení efektivity prioritizace různých typů nekonzistence při splittingu Quotient MDP. Hypotéza byla, že prioritizací dělení na základě nekonzistence prvního typu tato metoda více napodobí vlastnosti původní metody pro POMDP. Toto využití různého charakteru jednotlivých typů nekonzistence mělo za účel zvýšit efektivitu syntézy kontrolérů.

Induktivní syntéza využívající splitting Quotient MDP preferující dělení na základě nekonzistencí prvního i druhého typu byla provedena nad modely *Recycling*, *DecTyger*, *BoxPushing* a *Grid3x3*. Výsledky těchto experimentů jsou v tabulce 5.4.

	Recycling	DecTyger	Boxpushing	Grid3x3
výchozí přístup	3 s	966 s	1940 s	6.33 s
preference nekonzistence 1. typu	3 s	1038 s	1982 s	5.81 s
preference nekonzistence 2. typu	3 s	1009 s	2000 s	5.77 s

Tabulka 5.4: Tabulka ukazující čas trvající nalezení optima pomocí metody výchozího přístupu pro splitting Quotient MDP a jeho modifikací preferujících dělení na základě nekonzistence 1. typu nebo nekonzistence 2. typu. Nalezené optimální hodnoty byly pro každou z metod identické.

Z výsledků daného experimentu lze vidět, že změna strategie výběru volby akcí na základě kterých je prováděn splitting Quotient MDP nemá na rychlost a kvalitu nalezených strategií výrazný vliv. U většiny modelů dokonce metoda bez dané úpravy vrací optimální

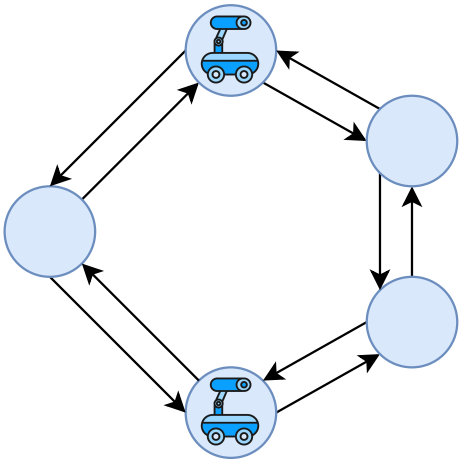
strategii v nejrychlejším čase. Největší časový rozdíl je u modelu DecTyger, kde je původní metoda rychlejší než metoda preferující dělení na základě nekonzistencí 1. typu až o 7% času. U modelu Boxpushing je původní metoda rychlejší až o 3% času. Naopak u modelu Grid3x3 metoda preferující nekonzistence 2. typu provedla zrychlení od původní metody přibližně o 9%. Tyto časové odchylky u různých verzí dané metody jsou způsobeny různou volbou nekozistencí pro dělení. Z toho důvodu jsou malé náhodné rozdíly mezi výsledky různých metod opodstatněny. Bohužel ale žádné z těchto rozšíření nezpůsobilo výrazné zrychlení, které bylo původně očekáváno. **Prioritizace různého typu nekonzistence pro slitting Quotient MDP tedy nemá na rychlost nalezení strategií a jejich kvalitu výrazný vliv.**

5.5 Demonstrace funkcionality bez discount transformace

V předchozí sekci byly ukázány výsledky experimentů porovnávající řešení Dec-POMDP pomocí induktivní syntézy se state-of-the-art přístupy. Všechny tyto metody pracují s discount faktorem $\gamma < 1$ a tedy i srovnání byla provedena nad modely s discount transformací. Řešení pomocí induktivní syntézy na rozdíl od ostatních přístupů dokáže pracovat s discount faktorem $\gamma = 1$. Tím pádem dokáže pracovat v nekonečném horizontu. Tato možnost práce v nekonečném horizontu je velice důležitá při práci s modely, kde se důležité rozhodnutí provede až v pozdější fázi rozhodování. Problémem práce s discount faktorem je, že se bere větší důraz na rozhodnutí, která jsou provedena dříve než rozhodnutí provedena později. Pro braní většího důrazu na pozdější rozhodnutí je v takovém případě potřeba zvýšit hodnotu discount faktoru. Čím větší je hodnota discount faktoru tím větší důraz je kladen na pozdější rozhodnutí, ale současně je tím navýšena složitost výpočtu. Při hledání strategie pro model, kde je potřeba brát větší důraz na pozdější rozhodnutí, je nutné správně nastavit hodnotu discount faktoru. Příliš malá hodnota discount faktoru by způsobila příliš malý důraz na důležité rozhodnutí provedené v pozdější fázi. Na druhou stranu příliš velké hodnota discount faktoru způsobí navýšení složitosti výpočtu optimální strategie. Metoda představená v této práci tento problém nemá jelikož dokáže pracovat i bez discount transformace a tedy i pro specifikace v nekonečném horizontu. V této sekci ukáži dva příklady, na kterých je demonstrována induktivní syntézy konečně stavových kontrolérů pro Dec-POMDP discount faktorem $\gamma = 1$.

5.5.1 Problém Circle

Prvním modelem, který jsem vytvořil pro demonstraci řešení Dec-POMDP problému s discount faktorem $\gamma = 1$, je model *Circle*. Jedná se o analogii známého problému *Grid* [4]. Model *Circle* popisuje chování dvou agentů na kruhové cestě. Ta se skládá z pěti polí. Tento problém je ilustrován na obrázku 5.1. Cílem těchto agentů je se za co nejkratší dobu potkat na stejném poli. Na začátku jsou oba agenti náhodně rozmístěni na odlišná pole. Každý z agentů je postaven v určitém směru. Buďto je postaven, tak že se pochodem dopředu pohybuje po směru hodinových ručiček a nebo naopak. Oba agenti ale neví, na kterém z polí se nachází. Také agenti neví, ve kterém směru jsou postaveni. Každý z agentů má možnost použití dvou akcí *front* a *back*. Akce *front* agenta s pravděpodobností 0.95 posune o jedno pole dopředu ve směru, ve kterém je postaven a s pravděpodobností 0.05 ho posune o jedno pole ve směru opačném. Celkem tedy existují 4 sdružené akce, kterými jsou $\langle front, back\rangle$, $\langle front, front\rangle$,$\langle back, front\rangle$ a $\langle back, back\rangle$, kde první část vždy představuje dílčí akci náležící prvnímu agentovi a druhá část představuje dílčí akci druhého agenta.



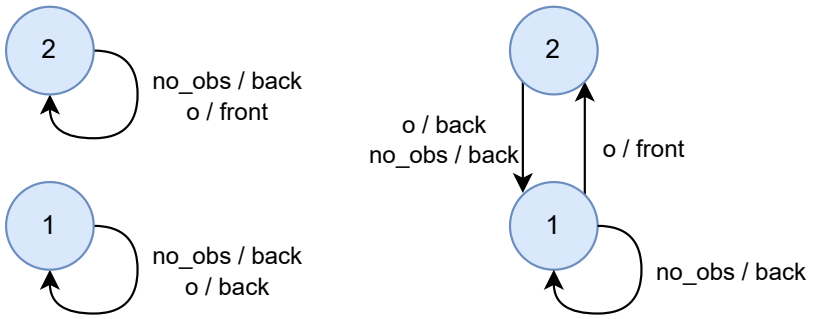
Obrázek 5.1: Obrázek ilustrující model *Circle*. Model má 5 polí, které jsou propojené do tvaru kruhu. Agenti se mohou pohybovat vždy pouze do vedlejších polí. Cílem agentů je nacházet se současně na stejném poli. Obrázky robotů jsou převzaty z *www.freepik.com*

Všimněme si, že pro modelování vzájemného hledání dvou agentů na této kruhové trati není potřeba brát v potaz konkrétní polohu agentů. Pro tyto účely postačí přesně modelovat vzájemnou polohu těchto agentů a jejich směr. Díky tomuto zjednodušení se velice zmenší počet stavů daného Dec-POMDP. Daný Dec-POMDP, který jsem vytvořil pro modelování daného problému, má tedy pouze 9 stavů $S = \{s_1, s_2, s_3, s_4, d_1, d_2, d_3, d_4, end\}$. Název těchto stavů se skládá z písmene a jeho indexu. Písmeno značí ve kterém směru jsou agenti vůči sobě postaveni. Stavy označeny písmenem s značí situace, kdy jsou agenti postaveni ve stejném směru. Stavy označeny písmenem d značí naopak situace, kdy jsou agenti postaveni ve směru opačném. Indexy stavů značí vzájemnou polohu agentů. Číslo indexu konkrétně značí o kolik polí v určitém směru je vzdálen druhý agent od agenta prvního. Počáteční stav je vybrán náhodně ze všech stavů až na stav *end*. Pravděpodobnost nastavení každého z těchto stavů jako počátečního stavu je nastavena na 0.125. To znamená že na počátku agenti neví v jaké vzájemné pozici se nacházejí a v jakém směru jsou postaveni. Jelikož oba agenti nemají žádné informace o tom, ve kterém stavu se nachází, tak po použití libovolné sdružené akce v každém stavu agenti získají vždy pouze jedno samé sdružené pozorování $\mathbf{O} = \langle o, o \rangle$. Pravidla změn stavů po použití konkrétních sdružených akcí jsou popsány v přechodové funkci $\mathcal{T}$, která je konkrétně popsána v tabulce 1. Cílem řešení tohoto problému, je najít takové strategie pro oba agenty, aby se minimalizoval očekávaný počet kroků potřebných k jejich setkání. Z tohoto důvodu je hodnota rewardu pro každou akci nastavena na 1 a cílem je najít strategii minimalizující očekávanou hodnotu rewardu.

Před prezentací výsledků syntézy kontrolérů pro daný problém je důležité připomenout, že v počátečním stavu agenti nemají k dispozici žádné pozorování. V rámci implementace v nástroji PAYNT tedy počátečnímu stavu přidáváme pozorování *no_obs*. Z tohoto důvodu i $1 - FSC$ dokáží reprezentovat i strategie zahrnující použití jiné akce v počátečním stavu, než ve stavech zbývajících. Toho dokáží i přes skutečnost, že v původním Dec-POMDP mají všechny stavy stejné sdružené pozorování.

Před uskutečněním syntézy FSC pro daný problém jsem uvažoval nad ideální strategií pro tento problém. Podle mého uvažování nad tímto problémem byla ideální kombinace strategií agentů taková, že jeden z agentů neustále provádí stejnou akci, tedy akci *front* nebo akci *back*. Druhý z agentů naopak střídá akci každé kolo. To znamená že by druhý

agent například každé liché kolo použil akci *front* a každé sudé kolo provedl akci *back*. Díky této kombinací strategií se zaručí setkání daných agentů, jelikož jeden z agentů bude neustále obcházet danou cestu zatím co druhý agent bude zastaven na dvou stavech, mezi kterými se bude pohybovat. Tato strategie funguje pro všechny možné počáteční nastavení. To znamená, že zaručí rychlé setkání agentů za podmínek kdy mají nastavený stejný směr cesty ale i pokud mají nastavený rozdílný směr a to i pro všechny možnosti jejich počátečních polí.



Obrázek 5.2: Grafy 2-FSC pro model *Circle*. Vlevo je FSC agenta d_1 a napravo FSC agenta d_2

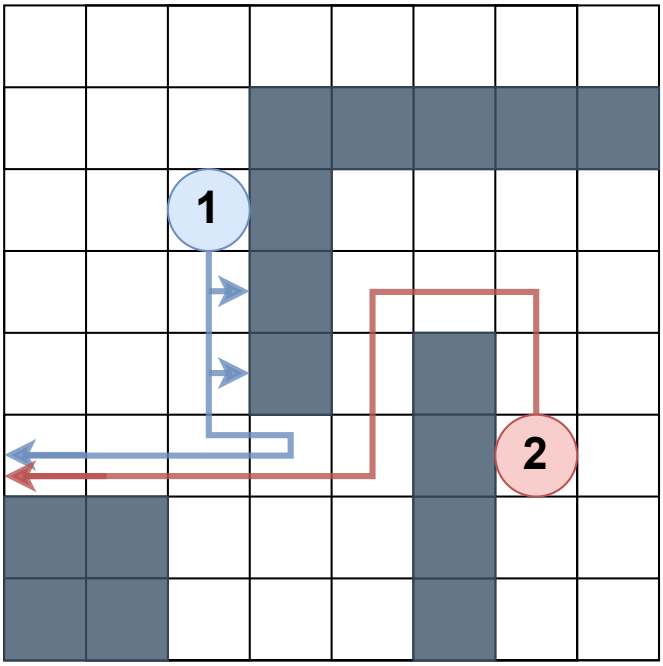
Nejdříve jsem pro model *Circle* provedl syntézu 1-FSC za účelem minimalizovat očekávanou hodnotu rewardu do dosažení cílového stavu *end*, tedy za účelem minimalizování očekávávaného počtu kroků obou agentů do stavu, kdy se oba agenti nacházejí na stejném poli. Výsledkem této syntézy jsou dvě 1-FSC, kde strategií prvního agenta je provádění akce *front* v každém stavu. To znamená, že tato strategie odpovídá mému předchozímu odhadu. U druhého agenta ale provedené mé odhadované strategie není pouze s 1-FSC možné. Výsledné FSC pro daného agenta reprezentuje strategii, kde v počátečním stavu agent provede akci *back* a nadále neustále prování akce *front*. Očekávaná hodnota rewardu je pro tuto strategii je rovna 23.36.

Poté jsem pro daný model provedl syntézu 2-FSC kontrolérů. Opět za účelem nalezení strategií minimalizujících počet kroků do setkání agentů na stejném poli. Tentokrát strategie prvního agenta zahrnovala neustálé provádění akce *back*. Druhý agent v počátečním pozorování *no_obs* provádí akci *front* a poté v pozorování *o* ve stavu paměti 1 provádí akci *back* a ve stavu paměti 2 provádí akci *front*. Po provedení těchto akcí v pozorování *o* a stavu paměti 1 se vždy změní stav pozorování 2. Obdobně po provedení akce v dílčím pozorování *o* a stavu paměti 2 se změní stav paměti na 1. Tato strategie tedy plně odpovídá mému původnímu odhadu ideální strategie. Očekávaná hodnota rewardu je pro tuto strategii je rovna 5.034. FSC obou agentů získané syntézou 2-FSC kontrolérů jsou zobrazeny na obrázku 5.2. Můžeme si všimnout že u prvního agenta se vůbec nepřejde do stavu paměti 2. Tento agent vůbec nepotřebuje využít tohoto stavu jelikož je jeho strategie reprezentovatelná i pomocí 1-FSC. Naopak FSC druhého agenta již druhý stav paměti využívá.

5.5.2 Problém Grid8x8

Druhý model, na kterém demonstruji použití induktivní syntézy je *Grid8x8* [13]. Jedná se opět o adaptaci problému *Grid* [4]. Tentokrát je model ale mnohem větší a komplexnější. Cílem dvou agentů je zde co nejrychlejší setkání v poli o velikosti $8x8$. Na tomto poli jsou umístěny překážky, přes které se agenti nedokáží pohybovat. Umístění daných překážek a

počáteční umístění agentů je popsáno na obrázku 5.3. Agent 1 podle svého pozorování dokáže rozpoznat pouze to, zda se napravo od něj nachází překážka. Agent 2 naopak dokáže rozeznat, zda se překážka nachází nalevo od něj. Oba agenti se pomocí akcí dokáží pohybovat o jedno políčko doprava, doleva, nahoru nebo dolů. Při provedení každé akce je 10% pravděpodobnost, že agent uklouzne a zůstane na stejném místě. Každý krok agentů do společného setkání má hodnotu rewardu 1. Cílem problému je minimalizovat očekávanou hodnotu rewardu.



Obrázek 5.3: Ilustrace problému grid8x8 adaptovaná z 5.3. Každému z agentů je znázorněna optimální strategie nalezená pomocí induktivní syntézy FSC.

Model *Grid8x8* má 4096 stavů Dec-POMDP a každý z agentů má 2 pozorování a 4 dílčí akce. Po převedení na Dec-POMDP s deterministickým pozorováním má daný model 9224 stavů a 590096 akcí. Rodina 2-FSC pro daný problém obsahuje přibližně 1099 miliard kontrolérů.

Pro daný problém se v rámci experimentu hledal kontrolér o velikosti k=2. Nalezení optimálního kontroléru dané velikosti trvalo pouhých 941 sekund. Následné prohledání všech kontrolérů dané rodiny FSC trvalo 4098 sekund. Získaná strategie je komplexní a na první pohled neintuitivní. Před provedením syntézy jsem nejdříve uvažoval nad strategiemi, kde se agenti budou pohybovat naproti sobě a po obejití překážek se potkají někde na půli cesty. Tato strategie ale selhává na možnosti uklouznutí některého z agentů a výsledné minutí. Druhý typ strategií, nad kterým jsem uvažoval, bylo doputování obou agentů do stejného stavu a zde následně počkat dokud nedorazí druhý agent. Tato strategie již zaručuje setkání obou agentů. Současně ale neminimalizuje počet provedených kroků do shledání agentů. Strategie vytvořená pomocí induktivní syntézy kombinuje tyto oba prvky. Agent 2 se co nejrychleji snaží dostat na společné místo, kde čeká agenta 1. Ten nejdříve zpomaluje svůj

postup pomocí narážení do překážek. Poté se vydá jedno pole naproti agentovi 2. Pokud by se v tomto bodě nesetkali, tak se agent přesune na finální místo, kde počká na druhého agenta. Tato kombinace tedy překonala mé vlastní pokusy o vytvoření optimální strategie. Očekávaný počet kroků při této strategii je 9.77. Tato společná strategie je znázorněna na obrázku 5.3.

Tento experiment ukazuje funkčnost aplikace dané metody na vlastnosti bez discount faktorizace. Současně se zde demonstruje aplikovatelnost induktivní syntézy FSC i na rozsáhlé modely jako je například model *Grid8x8*.

Kapitola 6

Závěr

Tato práce se zabývá řešením Dec-POMPD problémů pro specifikace v nekonečném horizontu. Tento problém se zde řeší pomocí rozšíření existujícího přístupu induktivní syntézy pro problémy POMDP tak, aby jej bylo možné použít pro řešení Dec-POMDP problémů. Získané strategie jsou zde reprezentovány pomocí konečně stavových kontrolérů. Konkrétně se tato práce zabývá rozšířením metody induktivní syntézy konečně stavových kontrolérů pro POMDP. Klíčovým objektem rozšíření dané metody je tvorba Quotient MDP a obarvení Quotient MDP. Tyto procesy byly modifikovány tak, aby je bylo možné aplikovat na práci s Dec-POMDP modely. Následně byla tato rozšíření implementována v rámci nástroje *PAYNT*. Tato nově získaná metoda pro řešení Dec-POMDP problémů na rozdíl od již existujících metod dokáže pracovat v nekonečném horizontu. Ostatní metody dokáží pracovat pouze v konečném horizontu nebo s discount faktorem. Z tohoto důvodu bylo potřeba pro porovnání kvality výsledků se state-of-the-art metodami provést discount transformaci daných modelů, tak aby jejich chování v nekonečném horizontu odpovídala práci s discount faktorem. V tomto srovnání nová metoda u většiny z modelů dosahovala výsledků obdobné kvality state-of-the-art přístupy. U modelů, kde je v některých stavech potřeba využít většího množství paměti, ale dosahovala výsledků výrazně horších. Problémem této metody je rychlé navýšení náročnosti výpočtu při přidání paměti hledaným FSC. V rámci práce se tento problém snaží zmírnit pomocí změny strategie splittingu Quotient MDP, tak aby lépe pracovala s vlastnostmi Dec-POMDP. Tato úprava ale efektivitu nezvýšila. I přes omezení velikosti kontrolérů, které je tato metoda schopná nalézt, jsou tyto kontroléry svou kvalitou schopny konkurovat state-of-the-art přístupům. Díky iterativnímu zvyšování velikosti hledaných FSC jsou získané kontroléry těmi nejmenšími pro dané strategie. Následně bylo demonstrováno využití dané metody při práci v nekonečném horizontu bez využití discount faktoru.

Do budoucna je možné se zaměřit na metody umožňující práci s FSC s větší pamětí. Jednou z těchto metod je například práce s redukovanými FSC. Ty umožňují přidávat paměť jen pro některé stavy a tím výrazně zmenšit prohledávanou rodinu FSC. Druhou možností jak umožnit prohledávání FSC s větší pamětí je prohledávání pouze části rodiny FSC. Takovým přístupem je například využití získaných informací o optimálním řešení nalezeném v předchozí iteraci prohledávání. Na základě těchto informací se může po přidání paměti hledat řešení pouze určité množině možných řešení. Jednou z možností této metody je například upřednostňování FSC které již v sobě obsahují chování optimálního řešení minulé iterace prohledávání a pouze ho rozšiřují o možnosti využití nové paměti. Druhým směrem v této metodě může být naopak nalezení akcí, které v minulé iteraci nebyly vůbec použity a v následující iteraci z prohledávání vyloučit všechna řešení, která tyto akce používají.

Literatura

[1] Amato, C., Bernstein, D. S. a Zilberstein, S. Optimizing Memory-Bounded Controllers for Decentralized POMDPs. In: Parr, R. a Gaag, L. C. van der, ed. *UAI 2007, Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence, Vancouver, BC, Canada, July 19-22, 2007*. AUAI Press, 2007, s. 1–8. DOI: 10.5555/3020488.3020489. Dostupné z: https://dl.acm.org/doi/10.5555/3020488.3020489.

[2] Amato, C., Bernstein, D. S. a Zilberstein, S. Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Autonomous Agents and Multi-Agent Systems*. Springer. 2010, sv. 21, s. 293–320.

[3] Amato, C., Bonet, B. a Zilberstein, S. Finite-state controllers based on Mealy machines for centralized and decentralized POMDPs. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2010, sv. 24, č. 1, s. 1052–1058.

[4] Amato, C., Dibangoye, J. a Zilberstein, S. Incremental policy generation for finite-horizon DEC-POMDPs. In: *Proceedings of the International Conference on Automated Planning and Scheduling*. 2009, sv. 19, s. 2–9.

[5] Andriushchenko, R., Ceska, M., Junges, S. a Katoen, J. Inductive synthesis of finite-state controllers for POMDPs. In: Cussens, J. a Zhang, K., ed. *Uncertainty in Artificial Intelligence, Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022, 1-5 August 2022, Eindhoven, The Netherlands*. PMLR, 2022, sv. 180, s. 85–95. Proceedings of Machine Learning Research. Dostupné z: https://proceedings.mlr.press/v180/andriushchenko22a.html.

[6] Andriushchenko, R., Češka, M., Junges, S., Katoen, J.-P. a Stupinskỳ, Š. PAYNT: a tool for inductive synthesis of probabilistic programs. In: Springer. *International Conference on Computer Aided Verification*. 2021, s. 856–869.

[7] Becker, R., Zilberstein, S., Lesser, V. a Goldman, C. V. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*. 2004, sv. 22, s. 423–455.

[8] Bernstein, D. S., Givan, R., Immerman, N. a Zilberstein, S. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research*. INFORMS. 2002, sv. 27, č. 4, s. 819–840.

[9] Bork, A., Junges, S., Katoen, J.-P. a Quatmann, T. Verification of Indefinite-Horizon POMDPs. In: Hung, D. V. a Sokolsky, O., ed. *Automated Technology for Verification and Analysis*. Cham: Springer International Publishing, 2020, s. 288–304. ISBN 978-3-030-59152-6.

[10] Cassandra, T. *Tony Cassandra's POMDP File Repository Page*. [citováno 6.5.2024]. Dostupné z: https://pomdp.org/examples/.

[11] Češka, M., Hensel, C., Junges, S. a Katoen, J.-P. Counterexample-driven synthesis for probabilistic program sketches. In: Springer. *International Symposium on Formal Methods*. 2019, s. 101–120.

[12] Češka, M., Jansen, N., Junges, S. a Katoen, J.-P. Shepherding hordes of Markov chains. In: Springer. *Tools and Algorithms for the Construction and Analysis of Systems: 25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6–11, 2019, Proceedings, Part II 25*. 2019, s. 172–190.

[13] Dibangoye, J. S., Amato, C., Buffet, O. a Charpillet, F. Optimally solving Dec-POMDPs as continuous-state MDPs. *Journal of Artificial Intelligence Research*. 2016, sv. 55, s. 443–497.

[14] Grzes, M., Poupart, P. a Hoey, J. Isomorph-Free Branch and Bound Search for Finite State Controllers. In: Rossi, F., ed. *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. IJCAI/AAAI, 2013, s. 2282–2290. Dostupné z: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6954.

[15] Grześ, M., Poupart, P., Yang, X. a Hoey, J. Energy efficient execution of POMDP policies. *IEEE Transactions on Cybernetics*. IEEE. 2014, sv. 45, č. 11, s. 2484–2497.

[16] Hauskrecht, M. Incremental Methods for Computing Bounds in Partially Observable Markov Decision Processes. In: Kuipers, B. a Webber, B. L., ed. *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island, USA*. AAAI Press / The MIT Press, 1997, s. 734–739. Dostupné z: http://www.aaai.org/Library/AAAI/1997/aaai97-114.php.

[17] Hensel, C., Junges, S., Katoen, J.-P., Quatmann, T. a Volk, M. The probabilistic model checker Storm. *International Journal on Software Tools for Technology Transfer*. Springer. 2022, s. 1–22.

[18] Kaelbling, L. P., Littman, M. L. a Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial intelligence*. Elsevier. 1998, sv. 101, 1-2, s. 99–134.

[19] Koops, W., Jansen, N., Junges, S. a Simão, T. D. Recursive Small-Step Multi-Agent A* for Dec-POMDPs. ijcai.org. 2023, s. 5402–5410. DOI: 10.24963/IJCAI.2023/600. Dostupné z: https://doi.org/10.24963/ijcai.2023/600.

[20] Kube, C. R. Task Modelling in Collective Robotics. *Auton. Robots*. 1997, sv. 4, č. 1, s. 53–72. DOI: 10.1023/A:1008859119831. Dostupné z: https://doi.org/10.1023/A:1008859119831.

[21] Kumar, A., Mostafa, H. a Zilberstein, S. Dual formulations for optimizing Dec-POMDP controllers. In: *Proceedings of the International Conference on Automated Planning and Scheduling*. 2016, sv. 26, s. 202–210.

[22] Kumar, A. a Zilberstein, S. Anytime Planning for Decentralized POMDPs using Expectation Maximization. In: Grünwald, P. a Spirtes, P., ed. *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, July 8-11, 2010*. AUAI Press, 2010, s. 294–301. Dostupné z: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2090&proceeding_id=26.

[23] Kumar, A. a Zilberstein, S. Point-based backup for decentralized POMPDs: Complexity and new algorithms. IFAAMAS. 2010.

[24] Kumar, A., Zilberstein, S. a Toussaint, M. Probabilistic inference techniques for scalable multiagent decision making. *Journal of Artificial Intelligence Research*. 2015, sv. 53, s. 223–270.

[25] Kwiatkowska, M., Norman, G. a Parker, D. Stochastic model checking. In: *International School on Formal Methods for the Design of Computer, Communication and Software Systems*. 2007, s. 220–270.

[26] Madani, O., Hanks, S. a Condon, A. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*. Elsevier. 2003, sv. 147, 1-2, s. 5–34.

[27] Nair, R., Tambe, M., Yokoo, M., Pynadath, D. a Marsella, S. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In: *IJCAI*. 2003, sv. 3, s. 705–711.

[28] Nair, R., Varakantham, P., Tambe, M. a Yokoo, M. Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In: *AAAI*. 2005, sv. 5, s. 133–139.

[29] Oliehoek, F. A. Decentralized POMDPs. In: Wiering, M. A. a Otterlo, M. van, ed. *Reinforcement Learning*. Springer, 2012, sv. 12, s. 471–503. Adaptation, Learning, and Optimization. DOI: 10.1007/978-3-642-27645-3_15. Dostupné z: https://doi.org/10.1007/978-3-642-27645-3_15.

[30] Oliehoek, F. A. a Amato, C. *A Concise Introduction to Decentralized POMDPs*. Springer, 2016. Springer Briefs in Intelligent Systems. ISBN 978-3-319-28927-4. Dostupné z: https://doi.org/10.1007/978-3-319-28929-8.

[31] Oliehoek, F. A. Sufficient Plan-Time Statistics for Decentralized POMDPs. In: Rossi, F., ed. *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. IJCAI/AAAI, 2013, s. 302–308. Dostupné z: http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6610.

[32] Pajarinen, J., Hottinen, A. a Peltonen, J. Optimizing spatial and temporal reuse in wireless networks by decentralized partially observable Markov decision processes. *IEEE Transactions on Mobile Computing*. IEEE. 2013, sv. 13, č. 4, s. 866–879.

[33] Pajarinen, J. a Peltonen, J. Periodic finite state controllers for efficient POMDP and DEC-POMDP planning. *Advances in neural information processing systems*. 2011, sv. 24.

[34] Pajarinen, J. a Peltonen, J. Periodic Finite State Controllers for Efficient POMDP and DEC-POMDP Planning. In: Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F. C. N. a Weinberger, K. Q., ed. *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*. 2011, s. 2636–2644. Dostupné z: https://proceedings.neurips.cc/paper/2011/hash/1f3202d820180a39f736f20fce790de8-Abstract.html.

[35] Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994. Wiley Series in Probability and Statistics. ISBN 978-0-47161977-2. Dostupné z: https://doi.org/10.1002/9780470316887.

[36] Quatmann, T., Jansen, N., Dehnert, C., Wimmer, R., Ábrahám, E. et al. Counterexamples for expected rewards. In: *International Symposium on Formal Methods*. 2015, s. 435–452.

[37] Szer, D., Charpillet, F. a Zilberstein, S. MAA*: A Heuristic Search Algorithm for Solving Decentralized POMDPs. In: *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*. AUAI Press, 2005, s. 576–590. Dostupné z: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=1221&proceeding_id=21.

[38] Van Hung, D. a Sokolsky, O. *Automated Technology for Verification and Analysis: 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19–23, 2020, Proceedings*. Springer Nature, 2020.

[39] You, Y., Thomas, V., Colas, F. a Buffet, O. Solving infinite-horizon Dec-POMDPs using finite state controllers within JESP. In: IEEE. *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. 2021, s. 427–434.

Přechod/akce	$\langle front, front \rangle$	$\langle front, back \rangle$	$\langle back, front \rangle$	$\langle back, back \rangle$
$s_1 \rightarrow s_1$	0.9025	0.0975	0.0975	0.9025
$s_1 \rightarrow s_3$	0.04875	0.9	0.0025	0.04875
$s_1 \rightarrow s_4$	0.04875	0.0025	0.9	0.04875
$s_2 \rightarrow s_2$	0.9025	0.0975	0.0975	0.9025
$s_2 \rightarrow s_4$	0.04875	0.9	0.0025	0.04875
$s_2 \rightarrow s_{end}$	0.04875	0.0025	0.9	0.04875
$s_3 \rightarrow s_1$	0.04875	0.0025	0.9	0.04875
$s_3 \rightarrow s_3$	0.9025	0.0975	0.0975	0.9025
$s_3 \rightarrow s_{end}$	0.04875	0.9	0.0025	0.04875
$s_4 \rightarrow s_1$	0.04875	0.9	0.0025	0.04875
$s_4 \rightarrow s_2$	0.04875	0.0025	0.9	0.04875
$s_4 \rightarrow s_4$	0.9025	0.0975	0.0975	0.9025
$d_1 \rightarrow d_1$	0.0975	0.9025	0.9025	0.0975
$d_1 \rightarrow d_3$	0.9	0.04875	0.04875	0.0025
$d_1 \rightarrow d_4$	0.0025	0.04875	0.04875	0.9
$d_2 \rightarrow d_2$	0.0975	0.9025	0.9025	0.0975
$d_2 \rightarrow d_4$	0.9	0.04875	0.04875	0.0025
$d_2 \rightarrow d_{end}$	0.0025	0.04875	0.04875	0.9
$d_3 \rightarrow d_1$	0.0025	0.04875	0.04875	0.9
$d_3 \rightarrow d_3$	0.0975	0.9025	0.9025	0.0975
$d_3 \rightarrow d_{end}$	0.9	0.04875	0.04875	0.0025
$d_4 \rightarrow d_1$	0.9	0.04875	0.04875	0.0025
$d_4 \rightarrow d_2$	0.0025	0.04875	0.04875	0.9
$d_4 \rightarrow d_4$	0.0975	0.9025	0.9025	0.0975
$d_{end} \rightarrow d_{end}$	1.0	1.0	1.0	1.0

Tabulka 1: Přechodová funkce $\mathcal{T}$ Dec-POMDP modelu Circle