Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

DOKTORSKA DISERTACIJA

Avtomatsko modeliranje večagentnih sistemov

Andraž Bežek

Mentor: akad. prof. dr. Ivan Bratko

Somentor: prof. dr. Matjaž Gams

Ljubljana, 2006

Zahvala

V prvi vrsti bi se rad zahvalil svojemu raziskovalnemu mentorju prof. dr. Matjažu Gamsu, ki mu je uspelo, da me je skozi celotni podiplomski študij uspešno krotil in me kljub pogostim nestrinjanjem bodril ter pomagal pri delu. Njegova neizmerna potrpežljivost ter pripravljenost pomagati sta ključno prispevali k nastanku disertacije.

Zahvalil bi se tudi fakultetnemu mentorju akademiku prof. dr. Ivanu Bratku, s katerim sem imel na žalost redke, a toliko bolj burne debate. Kljub vsemu mi je bil s svojo konstruktivno kritiko kažipot, ki me je usmerjal na razvejani in netlakovani poti raziskovalnega dela.

Zahvala gre tudi preostalim članom doktorske komisije: prof. dr. Blažu Zupanu, prof. dr. Andreju Dobnikarju in prof. dr. Vladislavu Rajkoviču, ki so mi s svojimi konstruktivnimi komentarji pomagali izboljšati disertacijo.

Posebej rad bi se zahvalil vsem sodelavcem Odseka za inteligentne sisteme, ki so s svojim delom nase prenesli breme vzdrževanja nefinanciranega mladega raziskovalca. Poleg tega bi se zahvalil tudi ostalim sodelavcem iz Instituta Jožef Stefan, ki so mi s svojimi konstruktivnimi komentarji zelo pomagali pri raziskovalnem delu. Še posebej bi se rad zahvalil cimri Tei Tušar, Mitju Luštreku, Aleksandru Pivku, Andreju Bratku, Domnu Marinčiču, Bogdanu Filipiču, Bernardu Ženku, Branku Kavšku, Janezu Branku, Marku Grobelniku, Dunji Mladenić, Petru Ljubiču, Juretu Ferležu, Ljupču Todorovskemu, Nadi Lavrač, Tomažu Erjavcu, Janezu Demšarju in Juriju Šilcu, s katerimi sem največ sodeloval in so mi bili v veliko pomoč pri reševanju marsikatere metodološke in tehnične zagate.

Peter Stone in Matthew Taylor iz The University of Texas at Austin, ZDA, sta mi nesebično odstopila njuno 3vs2 Keepaway strategijo, poleg tega pa sta prispevala ideje in komentarje, za kar se jima izrecno zahvaljujem. Zahvaljujem se tudi Galu Kaminki iz univerze Bar Ilan, Izrael, za konstruktivne ideje, komentarje in vzpodbude.

Ob tej priliki bi se zahvalil tudi nogometnim strokovnjakom dr. Marku Pocrnjiču, Marku Trebec in Janiju Resniku, ki so mi posredovala številne komentarje in nasvete v zvezi z nogometno tematiko.

Posebna zahvala gre staršem in ostalim članom družine, ki so me vedno, ne le v času študija, vzpodbujali in mi bili na voljo, ko sem jih potreboval. Brez njih te disertacije nikoli ne bi bilo.

Nenazadnje bi se zahvalil Diti, ki se morda sploh ne zaveda, kako mi je v kritičnem obdobju s svojo ljubeznijo pomagala in mi povrnila voljo do življenja.

Zahvaljujem se tudi vsem mojim prijateljem. Oni že vedo za kaj.

Kazalo

Seznam preglednic

Seznam slik

Povzetek

Ena najtežjih, do sedaj nerešenih nalog večagentnega modeliranja je iz sledenja nizkonivojskega obnašanja skupine agentov in zgolj osnovnega domenskega znanja ugotoviti, kakšno skupno strategijo izvajajo. Ta naloga je zahtevna iz dveh ključnih razlogov. Prvič, agenti so samostojne entitete, ki skušajo v nenehni interakciji s sodelujočimi agenti, nasprotnikovimi agenti ter okoljem skladno izvajati vnaprej dogovorjeno strategijo. Iz nizkonivojskega opisa delovanja posameznih agentov je zato težko izluščiti visokonivojsko strategijo skupine agentov. Drugič, pri učenju ima sistem poleg opazovanja nizkonivojskega obnašanja na voljo le osnovno domensko znanje. Tako osnovno domensko znanje je običajno na razpolago, naloga pa zahteva odkrivanje bolj poglobljenega znanja, kakršnega imajo le domenski strokovnjaki.

V disertaciji je predstavljen izvirni algoritem za strateško modeliranje MASDA (angl. *Multi-Agent Strategy Discovering Algorithm*), ki skuša iz sledenja nizkonivojskega obnašanja skupine agentov in zgolj osnovnega domenskega znanja ugotoviti, kakšno skupno strategijo izvajajo. MASDA uporablja postopek abstrakcije, ki omogoča ločevanje modelov agentnega delovanja, ki so posledica izvrševanja večagentne strategije, od modelov, ki so posledica agentnih odzivov na lokalne spremembe okolja. Zgrajeni model opisuje sodelovanje agentov, predstavljeno v grafični in simbolni obliki; grafični opis omogoča vizualno predstavitev večagentnega delovanja, simbolni opis pa na človeku razumljiv način predstavi pomembne značilnosti večagentne interakcije. S tem je omogočena človeška analiza, razlaga ter avtomatska klasifikacija.

Uspešnost algoritma smo v disertaciji prikazal na dveh večagentnih domenah. V domeni robotskega nogometa RoboCup zgrajeni modeli po mnenju domenskega strokovnjaka vsebinsko opisujejo del nogometne strategije. Meritve z osnovnimi kriteriji strojnega učenja na osnovi: klasifikacijske točnosti, priklica in natančnosti, potrjujejo primernost uporabe modelov MASDA pri avtomatski klasifikaciji. V domeni agentnega podajanja žoge 3vs2 Keepaway je bila deklarativna uspešnost modela potrjena v primerjavi z izvorno kodo treh znanih strategij, ujemanje posnemanja igranja pa s primerjanjem časovne uspešnosti ter s strojnim in človeškim primerjanjem ujemanja izvajanih akcij.

Ključne besede:

večagentni sistemi, modeliranje, agent, strategija, interakcija

Abstract

One of the most difficult unsolved tasks in the field of multi-agent modeling is to discover common agent strategy by knowing only low-level agent behavior and basic domain knowledge. This task is difficult for the following two reasons. First, agents are autonomous entities that try to accomplish in advance given strategy, while continuously interacting with team-agents, adversary-agents, and the environment. By giving only low-level descriptions of single agent behaviors, high-level strategy is therefore difficult to discover. Second, in addition to low-level agent behavior, the system utilizes only basic domain knowledge. Such knowledge is often available and the task is to discover high-level knowledge, usually available only to experts.

This dissertation presents a novel algorithm for strategic modeling MASDA (*Multi-Agent Strategy Discovering Algorithm*). By tracking low-level behavior of a group of agents and using only basic domain knowledge, it tries to discover common agent strategy. MASDA incorporates an abstraction process, which enables MASDA to separate models that are due to following of agent strategy, from models that are due to agent reactions to local environment changes. The created model describes collaboration of agents in a graphical and symbolical manner. Graphic descriptions visually present multi-agent activity, while symbolic descriptions present important characteristics of multi-agent interaction in a human-comprehendible way. By that human analysis, interpretation and automatic classification is possible.

Efficiency of the algorithm was shown on two multi-agent domains. In the domain of robotic soccer – RoboCup, the human expert confirmed that the created model semantically describes part of real-life soccer strategy. Measurements of classification accuracy, recall, and precision confirm suitability of MASDA models for computer classification. In the domain of ball keep away problem – 3vs2 Keepaway, the declarative efficiency of the model was confirmed by comparing it with source code for three known strategies and the strategy imitation performance was shown by measuring time efficiency and by human and computer comparison of played actions.

Keywords:

multi-agent systems, modeling, agent, strategy, interaction

1. Uvod

Kdor je slab strateg in podcenjuje nasprotnika, bo poražen. Zatorej pravim: če poznaš tako nasprotnika kot sebe, ti zmaga ne uide.

Sun Tzu, Umetnost vojskovanja

1.1 Motivacija in cilji

Modeliranje večagentnih sistemov je aktivno raziskovalno področje, ki je kljub veliki popularnosti večagentnih sistemov zaradi precejšne zahtevnosti še vedno dokaj neraziskano. Ena najtežjih, do sedaj nerešenih nalog večagentnega modeliranja je iz sledenja nizkonivojskega obnašanja skupine agentov in zgolj osnovnega domenskega znanja ugotoviti, kakšno skupno strategijo izvajajo. Ta naloga je zahtevna iz dveh ključnih razlogov:

- Agenti so **samostojne** entitete, ki skušajo v nenehni **interakciji** s soigralci, nasprotniki in okoljem skladno izvajati vnaprej dogovorjeno strategijo. Iz nizkonivojskega opisa delovanja posameznih agentov je zato težko izluščiti visokonivojsko strategijo skupine agentov.
- Pri učenju ima sistem poleg opazovanja nizkonivojskega obnašanja na voljo le **osnovno domensko znanje**. Tako osnovno domensko znanje je običajno na voljo, naloga pa zahteva odkrivanje konkretnega poglobljenega znanja, kakršnega imajo na voljo domenski strokovnjaki.

Cilj disertacije je razviti domensko neodvisen postopek, ki iz zaporedja osnovnih akcij večagentnega sistema in z uporabo osnovnega domenskega znanja zgradi človeku razumljive opise strategije delovanja večagentnega sistema. Ker je delovanje agentov posledica njihove strategije ter odzivov na lokalne spremembe okolja, je bil dodaten cilj uporabiti **abstrakcijo pridobljenih modelov**, ki omogoča ločevanje modelov, ki so posledica strategije, od modelov, ki so posledica lokalnega agentnega odločanja. Rezultat modeliranja je večagentni model, ki:

- **opisuje zaporedja agentnih akcij, ki so del večagentne strategije**. Vsak opis agentne akcije opisuje agentno akcijo, agenta, njegovo vlogo, agentno stanje in značilne lastnosti okolja.
- **je človeku razumljiv ter primeren za analizo in interpretacijo modela**. Zato ga želimo uporabiti za razlago in vpogled v sicer nepoznana pravila in vzorce delovanja večagentnega sistema. Poleg tega želimo v modelu zaobjeti tudi velikokrat zapostavljeni vidik agentnega modeliranja – to je vizualizacija agentnega delovanja, ki omogoča vizualno identifikacijo aktivnosti agentnega sistema.
- **je primeren za avtomatsko klasifikacijo**, torej omogoča identifikacijo trenutnih in napovedovanje prihodnjih agentnih akcij z zadovoljivo kvantifikativno točnostjo.

Ker do sedaj še ni poznanega splošnega in domensko neodvisnega postopka za modeliranje večagentnih sistemov, je naš cilj visoko zastavljen.

1.2 Pregled vsebine

Disertacija je sestavljena iz treh glavnih delov. Prvi del predstavlja splošen uvod v modeliranje večagentnih sistemov. V poglavju 2 so opisane značilnosti in lastnosti večagentnih sistemov, poleg tega pa sta predstavljeni tudi večagentni domeni RoboCup in 3vs2 Keepaway. V poglavju 3 je podan izčrpen pregled sorodnega dela, najprej iz splošnega področja modeliranja večagentnih sistemov, nato pa specifično iz modeliranja na domenah RoboCup in 3vs2 Keepaway. Problem modeliranja večagentnih sistemov je formalno opisan v poglavju 4.

Drugi del predstavlja glavni del disertacije. V poglavju 5 je podan splošen opis sistema za strateško modeliranje, poleg tega pa je opisan tudi postopek predobdelave podatkov ter predstavljena oblika domenskega znanja. Poglavje 6 izčrpno predstavi algoritem za strateško modeliranje večagentnih sistemov MASDA (angl. *Multi-Agent Strategy Discovering Algorithm*), kjer je vsak korak v algoritmu predstavljen v svojem podpoglavju. Opis uporabe modelov je predstavljen v zadnjem podpoglavju.

Tretji del disertacije predstavlja ovrednotenje pristopa. V poglavju 7 je podano ovrednotenje na domeni RoboCup, v poglavju 8 pa na domeni 3vs2 Keepaway. Diskusija o prispevkih disertacije je podana v poglavju 9, skupni zaključki pa v poglavju 10.

1.3 Prispevki znanosti

Glavni prispevek k znanosti je nov, domensko neodvisen algoritem za modeliranje večagentnih sistemov, ki iz opazovanja obnašanja večagentnega sistema ob le osnovnem predznanju zgradi človeku razumljivo delno ali celotno strategijo skupine agentov.

Prispevki k znanosti so:

- Ob uporabi le osnovnega predznanja algoritem uspešno odkriva visokonivojsko znanje neodvisno od domene – isti algoritem je bil uspešno uporabljen na dveh različnih domenah, pri čemer so bili spremenjeni le vhodni podatki o agentnem obnašanju in domenskem predznanju.
- V algoritmu je razvit nov postopek abstrakcije, ki omogoča gradnjo poljubno abstraktnih opisov strategije večagentnega sodelovanja. S tem je možno ločiti modele agentnega delovanja, ki so posledica izvrševanja večagentne strategije, od modelov, ki so posledica agentnih odzivov na lokalne spremembe okolja.
- Zgrajeni model opisuje sodelovanje agentov predstavljeno v grafični in simbolni obliki. Grafični opis omogoča vizualno predstavitev večagentnega delovanja, simbolni opis pa na človeku razumljiv način predstavi pomembne značilnosti večagentne interakcije. To omogoča lažjo analizo in razlago strategij.
- Uspešnost algoritma je izkazana na dveh večagentnih domenah – v domeni RoboCup z zgrajenimi modeli, ki po mnenju domenskega strokovnjaka opisujejo del nogometne strategije, ter z osnovnimi merili strojnega učenja: klasifikacijske točnosti, priklica in natančnosti. V domeni 3vs2 Keepaway je bila deklarativna uspešnost modela potrjena s primerjavo simbolnih opisov modelov in izvorne kode treh znanih strategij, ujemanje posnemanja igranja pa s primerjanjem časovne uspešnosti ter primerjanjem ujemanja izvajanih akcij.

2. Večagentni sistemi

> *We want to build intelligent actors, not just intelligent thinkers. Indeed, it is not even clear how one could assess intelligence in a system that never acted - or, put otherwise, how a system could exhibit intelligence in the absence of action.*
>
> *Martha Pollack, na predavanju ob podelitvi The Computers and Thought Award, IJCAI-91.*

V strokovni literaturi [39] ni točno sprejete definicije termina agent. Tako Russell in Norvig v 2. poglavju njune knjige [92] določita, da je agent vse, kar lahko zaznava okolje z uporabo senzorjev in deluje v okolju z uporabo izvajalnih struktur (angl. *effectors*), kar pa praktično ne loči splošnega programa od agenta. Jennings in Wooldridge [124] ločita dve definiciji agenta, šibko in močno. Šibka določa za agenta vse programske entitete, ki kažejo lastnosti avtonomije (delovanje brez neposrednega človeškega nadzora), socialne zmožnosti (sposobnost komuniciranja z ostalimi agenti), reaktivnosti (zmožnost reagirati na spremembe okolja) in proaktivnosti (zmožnost delovanja na lastno pobudo). Bolj zanimiva je močna definicija agenta, ki pravi, da je agent računalniški sistem, ki ima poleg lastnosti šibkega agenta tudi lastnosti, ki so opisane z uporabo človeških lastnosti. Shoham [98] te človeške lastnosti omeji na: znanje, prepričanje, namen in zavezo.

Ne glede na ohlapne definicije agenta pa obstajajo bolj točne definicije večagentnega sistema. Bistvena razlika večagentnega sistema glede na splošen sistem je v tem, da so agenti v sistemu avtonomne in inteligentne entitete, ki delujejo v danem okolju z namenom, da dosežejo cilje, ki jih sami ne morejo doseči. Za doseganje takih ciljev je nujno potrebna interakcija med agenti, ki jo nekateri [42, 119] postavljajo kot pogoj za dosego inteligence. Ali, kot trdi Gerhard Weiß [120], inteligenca je globoko in neizbežno povezana z interakcijo.

Definicija 1. ***Večagentni sistem*** je programsko okolje, v katerem deluje množica avtonomnih programskih entitet, imenovanih agenti, ki delujejo z namenom, da dosežejo lastne in skupne cilje. Okolje daje na voljo agentom infrastrukturo, ki omogoča njihovo delovanje, zaznavanje in medsebojno komunikacijo.

2.1 Lastnosti večagentnih sistemov

Čeprav obstaja več različnih vrst večagentnih sistemov [39], so osnovne lastnosti večagentnega sistema [110] sledeče:

- **Agenti imajo nepopolno informacijo in omejene zmožnosti, da sami rešijo zastavljeni problem.** Vzrok za nepopolno informacijo je lahko v preveliki količini informacij, ki bi jih agent lahko hranil (npr. internet), ali v sami nedostopnosti informacij (npr. podatki niso neposredno na voljo). Nezmožnost za samostojno reševanje problema izvira iz same narave problema, ki običajno zahteva sočasno udejstvovanje več agentov ali delovanje na različnih lokacijah.

- **Ni centralnega in globalnega nadzora**. To pomeni, da morajo agenti sami nadzirati in usklajevati skupne akcije. Tako lokalno odločanje je sicer pogosto neoptimalno, vendar omogoča večjo prilagodljivost pri reševanju problemov. Pomembna posledica medagentnega usklajevanja je v naravni dekompoziciji problemov. Če ima centralizirani sistem nadzor nad izvajanjem vseh korakov, ki so potrebni za reševanje problema, je v primeru agenta, ki deluje v večagentnem sistemu to nemogoče. Zato je običajni postopek reševanja nekega problema njegova dekompozicija na manjše podprobleme, ki jih rešujejo ostali agenti.
- **Podatki so decentralizirani**, torej je dostop do njih posreden in časovno potraten. Zato morajo imeti agenti mehanizme, ki jih omogoča iskanje odgovarjajočih informacijskih ponudnikov. Če agent ni sposoben zagotoviti potrebnih oz. pravih podatkov, je njegovo delovanje zato omejeno.
- **Izvajanje je asinhrono**, kar pomeni, da poteka izvajanje enega agenta neodvisno od izvajanja ostalih. Vzporedno delovanje lahko pospeši reševanje kompleksnih problemov, ki omogočajo dekompozicijo in vzporedno reševanje. Vendar pa ima asinhrono izvajanje tudi dve pomembni negativni posledici. Prva so sinhronizacijski problemi pri usklajevanju skupnih aktivnosti, saj časovno prepleteno izvajanje agentnih akcij vpliva na potek vseh akcij. Okolje mora zato agentom nuditi protokole, ki omogočajo agentom usklajevanje svojih aktivnosti z ostalimi. Drugi problem je sočasna uporaba skupnih virov. Zaradi posledičnega zaklepanja dostopa do virov prihaja do t.i. tekmovanja za proste vire.
- **Obstaja komunikacija med agenti**. Osnovna oblika komunikacije je fizična interakcija z okoljem, kjer spremembe v okolju kažejo na pomen sporočila. Tak način komunikacije je poznan kot stigmergija (angl. *stigmergy*) [49] in se npr. pojavlja pri kolonijah mravelj, kjer mravlje s feromoni označujejo poti do hrane. Bolj napredna je neposredna komunikacija med agenti. Slednja zahteva skupen komunikacijski protokol, ki je neke vrste skupno znanje, ki opisuje obliko in pomen sporočil. V kompleksnih večagentnih sistemih se pogosto uporabljata obe obliki komunikacije.

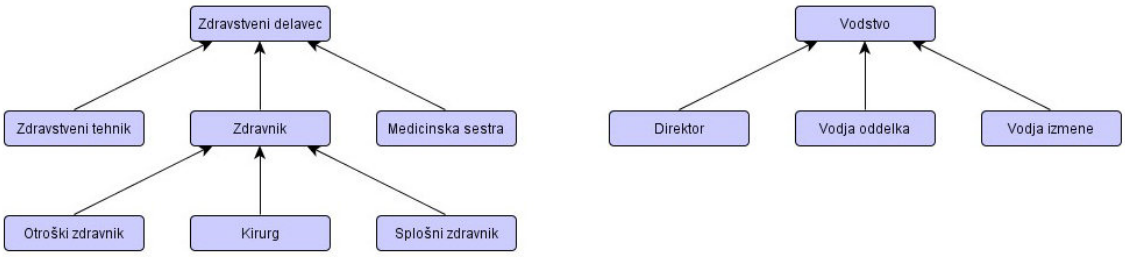
V večagentnih sistemih je organizacija in delovanje agentov opredeljena z agentnimi vlogami. Le-te določajo relacije in vzorce obnašanja do ostalih vlog. Ko agent prevzame določeno vlogo, je njegovo obnašanje opisano s pravili, ki jih določa vloga.

Za lažjo predstavo se večagentni sistem pogosto primerja s pojmom človeške družbe. Taka prispodoba je uporabna tudi, ko vpeljemo koncept agentne vloge, ki ga lahko razložimo s primerom človeške organizacije. Za primer lahko vzamemo organizacijo v bolnišnici, kjer obstaja vrsta različnih vlog, kot so direktor, vodja izmene, vodja oddelka, splošni zdravnik, otroški zdravnik, kirurg, medicinska sestra in zdravstveni tehnik. Vsaka vloga opisuje naloge, pravice, načine komunikacije in pravila obnašanja. V resničnem svetu zasedajo vloge resnične osebe, ki opravljajo delo, ki je določeno z njihovo vlogo. Vloge niso statične, saj lahko agenti dinamično prevzemajo in menjujejo vloge [73], da učinkoviteje dosežejo zastavljeni cilj. Tako je v primeru bolnišnice lahko več oseb splošnih zdravnikov, medtem ko lahko vodja oddelka, na primer, opravlja tudi vlogo kirurga.

V metodologiji za večagentno analizo in načrtovanje GAIA postavi Wooldridge s sodelavci [123] vlogo kot ključno enoto pri opisu večagentnega sistema. Vloge določi z agentno funkcionalnostjo, s pravicami, ki izhajajo iz vloge, z naborom agentnih aktivnosti in s protokoli, ki določajo načine interakcije z ostalimi vlogami. Vloge prav

tako niso togo omejene samo na medsebojno izključujoče vzorce obnašanja. Kendall [57] nadgradi koncept agentnih vlog s hierarhijo agentnih vlog. Posamezna agentna vloga je lahko definirano kot:

- združitev več vlog, torej predstavlja konceptualno bolj splošen opis agentne vloge;
- izpeljana iz obstoječe vloge, torej predstavlja konceptualno bolj specifično opredelitev agentne vloge.



Slika 1. Primer hierarhij vlog v bolnišnici.

Primer hierarhij vlog v bolnišnici je predstavljen na sliki 1, kjer bolj splošni opisi vlog zasedajo višji položaj v hierarhiji kot bolj specifični opisi. Predstavitev z uporabo hierarhije agentnih vlog omogoča učinkovit popis celotnega spektra možnih agentnih obnašanj. Zaradi domenske specifičnosti agenti vlog pa je njihov dejanski opis mogoč šele ob poznavanju dane večagentne domene.

2.2 Robotski nogomet

Mackworth [67] je že leta 1992 uvedel idejo o uporabi nogometnih robotov v raziskavah. Ideja se ni prijela, dokler niso Kitano, Asada in Kuniyoshi prijavili prevzeto in izboljšano različico kot japonski raziskovalni program, imenovan *Robot J-League2*. V jeseni 1993 so ameriški raziskovalci pokazali interes za *Robot J-League2* ter ga posledično spremenili v *Robot World Cup Initiative* ali na kratko v *RoboCup*. RoboCup je v literaturi včasih naslovljen tudi kot "RoboCup izziv" ali "RoboCup domena".

V letu 1995 so Kitano in sodelavci [58] predlagali prvi RoboCup dogodek kot *The First Robot World Cup Soccer Games and Conferences*, ki se je prvič dogodil leta 1997. Cilj RoboCup domene je predstaviti novi standardni problem s področja umetne inteligence [20] (v nadaljevanju UI) in robotike. RoboCup se razlikuje od ostalih raziskovalnih problemov UI s poudarjanjem iskanja porazdeljenih rešitev, kar predstavlja odmik od klasičnih centraliziranih rešitev tradicionalne UI. Prav tako vzpodbuja raziskovalce iz različnih področij, kot so umetna inteligenca, robotika, sociologija, vzporedni sistemi, sistemi v realnem času, ipd., da sodelujejo v raziskavah. Za primerjavo s standardnim problemom s področja UI so Kitano in sodelavci [58] izpostavili šah. Primerjava šaha z domeno RoboCup je prikazana na preglednici 1.

	Šah	RoboCup
Okolje	statično	dinamično
Spremembe okolja	izmenično potezne	v realnem času
Zaznava okolja	popolna	nepopolna
Opis okolja	simbolen	nesimbolen
Nadzor	centralen	porazdeljen

Preglednica 1. Primerjava domene RoboCup s šahom kot primerom standardnega problema v umetni inteligenci.

Z namenom koordinacije množice raziskovalcev so ustanovili v Švici registrirano zvezo RoboCup, imenovano *RoboCup Federation*. Njen cilj je promocija RoboCup domene, kar med drugim dosega s prirejanjem vsakoletnih svetovnih prvenstev. Člani zveze RoboCup so aktivni raziskovalci s številnih univerz in velikih računalniških podjetij. Ker so vključeni številni raziskovalci s celega sveta, so ustanovljeni lokalni odbori, ki promovirajo z RoboCup-om povezane dogodke v njihovem lokalnem okolju.

Zveza RoboCup je določila osnovne cilje in časovni potek raziskav. S formaliziranjem testnih okolji poskuša zveza tako aktivno pospeševati tehnološko napredne raziskave. Poleg splošnega napredka in dviga tehnološke osveščenosti družbe so sestavili tudi bolj pragmatičen cilj [90]:

> *"Do sredine 21. stoletja bo moštvo popolnoma avtonomnih humanoidnih robotskih igralcev nogometa premagalo zmagovalce takratnega svetovnega nogometnega prvenstva v igri, ki bo ustrezala uradnim FIFA pogojem [36]."*

Zveza RoboCup je hitro prerasla okvir robotskega nogometa in sedaj združuje več tematsko različnih področij z več ligami, kot je to predstavljeno na preglednici 2. Velja poudariti, da kljub navidezni različnosti vsa področja RoboCup predstavljajo različice osnovnega izhodiščnega problema, to je iskanje porazdeljenih rešitev za zastavljene probleme.

RoboCup področje	Lige v področju
RoboCupSoccer	*Soccer Simulation League* *Small-Size Soccer Robot (F-180) League* *Middle-Size Soccer Robot (F-2000) League* *4-Legged Robot League* *Humanoid League* *@Home*
RoboCupRescue	*Rescue Robot League* *Rescue Simulation League*
RoboCupJunior	*Soccer* *Rescue* *Dance*

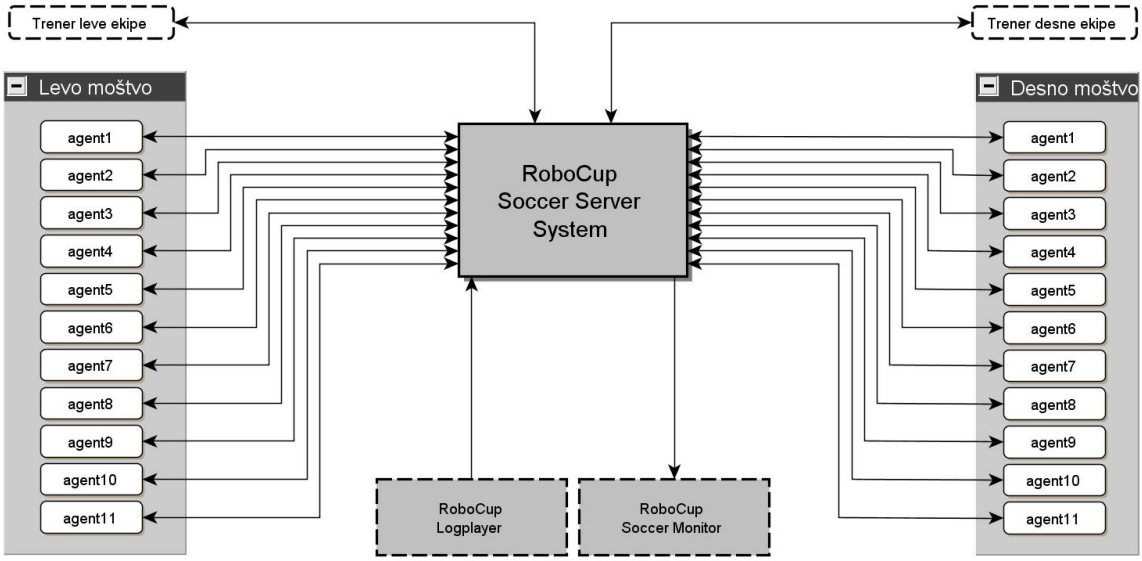
Preglednica 2. RoboCup področja in lige.

V večini RoboCup lig moštva robotov ali programov sodelujejo z namenom premagati nasprotnikovo ekipo. V tem pogledu se lige *RoboCup Rescue* razlikujejo, saj je cilj v domeni *RoboCup Rescue* v koordiniranju naporov med raznovrstnimi agenti, ki pomagajo reševati krizne razmere simuliranih katastrof.

2.2.1 Domena RoboCup Soccer Simulated League

Domena *RoboCup Soccer Simulated League* [59, 75] je liga, ki omogoča programsko simulirano večagentno okolje, v katerem dve ekipi z 11 agenti igrata simulirani nogomet. Jedro okolja je prosto dostopen simulator igre *RoboCup Soccer Server System* (v nadaljevanju RCSSS) [91], zasnovan kot sistem odjemalec-strežnik (angl. *client-server*), ki simulira nogomet med računalniškimi igralci. Vsakega igralca nadzira natanko en porazdeljen klient - agent, ki je računsko neodvisen od ostalih agentov. Domeno *RoboCup Soccer Simulated League*, shematsko predstavljeno na sliki 2, sestavljajo:

- strežnik *RoboCup Soccer Server*,
- 2 moštvi po 11 agentov,
- 2 agentna trenerja (opcijsko),
- prikazovalnik igre *RoboCup Soccer Monitor* (opcijsko),
- predvajalnik shranjenih iger *RoboCup Logplayer* (opcijsko).



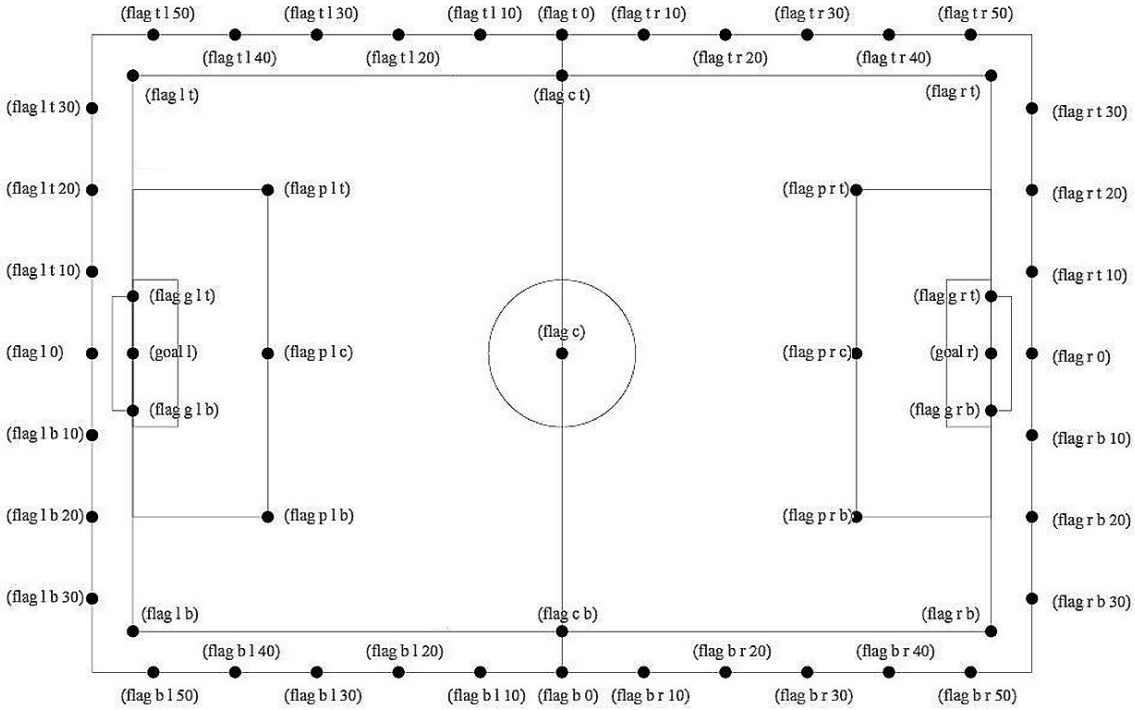
Slika 2. Shema okolja RoboCup Soccer Server System. Smeri puščic ustrezajo smeri komunikacije med entitetami v sistemu. Obvezni deli sistema so obkroženi s polno črno črto, neobvezni deli sistema pa s črtkano črto.

Ena izmed prednosti RCSSS je v abstraktnosti pristopa, saj sprosti raziskovalce, da se ne ukvarjajo s tehničnimi problemi zaznave objektov, komunikacije in strojnimi omejitvami, da dosežejo npr. premik robota. Abstrakcija tako omogoča raziskovalcem osredotočenje na višjenivojske cilje, kot so večagentno planiranje, sodelovanje in učenje.

Ker v nadaljevanju doktorata ne bomo obravnavali ostalih RoboCup lig oz. področji, bomo z omembo termina RoboCup vedno privzeli, da se le-ta nanaša le na *RoboCup Soccer Simulated League*. Prav tako bomo včasih uporabili izraz igralec, čeprav se le-ta nanaša na agenta, ki upravlja igralca v RCSSS.

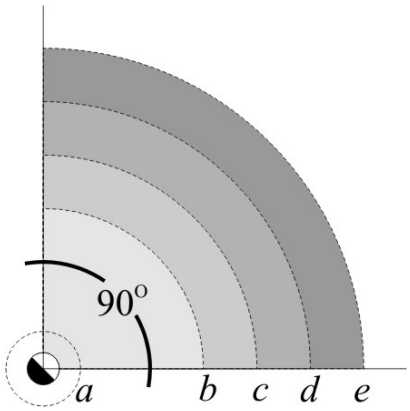
Simulirano okolje RCSSS predstavlja standardno nogometno igrišče, kot je prikazano na sliki 3. Naloga strežnika RCSSS [27] je, da simulira gibanje igralcev in žoge na igrišču ter da zagotavlja izpolnjevanje pravil nogometa [36]. Strežnik sledi

trenutnemu stanju agentov in simuliranega okolja, izvaja akcije agentov, fizikalno natančno izračunava novo stanje agentov in okolja ter pošilja agentom šumno in nepopolno informacijo o okolju. Okolje je v osnovi porazdeljeno in večagentno, saj sta edino izid in ura globalno poznani vrednosti. Za vse ostale informacije pa se morajo agenti zanesti na njihove zaznavne sposobnosti in na komunikacijo s soigralci.



Slika 3. Slika simuliranega nogometnega igrišča s pozicijskimi zastavicami, ki jih vidijo agenti. Zastavice uporabljajo agenti za določitev njihove pozicije na igrišču.

Agenti komunicirajo neposredno s RCSSS preko standardnega in dobro definiranega komunikacijskega protokola [27], ki poteka preko TCP/IP mrežnega protokola. Agenti periodično prejemajo šumno informacijo o smeri, razdalji in hitrosti objektov na igrišču (to so: ostali agenti, žoga, gola in pozicijske zastavice), vendar le za objekte, ki so v njihovem omejenem vidnem polju. Agentovo vidno zaznavanje je odvisno od širine vidnega polja (ozko - 45°, normalno - 90° ali široko - 180°) in kvalitete vida (nizka ali visoka). Z večanjem širine in kvalitete se ustrezno zmanjša frekvenca sporočanja vidnih podatkov, ki je 150 ms pri normalni širini in visoki kvaliteti. Poleg tega omejuje vidne informacije tudi oddaljenost objektov, kot prikazuje slika 4, saj obstaja več regij znotraj vidnega polja agenta. Agent zazna vse objekte znotraj črtkanega kroga z radijem ***a*** in znotraj sivega polja z radijem ***b***. Znotraj sivega polja med ***b*** in ***c*** vidijo agenti vedno ime moštva, toda številko igralca le z verjetnostjo med 1 in 0, ki pada proti 0 pri radiju ***c***. V sivem polju do radija ***d*** je vedno vidno ime moštva, ki pa se z verjetnostjo med 1 in 0 manjša proti 0, ko se oddaljenost bliža ***e***. Agent ne zazna ničesar izven vidnega polja z radijem ***e***. Privzete vrednosti teh parametrov so: ***a***=3m, ***b***=20m, ***c***=40m, ***d***=40m in ***e***=60m.



Slika 4. Prikaz normalno širokega vidnega polja agentov v RoboCup domeni.

Simulator simulira zvezno dogajanje z diskretnimi koraki, ki se v privzeti vrednosti izvajajo na 100 ms. V vsakem koraku dobi agent možnost, da sporoči RCSSS strežniku ukaz za izvajanje eno ali več parametriziranih ***osnovnih agentih akcij***, ki so:

- ***turn(moment):*** spremeni smer igralčevega telesa s hitrostjo obrata *moment*. Če se igralec ne premika, je *moment* enak spremembi kota igralca, sicer se upošteva koncept inercije, ki otežuje rotiranje telesa linearno s hitrostjo igralca.
- ***dash(power):*** pospeši igralca z močjo *power* v smeri orientacije igralčevega telesa. Vsak igralec ima na voljo določeno količino vzdržljivosti, ki se z izvajanjem akcije *dash* porablja z vrednostjo parametra *power*. Če je moč, potrebna za pospeševanje, večja od igralčeve vzdržljivosti, se moč zmanjša skladno s preostalo vzdržljivostjo. Če je *power* < 0, potem igralec pospešuje nazaj (zavira), kar pa je energijsko dvakrat bolj potratno.
- ***kick(power, direction):*** igralec brcne žogo z močjo *power* v smeri *direction* glede na smer igralčevega telesa. Moč udarca se zmanjša z oddaljenostjo igralca do žoge in z odmikom kota telesa glede na pozicijo žoge. Akcija se ne bo izvedla, če je žoga preveč oddaljena.
- ***turn_neck(angle):*** relativno glede na kot telesa spremeni kot glave (pogleda) v *angle*. Ta akcija se lahko izvaja vzporedno z ostalimi.
- ***catch(direction):*** igralec, ki opravlja vlogo vratarja, ujame žogo v smeri *direction*, če je žoga v primerni oddaljenosti od vratarja.
- ***say(message):*** igralec pošlje krajše sporočilo *message* vsem igralcem (tudi nasprotnikom). Domet sporočila je omejen z razdaljo.

Izvajanje akcij *turn*, *dash*, *kick* in *catch* se medsebojno izključuje, medtem ko se akcije *turn_neck* in *say* lahko izvajajo vzporedno. Ker se informacija o vizualnih zaznavah sporoča agentom s periodo 1,5 koraka (v privzetih nastavitvah), lahko agenti v standardni 10 minutni igri izvedejo 6000 osnovnih akcij ter prejmejo 4000 zaznav okolice.

Okolje v RoboCup domeni se lahko spremeni zelo hitro. Glede na največjo agentno hitrost (10 m/s) in omejeno velikost igrišča (diagonala je 123 m) velja ocena, da se v 15 sekundah lahko okolje spremeni v poljubno stanje.

Agenti lahko med seboj komunicirajo z uporabo akcije *say*, vendar je komunikacija količinsko omejena in nezanesljiva. V privzetih nastavitvah je domet komunikacije

omejen na radij 50 m okoli agenta (to je približno polovico igrišča), poleg tega pa lahko v vsakem koraku simulacije agent sprejme največ eno sporočilo.

Pri izračunavanju fizikalnega modela doda RCSSS šum, ki naredi domeno manj predvidljivo in bližje realnemu okolju. Manjše količine šuma se dodajo pri sporočanju pozicij objektov (objekti na igrišču, igralci in žoga), pri izračunu gibanja objektov (igralci in žoga), pri izračunu kota brce na žogo in pri upoštevanju vetra na igrišču. Zaradi šuma so agentna mnenja o lokaciji objektov v okolju lahko med seboj nekonsistentna.

Višjenivojske nogometne akcije (v nadaljevanju samo akcije), kot so npr.: podaja soigralcu, tek do določene pozicije ali preigravanja, so odigrane s časovno usklajenim kombiniranjem osnovnih agentnih akcij enega ali več agentov. Na primer, tek do določene pozicije na igrišču zahteva kombinacijo osnovnih akcij enega agenta *turn* in *dash*, medtem ko uspešna podaja žoge običajno zahteva kombinacijo osnovnih akcij *turn*, *dash* in *kick* prvega agenta ter podaji-usklajen tek do žoge drugega agenta. Tak proces zahteva časovno usklajeno izvajanje osnovnih akcij, kar v kombinaciji s šumnim zaznavanjem in izvajanjem akcij privede do nepredvidljivosti pri izvajanju višjenivojskih akcij.

Poseben agent, imenovan trener (angl. *coach agent*), nadzira igro svojega moštva in lahko pošilja ukaze agentom v posebnem trenerskem jeziku CLang [27]. Trener dobi od strežnika popolno sliko igrišča s točnimi pozicijami agentov in brez vidnih omejitev. Trenerjeva sporočila so vidna za vse člane njegove ekipe ne glede na njihov položaj, vendar je taka komunikacija mogoča le na vsakih 30 sekund oz. kadar se igra ustavi (zaradi npr. prekrška, kota, ...). Ta omejitev onemogoča mikro-nadziranje agentov in s tem trenerju vsiljuje bolj strateški nivo komuniciranja.

Ker predstavlja domena RoboCup računalniško verzija nogometa, se vloge agentov običajno modelirajo po vzoru vlog nogometnih igralcev v nogometnih moštvih. V svojem delu sta Stone in Veloso [107] na domeni RoboCup pokazala, da je pristop, kjer uporabljajo agenti različne vloge, boljši kot pristop brez vlog. Agentne vloge se določajo na podlagi trenutno izbrane formacije ekipe, ki se izbira na podlagi vnaprej določenih pogojev. Čeprav je možno statično določiti vloge agentov, je tak pristop neprilagodljiv in ne omogoča izkoriščanja nepričakovanih priložnosti [50]. Optimalna dinamična razporeditev agentnih vlog v RoboCup domeni je zato postala precejšen izziv [44].

2.2.2 Primernost domene RoboCup

Domena RoboCup je bila zasnovana z namenom, da širši znanstveni javnosti ponudi referenčno domeno s področja večagentnih sistemov. Z letnim organiziranjem svetovnih in regijskih prvenstev ter spremljajočih znanstvenih konferenc je RoboCup zveza omogočila primerjavo dosežkov različnih raziskovalnih skupin s celega sveta in s tem spodbudila znanstveno konkurenco v pozitivnem pomenu besede. Poleg tega je zaradi relativno enostavne primerjave kvalitete ekip (število golov oz. zmag na prvenstvu) ter zaradi splošne priljubljenosti nogometa domena tako postala *de facto* referenčna domena s področja večagentnih sistemov.

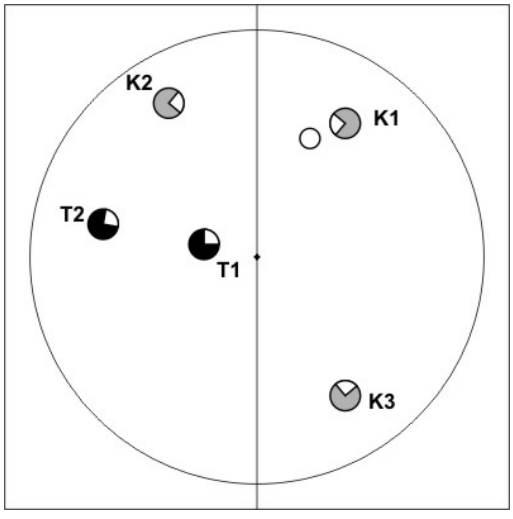
Cohen [29] celo predlaga opustitev znamenitega Turingovega testa kot klasičnega problema v umetni inteligenci. Namesto tega predlaga štiri domene, med katerimi kot

prvo izpostavi prav domeno RoboCup. Predlog utemelji s trditvijo, da ima domena vse potrebne lastnosti, ki jo opredeljujejo kot primeren problem umetne inteligence.

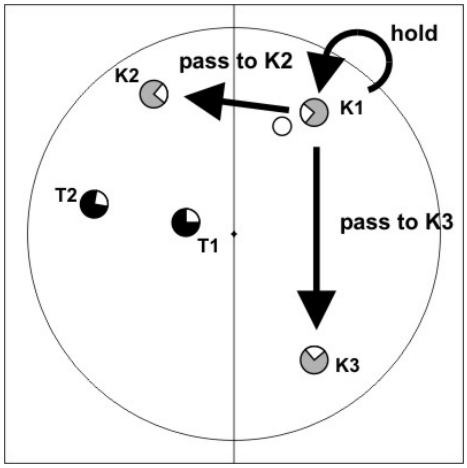
2.2.3 Domena 3vs2 Keepaway

Zaradi relativne kompleksnosti učnih problemov v domeni RoboCup so raziskovalci [104] začeli razmišljati o učenju v omejeni RoboCup domeni, ki bi vključevala manjše število igralcev. Določili so nalogo zadrževanja žoge pred napadalci (angl. *keep away*) [105] kot podproblem RoboCup domene, kjer ekipa branilcev (angl. *keepers*) poskuša obdržati posest žoge v omejenem prostoru, medtem ko ekipa napadalcev (angl. *takers*) poskuša pridobiti posest nad žogo. Cilj igre je maksimirati povprečno trajanje epizode, ki je določena kot igra, v kateri imajo posest nad žogo branilci. Če se žoge polastijo napadalci ali pa zapusti igralno polje, se epizoda konča in igra ponovno požene. Domena lahko vsebuje različno število napadalcev in branilcev ter različno veliko igralno polje. V doktoratu bomo uporabljali domeno *3vs2 Keepaway* [102], ki jo določa kvadratno igralno polje velikosti 20x20m, 3 branilci in 2 napadalca. Številčne oznake branilcev oz. oznake njihovih vlog (označeni s K1, K2 in K3) in napadalcev (označena s T1 in T2) so urejene po naraščajoči oddaljenosti do žoge, kot je to prikazano na sliki 5. Branilec z žogo ima tako vedno oznako K1.

Domena je poenostavljena tudi v smislu agentih akcij, saj so le visokonivojske. Akcije lahko izvaja le branilec, ki ima v posesti žogo (K1). Branilec K1 lahko žogo zadrži (akcija: **hold**) ali pa jo poda enemu od soigralcev (akciji: **pass to K2** in **pass to K3**). Slika 6 shematsko predstavlja vse tri možne akcije. Trajanje akcije **hold** je en časovni korak, medtem ko je trajanje podaje lahko daljše, saj je odvisno od usmerjenosti agenta K1 in poravnanosti žoge glede na smer podaje. Zaporedje osnovnih agentnih akcij pri izvedbi podaje tako vključuje več premikov in obračanj igralca, da doseže primerno pozicijo za podajo žoge. Izvajanje akcij je popolnoma avtomatsko, saj sistem sam pretvori izvajanje visokonivojskih akcij v odgovarjajoče zaporedje osnovnih agentih akcij.



Slika 5. Domena 3vs2 Keepaway. Sivi krogi z oznakami K1, K2 in K3 predstavljajo branilce (*keepers*), medtem ko črna kroga z oznakama T1 in T2 predstavljata napadalca (*takers*). Bel izrez v črnih in sivih krogih predstavlja smer pogleda agenta. Manjši bel krog predstavlja žogo.

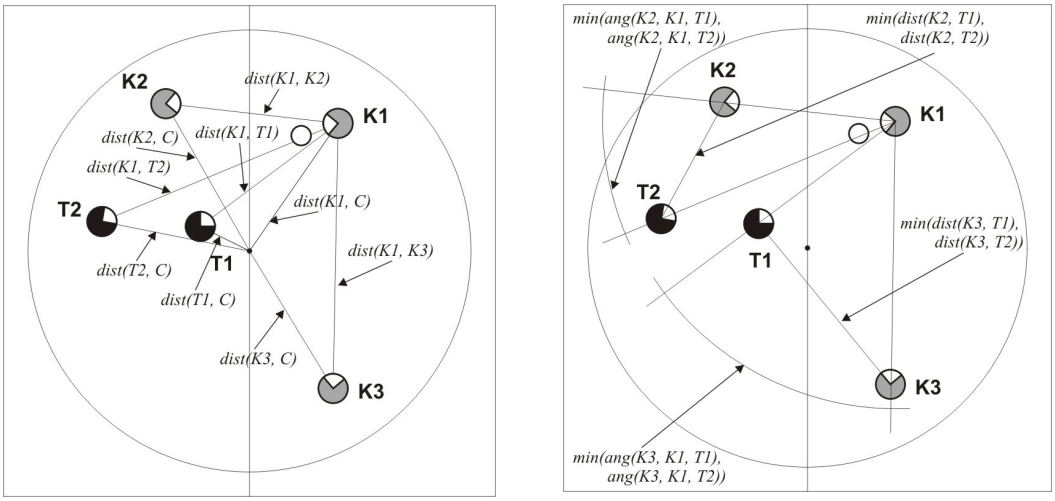


Slika 6. Možne akcije agenta z žogo K1 v 3vs2 Keepaway domeni (*hold*, *pass to K2* in *pass to K3*).

Poenostavitev je vpeljana tudi v agentnem zaznavanju okolja, saj vsi agenti "vidijo" opisan prostor z enakim naborom spremenljivk. Če je *dist(a, b)* razdalja med agentoma *a* in *b* in če je *ang(a, b, c)* kot med agenti *a* in *c* s krajiščem v agentu *b* in je *C* oznaka za središče igrišča, potem je domenski prostor opisan z naslednjimi 13 spremenljivkami[62]:

- *dist(K1, C), dist(K2, C), dist(K3, C),*
- *dist(T1, C), dist(T2, C),*
- *dist(K1, K2), dist(K1, K3), dist(K1, T1), dist(K1, T2),*
- *min(dist(K2, T1), dist(K2, T2)), min(dist(K3, T1), dist(K3, T2)),*
- *min(ang(K2, K1, T1), ang(K2, K1, T2)), min(ang(K3, K1, T1), ang(K3, K1, T2)).*

V sliki 7a so grafično prikazane vrednosti 9 spremenljivk iz prvih treh zgoraj naštetih alinej, v sliki 7b pa 4 spremenljivke zadnjih dveh alinej. Velja omeniti, da je opis okolja s podanimi spremenljivkami neodvisen od rotacije in zrcaljenja čez središče igrišča.



a) b)

Slika 7. Grafični prikaz vrednosti domenskih spremenljivk v domeni 3vs2 Keepaway.

3vs2 Keepaway okolje [65] je prosto dostopno v obliki izvorne kode za okolje Linux in omogoča preprosto vključitev agentov z različnimi strategijami delovanja. V okolju so že pripravljene naslednje tri referenčne agentne strategije:

- Vedno zadrži žogo (oznaka: **allwayshold**): agent vedno izbere akcijo **hold**.
- Naključna (oznaka: **rand**): agent naključno izbere eno akcijo.
- Ročno zgrajena (oznaka: **hand**): je preprosta, toda zelo učinkovita strategija. Če je T1 od K1 oddaljen več kot 5m (dist(K1, T1) > 5m), potem izbere akcijo **hold**. Sicer oceni kot in razdaljo med soigralci in napadalci in, če je soigralec dovolj odkrit, poda najbolj odkritemu. Če noben soigralec ni dovolj odkrit, izbere akcijo **hold**.

Omenjene agentne strategije pojmujemo kot referenčne in služijo raziskovalcem za merodajno primerjavo z njihovimi rezultati.

2.2.4 Primernost domene 3vs2 Keepaway

Namen raziskovalcev pri snovanju domene 3vs2 Keepaway je bil poenostaviti RoboCup domeno in hkrati obdržati njeno privlačnost. Če povzamemo, je 3vs2 Keepaway poenostavljen podproblem domene RoboCup, ki omogoča relativno enostavno implementacijo različnih agentnih strategij. Glavne poenostavitve so sledeče (v oklepajih so podana razmerja glede na RoboCup domeno):

- manjše število vključenih igralcev (5 igralcev : 22 igralcev + 2 trenerja),
- igra v manjšem prostoru (20 m x 20 m : 105 m x 68 m),
- majhen nabor akcij (3 višjenivojske : 6 osnovnih),
- zaznavanje okolja je za vse agente enako (13 spremenljivk : šumno ter z oddaljenostjo omejeno vidno in slušno zaznavanje),
- preproste agentne vloge - vloge se dodeljujejo glede na oddaljenost do žoge (5 vlog - K1, K2, K3, T1 in T2 : večje število nogometnih vlog, ki jih narekuje formacija, dinamika in strategija igre),
- igralci so osredotočeni samo na en cilj in jim ni potrebno izbirati med napadalnimi in obrambnimi podcilji (maksimiranje trajanja epizode : zmaga nogometne igre).

Prednosti 3vs2 Keepaway domene glede na RoboCup domeno so predvsem v enostavnejši implementaciji različnih agentnih strategij, v lažji primerjavi kvalitete agentnih strategij ter v lažji interpretaciji dobljenih rezultatov. Domena kljub poenostavitvam ohranja vse pomembne lastnosti večagentnih domen, saj posamezni agent nima možnosti, da sam reši problem, ni centralnega nadzora in izvajanje je asinhrono.

3. Pregled sorodnega dela

When I am dead, I hope it is said,
'His sins were scarlet, but his books were read'.

Hilaire Belloc

Modeliranje dinamičnih sistemov je eden od značilnih problemov umetne inteligence, zato je bila na tem področju opravljena vrsta raziskav. Ker je področje preširoko, da bi lahko omenili vse pomembne prispevke, se v tem poglavju omejimo predvsem na dela, ki se dotikajo širšega področja modeliranje večagentnih sistemov s poudarkom na razumljivosti dobljenega modela.

Pregled bomo začeli z modeliranjem v preprostejših agentnih domenah. V poteznih igrah s popolno informacijo, kot sta na primer dama in šah, je standardna tehnika preiskovanje prostora stanj z minimax postopkom in z uporabo domensko pogojenih hevurističnih funkcij, kot je na primeru dame pokazal Schaeffer s sodelavci [95]. Ko Carmel in Markovitch [23] predstavita M*, modificirano minimax iskanje z vgrajenim modelom nasprotnika, se izkaže, da v primeru neoptimalnega nasprotnika modeliranje nasprotnika pripomore k boljšim rezultatom. Kasneje Billings in sodelavci [16] pokažejo, da je v igrah, ki vključujejo zavajanje in delno poznavanje situacije, kot je na primer poker, modeliranja nasprotnika nujno za doseganje boljših rezultatov. Njihov sistem uporablja parametrizirano verzijo agentove privatne ocenjevalne funkcije, da oceni, kako nasprotnik vrednoti njihove karte. To informacijo nato uporabijo pri iskanju in ocenjevanju možnih potez.

Ponavljajoče matrične igre so prav tako popularna domena pri modeliranju agentov. Carmel in Markovitch [24, 25] sta razvila algoritem US-L* za iskanje modela nasprotnika, ki je skladen z znanim vhodom in izhodom modelirane entitete. V njunih raziskavah je interakcija med agenti predstavljena kot ponavljajoča se igra za dva igralca, kjer je cilj agentov maksimirati pričakovano vsoto nagrad v igri. Pri tem privzameta, da se lahko agentne strategije modelirajo z diskretnimi končnimi avtomati (angl. *discrete finite automatons - DFA*). Njun algoritem je sposoben zgraditi model nasprotnikovega DFA iz podatkov o njegovem vhodu in izhodu. Ker je iskanje minimalnega DFA NP-poln problem, sta razvila algoritem, ki v polinomskem času poišče DFA, ki je konsistenten z danim vhodom in izhodom. Za dani DFA model nasprotnika pa lahko v polinomskem času [76] poiščemo DFA z optimalnim odgovorom. Vendar pa transformacija dane dvoagentne domene v splošen večagentni sistem ostaja nepoznana.

Gmytrasiewicz in Durfee [46] sta formalizirala rekurzivno modeliranje, kjer vsak agent modelira tudi mišljenje ostalih agentov. To okolje je kasneje pripeljalo do zanimivih rezultatov iz teorije kompleksnosti. Fortnow in Whang [38] ter Freund s sodelavci [40] so raziskovali agentne strategije, ki so v igri proti nasprotniku s fiksnim strategijam blizu optimalnim. Preizkušene fiksne strategije so bili Turingovi stroji, logične funkcije in funkcije, ki upoštevajo statistiko zgodovine. Izkaže se, da pojem "strategije, ki je blizu optimalni" zahteva natančno definicijo in da obstaja več definicij, ki so uporabne za različne namene. Freund in Schapire [41] kasneje objavita algoritem, za katerega je povprečna izguba zagotovljeno blizu minimalni izgubi, ki jo

lahko doseže poljubna fiksna strategija. Izkaže se, da je konvergenca algoritma asimptotično optimalna in da velja za poljubnega nasprotnika.

Agentni modeli v omenjenih raziskavah sicer kažejo želene teoretične lastnosti, vendar so uporabni le na omejenih diskretnih agentnih domenah. Vendar pa ima mnogo pomembnih večagentnih domen lastnosti, ki jih ne moremo modelirati z omenjenimi diskretnimi modeli. Modeliranje domen s (skoraj) zveznim prostorom in z upoštevanjem časovne dinamike agentnih akcij zahteve drugačne modele. Podobno velja tudi za definicijo funkcije uspešnosti, ki mora biti bolj ohlapna.

Modeliranje v domenah z realnim časom (angl. *real-time*) in z zveznim domenskim ter akcijskim prostorom sta preučevala Tambe in Rosenbloom [113], ki sta v domeni bojevanja v zraku (angl. *air combat*) uporabila agente, ki sledijo nasprotnikovim akcijam in predvidevajo njihove prihodnje akcije, kar vpliva na odločitev pri izbiri trajektorije avtomatskega pilota. Njuna zaključka sta, da potrebujejo agenti lastne mehanizme, ki omogočajo modeliranje nasprotnikovih odzivov, ter da za modeliranje agentne interakcije potrebujejo skupno in celovito predstavitev. Laird [64] opiše agenta za prvoosebno 3D strelsko igro Quake II, ki z uporabo pravkar omenjenega sledenja agentom [113] poskuša predvidevati nasprotnikove akcije. Modelirano nasprotnikovo gibanje uporabi za odločitev, ali lahko prehiti nasprotnika pri pobiranju izboljšav in ali lahko pričaka nasprotnika v zasedi.

Kaminka s sodelavci v svojem delu [55] naslovi problem nenadzorovanega učenja sekvenčnega agentnega obnašanja. Opazovanja kompleksnega domenskega okolja prevede v časovno serijo prepoznanih osnovnih akcij (angl. *atomic actions*). Časovne serije nato analizirajo, da odkrijejo ponavljajoče in statistično pomembne podsekvence, ki karakterizirajo agentno moštvo. Njihovo delo je blizu našemu v smislu identificiranja pogostih akcijskih sekvenc, vendar pa je njihov pristop omejen, saj se omeji le na iskanje akcijskih zaporedij in popolnoma zanemari časovni in prostorski vidik odigranih akcij.

Kaminka in Avrahami [54] sta kasneje uvedla vedenjski pristop (angl. *behavior-based approach*) za prepoznavanje planov, v katerem so hierarhična obnašanja uporabljena za modeliranje robotskega obnašanja. Predstavita učinkovita algoritma za poizvedovanje ključnih opisov o obnašanju. Kljub uporabi hierarhije domenskih značilk, je njihov pristop omejen v tem, da jih uporablja zgolj kot knjižnico za iskanje ujemajočih opisov in ne za konstruiranje novih akcijskih opisov.

Sukthankar in Sycara v svojem delu [108] predstavita pristop z minimizacijo cene (angl. *cost minimization approach*) v problemu prepoznavanja človeškega obnašanja v vojaški domeni. Z uporabo podatkov, dobljenih iz telesnih senzorjev premika (angl. *full-body motion-capture*), je sistem sposoben prepoznati obnašanje človeka. Nizkonivojska klasifikacija je narejena z uporabo metode podpornih vektorjev (angl. *support vector machines*) in Markovega modela s skritimi stanji (angl. *hidden Markov model - HMM*). Izhod klasifikatorja je uporabljen kot vhodna značilka za prepoznavalnik obnašanja. Njihov sistem je sposoben obdelovati sekvence, ki so pogosto prekinjene in nedokončane. Opisani pristop je sposoben zaznati različno agentno obnašanje z uporabo vedenjske knjižnice, kar omejuje njegovo uporabnost, saj ni sposoben zaznati vedenja, ki ni zapisano v vedenjski knjižnici.

Na področju vzpodbujevalnega učenja (angl. *reinforcement learning*) je aktualno reševanje večjih problemov z uporabo hierarhičnih pristopov [6, 33]. Glavno vodilo pri hierarhičnih pristopih je v tem, da omogočajo agentu, da ignorira dele stanj, ki so nepomembni za trenutne odločitve [7]. Rezultati so sicer vzpodbudni, vendar pa niso uporabni pri kvalitativnem opisovanju modelov, saj je rezultat modela množica uteži, katerih pomen za človeka ni enostavno predstavljiv.

Problem predstavitve večagentnih planov je obdelovalo več raziskovalcev, ki so predlagali večagentne diagrame vpliva (angl. *multi-agent influence diagrams*) [61] in mrežne diagrame vpliva (angl. *influence diagram networks*) [109]. Toda vsi ti predlogi se ukvarjajo z modeliranjem agentnega odločanja v procesu načrtovanja večagentnih sistemov in ne z modeliranjem opaženega agentnega delovanja. Zaradi prikaza vzročnosti agentnega delovanja tudi popolnoma zanemarijo prostorski in časovni vidik izvajanih akcij in zato niso primerni za prikaz modelov agentnega obnašanja.

V zadnjem času je bilo predlaganih več modelov večagentnih sistemov [11, 18, 77, 80, 126], katerih skupna lastnost je, da je izhod modela skupna večagentna akcijska strategija, ki jo lahko razumemo kot univerzalen plan za agente. Izkaže se, da njihova prevelika splošnost povzroča visoko dimenzionalnost večagentnih modelov, kar onemogoča učinkovito grajenje in uporabo strategij.

Riley v doktorskem delu [89] obdela učenje in modeliranje agentov za potrebe dajanja nasvetov. V njem iz preprostih časovnih mrež (angl. *simple temporal network*), ki se uporabljajo za reševanje časovnih razporedov [71], razvije t.i. preproste večagentne časovne mreže (angl. *multi-agent simple temporal networks - MASTN*), ki so nov pristop za predstavitev in izvajanje večagentnega plana. Pri izvajanju plana uporablja algoritme, ki jih je predstavil Muscettola s sodelavci [72]. MASTN uvaja kompaktno obliko predstavitve skupnega večagentnega plana in poleg tega omogočajo iskanje odgovorov na vprašanje o strukturi nasveta, kar lahko učinkovito uporabi pri dajanju nasvetov agentom. Poleg tega razvije algoritem za učenje abstraktnega procesa Markovega odločanja (angl. *Markov Decision Process*), ki se uči iz sledi igre, podanih abstrakcij stanj in delnih akcijskih vzorcev. Učenje in uporaba sta eksperimentalno preverjena v različnih scenarijih, med njimi tudi v domeni RoboCup, kjer je razvil tri vrste večagentnih modelov. Prvi model upošteva plan akcij lastne ekipe in verjetnostno napoveduje gibanje nasprotnika. Naučeni model sicer uspešno izbira najbolj verjetne opise nasprotnikovega vedenja, vendar pa so le-ti ročno zgrajeni. Drugi model je sposoben opisati prostorske relacije med agenti v moštvu in tako določiti nogometno postavitev, kar uspešno uporabi pri igri z znanim nasprotnikom, ko imitira postavitve uspešnih moštev. Tretji model opisuje akcijske vzorce agentnega moštva. Z združevanjem v gruče (angl. *clustering*) in uporabo odločitvenih dreves (angl. *decision trees*) je model sposoben opisati karakteristične vzorce podaj, ki so bili zaznani v odigranih igrah. Modele uporabi za formiranje nasvetov agentom med samo igro, kar pa merljivo ne doprinese k izboljšavi igre.

Ker daje naše delo velik poudarek na človeškemu razumevanju dobljenih modelov, velja omeniti kvalitativno modeliranje (angl. *qualitative modeling*) [94], kjer je naloga zgraditi model, ki je konsistenten s podanim kvalitativnem znanjem o domeni. Za razliko od numeričnih modelov so kvalitativni modeli bolj razumljivi človeku in posledično lažje razložljivi. Šuc in Bratko [21, 111] sta razvila QUIN algoritem za kvalitativno učenje iz numeričnih podatkov. QUIN se uči kvalitativnih dreves iz

numeričnih podatkov in je bil uporabljen v vrsti dinamičnih domen, pa tudi v klasičnih, nedinamičnih domenah. Pristop so kasneje nadgradili v kvalitativno zvesto učenje [112] (angl. *Q2 learning*), kjer je numerično učenje usmerjano z naučenim kvalitativnim modelom. Pristop je za nas zanimiv predvsem zato, ker se je izkazal pri rekonstrukciji operaterjevih veščin (npr.: veščine nadziranja žerjava, acrobata in vožnje kolesa), kar je sorodno modeliranju delovanja agentov. Zaradi časovne kompleksnosti, ki je kubična glede na število atributov, pa se nam zdi pristop manj uporaben v večagentnih sistemih, saj je tam število atributov tipično zelo veliko.

3.1 RoboCup

Agentno modeliranje v domeni simuliranega robotskega nogometa so raziskovali Wünstel in sodelavci [125], ki so z uporabo t.i. samo-organizirajočih zemljevidov (angl. *self organizing maps*) klasificirali gibanje agentov. Miene in sodelavci [69] predstavijo domensko neodvisen pristop, kjer z diskretiziranim opisom stanja in akcij gradijo kvalitativne opise gibanja, ki so uporabljeni pri predvidevanju prepovedanega položaja.

V domeni RoboCup Small-Size Soccer Robot League uporabita Han in Veloso [47] ogrodje, ki uporablja HMM za predstavitev in prepoznavanja strateškega robotskega obnašanja. Stanja v HMM ustrezajo abstraktnemu modelu robotskega obnašanja. Z uporabo več naučenih HMM uspejo prepoznati različne tipe robotskega obnašanja.

Bowling s sodelavci [19] uvede taktike, igre in zbirke iger (angl. *play book*) kot sestavni del večagentnih planov. V kontekstu robotov iz RoboCup Small-Size Soccer Robot lige definira taktike kot enoagentno obnašanje, ki je večinoma odzivnega značaja. Igre so določene kot sekvence akcij in odgovarjajočih agentnih vlog, ki se lahko uporabljajo v različnih situacijah. Koncept zbirke iger je predstavljen kot metoda za brezšivno kombiniranje večagentnih planov, ki so predstavljeni kot zbirka alternativnih iger.

Stone v svojem doktoratu [100] in kasneje v knjigi [101] naslovi problem večnivojskega učenja (angl. *layered learning*) v večagentnih sistemih. Osnovna teza njegovega pristopa je, da je obnašanje večagentnih sistemov preveč kompleksno, da bi se lahko neposredno naučili preslikave iz vhodnih senzorjev do izhoda v obliki agentnih akcij. Stone zato predlaga razgraditev problema na več nivojev, od nižjenivojskih do višjenivojskih, ki vsak zase omogočajo učinkovito učenje. Učenje poteka od spodaj navzgor, kjer je naučen model na enem nivoju tudi vhod za učenje na višjem nivoju. Sam pristop večnivojskega učenja ne omogoča avtomatske hierarhične razgradnje nalog, temveč to prepušča uporabniku. Zaradi specifičnosti večagentnih domen Stone prepušča odločitev o uporabi specifičnih metod strojnega učenja uporabniku. Naučene veščine uporabi v uspešnem RoboCup moštvu [101], kjer se je prestrezanje žoge naučil z uporabo nevronske mreže z vzvratnim razširjanjem napake (angl. *back-propagation neural network*), ocenjevanje primernosti podaje z algoritmom za grajenje odločitvenih dreves C4.5 [81] in odločanje o cilju podaje z razvitim TPOT-RL (angl. *team-partitioned opaque-transition reinforcement learning*) postopkom.

Steffens [99] modelira nasprotnike v RoboCup domeni s postopkom FBDOM (angl. *feature-based declarative opponent-modelling*). Osnova njegovega pristopa so

značilke, ki jih definira kot taktične sekvence akcij. S pomočjo sklepanja na podlagi primerov (angl. *case-based reasoning*) je sistem sposoben klasificirati različne nogometne situacije. Glavni izsledki njegovih raziskav nakazujejo, da lahko večagentne taktike identificiramo zgolj z manjšim številom tipičnih značilk, ki so neodvisne od trenutnega nasprotnika. Njegov zaključek je, da lahko model neke ekipe dobimo zgolj z opazovanjem iger ene ekipe s poljubnim nasprotnikom.

Na domeni RoboCup so bile uspešno uporabljene tudi ostale tehnike strojnega učenje. Dve raziskavi sta poskušali z genetskim programiranjem (angl. *genetic programming*) naučiti celotno moštvo v RoboCup domeni [8, 66]. Čeprav sta bila poskusa v začetku namenjena pokazati, da je genetsko programiranje sposobno opisati nalogo polnega kooperativnega delovanja agentov, je prvi sistem uspel zgolj izboljšati ročno grajene nizkonivojske akcije. Drugemu sistemu je uspelo uspešno modelirati nekaj igralčevih obnašanj, ne pa tudi ostalih kooperativnih obnašanj.

Tudi nevronske mreže (angl. *neural networks*) in odločitvena drevesa (angl. *decision trees*) so bila uporabljena na različnih nogometnih podproblemih, kot je na primer odločanje o strelu ali podaji v bližini gola [74] in učenje, kako podajati in prestrezati žogo [68].

Čeprav v domeni RoboCup potekajo obsežne raziskave o kvantitativnem modeliranju (angl. *quantitative modeling*), se je s problemom konstruiranja človeško berljivih opisov večagentne aktivnosti ukvarjalo relativno malo raziskav. Omenimo delo Rainesa, ki je s sodelavci [83] razvil avtomatskega asistenta za analizo agentnih moštev. Njihov sistem ISAAC izvaja naknadno (angl. *off-line*) analizo moštev iz zapisov agentnega delovanja. Analiza uporablja pristop od spodaj navzgor (angl. *bottom-up*), ki vključuje tehnike odkrivanja znanj iz podatkov (angl. *data mining*) in induktivne učne tehnike. ISSAC je sposoben odkriti zanimive vzorce, ki so osnovani na uporabniško podanih značilkah, torej kombinira analitične sposobnosti sistema z domenskim znanjem uporabnika.

Kuhlmann in sodelavci so v svojih raziskavah [63] v RoboCup domeni predstavili problem formuliranja nasvetov (angl. *advice-giving*) kot učni problem. Sposobni so se naučiti in predvideti agentno obnašanje na osnovi preteklih opažanj in avtomatsko generirati nasvete za izboljšavo delovanja njihovega moštva. Čeprav se njihovo delo skoncentrira na problem dajanja nasvetov, uspešno skonstruirajo pravila, ki napovedujejo nekatere naslednje pomembne akcije.

Iz domen sorodnih robotskemu nogometu velja omeniti delo Hirana in Tsumota [48], ki sta predstavila novo metodo za iskanje zanimivih vzorcev podaj, ki jih dobita iz ročno označenih posnetkov nogometnih tekem. Z upoštevanjem dveh značilnosti sekvence podaj - časovne neenakosti in zahteve po večstopenjskem (angl. *multiscale*) opazovanju - so razvili metodo za primerjanje sekvenc, ki je osnovana na večstopenjskem ujemanju (angl. *multiscale matching*). Metoda z uporabo hierarhičnega združevanje v gruče (angl. *hierarchical clustering*) zgradi gruče prostorsko podobnih sekvenc podaj. Eksperimentalni rezultati, dobljeni iz zapisov nogometnih tekem, kažejo da je metoda sposobna odkriti zanimive prostorske vzorce podaj, ki jih lahko povežemo z uspešnimi goli. Njihovo delo je podobno našemu v smislu odkrivanja prostorsko podobnih sekvenc podaj in uporabe hierarhičnega združevanja v gruče. Za razliko od našega dela pa je omenjeni pristop omejen le na

iskanje podobnih sekvenc podaj in popolnoma zanemari ostale vidike večagentnega modeliranja.

V domeni ameriškega nogometa obravnava Intille v svojem doktorskem delu [53] nalogo prepoznavanja večagentnih akcij. Iz podanih trajektorij izračuna lokalne dogodke (angl. *local events*). Z uporabo zbirke planov, kjer poveže lokalne dogodke s časovnimi relacijami (npr.: pred, potem), domenskim znanjem o ameriškem nogometu in propagiranem nezanesljivega sklepanja (angl. *uncertain reasoning*) je njegov sistem sposoben določiti, ali je predstavljen del igre primer naučene akcije ali ne. Pomanjkljivost njegovega pristopa je predvsem v tem, da uporablja ročno zgrajeno zbirko planov in ni zmožen odkriti novega obnašanja.

3.2 Keepaway

Keepaway domena v povezavi z RoboCup Soccer Server simulatorjem je bila že leta 2001 predlagana kot testno okolje s področja strojnega učenja [105]. Od takrat je bila uporabljena v različnih raziskavah iz širšega področja strojnega učenja: časovno diferenčnega vzpodbujevalnega učenja s funkcijsko aproksimacijo (angl. *temporal difference reinforcement learning with function approximation*) [106], evolucijskega učenja [78], relacijskega vzpodbujevalnega učenja (angl. *relational reinforcement learning*) [118] in prenosa obnašanja (angl. *behavior transfer*) [114].

Whiteson & Stone [121] sta uporabila t.i. neuroevolucijo (angl. *neuroevolution*) pri učenju branilcev v domeni SoccerBots [10]. Igralci so se bili zmožni naučiti več konceptualno različnih nalog od osnovnih veščin do višjenivojskega sklepanja z uporabo hierarhičnega pristopa, ki ga poimenujeta "*concurrent layered learning*". Ročno zgrajeno odločitveno drevo je bilo uporabljeno na najvišjem odločitvenem nivoju. Branilce sta ocenjevala glede na število uspešnih podaj. Hsu & Gustafson [51] sta predelala branilce iz 3vs2 Keepaway domene v dosti bolj preprosto in abstraktno domeno, imenovano TeamBots simulator [9], kjer se igralci premikajo po mreži in izvajajo diskretne akcije. Napadalci se premikajo dvakrat hitreje kot branilci in žoga potuje dvakrat hitreje kot napadalci. Branilce sta naučila, da minimizirajo število preobratov v igrah s fiksnim trajanjem. Primerjava omenjenih pristopov z našim je težavno, ker uporabljajo različno funkcijo uspešnosti in drugačno dinamiko igre.

Pietro in sodelavci [78] so uporabili evolucijske algoritme (angl. *evolutionary algorithms*) pri učenju treh branilcev in dveh napadalcev v RoboCup simulatorju. Podobno kot pri nas so se raziskovalci koncentrirali na učenje akcij branilcev, ki imajo v posesti žogo. Na voljo so imeli enak nabor višjenivojskih akcij kot v našem primeru. Za mero uspešnosti uporabljajo, tako kot mi, povprečno dolžino epizode. Ker so njihove višjenivojske akcije in nižjenivojsko obnašanje implementirane neodvisno, je primerjava rezultatov s to disertacijo zapletena.

Riedmiller je s sodelavci [88] uporabil vzpodbujevalno učenje (angl. *reinforcement learning*) za učenje nižjenivojskih veščin, kot so strel, prestrezanje in preigravanje, kot tudi višjenivojskega sodelovanja agentov, v katerem 2 napadalca preigravata 2 branilca. Njihov pristop uporablja celoten senzorski prostor (angl. *full sensor space*) kot vhod v nevronsko mrežo, ki je uporabljena za aproksimiranje funkcij. Pri fiksnih obnašanjih so se uspeli naučiti sodelovanje dveh napadalcev. Kasneje so pristop nadgradili in uspeli naučiti 3 napadalce proti 4 branilcem v več različnih napadalnih

scenarijih [87]. Te raziskave so del raziskav, katerih cilj je implementirati celotno nogometno moštvo z uporabo različnih tehnik vzpodbujevalnega učenja.

V prvem iz serije člankov Stone in Sutton [104] opišeta učenje v Keepaway domeni z epizodnim SMDP Sarsa(λ) algoritmom za vzpodbujevalnim učenjem in z uporabo linearno prekrivnih kodirnih funkcijskih približkov (angl. *linear tile-coding function approximation*) CMAC [3]. Odločita se, da bosta modelira višjenivojske akcije, ki so bile implementirane v moštvu CMUnited-99 [103]. Čeprav je agentno vidno polje običajno šumno in omejeno na 90°, sta v poskusih uporabila poenostavitev v smislu 360° pogleda in brezšumnega ter dolžinsko neomejenega zaznavanja. Učenje agentov je potekalo neodvisno drug od drugega. V poskusih sta poleg tega neodvisno spreminjala velikost učnega in testnega igralnega polja. Pokazala sta, da se je njun pristop sposoben naučiti strategije igre, ki je boljša od preprostih ročno kodiranih strategij.

Stone in sodelavci so kasneje dopolnili njihov Keepaway simulator ter ga kot Keepaway okolje prosto ponudili raziskovalni skupnosti [102, 105]. Okolje raziskovalcem olajša implementacijo različnih poskusov ter omogoča primerjavo rezultatov.

Z uporabo Keepaway okolja so Kuhlmann in sodelavci [62] razširili pristop iz [104], tako da so ponovno uvedli običajno omejitev agentnega zaznavanja ter spreminjali velikosti ekip. V poskusih so testirali ekipe velikosti 3:2, 4:3 in 5:4. Njihovo ključno odkritje je zaključek, da je učenje skupine agentov težje od učenje enega agenta. Poskusi so pokazali, da se en agent v okolju z že naučenimi agenti uči signifikantno hitreje kot pri simultanem učenju celotne skupine agentov, čeprav se v obeh primerih vsak agent uči neodvisno od ostalih. Toda učenje dveh ali treh agentov v okolju z že naučenimi agenti ni nič hitrejše, kot učenje celotne skupine. Rezultati torej kažejo, da je, vsaj v tej domeni, večagentno učenje v osnovi težje kot učenje enega samega agenta. Število agentov sicer ne vpliva na kvaliteto naučene strategije, temveč zgolj na hitrost učenja. Kljub obetavnim rezultatom pa ima njihov pristop pomembno pomanjkljivost - zaradi pristopa z vzpodbujevalnim učenjem raziskovalci niso bili sposobni interpretirati naučenih strategij. Avtorji tako sami priznavajo, da nimajo primerne razlage naučenih strategij. Njihovo razlago so okvirno podali zgolj z gledanjem posnetih iger in ne z razumevanjem naučenega modela.

4. Modeliranje večagentnih sistemov

V splošnem je poveljevati mnogim kot poveljevati peščici; treba jih je le prav razporediti v skupine.

Sun Tzu, Umetnost vojskovanja

4.1 Problem

Problem modeliranja večagentnih sistemov lahko obravnavamo kot problem modeliranja skupine agentov ali kot problem modeliranja posameznih agentov. V prvem primeru je poudarek na modeliranju skupnega delovanja, medtem, ko je poudarek v drugem primeru na modeliranju delovanja posamičnega agenta. Poleg tega lahko modeliramo večagentne aktivnosti na različnih nivojih abstrakcije. Najvišje v hierarhiji so cilji, ki jih želi večagentni sistem doseči. Do ciljev vodijo strategije, ki so v kompleksnih večagentnih domenah sestavljene iz različnih podstrategij oz. taktik. Le te so sestavljene iz višjenivojskih akcij, ki so sestavljene iz osnovnih agentnih akcij. Delitev na osnovne in višjenivojske akcije na prvi pogled ni smiselna, vendar pomembno olajša modeliranje. Tu lahko predstavimo primer iz človeškega sveta, kjer gibi rok in nog predstavljajo osnovne človeške akcije. Izmenično premikanje leve in desne noge, ter nato odpiranje vrat lahko tako predstavlja naslednje višjenivojske akcije: hoja do avta, sprehod do trgovine ali obisk zdravnika. Pri tem tudi velja, da osnovne in višjenivojske agentne akcije opisujejo delovanje zgolj enega agenta, medtem ko taktike in strategije opisujejo skupno delovanje več agentov. Primer hierarhije opisov večagentnih aktivnosti je prikazan na sliki 8.

Slika 8. Hierarhija opisov večagentnih aktivnosti.

V našem primeru smo preučevali problem modeliranja skupine agentov in podrobneje, modeliranje strateškega in taktičnega obnašanja skupine agentov. Ker je težko natančno določiti, kaj je strategija in kaj samo taktika, bomo s terminom ***modeliranje strateškega delovanja*** vedno vključevali tudi modeliranje taktik, torej ne bomo striktno ločevali modeliranja strateškega in taktičnega delovanja. Strateška dejavnost je tista agentna dejavnost, ki neposredno sledi iz strategije večagentnega sistema. ***Model strateškega delovanja*** je opis višjenivojskih napotkov za delovanje agentov, ki so del večagentne strategije.

Strategija tako točno ne opredeli agentnega obnašanja, temveč zgolj opiše višjenivojsko obnašanje, ki je potrebno za doseganja končnih ciljev. Pri tem velja, da so nižjenivojski opisi potrebni za opis višjenivojskih oz. da so višjenivojski opisi sestavljeni iz nižjenivojskih.

Na primer, v domeni RoboCup so strategije moštev sestavljene iz taktik za napadalne in obrambne situacije. Čeprav se nanašajo na različne situacije, med seboj vseeno niso neodvisne. Če na primer obrambna taktika zahteva specifično postavitev igralcev na igrišču, ki ni optimalna za napadalno situacijo, potem morajo igralci ob prevzetju žoge opraviti časovno potratno prerazporeditev. To pomeni, da trener (oz. tisti, ki določa strategijo) ne more poljubno prosto izbirati različnih delov strategij, ne da bi preučil učinka v ostalih situacijah. Primer akcijskih opisov v domeni RoboCup je predstavljen v preglednici 3.

Akcijski opisi	Primeri iz domene RoboCup
Cilji	glavni cilj je zmaga nad nasprotnikom, ki je sestavljen iz dveh podciljev: zadeti nasprotnikov gol in preprečiti lasten gol
Strategije in taktike	obrambne (individualna obramba, conska obramba, ...) in napadalne taktike (prodor po krilih, streli v kot gola, ...)
Višjenivojske akcije	preigravanje, podaja igralcu, strel na gol, ...
Osnovne agentne akcije	turn, dash, kick, turn_neck, catch in say

Preglednica 3. Primeri opisov v domeni RoboCup za različne nivoje abstrakcije.

Pri modeliranju večagentnih sistemov ne smemo zanemariti interakcije med agenti. Opis strategije mora tako upoštevati, da na primer v domeni RoboCup podaja žoge med igralcema zahteva strel prvega igralca in usklajeno prestrezanje žoge drugega igralca. Analiza igre mora prav tako slediti namenom več igralcev in ločiti morebitne dvoumne dogodke, kot je na primer premikanje igralca, ki ga lahko razumemo kot kritje nasprotnika, zasedanje obrambne pozicije ali odkrivanje za sprejem žoge. Bistvo modeliranja večagentnih sistemov je prav v odkrivanju teh vzorcev medsebojne interakcije.

Posebno pozornost pri modeliranju večagentnih sistemov moramo posvetiti agentnim vlogam. Ker določajo agentne vloge osnovne vzorce obnašanja, je za osnovno enoto modeliranja smiselno vzeti agentno vlogo. Opis večagentnega delovanja tako ne opisuje fizičnih instanc agentov, temveč zgolj njihove vloge. Tak pristop je primerljiv s človeškimi opisi realnega sveta, kjer pri opisovanju človeškega delovanja običajno opisujemo akcije v povezavi s človeškimi vlogami. Tako na primer rečemo, da nas je ustavil policist ali operiral kirurg. Imena oseb uporabljamo zgolj v primeru, ko želimo eksplicitno poudariti specifično osebo. Tak način opisovanja agentov bomo prevzeli tudi v našem delu, kjer bomo opisovali delovanje in sodelovanje agentnih vlog in ne posameznih instanc agentov.

Ker v disertaciji omenjamo več terminov, ki so si na prvi pogled podobni, jih bomo najprej opisno predstavili. ***Agentna akcija*** oz. na kratko ***akcija*** je opis neke zaključene agentne dejavnosti. Akcija ima lahko parametre, ki spreminjajo izvajanje agentne dejavnosti. ***Agentna vloga*** oz. na kratko ***vloga*** je popis agentnega obnašanja v

večagentnem sistemu, ki določa tudi nabor možnih agentnih akcij. Ko agent prevzame določeno vlogo, je njegovo obnašanje opisano s pravili, ki jih določa vloga. ***Instanca akcije*** je udejanjena (opravljena oz. odigrana) agentna akcija, ki jo je opravil nek agent. Predstavljena je s parom ***agent:akcija***. Kjer točna instanca agenta ni pomembna, je predstavljena s parom ***vloga:akcija***. ***Akcijski koncept*** je opis akcije, ki se ga želimo naučiti in opisuje agenta oz. njegovo vlogo ter agentno akcijo v nekem omejenem prostoru stanja in parametrov akcije. ***Makroakcija*** je neko neprekinjeno zaporedje akcij, ki je del agentne strategije. ***Koncept makroakcije*** je opis makroakcije, ki ga želimo določiti na osnovi primerov zaporedij akcijskih instanc. ***Večagentni model*** je model delovanja večagentnega sistema. ***Agentna strategija*** oz. na kratko ***strategija*** je nabor pravil, ki določajo način doseganja ciljev.

4.2 Formalni opis modeliranja večagentnih sistemov

Definicija 2. ***Osnovni večagentni sistem*** MAS-basic je določen z <AG, A^{basic}, S^{basic}, F^{basic}, G^{basic}, h>, kjer je:

– AG = {ag_1, ag_2, ..., ag_n} množica agentov;

– A^{basic} = { a_0^{basic}, a_1^{basic}, ..., a_l^{basic} } množica osnovnih agentnih akcij, ki vsebuje posebno akcijo a_0^{basic}, ki ne izvaja nič;

– S^{basic} = { s_1^{basic}, s_2^{basic}, ..., s_z^{basic} } množica stanj osnovnega večagentnega sistema, kjer je vsako stanje $s_i^{basic} \in S^{basic}$ sestavljeno iz stanja okolja $s_{env} \in S_{env}$ in osnovnih stanj vseh agentov $s_{ag_j}^{basic} \in S_{ag_j}^{basic}$, j = 1, 2, ..., n:

$$S^{basic} = S_{env} \times S_{ag_1}^{basic} \times S_{ag_2}^{basic} \times \ ... \times S_{ag_n}^{basic},$$

kjer vsako osnovno stanje agenta $s_{ag_j}^{basic} \in S_{ag_j}^{basic}$ opisuje agentovo notranje stanje ter vključuje informacijo o agentovi identiteti $ag_j \in$ AG in lastni osnovni predstavi okolja (vrednosti njegovih osnovnih zaznavnih senzorjev);

– F^{basic} = { f_1^{basic}, f_2^{basic}, ..., f_n^{basic} } množica funkcij nižjenivojske zaznave okolja posameznih agentov, kjer f_i^{basic} glede na stanje okolja $s_{env} \in S_{env}$ ustrezno popravi lastno predstavo o okolju v osnovnem stanju agenta $s_{ag_i}^{basic} \in S_{ag_i}^{basic}$:

$$f_i^{basic} : S_{env} \times S_{ag_i}^{basic} \rightarrow S_{ag_i}^{basic};$$

– G^{basic} = { g_1^{basic}, g_2^{basic}, ..., g_n^{basic} } možica nižjenivojskih odločitvenih funkcij posameznih agentov, kjer je g_i^{basic} odločitvena funkcija agenta ag_i, ki na podlagi lastnega osnovnega stanja izbere eno osnovno agentno akcijo:

$$g_i^{basic} : S_{ag_i}^{basic} \rightarrow A^{basic};$$

– h funkcija prehodov stanj sistema, kjer je novo stanje sistema posledica skupnega učinka izvajanih akcij vseh agentov:

$$h: S_{env} \times (A^{basic})^n \rightarrow S_{env}.$$

Osnovni večagentni sistem deluje na sledeč način. Za vsakega agenta $ag \in$ AG se najprej izvede funkcija nižjenivojske zaznave okolja f_{ag}^{basic} in nato še nižjenivojska odločitvena funkcija g_{ag}^{basic}, ki na podlagi lastne nižjenivojske strategije izbere eno osnovno agentno akcijo. Ko sistem dobi odločitve o osnovnih agentnih akcijah vseh agentov, izvrši funkcijo *h*, ki izračuna skupni učinek osnovnih agentnih akcij in ustrezno spremeni stanje okolja. Opisani postopek predstavlja en korak izvajanja osnovnega večagentnega sistema, ki se ponavlja, dokler izvajanja ne prekine uporabnik oz. dokler sistem ne pride v uporabniško določena končna stanja.

Primer 1: Sistem, ki ga predstavlja RoboCup Soccer Simulated League, opisan na strani 7, je osnovni večagentni sistem MAS-basic, ki je določen z <AG, A^{basic}, S^{basic}, F^{basic}, G^{basic}, *h*>, kjer je:

- AG = {$levi_1$, $levi_2$, ..., $levi_{11}$, $desni_1$, $desni_2$, ..., $desni_{11}$} množica 22 agentov iz dveh moštev (levi in desni);
- A^{basic} = {*turn(moment), dash(power), kick(power, direction), catch(direction), say(message), turn_neck(angle)*} množica osnovnih agentnih akcij s parametri;
- S^{basic} množica stanj celotnega osnovnega večagentnega sistema, ki je sestavljena iz stanja okolja S_{env}, ki vsebuje informacijo o trenutni igri (rezultat, čas igre, stanje igre), fizikalnih lastnostih agentov in žoge (položaj, velikost, orientacija, vektor hitrosti, masa) ter osnovnih stanj agentov S_{ag}^{basic}, ki vključujejo informacijo o agentovi identiteti $ag \in$ AG in lastni predstavi o stanju na igrišču;
- F^{basic} = {f_1, f_2, ..., f_{22}}, $f_1 = f_2 = ... = f_{22}$, množica funkcij nižjenivojske zaznave posameznih agentov, ki predstavlja vidno in slušno zaznavo agentov v RoboCup okolju. Vidna zaznava je šumna in prostorsko omejena, kot je to prikazano na sliki 4, stran 9;
- G^{basic} = {g_1, g_2, ..., g_{22}} možica nižjenivojskih odločitvenih funkcij posameznih agentov; funkcije so implementirane v obliki podprograma v agentnih programih, ki se poženejo pred začetkom igre in med delovanjem vsak simulacijski korak sporočijo svoje odločitve o izbranih osnovnih akcijah strežniku RoboCup Soccer Server System, kot je to prikazano na shemi okolja na sliki 2 na strani 7;
- *h* funkcija prehodov stanj sistema, ki jo določa simulator RoboCup Soccer Server System, ki simulira gibanje igralcev in žoge na igrišču ter zagotavlja izpolnjevanje pravil nogometa.

Definicija 3. ***Večagentni sistem*** MAS je določen z <AG, A^{basic}, S^{basic}, F^{basic}, *h*, A, R, S, F, G, G'^{basic}>, kjer je:

- AG množica agentov, A^{basic} množica osnovnih akcij, S^{basic} množica stanj osnovnega večagentnega sistema, F^{basic} množica funkcij nižjenivojske zaznave posameznih agentov in *h* funkcija prehodov stanj osnovnega večagentnega sistema, kot jih določa definicija osnovnega večagentnega sistema;
- A = {a_0, a_1, ..., a_k} množica agentnih akcij, kjer je $a_i = a_i$(*precondition*, *parameters*, *effect*), kjer *precondition* določa pogoje za pričetek izvajanja akcije, *parameters* določa parametre akcije in *effect* končni učinek akcije, predstavljen kot opis sprememb stanj po zaključku akcije; akcija a_0 je določena kot a_0(*"vedno"*, *""*, *"ni sprememb"*) in rezultira v osnovni agentni akciji a_0^{basic}, ostale akcije a_i, $i >$ 0, pa rezultirajo v zaporedju osnovnih agentnih akcij: $a_{i,1}^{basic}$, $a_{i,2}^{basic}$, ... , $a_{i,p}^{basic}$, kjer $a_{u,v}^{basic} \in A^{basic}$;
- R = {r_1, r_2, ..., r_m} množica agentnih vlog;

– $S = \{s_1, s_2, ..., s_z\}$ množica stanj večagentnega sistema MAS, kjer je vsako stanje $s \in S$ sestavljeno iz stanja okolja $s_{env} \in S_{env}$ in notranjih stanj agenta $s_{ag_i} \in S_{ag_i}$, ki vključujejo informacijo o njegovi identiteti $ag_i \in AG$, njegovi trenutni vlogi $r \in R$ in lastni predstavi okolja (vrednosti njegovih zaznavnih senzorjev):

$$S = S_{env} \times S_{ag_1} \times S_{ag_2} \times ... \times S_{ag_n};$$

– $F = \{f_1, f_2, ..., f_n\}$ množica funkcij višjenivojske zaznave posameznih agentov, kjer f_i glede na osnovno stanje agenta $s_{ag_i}^{basic} \in S_{ag_i}^{basic}$ ustrezno popravi vlogo in lastno predstavo okolja v notranjem stanju agenta $s_{ag_i} \in S_{ag_i}$:

$$f_i : S_{ag_i}^{basic} \times S_{ag_i} \rightarrow S_{ag_i};$$

– $G = \{g_1, g_2, ..., g_n\}$ množica višjenivojskih odločitvenih funkcij posameznih agentov, ki predstavljajo agentovo višjenivojsko strategijo. Posamezna odločitvena funkcija $g_i \in G$ glede na agentovo strategijo in njegovo notranje stanje izbere akcijo $a \in A$, pri čemer mora biti pogoj za pričetek akcije *a.precondition* izpolnjen:

$$g_i: S_{ag_i} \rightarrow A;$$

– $G'^{basic} = \{g'^{basic}_1, g'^{basic}_2, ..., g'^{basic}_n\}$ množica prevedbenih funkcij, kjer vsaka funkcija g'^{basic}_i prevede izvajanje trenutne agentne akcije $a \in A$ v nižjenivojsko agentno akcijo, ki je skladna s podanimi parametri akcije *a.parameters* in naj bi privedla do uresničenega končnega učinka akcije *a.effect*:

$$g'^{basic}_i : S_{ag_i}^{basic} \times A \rightarrow A^{basic}.$$

Večagentni sistem deluje na dveh nivojih. Višji nivo na podlagi višjenivojskega znanja in višjenivojskega odločitvenega procesa izbere agentno akcijo, ki se izvrši na nižjem nivoju kot zaporedje osnovnih agentnih akcij. Osnovne akcije a_i^{basic}, funkcije nižjenivojske zaznave f_i^{basic} in funkcija prehodov stanj *h* se izvršijo v vsakem koraku izvajanja MAS kot pri MAS-basic s to razliko, da se namesto g_i^{basic} uporablja g'^{basic}_i, ki izbere naslednjo osnovno akcijo z upoštevanjem trenutno izvajane višjenivojske akcije a_i. Zaporedje izvajanj nižjenivojskih akcij posameznega agenta se prekine, ko je višjenivojska akcija a_i opravljena oz. ko je prišlo do nezmožnosti njene izvedbe. Tedaj se izvrši agentova funkcija višjenivojske zaznave f_i in funkcija višjenivojske odločitve g_i, ki določi novo višjenivojsko akcijo. Nato se začne izvrševati nova akcija z novim zaporedjem osnovnih agentnih akcij. Enaka akcija pri istem agentu se glede na okolje in interakcijo z ostalimi agenti lahko izvede v različno dolgem zaporedju različnih osnovnih agentnih akcij.

Definicija 4. ***Strategija*** MAS <AG, A^{basic}, S^{basic}, F^{basic}, *h*, A, R, S, F, G, G'^{basic}> je množica odločitvenih funkcij G.

Primer 2: Domeno RoboCup, ki jo uporabljamo v disertaciji, modeliramo kot večagentni sistem, ki je določen z <AG, A^{basic}, S^{basic}, F^{basic}, *h*, A, R, S, F, G, G'^{basic}>, kjer so:

- AG, A^{basic}, S^{basic}, F^{basic}, in *h* določeni enako kot v primeru 1,
- A = {Intercept(*precondition*= "žogo ima nasprotnik v bližini agenta", *parameters*= "vložena energija", *effect*= "žoga v lasti agenta"), Pass(*precondition*= "žogo ima v lasti agent", *parameters*= "prejemnik žoge", *effect*= "žoga v lasti prejemnika žoge"), ...}, ostale akcije so tu navedene le poimensko in so podrobneje predstavljene v preglednici 32 v prilogi A na strani 138: Kick, To-space, Through-pass, Straight-through-pass, Diagonal-through-pass, Cross-pass, Square-pass, To-player, Successful-shoot, Miss, Goal-line, Corner, Sideline, To-opponent, Unsuccessful-shoot, Movement, Offensive-movement, Movement-on-the-ball, Long-dribble, Speed-dribble, Control-dribble, Movement-off-the-ball, Attack-support, Creating-space, Undecided-movement, Run-for-ball, Defensive-movement, Tackle, Cover, Defensive-support, An-opponent, Passing-on, Tracking in Area-of-the-field;
- R = {GK, FB, R-FB, C-FB, RC-FB, LC-FB, L-FB, ST, SW, MF, R-MF, C-MF, RC-MF, LC-MF, L-MF, FW, C-FW, R-FW, RC-FW, LC-FW, L-FW} množica agentnih vlog, ki so podrobneje predstavljene v preglednici 33 v prilogi A,
- S množica stanj večagentnega sistema, ki je sestavljena iz stanja okolja S_{env}, določenega enako kot v primeru 1, in notranjih stanj agentov S_{ag}, ki vključujejo informacijo o agentovi identiteti ag $\in$ AG, trenutni agentni vlogi r $\in$ R in višjenivojski predstavi o stanju na igrišču;
- F= {f_1, f_2, ..., f_n} je množica funkcij višjenivojske zaznave agentov, ki je pogojena z implementacijo agentov in predstavlja pretvorbo nižjenivojskega opisa agentnega stanja v modificiran višjenivojski opis;
- G = {g_1, g_2, ..., g_n} možica višjenivojskih odločitvenih funkcij posameznih agentov, ki so implementirane v obliki agentnih podprogramov, ki izbirajo agentne akcije na podlagi implementiranih nogometnih strategij.
- G'^{basic} je množica nižjenivojskih odločitvenih funkcij posameznih agentov, ki pretvarjajo izvajanje agentnih akcij v izvajanje dinamičnega zaporedja osnovnih agentnih akcij.

Primer 3: 3vs2 Keepaway okolje, predstavljeno v poglavju 2.2.3 na strani 11 je večagentni sistem <AG, A^{basic}, S^{basic}, F^{basic}, *h*, A, R, S, F, G, G'^{basic}>, kjer so:

- AG {keeper1, keeper2, keeper3, taker 1, taker2} množica 5 agentov,
- A^{basic}, S^{basic}, F^{basic}, *h* določeni enako kot v primeru 1,
- A={hold(*precondition*= "žoga v lasti K1", *parameters*= "", *effect*= "žoga v lasti K1"), pass to K2(*precondition*= "žoga v lasti K1", *parameters*= "", *effect*= "žoga v lasti K2"), pass to K3(*precondition*= "žoga v lasti K1", *parameters*= "", *effect*= "žoga v lasti K3")} množica agentnih akcij,
- R={K1, K2, K3, T1, T2} množica agentnih vlog,
- S množica stanj večagentnega sistema, sestavljeno iz stanja okolja S_{env}, določenega enako kot v primeru 1, in notranjih stanj agentov S_{ag}, ki so predstavljeni z vrednostmi 13 spremenljivk, ki opisujejo 3vs2 Keepaway domenski prostor, kot je to opisano na strani 12;
- G = {g_1, g_2, ..., g_n}, g_1=g_2=g_3, g_4=g_5, možica višjenivojskih odločitvenih funkcij posameznih agentov, kjer g_1, g_2 in g_3 predstavljajo odločitvene funkcije branilcev in g_4 in g_5 odločitvene funkcije napadalcev; odločitvene funkcija g_1=g_2=g_3

branilcev je implementirana v obliki agentnega podprograma, ki izbira agentne akcije na podlagi keepaway strategije - primeri referenčnih keepaway strategij so predstavljeni na strani 13, primer kompleksnejše keepaway strategije *hand3-6-9* pa je v obliki algoritma predstavljen na sliki 70;

– $F = \{f_1, f_2, ..., f_5\}$, $f_1 = f_2 = ... = f_5$, množica funkcij zaznave posameznih agentov, ki stanje okolja S_{env} preslika v lastno predstavo okolja s 13 spremenljivkami;
– G'^{basic} je množica nižjenivojskih odločitvenih funkcij posameznih agentov, ki pretvarjajo izvajanje agentnih akcij v zaporedje osnovnih agentnih akcij.

Bistvena razlika med ***osnovnim večagentnim sistemom*** in ***večagentnim sistemom*** je v načinu izvajanja agentnih akcij. V prvem primeru so akcije osnovne in trajajo natanko eno časovno enoto oz. en korak. Akcije vseh agentov v sistemu v enem koraku skupaj s spremembami okolja določajo naslednje stanje sistema. Zato lahko tak sistem modeliramo kot enoagentnega, kjer "super agent" izvaja "super akcije", ki predstavljajo osnovne akcije za vse agente iz modeliranega sistema v nekem trenutku. Akcije v večagentnem sistemu pa so višjenivojske in so sestavljene iz zaporedja osnovnih akcij, ki se spreminja skladno z doseganjem želenega učinka akcije ter agentnim odzivanjem na zaznane spremembe stanja. Posledično zato poznavanje vseh agentnih akcij v nekem trenutku ne omogoča točnega poznavanja sledečega stanja sistema. Tak sistem ne moremo modelirati z enoagentnim, saj zaradi časovne prepletenosti višjenivojskih akcij njihovo združevanje v "super akcije" ni mogoče. Na preskok iz osnovnih na višjenivojske agentne akcije lahko gledamo kot na abstrakcijo agentnih akcij. Taka abstrakcija je zaželena, saj razstavi "enoagentno" domeno na več samostojnih delov – agentov.

Razlike med osnovnim in višjenivojskim agentnim akcijam lahko ponazorimo s primerom iz človeškega sveta. Osnovne akcije lahko primerjamo z osnovnimi človeškimi gibi, kot so premik nog, rok in glave, za katere privzamemo, da trajajo natanko eno sekundo. Primer akcije je tako na primer hoja do soseda, parameter akcije pa vložena energija, kar se kaže v hitrost hoje in ustrezno vpliva na trajanje izvajanja akcije. V praksi vemo, da rezultat hoje ni točno napovedljiv, saj lahko našo hojo zavleče nenadejani telefonski pogovor ali pa jo prekine ugotovitev, da se je sosed ravnokar odpeljal.

Tu velja omeniti dve pomembni razliki, ki ju v primerjavi z običajnimi enoagentnimi sistemi prinaša definicija večagentnih sistemov:

- Agentna akcija je sama po sebi samo opis neke agentne dejavnosti, zato moramo za poznavanje sprememb stanja poznati tudi agenta, ki jo izvaja. Odigrano instanco agentne akcije zato vedno opišemo s parom *agent:akcija* oz. *vloga:akcija*, kjer ni pomembna točna instanca agenta.
- Za ugotavljanje sprememb stanja sistema je potrebno upoštevati vse trenutno izvajane agentne akcije. Delovanje ostalih agentov pomembno vpliva na agentovo izvajanje akcij. Ker agent ne more natančno predvideti delovanja ostalih agentov v sistemu, rezultat akcije za agenta zato ni točno predvidljiv.

Definicija 5. ***Sled delovanja MAS*** v časovnem intervalu *T* je $trace=\{s_0, s_1, ..., s_T\}$, $s_i \in S_{env}$, ki opisuje zaporedje stanj okolja v danem časovnem intervalu T. Sled delovanja *trace* tabelarično predstavimo kot seznam atributov, ki predstavljajo merljive lastnosti okolja, in njihove vrednosti za vse časovne točke iz intervala T. Podatkovni tipi atributov ustrezajo tipom lastnosti okolja.

Primer sledi delovanja *trace* je predstavljen v preglednici 4.

čas	$atribut_1$	$atribut_2$	...	$atribut_n$
t_1	$v_{1,1}$	$v_{2,1}$	...	$v_{n,1}$
t_2	$v_{1,2}$	$v_{2,2}$		$v_{n,2}$
...	...	...		...
t_{max}	$v_{1,max}$	$v_{2,max}$		$v_{n,max}$

Preglednica 4. Tabelarična predstavitev sledi izvajanja MAS. Vsebuje vrednosti atributov, ki opisujejo stanje sistema za vse časovne točke iz intervala T.

V sledi izvajanja MAS ni opisov notranjih stanj agentov ampak zgolj njihove zunanje merljive lastnosti (npr.: pozicija in smer gibanja), ki so predstavljene v stanju okolja. Sled izvajanja MAS tako opisuje le del pripadajočega MAS-basic.

Definicija 6. ***Modeliranje strategije večagentnega sistema*** MAS<AG, A^{basic}, S^{basic}, F^{basic}, *h*, A, R, S, F, G, G'^{basic}> je iskanje približka strategije G na osnovi sledi delovanja MAS *trace*. Pri tem je podano:

- sled delovanja MAS,
- množica agentov AG,
- množica agentnih vlog R,
- množica agentnih akcij A in
- množica osnovnih agentnih akcij A^{basic}.

Naloga je poiskati množice zaporedij pogosto odigranih akcij $Z=\{z_1, z_2, ..., z_m\}$, kjer je $z_x=a_{x,1}, a_{x,2}, ..., a_{x,c}$, in čim boljši približek višjenivojskih odločitvenih funkcij G v obliki $\Gamma=\{\gamma_1, \gamma_2, ..., \gamma_n\}$, kjer je γ_i približek g_i.

Pri tem je množica Z približek fizične manifestacije delovanja MAS, t.j. interakcije posameznih agentov v večagentnem okolju. Množica Γ pa je približek notranjih odločitvenih funkcij agentov.

5. Sistem za strateško modeliranje večagentnih sistemov - MASDS

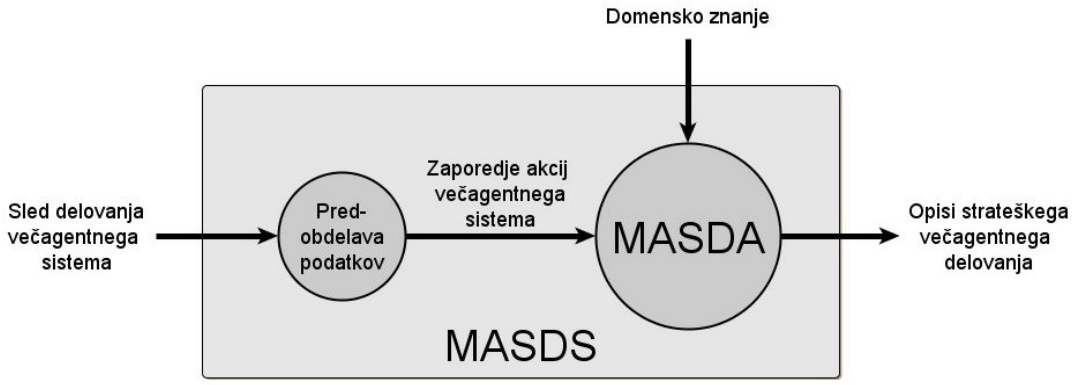
We are ready for any unforeseen event that may not occur.

George W. Bush

5.1 Splošen opis sistema

MASDS (angl. *Multi-Agent Strategy Discovering System*) je sistem, ki omogoča modeliranje večagentnih sistemov. Vhod v MASDS je sled izvajanja večagentnega sistema in osnovno domensko znanje, ki omogoča domensko specifičen opis in primerjavo akcijskih konceptov. MASDS zgradi in opiše model strateškega večagentnega delovanja.

MASDS sestavlja modul za predobdelavo podatkov in modul, ki izvaja modeliranja večagentnih sistemov z MASDA (angl. *Multi-Agent Strategy Discovering Algorithm*). Delovanje celotnega sistema je shematsko predstavljeno na sliki 9. V tem poglavju bomo najprej podali splošen opis sistema, nato bomo opisali korak predobdelave podatkov ter obliko domenskega znanja. Opis MASDA bomo podrobneje predstavili v sledečem poglavju.



Slika 9. Shematska predstavitev delovanja MASDS sistema.

Posamezne naloge, ki jih pri modeliranju večagentnih sistemov izvaja MASDS, so predstavljene na sliki 10, kjer vsak razdelek razen prvega predstavlja eno nalogo v MASDS. Smer poteka nalog je od leve proti desni in od vrha navzdol. Vsak razdelek na vrhu opisuje nalogo, v sredini slikovni in na dnu tekstovni opis izhoda posamezne naloge. Slike predstavljajo primere rezultatov posameznih nalog na domeni RoboCup.

Modul za predobdelavo podatkov pretvori vhodne podatke, zapisane kot sled izvajanja večagentnega sistema, v obliko primerno za vhod v MASDA. V podani sledi delovanja večagentnega zazna osnovne agentne akcije (P.1), iz katerih nato generira ustrezno zaporedje agentnih akcij (P.2), ki predstavlja vhod v MASDA. Vhod v MASDA je tudi domensko znanje, ki predstavlja znanje o agentnih akcijah, vlogah,

opisu prostora ter razdaljah v domenskem prostoru. Znanje je podano v obliki hierarhičnih struktur, kar olajša kasnejši proces abstrakcije.

Postopek MASDA modeliranja lahko v grobem razdelimo na dva koraka. V prvem koraku (I.) zgradi MASDA grafičen model večagentnega delovanja, ki ga v drugem koraku (II.) še dodatno opiše. Vsak korak je sestavljen iz več nalog:

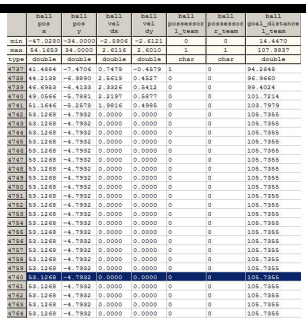
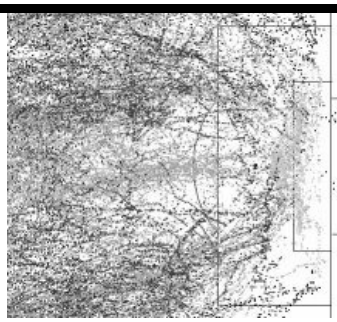
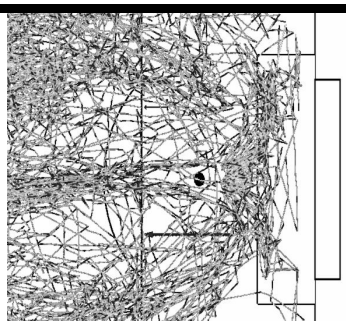
- V I. koraku MASDA iz zaporedja agentih akcij ter podanega domenskega znanja zgradi podatkovno strukturo, imenovano akcijski graf (I.1), ki omogoča hranjenje, predstavitev in poizvedovanje o večagentnem delovanju. Sledi proces abstrakcije (I.2), kjer z uporabo strukture imenovane abstrakten akcijski graf iterativno gradi bolj abstraktne opise agentnega delovanja. Proces abstrakcije uporablja hierarhično združevanje v gruče, kjer združujemo opise akcijskih konceptov, ki so si blizu v prostoru agentih vlog, akcij in stanj. Proces je omejen s podanim parametrom abstrakcije, ki določa abstraktnost opisov v abstraktnem akcijskem grafu. Sledi izbira delov abstraktnega akcijskega grafa (I.3), ki predstavljajo strateške dejavnosti večagentnega sistema.
- V II. koraku MASDA dodatno opiše dobljeni akcijski model. Najprej mu priredi ustrezen opis agentnih akcij in vlog (II.1), nato generira učne primere (II.2), ki jih uporabi kot vhod za algoritem za induciranje odločitvenih pravil (II.3). Dobljena pravila dodatno opisujejo akcijski model. Končni izhod MASDA je model konceptov makroakcij, ki skupaj opisujejo del strategije večagentnega sistema.

Pomembna lastnost MASDS je v stalni abstrakciji podatkov, ki jih vršijo posamezne naloge znotraj MASDS. Preglednica 5 prikazuje okvirno število in tipe objektov, s katerimi operirajo posamezne naloge v MASDS. Število objektov s katerimi operirajo posamezne naloge se iz naloge v nalogo v MASDS manjšajo, kar nakazuje na naraščajočo abstraktnost podatkov, s katerimi operira MASDS,

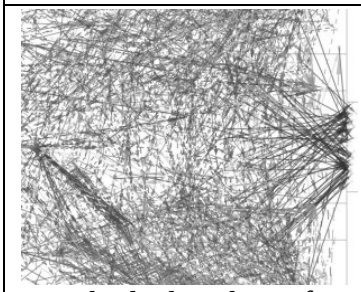
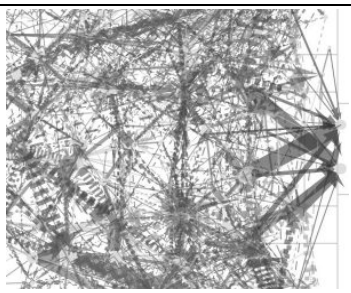
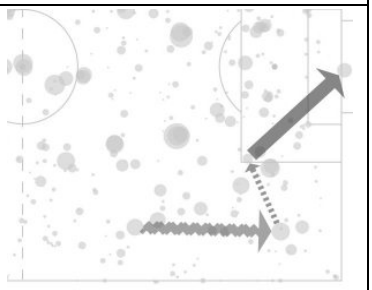
Naloga	Rezultat naloge	Tip objektov	Število objektov	
vhod	sled izvajanja večagentnega sistema	numerične vrednosti	~3.000.000	Naraščajoča abstrakcija
P.1	seznam osnovnih agentnih akcij	osnovne agentne akcije	~300.000	
P.2	zaporedje agentnih akcij	agentne akcije	~1.000	
I.1	akcijski graf	povezave v grafu	~1.000	
I.2	abstrakten akcijski graf	povezave v grafu	~500	
I.3	koncepti makroakcij	poti v grafu	~200	
II.1	karakterni opisi konceptov makroakcij	opisane poti	~200	
II.2	učni primeri	učni primeri - instance akcij	~100	
II.3	opisi konceptov makroakcij	pravila	~10	

Preglednica 5. Prikaz abstrakcije v MASDS. Predstavljeni so koraki, rezultati korakov, tipi objektov, ki jih uporabljajo posamezni koraki, in okvirno število objektov v primeru analize ene igre v domeni RoboCup.

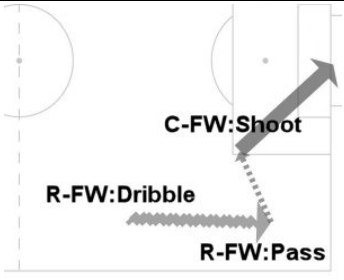
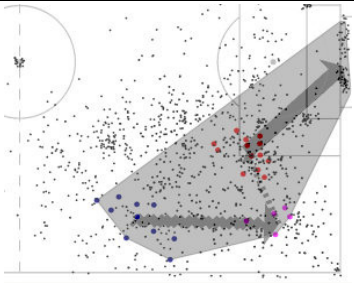
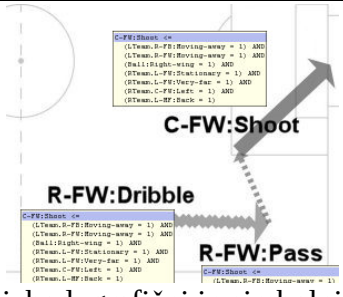
Predobdelava podatkov

Vhod v sistem	**P.1** Zaznava osnovnih agentnih akcij	**P.2** Zaznava agentnih akcij
		
vhod: sled izvajanja večagentnega sistema	izhod: seznam osnovnih agentnih akcij	izhod: zaporedje agentnih akcij

MASDA I. korak: Gradnja grafičnega akcijskega modela

I.1 Gradnja akcijskega modela	**I.2** Gradnja abstraktnega akcijskega modela	**I.3** Izbira strateških konceptov makroakcij
		
izhod: akcijski graf	izhod: abstrakten akcijski graf	izhod: koncepti makroakcij

MASDA II. korak: Simbolni opis akcijskega modela

II.1 Določitev agentnih vlog in akcij	**II.2** Določitev učnega problema	**II.3** Indukcija pravil
		
izhod: karakterni opisi konceptov makroakcij	izhod: učni primeri	izhod: grafični in simbolni opisi konceptov makroakcij

Slika 10. Predstavitev korakov in nalog v MASDS. Vsak razdelek razen prvega predstavlja eno nalogo v MASDS, kjer je na vrhu opisan proces, v sredini grafični in na dnu tekstovni opis izhoda posamezne naloge.

5.2 Predobdelava podatkov

V koraku predobdelave podatkov je potrebno pripraviti podatke, ki jih za vhod uporablja MASDA. Ta korak izvaja dve nalogi. V prvi zazna osnovne agentne akcije in generira seznam osnovnih agentnih akcij, ki jih uporabi v drugi nalogi, kjer zazna agentne akcije in generira ustrezno zaporedje agentnih akcij.

5.2.1 Zaznava osnovnih agentnih akcij (P.1)

Vhod v sistem za strateško modeliranje večagentnih sistemov je sled delovanja večagentnega sistema *trace*, ki jo dobimo z opazovanjem delovanja večagentnega sistema v nekem časovnem intervalu T. Taka sled delovanja večagentnega sistema je predstavljena kot seznam vrednosti atributov, ki opisujejo domensko stanje skozi dani časovni interval. Vrednosti atributov v sledi delovanja večagentnega sistema za dano časovno točko bolj ali manj podrobno opisuje stanje sistema, kot je to prikazano na preglednici 6. Ker vsaka osnovna agentna akcija spremeni domensko stanje na točno določen način, je možno ob poznavanju učinka osnovnih agentnih akcij in sprememb stanja povratno sklepati na opravljene osnovne agentne akcije.

čas	$atribut_1$	$atribut_2$	...	$atribut_m$	
...	...	...		...	
t_i	$v_{1,i}$	$v_{2,i}$	...	$v_{m,i}$	$\approx S(t_i)$
t_{i+1}	$v_{1,i+1}$	$v_{2,i+1}$		$v_{m,i+1}$	$\approx S(t_{i+1})$
...	...	...		...	

Preglednica 6. Vrednosti atributov v sledi delovanja opisujejo stanje sistema.

Če poznamo stanje v časovni točki t_i in če poznamo stanje v naslednji časovni točki t_{i+1}, potem poznamo točno spremembo stanja, ki so ga skupaj povzročile osnovne agentne akcije vseh agentov v sistemu:

$$S(t_i) \xrightarrow{(ag_1,\ a_{i,1}^{basic}) \times (ag_2,\ a_{i,2}^{basic}) \times ... \times (ag_n,\ a_{i,n}^{basic})} S(t_{i+1})$$

Predpostavimo, da sprememba stanja enolično določa nabor instanc osnovnih agentnih akcij:

$$(S(t_i), S(t_{i+1})) \Rightarrow (a_{i,1}^{basic},\ a_{i,2}^{basic},\ ...,\ a_{i,n}^{basic})$$

Učinkoviti postopki določanja nabora primernih osnovnih agentnih akcij so raznoliki in domensko odvisni, najpreprostejši domensko neodvisni postopek pa simulira izvajanje vseh možnih kombinacij osnovnih akcij in preveri, če iz začetnega stanja $S(t_i)$ povzročijo novo stanje, ki je enako $S(t_{i+1})$. Če postopek identifikacije osnovnih agentnih akcij ponovimo za vse pare časovnih točk (t_i, t_{i+1}) dobimo seznam vseh opravljenih osnovnih akcij v časovnem intervalu T, kot to prikazuje preglednica 7. Postopek pretvorbe je kot algoritem predstavljen na sliki 11.

čas	ag_1	ag_2	...	ag_n
t_0	$a_{0,1}^{basic}$	$a_{0,2}^{basic}$		$a_{0,n}^{basic}$
...	...	...		...
t_i	$a_{i,1}^{basic}$	$a_{i,2}^{basic}$	...	$a_{i,n}^{basic}$
...	...	...		...
t_{max}	$a_{max,1}^{basic}$	$a_{max,2}^{basic}$		$a_{max,n}^{basic}$

Preglednica 7. Seznam osnovnih agentnih akcij v nekem časovnem intervalu.

```
Vhod:
- število atributov: NumAttr
- časovni interval: 0...MaxT-1
- število agentov: NumAgents
- število osnovnih akcij: NumBasicActions
- množica osnovnih akcij: basicActions[NumBasicActions]
- sled delovanja: trace[MaxT][NumAttr]
- funkcija FindBasicActions(...), ki iz spremembe vrednosti atributov
v sledi delovanja zazna vse opravljene osnovne agentne akcije

Algoritem:
// obdelamo celoten časovni interval
for (t = 0; t < MaxT-1; t++) {
  // poišči primeren nabor osnovnih akcij theActions[NumAgents],
  // ki iz trace[t][] privedejo v stanje trace[t+1][]
  FindBasicActions(trace[t], trace[t+1], &theActions[]);
  // shrani primerne osnovne akcije v seznam
  for (i = 0; i < NumAgents; i++) {
    basicActionList[t][i] = theActions[i];
  }
}

Izhod:
- seznam osnovnih agentnih akcij: basicActionList[MaxT-1][NumAgents]
```

Slika 11. Algoritem, ki pretvori sled delovanja večagentnega sistema v seznam osnovnih agentnih akcij.

5.2.2 Zaznava agentnih akcij (P.2)

Naslednja stopnja predobdelave podatkov je zaznava in generiranja instanc akcij, kar je v osnovi podobno nalogi zaznave osnovnih agentnih akcij. Tu je potrebno iz sledi izvajanja večagentnega sistema in z uporabo seznama osnovnih agentnih akcij v nekem časovnem intervalu določiti zaporedje izvajanih agentnih akcij. Ker na izvajanje neke agentne akcije vplivajo parametri akcije in delovanje ostalih agentov, je njen vpliv na spremembe okolja in na samo zaporedje osnovnih agentnih akcij spremenljiv, kar otežuje njeno eksaktno zaznavanje. V splošnem pri zaznavi instanc akcij ugotavljamo značilne spremembe stanja, ki so karakteristične za posamezno akcijo, zato je zaznava instanc agentnih akcij izrazito domensko odvisen problem. Ker je točno zaznavanje instanc akcij ter posledično določanje akcij in njihovih parametrov pogosto nemogoče, se pri zaznavi zadovoljimo z določitvijo akcije, začetnega čas, trajanja akcije, agenta, ki jo je izvajal in njegove vloge, točne vrednosti parametrov akcije pa ne ugotavljamo.

Rezultat zaznave so zaporedja instanc akcij, ki za vsakega agenta opisujejo akcije, ki jih je izvajal v danem časovnem intervalu. Vsaka instanca akcije je opisana s peterko <*action, start, duration, agent, role*>, kjer je *action* ime akcije, *start* čas začetka akcije, *duration* trajanje akcije, *agent* ime agenta in *role* vloga agenta, ki je opravil akcijo. Zaporedja so urejena po naraščajoči vrednosti časa začetka akcije *start*. Ker v splošnem velja, da se akcije posameznega agenta lahko izvajajo tudi vzporedno, za akciji <$action_i$, $start_i$, $duration_i$, *agent*, $role_i$> in <$action_{i+1}$, $start_{i+1}$, $duration_{i+1}$, *agent*, $role_{i+1}$> ne velja lastnost: $start_i + duration_i <= start_{i+1}$. Zaporedje instanc akcij za n agentov je predstavljeno v preglednici 8. Opisani postopek je kot algoritem predstavljen na sliki 12

```
Vhod:
- število atributov: NumAttr
- časovni interval: 0...MaxT-1
- število agentov: NumAgents
- število osnovnih akcij: NumBasicActions
- število akcij: NumActions
- množica osnovnih akcij: basicActions[NumBasicActions]
- množica akcij: actions[NumActions]
- sled delovanja: trace[MaxT][NumAttr]
- seznam osnovnih agentnih akcij: basicActionList[MaxT-1][NumAgents]
- funkcija DetectAction(...), ki iz sledi izvajanja, začetnega časa
in seznama osnovnih akcij agenta ugotovi agentno akcijo, njeno
trajanje in trenutno vlogo agenta

Algoritem:
// nastavi število zaznanih akcij za vsakega agenta na 0
for (agent = 0; agent < NumAgents; agent++) {
  NumDetectedActions[agent] = 0;
}
// preverimo celotni časovni interval
for (startT = 0; startT < MaxT-1; startT++) {
  // za vsakega agenta posebej
  for (agent = 0; agent < NumAgents; agent++) {
    // ugotovimo ali je v času startT začel izvajati kakšno akcijo
    if (DetectAction(agent, startT, trace[], basicActionsList[],
         &action, &duration, &role)) {
     // v času startT je agent izvedel akcijo 'action'
     // povečajmo število zaznanih akcij tega agenta
     pos = NumDetectedActions[agent]++;
     // shranimo podatke o akciji
     actionList[pos][agent].action = action;
     actionList[pos][agent].start = startT;
     actionList[pos][agent].duration = duration;
     actionList[pos][agent].agent = agent;
     actionList[pos][agent].role = role;
    }
  }
}

Izhod:
- zaporedje agentnih akcij: actionList[][NumAgents]
```

Slika 12. Algoritem, ki pretvori seznam osnovnih agentnih akcij v zaporedje agentnih akcij.

ag_1	ag_2		ag_n
$<a_{1,1}, s_{1,1}, d_{1,1}, ag_1, r_{1,1}>$	$<a_{2,1}, s_{2,1}, d_{2,1}, ag_2, r_{2,1}>$		$<a_{n,1}, s_{n,1}, d_{n,1}, ag_n, r_{n,1}>$
$<a_{1,2}, s_{1,2}, d_{1,2}, ag_1, r_{1,2}>$	$<a_{2,2}, s_{2,2}, d_{2,2}, ag_2, r_{2,2}>$		$<a_{n,2}, s_{n,2}, d_{n,2}, ag_n, r_{n,2}>$
...	...	...	...
	$<a_{2,v}, s_{2,v}, d_{2,v}, ag_2, r_{2,v}>$		
$<a_{1,u}, s_{1,u}, d_{1,u}, ag_1, r_{1,u}>$			$<a_{n,w}, s_{n,w}, d_{n,w}, ag_n, r_{n,w}>$

Preglednica 8. Zaporedje instanc akcij za n agentov.

5.3 Vključitev domenskega znanja

Oblika domenskega znanja, ki je vključeno v proces modeliranja, pomembno vpliva na proces modeliranja in na izraznost dobljenega modela. Hierarhični pristopi se pogosto uporabljajo [6, 33, 54], kadar je potrebno modelirati domene, ki vključujejo predstavitev podatkov na različnih nivojih abstrakcije. Ker je bil eden izmed osnovnih ciljev pri razvoju MASDA, da v postopek modeliranja vključimo tudi proces abstrakcije, uporablja MASDA domensko znanje v obliki hierarhičnih struktur, ki omogočajo opis modela na različnih nivojih abstrakcije.

Definicija 7. ***Hierarhija H*** je hierarhično urejena struktura elementov. Neposredni predniki elementa iz hierarhije $a \in H$ so množica $parent_H(a)$. Vsi predniki elementa iz hierarhije $a \in H$ so elementi $ancestors_H(a)=\{b \in H; \exists (e_1, e_2, ..., e_n) \in H^n: e_1 \in parent_H(a) \wedge e_2 \in parent_H(e_1) \wedge ... \wedge b \in parent_H(e_n)\}$. Neposredni potomci elementa iz hierarhije $a \in H$ so množica $children(a)$. Vsi potomci elementa iz hierarhije $a \in H$ so elementi $descendents_H(a)=\{b \in H; \exists(e_1, e_2, ..., e_n) \in H^n: e_1 \in parent_H(b) \wedge e_2 \in parent_H(e_1) \wedge ... \wedge a \in parent_H(e_n)\}$. Element hierarhije $a \in H$, ki nima prednikov, torej je $parent_H(a) = \{\}$, imenujemo koren hierarhije H in ga označimo z a_{rootH}. Hierarhija ima natanko en koren hierarhije. Elemente, ki nimajo potomcev, imenujemo listi hierarhije. Hierarhija nima ciklov, kar pomeni da velja:

$$\forall a \in H: ancestors_H(a) \cap descendents_H(a) = \{\}.$$

Definicija 8. Globina elementa v hierarhiji $a \in H$, $depth_H(a)$, je najkrajša razdalja od korena hierarhije do elementa a:

$$depth_H(a) = \begin{cases} 0; a=a_{rootH} \\ n+1; n=\min\limits_{e \in parents(a)}(depth_H(e)) \end{cases}$$

Definicija 9. Globina hierarhije H je $depth_{Hmax} = \max\limits_{a \in H}(depth_H(a))$.

Definicija 10. ***Najbližji skupni prednik*** elementov v hierarhiji $a, b \in H$ je rezultat funkcije $common\_ancestor_H(a, b)$, ki je definirana sledeče:

$$common\_ancestor_H(a, b) = \max\arg\limits_{c \in (ancestors_H(a) \cap ancestors_H(b))}(depth_H(c))$$

Definicija 11. Razdalja med elementoma v hierarhiji $a, b \in H$ je $distance_H(a, b)$, ki je definirana kot vsota dolžin poti od elementov a in b do najbližjega skupnega prednika:

$$distance_H(a, b) = depth_H(a) + depth_H(b) - 2*depth_H(common\_ancestor_H(a, b))$$

5.3.1 Opis akcij in agentov

MASDA uporablja opis agentnih vlog in akcij v obliki hierarhije, kar omogoča učinkovit popis vseh akcij in vlog. Poleg tega taka predstavitev omogoča iskanje bolj splošnih opisov agentnih akcij in vlog, kar je uporabno v kasnejšem procesu abstrakcije.

Definicija 12. ***Hierarhija agentnih akcij*** je hierarhija $\boldsymbol{H_{action}}$, katere elementi so opisi agentnih akcij. Če za elementa hierarhije agentnih akcij $a, b \in H_{action}$ velja $a \in parent_{Haction}(b)$, potem velja, da je opis agentne akcije a bolj splošen od opisa agentne akcije b. Koren hierarhije agentnih akcij je element, ki predstavlja najbolj splošen opis agentne akcije.

Definicija 13. ***Hierarhija agentnih vlog*** je hierarhija $\boldsymbol{H_{role}}$, katere elementi so opisi agentnih vlog. Če za elementa hierarhije agentnih vlog $a, b \in H_{role}$ velja $a \in parent_{Hrole}(b)$, potem velja, da je opis agentne vloge a bolj splošen od opisa agentne vloge b. Koren hierarhije agentnih vlog je element, ki predstavlja najbolj splošen opis agentne vloge.

Hierarhične opise agentnih akcij in vlog bomo uporabljali v kasnejšem procesu abstrakcije, kjer bomo združevali konceptualno podobne opise. Zato moramo določiti ustrezno konceptualno razdaljo med elementi v hierarhiji H, ki jo imenujemo ***hierarhična razdalja*** in jo označimo s $dist_H$. Določitev hierarhične razdalje pa še zdaleč ni enoličen proces [34]. V velikih hierarhičnih strukturah (npr. WordNet [70]) je možno definirati statistično utemeljene metrike, ki z uporabo različnih verjetnostnih porazdelitev [86, 93] ali entropije [22, 85] računajo hierarhično razdaljo. Le-te pa niso primerne za manjše, ročno grajene hierarhije, kjer majhno število elementov preprečuje učinkovito uporabo statističnih metod. V splošnem velja [2], da mora biti hierarhična razdalja občutljiva na:

- dolžino najkrajše poti med elementoma hierarhije,
- globino v hierarhiji; bližnji elementi globlje v hierarhiji so bolj podobni kot enako oddaljeni elementi v vrhu hierarhije,
- gostoto elementov v hierarhiji; elementi v gostejšem delu so relativno bližje, kot tisti v redkejšem delu hierarhije,
- metrika mora biti neodvisna od velikosti hierarhije.

Ustrezni hierarhični razdalji med elementi hierarhij akcij $dist_{Haction}$ in $dist_{Hrole}$, ki merita razdaljo med elementi hierarhije akcij H_{action} in hierarhije vlog H_{role}, lahko določimo le na podlagi vsebinskega razumevanja posamezne domene, zato ju v praksi določi domenski ekspert.

5.3.2 Opis stanja

V strojnem učenju (angl. *machine learning*) je običajen opis prostora stanj z uporabo domenskih atributov. Atributi so spremenljivke, ki opisujejo neko domensko

značilnost in so lahko različnih tipov. V splošnem poznamo zvezne (običajno podmnožica realnih števil) in diskretne tipe, slednji se delijo na nominalne, kjer ne obstaja relacija urejenosti (npr.: barve), in ordinalne, kjer je določena relacija urejenosti (npr.: nizek, srednji, visok).

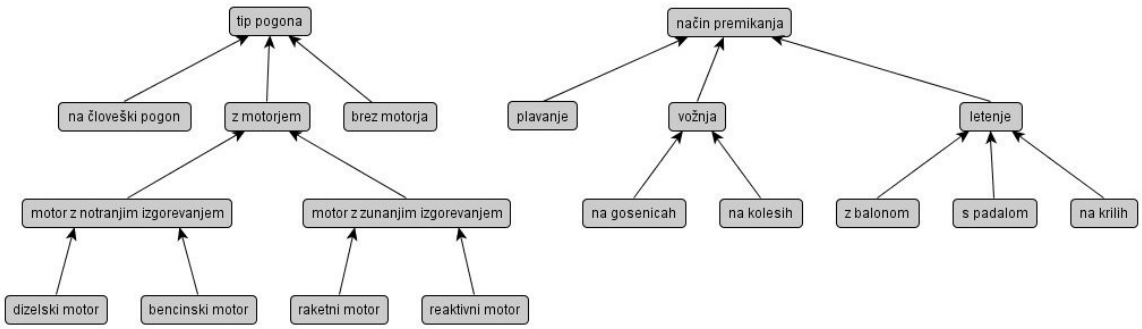
Pri uporabi atributov za predstavitev domenskega stanja potrebujemo za predstavitev nekega podprostora množico atributov in njihovih vrednostnih omejitev. Ker je množica vrednostnih omejitev v kompleksni domeni za človeka običajno težko predstavljiva, bomo za predstavitev domenskega podprostora uporabljali predstavitev z uporabo binarna značilk. Značilka (angl. *feature*) [60] je definirana z atributom oz. atributi in njihovimi vrednostnimi omejitvami. Binarna značilka je resnična, če je resničen tudi njen vrednosti pogoj, sicer je neresnična. Ker v disertaciji uporabljamo zgolj binarne značilke, bomo s terminom značilka vedno označevali binarno značilko. Pretvorba atributov v značilke običajno ni enostavna, saj je zelo odvisna od tipa atributa, porazdelitve vrednosti in domenskih značilnosti. Na primer, če bi v domeni prevoznih sredstev imeli na voljo spremenljivke *moč_motorja*, *teža_vozila* in *barva_vozila*, bi lahko določili naslednje značilke: *športni_avto* (*moč_motorja* ≥ 100 KW in *teža_vozila* < 2000 kg), *osebni_avto* (*moč_motorja* < 100 KW in *teža_vozila* < 2000 kg), *gospodarsko_vozilo* (*teža_vozila* ≥ 2000 kg), *moder* (*barva_vozila* = modra), *rdeč* (*barva_vozila* = rdeča) in *črn* (*barva_vozila* = črna). Opis *črn* in *osebni_avto* je tako človeku bolj razumljiv, kot opis z uporabo atributov: *moč_motorja* = 75 KW in *teža_vozila* = 1300 kg in *barva_vozila* = črna.

Čeprav se z uporabo značilk pri opisu prostora stanja poveča berljivost opisov, pa ti opisi ne podajajo relacij, ki bi določale, ali je neka značilka bolj abstraktna od druge. V ta namen vpeljemo združevanje značilk v hierarhijo, ki določa relacijo abstraktnosti med značilkami.

Definicija 14. ***Hierarhija značilk za domensko lastnost*** je hierarhija $\boldsymbol{H_{lastnost}}$ v obliki drevesa značilk, kjer predstavlja vsaka značilka v drevesu bolj abstrakten opis domenske lastnosti od njej podrejenih značilk. Koren hierarhije določa lastnost domene, ki jo opisuje hierarhija. Vsak element iz hierarhije opišemo s parom ***lastnost:ime značilke***.

Hierarhijo značilk za domensko lastnost si lahko predstavljamo kot popis neke lastnosti domene, ki vključuje relacijo abstraktnosti. Točna definicija lastnosti domene je odvisna od podanega načina popisa domene in je lahko predstavljena kot: opis v naravnem jeziku, (sestavljen) atribut, del ontologije ali kako drugače predstavljeno znanje o domeni.

V splošnem je neka domena predstavljena z več domenskimi lastnostmi, torej z več hierarhijami značilk. Primer opisa domene prevoznih sredstev z uporabo dveh domenskih lastnosti: tip pogona in način premikanja je predstavljen s hierarhijama značilk $H_{\text{tip pogona}}$ in $H_{\text{način premikanja}}$, ki sta predstavljeni na sliki 13. V tem primeru opišemo gospodarsko vozilo z naslednjimi značilkami: "tip pogona:dizelski motor", "tip pogona:motor z notranjim izgorevanjem" in "tip pogona:z motorjem", "način premikanja:na kolesih" in "način premikanja:vožnja". Prednosti hierarhije značilk se pokažejo pri skupnem opisu več različnih primerov. Na primer, ptice in jadralni avion opišemo z značilkami: "tip pogona:brez motorja" in "način premikanja:letenje", medtem ko avto, bager in tovornjak opišemo z: "tip pogona:motor z notranjim izgorevanjem", "tip pogona:z motorjem" in "način premikanja:vožnja".



Slika 13. Primera hierarhij domenskih značilk na domeni prevoznih sredstev. Leva hierarhija $H_{\text{tip pogona}}$ prikazuje značilke, ki opisujejo tip pogona, desna hierarhija $H_{\text{način premikanja}}$ opisuje način premikanja prevoznih sredstev.

Hierarhije značilk sicer omogočajo določitev relacije abstraktnosti med značilkami ter učinkovito popišejo domenske lastnosti, vendar običajno primerne hierarhije značilk določi domenski ekspert, kar omejuje domensko neodvisnost in splošnost pristopa. Poleg tega imamo običajno na voljo le atributni opis prostora, ki onemogoča določanja relacije abstraktnosti. Če nimamo na voljo domenskega eksperta, ki bi iz atributov določil hierarhije domenskih značilk, lahko atribute pretvorimo v značilke z diskretizacijo atributov na vrednostne intervale. Ker v splošnem ne poznamo primernih vrednostnih intervalov, je rešitev v uporabi hierarhije domenskih značilk za atribut *a*, ki opisuje značilke, ki omejujejo domenski prostor atributa *a* na poljubnem intervalu atributnih vrednosti.

Definicija 15. ***Hierarhija značilk za atribut a*** s podano množico mej intervalov $\{v_0, v_1, ..., v_{a\text{-}max}\}$ je hierarhija v obliki dvojiškega drevesa značilk $\boldsymbol{H_a(a, \{v_0, v_1, ..., v_{a\text{-}max}\})}$ z lastnostjo, da predstavlja vsaka značilka v hierarhiji unijo njej podrejenih značilk. Listi hierarhije značilk H_a so značilke, ki opisujejo spremenljivko *a* na osnovnih intervalih: $a \in [v_i, v_{i+1})$; $0 \leq i <$ a-max. Koren hierarhije značilk je značilka, ki pokriva celotni interval spremenljivke $a \in [v_0 , v_{a\text{-}max})$. Oznaka za značilko iz H_a, ki pokriva interval $a \in [min, max)$ je *a_min_max*. Značilka *a_min_max* je resnična, če je vrednost atributa $a \in [min, max)$, sicer je neresnična. Resničnost značilke *a_min_max* za dano vrednost *b* določa funkcija *true(a_min_max, b)*, ki je resnična, če je $min \leq b < max$. Na hierarhijo značilk za atribut *a* lahko gledamo kot na hierarhijo značilk za naslednjo domensko lastnost: vrednost atributa *a* je v nekem vrednostnem intervalu. Atributu *a* pravimo tudi izhodiščni atribut hierarhije značilk za atribut *a*.

```
CreateHierarchy(Node *parent, bool isLeft, void * attr,
                double range[], int left, int right) {
  Node *leftNode, *rightNode;
  int newLeft, newRight;
  double min, newMin, max, newMax;

  if (left+1 >= right) {
    return;
  }
  // izračunaj meje intervalov in vrednosti mej
  newLeft = left+1;
  newRight = right-1;
  min = range[left];
  newMin = range[newLeft];
  newMax = range[newRight];
  max = range[right];
  if (isLeft) { // zgradi podhierarhijo levega otroka
    leftNode = parent->AddChild(attr, min, newMax);
    CreateHierarchy(leftNode, true, attr, range, left, newRight);
    rightNode = parent->AddChild(attr, newMin, max);
    CreateHierarchy(rightNode, false, attr, range, newLeft, right);
  }
  else { // zgradi podhierarhijo desnega otroka
    leftNode = (parent->GetParent()->LeftChild()->RightChild();
    parent->AddChild(leftNode);
    rightNode = parent->AddChild(attr, newMin, max);
    CreateHierarchy(rightNode, false, attr, range, newLeft, right);
  }
}
```

Slika 14. Procedura za gradnjo hierarhije domenskih značilk za podani atribut.

Procedura, ki rekurzivno gradi hierarhijo domenskih značilk za dani atribut *attr* in njegove vrednostne intervale podane v seznamu *range[]*, je predstavljena na sliki 14. Postopek sproži klic procedure z naslednjimi parametri, kjer je *root* korensko vozliče in *range[]* seznam mej intervalov: *CreateHierarchy(root, true, attr, range[], 0, sizeof(range[])-1).*

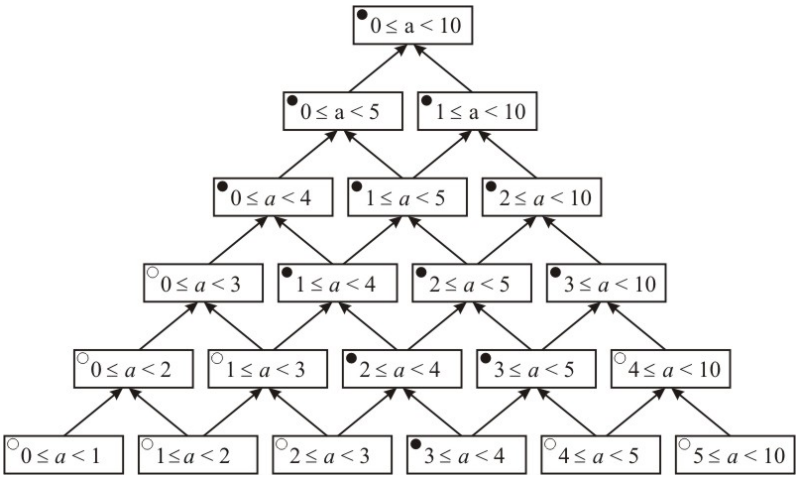
Definicija 16. ***Najmanj splošna značilka*** atributa *a* za vrednost *b* v hierarhiji značilk H_a, označeno z ***least_general*($H_a(a, \{v_0, v_1, ..., v_{a\text{-}max}\}), v)$**, je:

$$\text{least\_general}(H_a(a, \{v_0, v_1, ..., v_{a\text{-}max}\}), b) = x;$$
$$x \in H_{feature} \wedge true(x, b) \wedge (\forall y \in H_a : true(y, b) \Rightarrow y = common\_ancestor_{Ha}(x, y))$$

Definicija 17. ***Najmanj splošna skupna značilka*** značilk $b_1, b_2, ..., b_m$ v hierarhiji značilk $H_{feature}$, označeno z ***least_general*(H_a, $\{b_1, b_2, ..., b_m\}$)**, je:

$$least\_general(H_a, \{b_1, b_2, ..., b_m\}) =$$
$$common\_ancestor_{Ha}(b_1, common\_ancestor_{Ha}(b_2, ...common\_ancestor_{Ha}(b_{m-1}, b_m)...))$$

Primer hierarhije značilk za atribut *a* je predstavljen na sliki 15. Na primeru je predstavljena tudi resničnost značilk, če ima atribut *a* vrednost 3, kjer črne pika označujejo resnične, bele pike pa neresnične značilke.



Slika 15. Primer hierarhije značilk $H_a(a, \{0, 1, 2, 3, 4, 5, 10\})$, ki opisujejo atribut *a* na vseh možnih intervalih, ki so določeni s podanimi mejami intervalov. Črne pike označujejo resnične značilke in bele pike neresnične, če ima atribut *a* vrednost 3.

V preglednici 9 je predstavljen primer, ki prikazuje resnične značilke in odgovarjajoče najmanj splošne značilke za naslednje vrednosti atributa a=1, a=2, a=3 in $a \in \{1, 2, 3\}$. Na primeru vidimo, da je za opis najbolj primerna uporaba najmanj splošne značilke. To je še posebej očitno pri treh vrednostih atributa $a \in \{1, 2, 3\}$, kjer značilka a_1_4 nadomešča opis z atributi: $(a \geq 1) \wedge (a < 4)$.

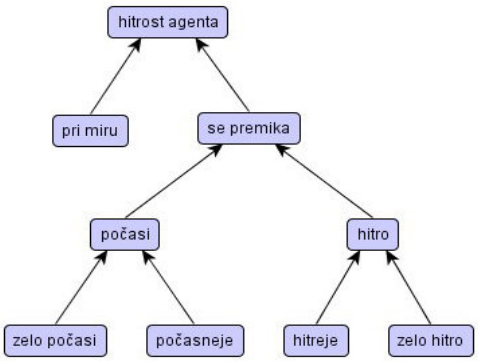
Vrednost atributa *a*	Resnične značilke	Najmanj splošna skupna značilka
1	a_1_2, a_0_2, a_1_3, a_0_3, a_1_4,a_0_4, a_1_5, a_0_5, a_1_10, a_0_10	a_1_2
2	a_2_3, a_1_3, a_2_4, a_0_3, a_1_4, a_2_5, a_0_4, a_1_5, a_2_10, a_0_5, a_1_10, a_0_10	a_2_3
3	a_3_4, a_2_4, a_3_5, a_1_4, a_2_5, a_3_10, a_0_4, a_1_5, a_2_10, a_0_5, a_1_10, a_0_10	a_3_4
{1, 2, 3}	a_1_2, a_2_3, a_3_4, a_0_2, a_1_3, a_2_4, a_3_5, a_0_3, a_1_4, a_2_5, a_3_10, a_0_4, a_1_5, a_2_10, a_0_5, a_1_10, a_0_10	a_1_4

Preglednica 9. Primer opisa z značilkami in najmanj splošnimi skupnimi značilkami iz hierarhije značilk za atribut *a*.

Definicija 18. ***Hierarhija značilk za agentno lastnost*** je hierarhija $\boldsymbol{H_{agentna\text{-}lastnost}(agent)}$ v obliki drevesa značilk, kjer predstavlja vsaka značilka v drevesu bolj abstraktno agentno lastnost od njej podrejenih značilk. Koren hierarhije določa lastnost agenta, ki jo opisuje hierarhija. Vsako element iz hierarhije značilk za agentno lastnost opišemo s trojko ***agent:agentna-lastnost:značilka***.

Hierarhija značilk za agentno lastno je nabor značilk, ki opisujejo neko agentno lastnost, ki je skupna vsem agentom v sistemu. Primer za tako hierarhijo bi bil primer na sliki 16, kjer je prikazana hierarhija značilk za agentno lastnost: *hitrost agenta*. V primeru, da imamo agenta A, ki se premika zelo počasi, in agenta B, ki se premika

zelo hitro, bi stanje opisali z naslednjimi značilkami: {"A:hitrost agenta:zelo počasi", "A:hitrost agenta:počasi", "A:hitrost agenta:se premika", "B:hitrost agenta:zelo hitro", "B:hitrost agenta:hitro", "B:hitrost agenta:se premika"}. Prednost takega opisa je v možnosti iskanja najmanj splošne skupne značilke, saj lahko na podlagi prejšnjega opisa stanj ugotovimo, da je skupna lastnost agentov A in B, da se premikata {"A in B:hitrost agenta:se premika"}.



Slika 16. Primer hierarhije značilk za hitrost agenta, $H_{hitrost}(agent)$.

Ker je večagentni sistem sestavljen iz okolja in agentov, je opis celotnega stanja sistema podan s hierarhijami značilk, ki opisujejo okolje, in s hierarhijami značilk, ki opisujejo agentne lastnosti. Večagentni sistem z n agenti, N agentnimi lastnostni in M lastnostmi okolja je opisan s hierarhijami značilk, ki so predstavljene v preglednici 10. Skupno število hierarhij značilk za opis večagentnega sistema je torej M+n*N.

	Hierarhije značilk
okolje	$H_{lastnost1}$, $H_{lastnost2}$, ..., $H_{lastnostM}$
agent a_1	$H_{agentna\text{-}lastnost1}(a_1)$, $H_{agentna\text{-}lastnost2}(a_1)$, ..., $H_{agentna\text{-}lastnostN}(a_1)$
agent a_2	$H_{agentna\text{-}lastnost1}(a_2)$, $H_{agentna\text{-}lastnost2}(a_2)$, ..., $H_{agentna\text{-}lastnostN}(a_2)$
...	...
agent a_n	$H_{agentna\text{-}lastnost1}(a_n)$, $H_{agentna\text{-}lastnost2}(a_n)$, ..., $H_{agentna\text{-}lastnostN}(a_n)$

Preglednica 10. Opis večagentnega sistema s hierarhijami značilk za večagentni sistem z n agenti, N agentnimi lastnostmi in M lastnostmi okolja.

Uporaba hierarhij značilk ima več prednosti. Ker predstavlja hierarhija značilk množico značilk, ki opisuje le eno domensko lastnost oz. atribut, s tem omogoča možnost izbire najbolj primernih značilk za opis danega stanja MAS. Poleg tega so značilke v hierarhiji urejene po abstraktnosti, kar omogoča predstavitev stanja z bolj ali manj splošnim opisom. Čeprav lahko z večjim številom značilk bolj podrobno opišemo neko stanje, pa je to obenem tudi slabost, saj vsebuje množica značilk tudi nepotrebne značilke, ki slabo vplivajo na uspešnost algoritmov za strojno učenje [17]. Prednost hierarhije značilk je tudi v splošnosti pristopa, saj omogoča popis različno kompleksnih domenskih pojmov. V najpreprostejšem primeru predstavljajo značilke le neposredno preslikavo iz atributov (na primer: hierarhije značilk, ki opisujejo domeno 3vs2 Keepaway v poglavju 8.1), v bolj kompleksnih primerih pa lahko domenski ekspert z značilkami opiše strateške lastnosti večagentnega obnašanja (na primer: hierarhija značilk $H_{Teamplay}$ na sliki 80, ki opisuje tip igre v domeni RoboCup).

6. Algoritem za strateško modeliranje večagentnih sistemov - MASDA

Kar je dobro znati, se je težko naučiti.

Grški pregovor

MASDA (angl. *Multi-Agent Strategy Discovering Algorithm*) [13-15] je domensko neodvisen algoritem, ki omogoča modeliranje večagentnih sistemov. Vhod v MASDA je zaporedje akcij večagentnega sistema in osnovno domensko znanje, ki omogoča domensko specifičen opis in primerjavo agentnih vlog in akcij. MASDA zgradi in opiše model strateškega večagentnega delovanja.

Za opis večagentnega obnašanja uporablja MASDA skupno predstavitev, kar temelji na ugotovitvah, ki sta jih predstavila Tambe in Rosenbloom [113]. Ugotovila sta namreč, da za modeliranje agentne interakcije potrebujemo skupno in celovito predstavitev. Ker se interakcija med agenti izraža z usklajenim izvajanjem agentnih akcij, je celovita predstavitev primerna tudi za opis delovanja večagentnega sistema.

MASDA opisuje delovanje večagentnega sistema na podoben način, kot ga je predstavil Bowling [19], ki je definiral večagentno igro kot sekvenco akcij in odgovarjajočih agentnih vlog. Tako agente opisujemo z njihovimi vlogami, sekvence akcij pa preoblikujemo v usmerjen graf, kjer povezave predstavljajo akcijske koncepte. MASDA skupen opis agentov in njihovih akcij dopolni s pravili, ki podrobneje opisujejo domenski prostor.

Bistveni lastnosti MASDA sta, da ne uporablja znanja v obliki v naprej definiranih zbirk planov oz. strategij ter, da vključuje proces abstrakcije, kar omogoča opisovanje večagentnega delovanja na različnih nivojih abstrakcije. Ker je osnovni princip abstrakcije v MASDA združevanje domensko in konceptualno podobnih akcij, lahko algoritem zazna le večagentno delovanje, ki se skozi čas ponavlja. Zato je za njegovo uspešno delovanje potrebno, da se strateško delovanje agentov skozi čas ponavlja v podobni obliki, za razliko od lokalno pogojenega delovanja agentov, ki se kaže v raznolikih oblikah. Ker je vhod v MASDA dejansko delovanje večagentnega sistema, lahko algoritem odkrije le tiste strategije, ki so bile v opazovanem času izvajane. MASDA tako ni sposoben odkriti morebitnih strategij, ki niso bile izvajane.

Algoritem je v obliki psevdo kode predstavljen na sliki 17.

```
Vhod:
- hierarhije značilk: H
- seznam zaporedja agentnih akcij: actionList[]
- funkcija razdalje: dist

Algoritem:
  // I. gradnja grafičnega akcijskega modela
  // I.1 gradnja akcijskega modela
  AG = createAG(actionList[], H)
  // I.2 gradnja ustreznega abstraktnega akcijskega modela
  AAG = convertAGtoAAG(AG)
  AAG_abs = AbstractionProces(AAG, dist, abs) // proces abstrakcije
  // I.3 izbira strateških konceptov makroakcij
  strategicPaths = {}
  for each (path ∈ AAG_abs) {
    if (isStrategic(path)) {
      strategicPaths = strategicPaths ∪ {path}
    }
  }
  // II. simbolni opis akcijskega modela
  for each (path ∈ strategicPaths) {
    // II.1 opisovanje konceptov makroakcij
    for each (e ∈ path.Edges) {
      e.action = common_parent(e.actions)
      e.role = common_parent(e.roles)
    }
    for each (e ∈ path.Edges) {
      // II.2 določitev učnih podatkov
      e.examples = chooseLearningExamples(e, AAG)
      // II.3 induciranje pravil
      data = generateLearnData(e.examples)
      e.rules = induceRules(data)
    }
  }

Izhod:
- strateški koncepti makroakcij: strategicPaths
```

Slika 17. Psevdo koda algoritma za strateško modeliranja MASDA.

6.1 Gradnja grafičnega akcijskega modela (I. korak)

6.1.1 Gradnja akcijskega modela (I.1)

Ker moramo pri modeliranju večagentnih sistemov upoštevati časovno komponento izvajanja akcij, uporabimo za teoretično osnovo pri modeliranju časovnega vidika agentnih akcij področje časovnega sklepanja (angl. *temporal reasoning*). Nanj lahko gledamo kot na poseben primer problema zadoščanja omejitvam (angl. *constraint satisfaction*) [116], kjer predstavljajo spremenljivke časovne entitete in omejitve množico možnih časovnih relacij med spremenljivkami. Predlaganih je bilo več pristopov [4, 97], ki omogočajo predstavitev in sklepanje o časovnih dogodkih. Za naše namene smo uporabili ***algebro časovnih točk*** (angl. *time point algebra*) [117]. To je relacijska algebra, ki je določena z elementi množice osnovnih časovnih relacij $T = \{<, >, =, \leq, \geq, \neq, <>, \emptyset\}$ ter operacijami inverza, preseka in unije množic. Algebra

časovnih točk omogoča definicijo ***časovno označenega grafa*** (angl. *temporally labeled graph*, krajše TL graf) [5, 43], ki omogoča predstavitev in sklepanje o časovno pogojenih dogodkih, ki so določenimi s točkami v času. Ker lahko začetek in konec agentnih akcij obravnavamo kot časovne dogodke, bomo uporabili algebro časovnih točk za določanje časovnih relacij in odvisnosti med agentnimi akcijami.

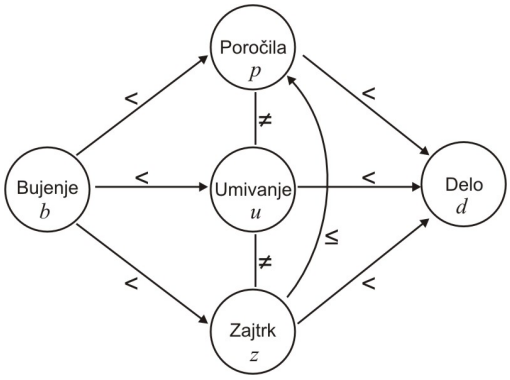
Definicija 19. Spremenljivka, ki predstavlja časovno točko nekega dogodka, je ***TP spremenljivka*** (angl. *time-point variable*).

Definicija 20. Množica, ki vsebuje vrednosti TP spremenljivk, je ***TP množica***.

Definicija 21. ***Časovno označen graf*** (krajše TL graf) $G(V, E)$ je graf z vozlišči V in množico označenih povezav E, kjer vsaka povezava $a \rightarrow b:(a, l, b) \in E$ povezuje vozlišči $a \in V$ in $b \in V$ z oznako l. Povezave so usmerjene in označene s '$<$' in '$\leq$' ali neusmerjene in označene s '$\neq$'. Vsako vozlišče ima lastno TP spremenljivko.

Oznaka povezave določa relacijo med TP spremenljivkama, ki jih določata začetno in končno vozlišče, in odgovarjajo osnovnim relacijam algebre časovnih točk. Slika 18 kaže primer TL grafa, ki prikazuje časovne odvisnosti na primeru jutranjih aktivnosti. Iz primera lahko sklepamo na naslednje zakonitosti:

- delo se vedno začne po bujenju ($b < u < d$),
- poročila gledamo ob ali po zajtrku ($z \leq p$),
- med zajtrkom ali gledanjem poročil se ne umivamo ($z \neq u \neq p$).



Slika 18. Primer TL grafa, ki prikazuje časovne odvisnosti med jutranjimi aktivnostmi.

Definicija 22. Pot v TL grafu je ***path$_{\leq}$***, če je vsaka oznaka povezave v poti $path_{\leq}$ ali '$\leq$' ali '$<$'.

Definicija 23. Pot v TL grafu je ***path$_{<}$*** , če je $path_{\leq}$ in je vsaj ena oznaka povezave v poti enaka '$<$'.

Definicija 24. Cikel v TL grafu je ***cycle$_{\leq}$*** (***cycle$_{<}$***), če je pot $path_{\leq}$ ($path_{<}$) dolžine n iz $v_0 \in V$ do $v_n \in V$, za $n \geq 1$ in je $v_0 = v_n$.

Definicija 25. TL graf je ***acikličen***, če ne vsebuje nobenega cikla $cycle_{\leq}$.

Primer na sliki 18 prikazuje acikličen TL graf z naslednjimi potmi $path_{<}$: b→p→d, b→t→d, b→z→d in b→z→p→d.

Definicija 26. Množica povezav, ki izhaja iz vozlišča $v \in V$ v grafu G(V, E), kjer je V množica vozlišč in E množica povezav, je ***out(v)*** = $\{\forall e \in E: v \rightarrow x ; v \in V, x \in V\}$.

Definicija 27. Množica povezav, ki vodi v vozlišče $v \in V$ v grafu G(V, E), kjer je V množica vozlišč in E množica povezav, je ***in(v)*** = $\{\forall e \in E: x \rightarrow v ; x \in V, v \in V\}$.

Definicija 28. Začetno vozlišče povezave $e \in E$ v grafu G(V, E), kjer je V množica vozlišč in E množica povezav, je ***source(e)*** = $\{v \in V; e \in out(v)\}$.

Definicija 29. Končno vozlišče povezave $e \in E$ v grafu G(V, E), kjer je V množica vozlišč in E množica povezav, je ***destination(e)*** = $\{v \in V; e \in in(v)\}$.

Ti konstrukti nam omogočajo, da definiramo osnovno podatkovno strukturo, ki bo opisala večagentno obnašanje.

Definicija 30. ***Akcijski graf AG(V, E)*** je acikličen TL graf z vozlišči *V* in množico označenih povezav *E*, kjer so vse povezave usmerjene in označene z '<'. Povezava $e \in E$ je določena s *e*:(*source, destination, agent, role, action, start, duration*), kjer je *source* začetno vozlišče, *destination* končno vozlišče, *agent* ime agent, *role* $\in H_{role}$ vloga agenta *agent* iz hierarhije agentnih vlog, *action* $\in H_{action}$ akcija iz hierarhije agentnih akcij, *start* čas začetka akcije in *duration* trajanje akcije. Vozlišče $v \in V$ je določeno s *v*:(*agent, role, action, tp, pos*), kjer je *agent* ime agent, *role* agentna vloga agenta *agent, action* opis agentne akcije, *tp* je TP spremenljivka in *pos* domenska pozicija agenta *agent* v času *tp*. Stanje sistema določimo s poizvedovanjem vrednosti atributov v sledi delovanja, ki opisujejo sistem v času *tp*.

Za vsako vozlišče akcijskega grafa *AG(V,E)* velja, da izhodne povezave predstavljajo isto instanco akcije:

$$\forall v \in V : (\forall a \in out(v), \forall b \in out(v) \Rightarrow$$
$$(a.agent = b.agent) \wedge (a.role = b.role) \wedge (a.action = b.action) \wedge$$
$$(a.start = b.start) \wedge (a.duration = b.duration))$$

V akcijskem grafu ločimo dva tipa vozlišč: vozlišča z izhodi in končna vozlišča. Vozlišče z izhodi *v* ima vsaj eno izhodno povezavo: $|out(v) > 0|$. Če vozlišče akcijskega grafa nima izhodne povezave, $|out(v) = 0|$, ga imenujemo končno vozlišče in označimo z $v_{terminal}$. Vrednosti spremenljivk v vozlišču z izhodi *v* so sledeča: $v.role = e.role: e \in out(v)$, $v.action = e.action: e \in out(v)$ in $v.tp = e.start: e \in out(v)$. Vrednosti spremenljivk v končnem vozlišču $v_{terminal}$ so: $v_{terminal}.role = e.role: e \in in(v)$, $v_{terminal}.action = e.action: e \in in(v)$ in $v_{terminal}.tp = (e.start+e.duration): e \in in(v)$. Če v končno vozlišče vodi več povezav, imajo vse akcije, ki jih predstavljajo vhodne povezave, enak čas zaključka.

Akcijski graf zgradimo iz akcijskega zaporedja na naslednji način. Za vse akcije agentov <*action, start, duration, agent, role*> kreiramo ustrezna vozlišča *node(agent, role, action, start, pos*), kjer je *pos* domenska pozicija agenta *agent*. Nato za vsako akcijo <*action, start, duration, agent, role*> in odgovarjajoče vozlišče *node(agent, role, action, start, pos*) poiščemo vozlišča *node'(agent', role', action', start', pos'*), kjer je *start' = start+duration* in jih povežemo s povezavo <*node, node', agent, role, action, start, duration*>. Če ne najdemo nobenega ustreznega vozlišča *node'*, kreiramo

novo končno vozlišče *node*$_{terminal}$*(agent, role, action, start+duration, pos*) ter ga povežemo s povezavo <*node, node*$_{terminal}$*, agent, role, action, start, duration*>. Opisani postopek je kot algoritem predstavljen na sliki 19.

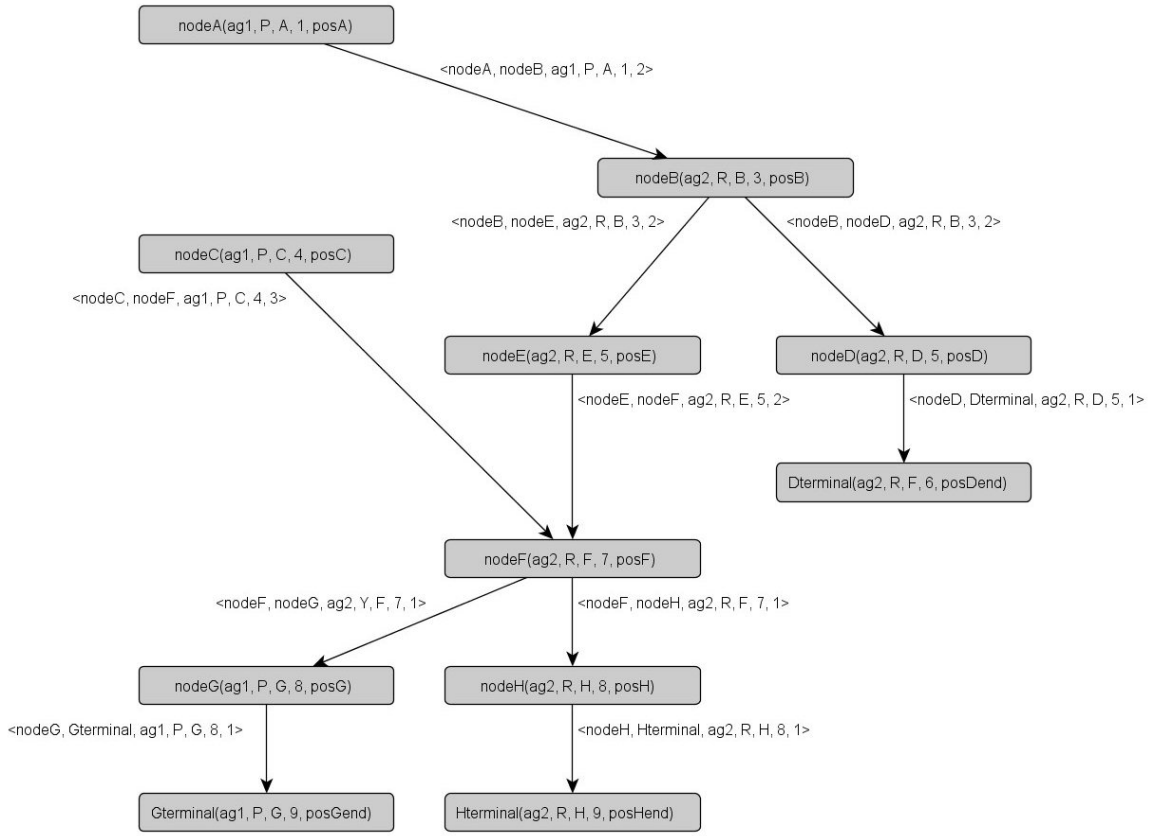
```
Vhod:
- število agentov: NumAgents
- zaporedje agentnih akcij: actionList[][NumAgents]
- število zaznanih akcij: NumDetectedActions[NumAgents]
- domenske pozicije agentov: position[MaxT][NumAgents]

Algoritem:
new(AG) // kreiramo prazen akcijski graf
for (agent = 0; agent < NumAgents; agent++) {
  for (pos = 0; pos < NumDetectedActions[agent]; pos++) {
    // za vse zaznane akcije agenta kreiramo ustrezna vozlišča
    AG.NewNode(agent, actionList[pos][agent].role,
               actionList[pos][agent].action,
               actionList[pos][agent].start,
               position[n.startTime][agent])
  }
}
for (agent = 0; agent < NumAgents; agent++) {
  // za vsa vozlišča poiščemo ustrezne izhodne povezave
  for (pos = 0; pos < NumDetectedActions[agent]; pos++) {
    // poiščemo ustrezno izhodiščno vozlišče
    n = AG.FindNode(agent, actionList[pos][agent].role,
                    actionList[pos][agent].action,
                    actionList[pos][agent].startTime)
    foundNext = false
    for each ((next ∈ AG.V) && (next != n) &&
              (next.startTime == n.startTime+n.duration)) {
      // poiščemo vsa sledeča vozlišča in jih povežemo
      AG.NewEdge(n, next, agent, actionList[pos][agent].role,
                 actionList[pos][agent].action,
                 actionList[pos][agent].start,
                 actionList[pos][agent].duration)
      foundNext = true
    }
    if (!foundNext) {
      // če ni sledečega vozlišča kreiramo končno vozlišče
      start = actionList[pos][agent].start +
              actionList[pos][agent].duration
      nTerminal = AG.NewNode(agent, actionList[pos][agent].role,
                             actionList[pos][agent].action, start,
                             position[start][agent])
      // povežemo izhodiščno in končno vozlišče
      AG.NewEdge(n, nTerminal, agent, actionList[pos][agent].role,
                 actionList[pos][agent].action,
                 actionList[pos][agent].start,
                 actionList[pos][agent].duration)
    }
  }
}

Izhod:
- akcijski graf: AG
```

Slika 19. Algoritem, ki pretvori zaporedje agentnih akcij v akcijski graf.

Gradnjo akcijskega grafa predstavimo na naslednjem primeru, ki prikazuje delovanje večagentnega sistema v časovnem intervalu T={1, 2, 3, 4, 5, 6, 7, 8}. Agenta ag_1 in ag_2 sta v tem času opravila zaporedje akcij, ki so predstavljene v časovno urejeni preglednici 11. Odgovarjajoči akcijski graf je predstavljen na sliki 20.

<table>
<tr><th>t</th><th>ag_1</th><th colspan="2">ag_2</th></tr>
<tr><td>1</td><td rowspan="2"><A, 1, 2, ag_1, P></td><td colspan="2"></td></tr>
<tr><td>2</td><td colspan="2"></td></tr>
<tr><td>3</td><td></td><td colspan="2" rowspan="2"><B, 3, 2, ag_2, R></td></tr>
<tr><td>4</td><td rowspan="3"><C, 4, 3, ag_1, P></td></tr>
<tr><td>5</td><td><D, 5, 1, ag_2, R></td><td rowspan="2"><E, 5, 2, ag_2, R></td></tr>
<tr><td>6</td><td></td></tr>
<tr><td>7</td><td></td><td colspan="2"><F, 7, 1, ag_2, R></td></tr>
<tr><td>8</td><td><G, 8, 1, ag_1, P></td><td colspan="2"><H, 8, 1, ag_2, R></td></tr>
</table>

Preglednica 11. Primer instanc akcij dveh agentov v časovnem intervalu t∈[1, 8].



Slika 20. Akcijski graf, ki ustreza instancam agentnih akcij iz preglednice 11.

Akcijski graf [12] je učinkovita podatkovna struktura, ki omogoča hranjenje, predstavitev in poizvedovanje o sekvenčnem, časovnem in akcijskem vidiku večagentnega obnašanja. Uporabimo ga lahko za predstavitev poljubnega časovnega in akcijskega vidika delovanja večagentnega sistema. To prikazuje primer na preglednici 12, kjer so predstavljene vse osnovne časovne relacije [43] iz intervalne

algebre [4] in ustrezni primeri akcij, predstavljenih v preglednici 11 in akcijskem grafu na sliki 20.

Časovna relacija	Primer relacije v časovnem diagramu							Ustrezni akciji iz primera v preglednici 11
X *before* Y	**X**	**X**	**X**		**Y**	**Y**	**Y**	X=<A, 1, 2, ag_1, P>, Y= <C, 4, 3, ag_1, P>
Y *after* X								
X *meets* Y	**X**	**X**	**X**	**X**				X=<F, 7, 1, ag_2, R>, Y=<H, 8, 1, ag_2, R>
Y *met-by* X					**Y**	**Y**	**Y**	
X *overlaps* Y			**X**	**X**	**X**			X=<B, 3, 2, ag_2, R>, Y=<C, 4, 3, ag_1, P>
Y *overlapped-by* X				**Y**	**Y**	**Y**		
X *during* Y			**X**	**X**	**X**			X=<D, 5, 1, ag_2, R>, Y=<C, 4, 3, ag_1, P>
Y *contains* X		**Y**	**Y**	**Y**	**Y**	**Y**		
X *starts* Y		**X**	**X**	**X**				X=<D, 5, 1, ag_2, R>, Y=<E, 5, 2, ag_2, R>
Y *started-by* X		**Y**	**Y**	**Y**	**Y**	**Y**		
X *finishes* Y				**X**	**X**	**X**		X=<E, 5, 2, ag_2, R>, Y=<C, 4, 3, ag_1, P>
Y *finished-by* X		**Y**	**Y**	**Y**	**Y**	**Y**		
X *equal* Y			**X**	**X**	**X**			X=<G, 8, 1, ag_1, P>, Y=<H, 8, 1, ag_2, R>
			Y	**Y**	**Y**			

Preglednica 12. Trinajst osnovnih časovnih relacij iz intervalne algebre in ustrezni primeri akcij iz akcijskega grafa na sliki 20.

Omejitev, da iz vozlišča izhajajo le povezave, ki predstavljajo isto instanco akcije, se zdi na prvi pogled neutemeljena. Če agent v danem trenutku začne vzporedno opravljati akcijo A in akcijo B, bi tako situacijo lahko predstavili z dvema izhodnima povezavama, ki vsaka predstavlja svojo akcijo. Taki predstavitvi se zavedno odpovemo, ker želimo, da izhajajoče povezave iz enega vozlišča opisujejo eno instanco akcije, kar s pridom izkoristimo v kasnejšem procesu abstrakcije. Primer predstavitve vzporednih akcij z akcijskim grafom je prikazan na primeru na sliki 20.

MASDA ne predpisuje načina izrisa akcijskega grafa, vendar lahko domenske pozicije, ki so vsebovane v vozliščih akcijskega grafa, uporabimo kot domensko specifične koordinate vozlišč. To se še posebej izkaže v domenah, kjer agenti delujejo v prostoru, ki je predstavljiv v dveh dimenzijah. Ker tak prikaz človeku precej olajša prostorsko razumevanje večagentnega obnašanja, bomo v nadaljevanju vse primere predstavljali, kot da agenti delujejo v dvodimenzionalnem prostoru.

Čeprav lahko v splošnem agenti opravljajo več različnih vlog hkrati, uporablja akcijski graf za predstavitev akcije samo eno vlogo. Ta omejitev omogoča MASDA, da v kasnejšem procesu abstrakcije učinkovito abstrahira opise agentnih vlog. V primeru, da modeliramo večagentno domeno, kjer lahko agenti opravljajo več vlog hkrati, zgradimo novo hierarhijo vlog z vlogami, ki ustrezajo vsem možnim kombinacijam izhodiščnih vlog.

6.1.2 Gradnja abstraktnega akcijskega modela (I.2)

Akcijski graf je podatkovna struktura, ki je primerna za predstavitev poljubnega večagentnega akcijskega zaporedja in omogoča tako programsko kot tudi človeško

analizo. Slednja je v primeru kompleksnega akcijskega grafa z mnogimi vozlišči in povezavami otežena, ker je tak graf težko grafično predstaviti v človeku sprejemljivi obliki. Ker ljudje lažje dojamemo manjše strukture kot večje, je smiselno poiskati način, ki bi zmanjšal število vozlišč in povezav v akcijskem grafu in obenem ohranil informacijo, ki jo ima osnovna predstavitev. Zato predstavimo novo podatkovno strukturo, imenovano ***abstrakten akcijski graf***, ki omogoča bolj kompaktno predstavitev kot akcijski graf.

Definicija 31. ***Abstrakten akcijski graf AAG(V, E)*** je graf z vozlišči V in množico usmerjenih povezav E. Povezava $e \in E$ je določena z e:(*source*, *destination, instances*), kjer je *source* začetno vozlišče, *destination* končno vozlišče in *instances* množica instanc akcij *instances:{(agent, role, action, start, duration)*$^+$*}*, pri čemer je *agent* agent, *role* $\in H_{role}$ agentna vloga iz hierarhije agentnih vlog, *action* $\in H_{action}$ agentna akcija iz hierarhije agentnih akcij iz hierarhije, *start* čas začetka akcije in *duration* trajanje akcije. Vozlišče $v \in V$ je določeno s v:(*agents*, *roles*, *actions*, *ts*, *pos*), kjer je *agents* množica agentov, *roles* množica agentnih vlog iz H_{role}, *actions* množica agentnih akcij iz H_{action}, *ts* je TP množica in *pos* je povprečna domenska pozicija agentov *agents* v časih, ki jih določa *ts*.

Za vsako vozlišče $v \in V$ iz abstraktnega akcijskega grafa AAG(V, A) velja naslednje:

$$\text{Če } |out(v)| > 0: \quad \begin{cases} v.roles = \{i.role;\ \forall e \in out(v), \forall i \in e.instances\} \\ v.actions = \{i.action;\ \forall e \in out(v), \forall i \in e.instances\} \\ v.ts = \{i.start;\ \forall e \in out(v), \forall i \in e.instances\} \end{cases}$$

$$\text{Če } |out(v)| = 0: \quad \begin{cases} v.roles = \{i.role;\ \forall e \in in(v), \forall i \in e.instances\} \\ v.actions = \{i.action;\ \forall e \in in(v), \forall i \in e.instances\} \\ v.ts = \{(i.start+i.duration);\ \forall e \in in(v), \forall i \in e.instances\} \end{cases}$$

Postopek, ki prevede opis iz AG v ekvivalenten opis v AAG, je podan z algoritmom, predstavljenim na sliki 21. AG iz katerega smo pridobili AAG, imenujemo ***izhodiščni akcijski graf***.

Tako dobljeni AAG sicer ne predstavlja kompaktnejše oblike opisa kot izhodiščni AG, omogoča pa, da to naredimo. Ker lahko v AAG z eno povezavo predstavimo več instanc agentnih akcij, je možno zgraditi bolj kompakten opis kot z uporabo akcijskega grafa. Ker lahko povezava iz AAG predstavlja več povezav iz AG, je zato njen opis agentne akcije abstraktnejši kot v AG. Posledično lahko predstavlja opis z AAG bolj abstrakten akcijski model kot opis z AG.

Predpostavka postopka abstrakcije je, da podobne akcije, ki jih izvajajo agenti s podobnimi vlogami v bližnjem domenskem prostoru, predstavljajo podoben akcijski koncept. Torej lahko zmanjšamo število vozlišč v AAG tako, da združujemo vozlišča, ki predstavljajo podobne akcijske koncepte. Nova vozlišča tako predstavljajo skupen opis akcijskih konceptov združenih vozlišč.

```
Vhod:
- akcijski graf: AG

Algoritem:
// kreiraj prazen AAG
new(AAG)
// pretvori vsa vozlišča
for each (v_AG ∈ AG.V) {
  v_AAG = AAG.NewNode()
  v_AAG.agents = {v_AG.agent}
  v_AAG.roles = {v_AG.role}
  v_AAG.actions = {v_AG.action}
  v_AAG.ts = {v_AG.tp}
  v_AAG.pos = v_AG.pos
}

// pretvori vse povezave
for each (e_AG ∈ AG.E) {
  e_AAG = AAG.NewEdge()
  e_AAG.source = e_AG.source
  e_AAG.destination = e_AG.destination
  e_AAG.instances = {(e_AG.role, e_AG.action, e_AG.start, e_AG.duration)}
}

Izhod:
- abstrakten akcijski graf: AAG
```

Slika 21. Algoritem, ki prevede akcijski graf v ekvivalenten abstrakten akcijski graf.

Za merjenje podobnosti akcijskih konceptov potrebujemo metriko podobnosti akcijskih opisov. To določimo kot heterogeno funkcijo razdalje [122] *dist(a, b)*, ki meri konceptualno razdaljo med akcijskima konceptoma v vozliščih $a, b \in$ AAG. Zanjo velja, da manjša, kot je vrednost *dist(a, b)*, bolj sta si akcijska koncepta v vozliščih *a* in *b* podobna. Če je *dist(a, b)*=0, potem sta akcijska koncepta v vozliščih *a* in *b* enaka. Vsaka metrika razdalje, torej tudi *dist(a, b)*, mora zadovoljiti trem aksiomom razdalje [22]:

1. Nenegativnost: $\forall\ a, b$: $(dist(a, b) \geq 0) \wedge (dist(a, b) = 0 \Leftrightarrow a = b)$
2. Simetrija: $dist(a, b) = dist(b, a)$
3. Trikotniška neenakost: $\forall\ a, b, c$: $dist(a, b) \leq dist(a, c) + dist(c, b)$

Problem iskanja ustrezne funkcije *dist(a, b)* je izrazito domensko specifičen, saj je razdaljo med simbolnimi opisi akcij in agentnih vlog v večdimenzionalnem domenskim prostoru potrebno določiti s poznavanjem njihovega resničnega pomena. Tako Aggarwal [1] trdi, da je izbira metrik razdalje v kompleksnih domenah uporabniško naravnana, saj je prav uporabnik tisti, ki ocenjuje in uporablja dobljene modele. Poleg tega pokaže, da je uspešnost parametričnih metrik v kompleksnih domenah boljša od uporabe fiksnih metrik.

Naša definicija razdalje med vozlišči v AAG je zato temu primerno splošna in upošteva razdaljo med hierarhičnimi opisi akcij $dist_{Haction}(a, b)$, razdaljo med hierarhičnimi opisi agentih vlog $dist_{Hrole}(a, b)$ in razdaljo med pozicijami agentov v domenskem prostoru $dist_{pos}(a, b)$.

Definicija 32. ***Razdalja med vozlišči*** $a \in V$ in $b \in V$ v abstraktnem akcijskem grafu *AAG(V, E)* je ***dist(a, b)***, ki je definirana kot:

$$dist(a, b) = w_{pos} \cdot \overline{dist_{pos}(a,b)} + w_{role} \cdot \overline{dist_{Hrole}(a,b)} + w_{action} \cdot \overline{dist_{Haction}(a,b)},$$

kjer je w_{pos} utež, ki določa razmerje s funkcijo povprečne oddaljenosti med pozicijami agentov v domenskem prostoru $\overline{dist_{pos}(a,b)}$, w_{role} utež, ki določa razmerje s funkcijo povprečne oddaljenosti med elementi *a.roles* in *b.roles* $\overline{dist_{Hrole}(a,b)}$, in w_{action} utež, ki določa razmerje s funkcijo povprečne oddaljenosti elementov *a.actions* in *b.actions* $\overline{dist_{Haction}(a,b)}$. Funkcija povprečne oddaljenosti med pozicijami agentov $\overline{dist_{pos}(a,b)}$ je definirana kot: $\overline{dist_{pos}(a,b)} = dist_{pos}(a.pos, b.pos)$. Funkcija razdalje $dist_{pos}(a.pos, b.pos)$ je domensko specifična, zato jo tu ne definiramo. Povprečna oddaljenost opisov agentnih vlog $\overline{dist_{Hrole}(a,b)}$ je definirana kot:

$$\overline{dist_{Hrole}(a,b)} = \sum_{\forall (x,y): x \in a.roles, y \in b.roles} \frac{dist_{Hrole}(x,y)}{|a.roles| \cdot |b.roles|}$$

Povprečna oddaljenost opisov akcij $\overline{dist_{Haction}(a,b)}$ je definirana sledeče:

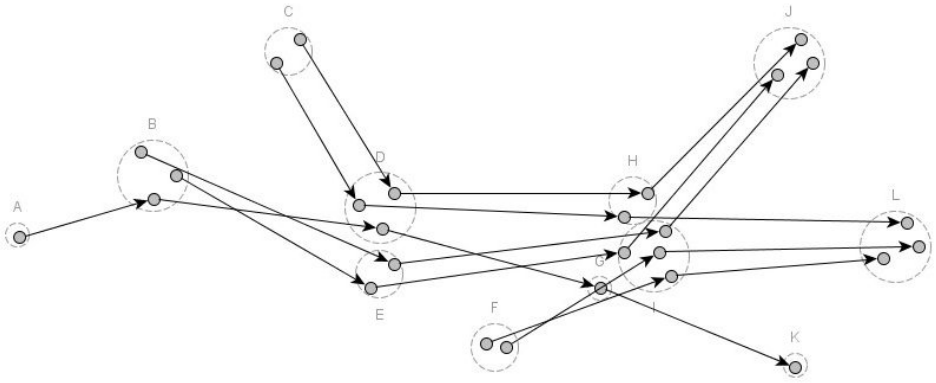
$$\overline{dist_{Haction}(a,b)} = \sum_{\forall (x,y): x \in a.actions, y \in b.actions} \frac{dist_{Hactions}(x,y)}{|a.actions| \cdot |b.actions|}$$

Če imamo na voljo ustrezno metriko razdalje, lahko uporabimo algoritem, ki dani AAG z združevanjem vozlišč pretvori v kompaktnejšo obliko. Osnova algoritma je predpostavka, da bližnja vozlišča predstavljajo podobne akcijske koncepte. Algoritem iterativno išče najbližji vozlišči istega tipa (obe končni ali obe z izhodiščnimi povezavami). Če je razdalja med njima manjša ali enaka ustavitveni vrednosti parametra *abs*, ju združi v novo vozlišče in ustrezno preveže povezave med združenimi vozlišči. Postopek se ponavlja dokler najmanjša razdalja med vozlišči ne preseže vrednosti *abs*. Parametru ***abs*** pravimo ***parameter abstrakcije***. Tak postopek združevanja najbližjih vozlišč v AAG imenujemo ***postopek abstrakcije***, saj se z vsako združitvijo dveh vozlišč v AAG njihov opis abstrahira. Rezultat je AAG, ki ima vse razdalje med vozlišči večje od *abs*. Algoritem, ki opravlja postopek abstrakcije, je predstavljen na sliki 23.

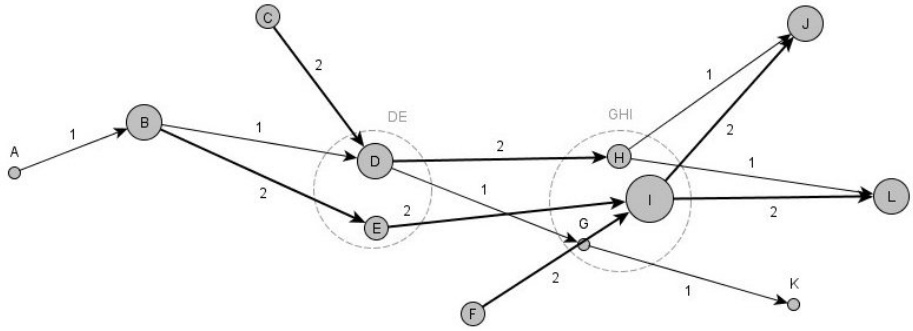
Definicija 33. ***Abstrakten akcijski graf z abstraktnostjo abs***, $\boldsymbol{AAG_{abs}(V,E)}$, je abstrakten akcijski graf *AAG(V, E)*, kjer velja:

$$\forall a, b \in V: dist(a, b) > abs$$

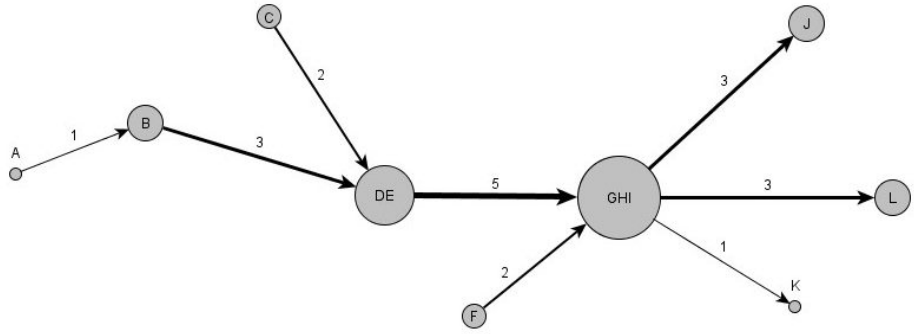
Metrika razdalje v sicer domensko neodvisen postopek abstrakcije vnese domensko znanje, ki določa podobnosti in razlike med akcijskimi koncept, zato tudi bistveno vpliva na končni AAG. S spreminjanjem metrike razdalje lahko gradimo AAG, ki prikazujejo različne značilnosti istega večagentnega sistema.



a) Shematski primer izhodiščnega AG. Svetli črtkani krogi prikazujejo vozlišča, ki so med seboj oddaljena manj od parametra abstrakcije abs_1.



b) Shematski primer AAG_{abs1}. Svetli črtkani krogi označujejo vozlišča, ki so med seboj oddaljena manj od parametra abstrakcije abs_2.



c) Shematski primer AAG_{abs2}.

Slika 22. Primer procesa abstrakcije AG v AAG_{abs1} in nato v AAG_{abs2}. Velikost vozlišč je sorazmerna s številom združenih vozlišč izhodiščnega AG. Številke na povezavah in njihova debelina odgovarjajo številu povezav, ki so povezovala združena vozlišča v izhodiščnem AG.

Iz danega AAG lahko zgradimo vrsto abstraktnih akcijskih grafov z naraščajočo abstraktnostjo. Tako imamo na voljo cel spekter akcijskih modelov, ki opisujejo dano večagentno delovanje na različnih nivojih abstrakcije. AAG_{abs} z večjo vrednostjo

parametra *abs* predstavlja bolj abstrakten model večagentnega delovanja, kot $AAG_{abs'}$ z manjšim parametrom abstrakcije *abs'* < *abs*.

Primer gradnje AAG bomo predstavili na naslednjem preprostem primeru. Recimo, da imamo na voljo izhodiščni AG, predstavljen na sliki 22a, kjer pozicije vozlišč ustrezajo domenskim pozicijam agentov. Za potrebe primera privzemimo, da opravljajo vsi agenti isto vlogo ter izvajajo samo eno akcijo - premik v prostoru. Parametri akcije premika sta kot obrata in dolžina premika. Domensko pozicijo vozlišča *pos* predstavimo kot koordinati v dvodimenzionalnem prostoru *pos:(x, y).* Metrika razdalja je evklidska razdalja med pozicijami vozlišč v dvodimenzionalnem domenskem prostoru: $dist(a,b) = \sqrt{(a.pos.x - b.pos.x)^2 + (a.pos.y - b.pos.y)^2}$. Pri določeni vrednosti parametra abstrakcije abs_1 zgradi postopek abstrakcije AAG_{abs1}, ki je predstavljen na sliki 22b. Postopek abstrakcije AAG_{abs1} lahko nato ponovimo z večjim parametrom abstrakcije abs_2, $abs_1 < abs_2$. Rezultat abstrakcije je AAG_{abs2}, ki je predstavljen na sliki 22c in predstavlja bolj abstrakten večagentni model od AAG_{abs1}, ki ne ločuje akcijskih konceptov, ki so v AAG_{abs1} opisani v vozliščih D in E ter G, H in I.

```
Vhod:
- abstrakten akcijski graf: AAG
- funkcija razdalje med vozlišči: dist(n1, n2)
- ciljna vrednost abstrakcije: abs

Algoritem:
// proces abstrakcije
while (true) {
  // poišči najbližji vozlišči
  minDist = ∞
  for each (a ∈ AAG.V) {
    for each (b ∈ AAG.V) {
      if (a == b) {
       continue
      }
      if ((dist(a, b) <= minDist) && (a.IsTerminal == b.IsTerminal)) {
        minA = a
        minB = b
        minDist = dist(a, b)
      }
    }
  }

  // preveri, če sta najdeni vozlišči dovolj blizu
  if (minDist > abs) {
    break // prekini proces abstrakcije
  }

  // združi vozlišči
  c = AAG.NewNode()
  c.actions = a.actions ∪  b.actions
  c.roles = a.roles ∪  b.roles
  c.ts = a.ts ∪ b.ts
  c.pos = avg(position of agents in c.instances)

  // ustrezno preveži povezave iz vozlišč a in b v vozlišče c
  for each (e ∈ (out(a) ∪ out(b))) e.source = c
  for each (e ∈ (in(a) ∪ in(b))) e.destination = c
  // združi povezave, ki povezujejo isti vozlišči
  for each (e,f ∈ (in(c) ∪ out(c))) {
    if ((e.source == f.source) && (e.destination == f.destination)) {
      e.instances = e.instances ∪ f.instances
      AAG.RemoveEdge(f)
    }
  }
  AAG.RemoveNode(a)
  AAG.RemoveNode(b)
}

Izhod:
- abstrakten akcijski graf: AAGabs
```

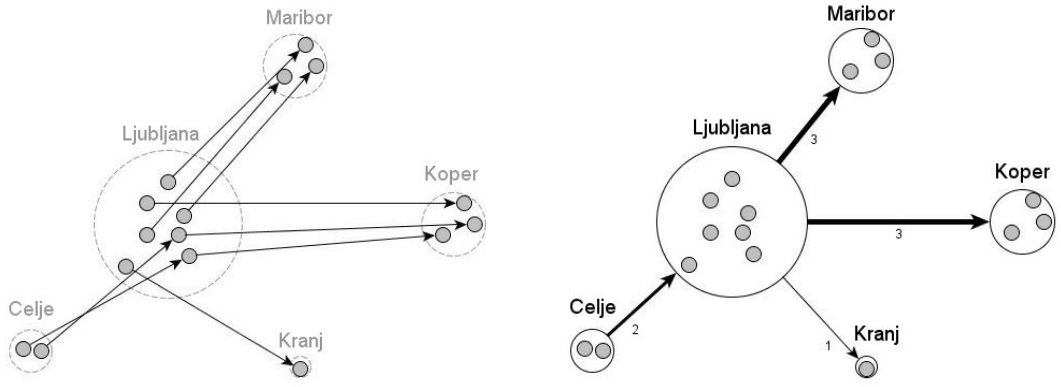
Slika 23. Algoritem za postopek abstrakcije v abstraktnem akcijskem grafu.

Če so vozlišča v izhodiščnem AG predstavljala začetke oz. zaključke, povezave pa celotne instance akcij, se je njihov pomen v AAG spremenil. MASDA v postopku

abstrakcije združuje vozlišča in povezave, zato le-ti predstavljajo nove akcijske koncepte. Vozlišče ne predstavlja zgolj koncepta začetka agentne akcije, saj lahko za razliko od izhodiščnega AG iz vozlišča v AAG izhaja več povezav, ki predstavljajo različne akcijske koncepte. Torej predstavlja vozlišče v AAG koncept začetka abstraktne akcije, ki je splošnejši od vseh akcijskih konceptov, ki jih predstavljajo izhodne povezave. Če iz vozlišča v AAG ne izhaja nobena povezava, predstavlja tako vozlišče koncept zaključka akcije, ki ga predstavljajo vhodne povezave. Povezava v AAG vsebuje množico pomensko podobnih instanc akcij, zato predstavlja akcijski koncept v domenskem podprostoru, ki ga omejuje začetno in končno vozlišče.

Z uporabo simbolov iz definicije večagentnega sistema, določa vozlišče $v \in V$ v abstraktnem akcijskem grafu AAG(V, E) podprostor: $AG \times R \times S_{start}$, kjer je $S_{start} \subseteq S$ množica stanj, ki jih opisujejo primeri vozlišča *v*. Podobno določa povezava $e \in E$ podprostor: $AG \times R \times S_{start} \times S_{end}$, kjer je $S_{start} \subseteq S$ množica stanj, ki jih opisuje vozlišče *source(e)* in $S_{end} \subseteq S$ množica stanj, ki jih opisuje vozlišče *destination(e)*.

Za ponazoritev teh pomenov vzemimo naslednji preprost primer. Recimo, da z večagentnim sistemom simuliramo promet med petimi slovenskimi mesti, naš namen pa je analiza agentnih akcij. Zaradi enostavnosti vsi agenti opravljajo isto vlogo prevoznika, ter izvajajo le eno akcijo: vožnjo z avtomobilom iz enega mesta v drugega, kjer cilj vožnje upoštevamo kot parameter akcije. Vhod v MASDA je sled, ki predstavlja dvodimenzionalni opis poteka vožnje. Iz sledi ugotovimo, da sta agenta iz Celja opravila dve vožnji preko Ljubljane v Koper. Trije agenti so se iz različnih koncev Ljubljane odpeljali v Maribor, eden se je odpeljal v Koper in eden v Kranj. Agentne akcije vožnje z avtomobilom lahko ponazorimo z akcijskim grafom, ki je poenostavljen predstavljen na sliki 24a. Za funkcijo razdalje vzamemo kar evklidsko funkcijo razdalje med dvema točkama v dveh dimenzijah, parameter abstrakcije pa nastavimo na abs=10km. S postopkom abstrakcije nato pretvorimo izhodiščni AG v odgovarjajoč AAG_{abs}, ki je shematsko predstavljen na sliki 24b. Vozlišča, ki so v izhodiščnem AG predstavljala domenske pozicije v oddaljenosti do 10 km, je postopek abstrakcije združil v nova vozlišča, ki predstavljajo posplošene domenske pozicije agentov - slovenska mesta.



Slika 24. a) Shematski primer izhodiščnega AG, ki predstavlja domeno vožnje z avtomobilom in b) ustrezni shematski primer AAG_{abs}, kjer števila na povezavah ustrezajo številu akcij v izhodiščnem AG, sivi krogci pa vozlišča v izhodiščnem AG.

Analiza AAG_{abs} pokaže, da povezave ustrezajo konceptu agentne akcije - vožnja iz enega mesta v drugega. V našem primeru koncepti akcij v povezavah opredelijo tudi začetno domensko pozicijo agenta in parameter akcije. Tako predstavlja povezava *Ljubljana-Koper* koncept agentne akcije: *vožnja iz Ljubljane v Koper*, kjer *Ljubjana* predstavlja domensko pozicijo agenta ob pričetku akcije, *Koper* pa vrednost parametra akcije. Iz primera vidimo, da vozlišče z izhodnimi povezavami predstavlja najmanj splošen koncept začetka akcije, ki še ustreza vsem akcijskim konceptom, ki jih predstavljajo izhodne povezave. V našem primeru prikazuje vozlišče *Ljubljana* splošen koncept začetka akcije, ki jo predstavljajo koncepti: *vožnja iz Ljubljane v Maribor*, *vožnja iz Ljubljane v Koper* in *vožnja iz Ljubljane v Kranj*. Vozlišče *Ljubljana* torej predstavlja koncept začetka agentne akcije: *vožnja iz Ljubljane*. V našem primeru vozlišče opredeli le začetno domensko pozicijo agenta. Podobno lahko sklepamo za vozlišča, ki imajo samo vhodne povezave in torej predstavljajo najmanj splošen koncept konca akcije, ki še ustreza akcijskim konceptom, ki jih predstavljajo vhodne povezave. Primer za to je vozlišče *Maribor*, ki opisuje koncept konca agentne akcije: *vožnja v Maribor*.

Povezava v AAG torej predstavlja akcijski koncept in določi akcijo v nekem omejenem prostoru parametrov akcije in domene. Vozlišče v AAG predstavlja koncept začetka oz. konca akcije in je določen z unijo omejitev prostora stanja ter parametrov akcij, ki ga določajo izhodne oz. vhodne povezave.

6.1.3 Izbira strateških konceptov makroakcij (I.3)

AAG je večagentni akcijski model, ki omogoča analizo večagentnega delovanja. Pri analizi večagentnega delovanja nas med drugim zanimajo zaporedja akcij, ki se pojavljajo bolj pogosto od ostalih [56]. Za taka zaporedja sklepamo, da so posledica izvrševanja večagentne strategije in ne posledica odzivov na lokalne spremembe okolja. Recimo, da modeliramo nek večagentni sistem, ki smo ga opazovali v daljšem časovnem obdobju, v katerem smo zasledili ponavljanje podobnih situacij. Sklepamo, da je del opravljenih akcij nastal zaradi izvajanja večagentne strategije, del pa zaradi lokalnega odzivanja agentov na spremembe okolja. Če ima večagentni sistem strategijo, ki je predstavljena kot neko delno urejeno zaporedje akcij, in če okolje omogoča agentom, da izvajajo strategijo, potem bodo akcije, ki ustrezajo strategiji, prej ali slej opravljene. Če v podobnih situacijah pogosto zasledimo podobna zaporedja akcij, potem lahko sklepamo, da podobna zaporedja akcij predstavljajo del večagentne strategije in ne zgolj agentni odziv na lokalne spremembe okolja. Zato privzamemo, da velja sledeča predpostavka: ***Načrtovano izvajanje strategije večagentnega sistema se v podobnih situacijah kaže v podobnih zaporedjih akcij, lokalno odzivanje agentov na "šumne" spremembe okolja pa v spremenljivih zaporedjih akcij.***

V AAG je vsako zaporedje agentnih akcij predstavljeno kot pot v grafu, kjer povezave v poti ustrezajo danemu zaporedju agentnih akcij. Pogosta akcijska zaporedja so tako predstavljena kot poti v AAG, katerih povezave predstavljajo večje število instanc akcij kot ostale povezave. Dolžina poti predstavlja število akcijskih konceptov, ki so vsebovane v akcijskem zaporedju.

Definicija 34. ***Pot v AAG***, označena s *path(V, E)*, je povezan podgraf v AAG, za katerega velja, da vsebujejo vozlišča z največ eno izhodno in največ eno vhodno povezavo:

$$\forall v \in path.V: (|out(v)| \leq 1) \wedge (|in(v)| \leq 1)$$

Če je pot *path* iz nekega AAG, torej je *path* podgraf AAG, to označimo kot *path* ∈ AAG.

Definicija 35. ***Dolžina poti*** *length(path)* je število povezav na poti *path*:

$$length(path) = |path.E|$$

Dovolj veliko število instanc akcij v povezavah pa ne zagotavlja, da predstavlja pot tudi dejansko neprekinjeno akcijsko zaporedje. Izvzeti moramo namreč primere, ko sicer sklenjena pot v AAG ne predstavlja sklenjenih poti v izhodiščnem AG. Omenjene pojme določata pojma šibke in močne povezanost poti.

Definicija 36. ***Šibka povezanost*** poti *path* v AAG, označena s $connection_{weak}$, je minimalno število instanc akcij, ki so opisane v povezavah v poti *path(V, E)*.

$$connection_{weak}(path) = \min_{\forall e \in path.E} (|e.actions|)$$

Definicija 37. ***Močna povezanost*** poti *path* v AAG, označena s $connection_{strong}(path)$, je število poti dolžine *length(path)* iz izhodiščnega AG, ki so v celoti vsebovane v poti *path*. Močna povezanost je vedno manjša ali enaka šibki povezanosti:

$$connection_{strong}(path) \leq connection_{weak}(path)$$

Če obstaja pot $path_{abs1} \in AAG_{abs1}$, potem za $abs_2 > abs_1$ obstaja pot $path_{abs2} \in AAG_{abs2}$, ki je sestavljena iz vozlišč oz. združenih vozlišč, ki so bila vsebovana v $path_{abs1}$. Za poti $path_{abs1}$ in $path_{abs2}$ veljajo naslednje neenakosti:

$$connection_{weak}(path_{abs1}) \leq connection_{weak}(path_{abs2})$$

$$connection_{strong}(path_{abs1}) \leq connection_{strong}(path_{abs2})$$

$$length((path_{abs1}) \geq length(path_{abs2})$$

Za razlago pojmov šibke in močne povezanosti si poglejmo že omenjeni primer na strani 52. Na sliki 22a je poenostavljen primer izhodiščnega AG iz katerega smo zgradili dva abstraktna akcijska grafa za različnima parametroma abstraktnosti: AAG_{abs1}, predstavljen na sliki 22b, in AAG_{abs2}, ki je predstavljen na sliki 22c. AAG_{abs1} vsebuje 6 poti dolžine, ki imajo šibko povezanost vsaj 2: B→E→I, E→I→J, B→E→I→J, C→D→H, F→I→J in F→I→L. Vendar pa imajo samo poti B→E→I, E→I→J, B→E→I→J, C→D→H in F→I→L tudi močno povezanost enako 2. AAG_{abs2} vsebuje 16 poti, ki imajo šibko povezanost vsaj 2 in so predstavljene v preglednici 13, skupaj z njihovimi dolžinami ter vrednostmi za šibko in močno povezanost. Na primeru vidimo, da poti vsebovane v AAG_{abs1} obstajajo tudi v AAG_{abs2} z ustrezno večjo šibko/močno povezanostjo. Prav tako vidimo, da ima pot B→DE→GHI→L šibko povezanost enako 3 in močno povezanost enako 0, kar pomeni, da zaporedje akcij, ki jih določa, nikoli ni bilo v celoti odigrano. Pot B→DE→GHI→J ima pri enako visoki šibki povezanost močno povezanost enako 2, torej je bilo zaporedje akcij v celoti odigrano natanko dvakrat.

Pot *path*	Dolžina poti *length(path)*	Šibka povezanost $connection_{weak}(path)$	Močna povezanost $connection_{strong}(path)$
B→DE	1	3	3
B→DE→GHI	2	3	3
B→DE→GHI→J	3	3	2
B→DE→GHI→L	3	3	0
C→DE	1	2	2
C→DE→GHI	2	2	2
C→DE→GHI→J	3	2	1
C→DE→GHI→L	3	2	1
DE→GHI	1	5	5
DE→GHI→J	2	3	3
DE→GHI→L	2	3	1
F→GHI	1	2	2
F→GHI→J	2	2	0
F→GHI→L	2	2	2
GHI→J	1	3	3
GHI→L	1	3	3

Preglednica 13. Poti iz AAG_{abs2}, predstavljenega na sliki 22c, ki imajo šibko povezanost ≥ 2. Sive vrstice predstavljajo poti, ki so bile vsebovane tudi v AAG_{abs1}.

Definicija 38. ***Makroakcija*** je neko zaporedje akcij, ki je vsebovano v večagentni strategiji.

Ker predstavlja pot v AAG neko zaporedje akcij, vozlišča pa z metriko podobnosti pogojene podobne situacije, sklepamo, da pot z dovolj visoko močno povezanostjo predstavlja koncept makroakcije. Pri tem se zastavi vprašanje, kakšna mora biti v resnici močna povezanost poti, da zaupamo ugotovitvi, da pot predstavlja del večagentne strategije. Odgovor na to vprašanje je domensko pogojen, saj nanj vplivajo naslednji dejavniki:

- Kompleksnost domenskega prostora; bolj kompleksen kot je domenski prostor, večjo nepredvidljivost in raznolikost agentnega delovanja lahko pričakujemo. Posledično lahko pričakujemo manjše vrednosti močne povezanosti poti, kot v enostavnejšem domenskem prostoru.
- Število podobnih situacij; šele ponavljanje podobnih situacij omogoča večkratno izvajanje podobnih delov strategij, kar šele omogoča zaznavanje zaporedij akcij, ki so del večagentne strategije. Večje število ponovitev poleg tega omogoča tudi boljšo pokritost domenskega prostora s primeri, torej lahko pričakujemo višje vrednosti močne povezanosti poti, ki predstavljajo koncepte makroakcij.
- Dolžina poti; daljša ko je pot, manjšo močno povezanost pričakujemo. Dolge poti z visoko močno povezanostjo so zato dobri kandidati za predstavnike makroakcij.

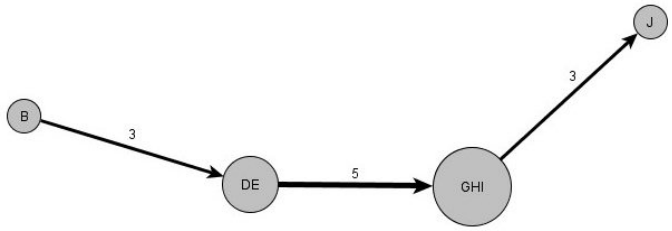
V splošnem velja, da je domenski prostor večagentnih sistemov izrazito kompleksen, zato je običajna zelo majhna pokritost prostora s primeri. Zato je število primerov, ki jih opisuje pot, tipično majhno v primerjavi s celotnim številom primerov.

Ker imajo krajše poti v povprečju višjo močno povezanost kot daljše poti, temelji naš pristop na izbiri poti, ki imajo pri fiksni dolžini poti najvišje vrednosti močne povezanosti. Tak pristop sicer ne zagotavlja, da izbrane poti res predstavljajo koncepte makroakcij, zagotavlja pa, da če AAG predstavlja koncepte makroakcij dane dolžine, jih s tem postopkom tudi izberemo. Opisani postopek je v obliki algoritma predstavljan na sliki 25.

```
Vhod:
- abstrakten akcijski graf: AAG
- ciljna dolžina makroakcije: len
- minimalna vrednost šibke povezanosti poti: minconn

Algoritem:
// množica strateških poti je na začetku prazna
strategicPaths = {}
for each (path ∈ AAG) {
  // če pot ustreza jo damo v množico strateških poti
  if ((length(path) == len) && (connection_strong(path) >= minconn)) {
    strategicPaths = strategicPaths ∪ path
  }
}

Izhod:
- strateški koncepti makroakcij dolžine len: strategicPaths
```

Slika 25. Algoritem, ki predstavlja postopek izbire strateških konceptov makroakcij.

Če bi v primeru AAG_{abs2}, predstavljenega na sliki 22c na strani 52, morali izbrati najbolj primerno pot, ki bi predstavljala koncept makroakcije s tremi akcijami, bi izbrali pot B→DE→GHI→J, predstavljeno na sliki 26. Ta pot ima med potmi dolžine 3 največjo vrednostjo močne povezanosti, poteg tega imata njeni podpoti B→DE→GHI in DE→GHI→J najvišjo vrednost močne povezanosti pri dolžini poti enaki 2, kar še dodatno podpira našo izbiro.



Slika 26. Pot, ki predstavlja najprimernejšega kandidata za predstavitev koncepta makroakcije s tremi akcijami. Številke na povezavah ustrezajo številu instanc akcij, ki jih predstavljajo povezave izhodiščnega AG.

6.2 Simbolni opis akcijskega modela (II. korak)

6.2.1 Določitev agentnih vlog in akcij (II.1)

AAG predstavlja večagentni model, kjer vozlišča in povezave predstavljajo različne akcijske koncepte. Brez poglobljene analize pa je njihov pomen človeku nerazumljiv. Eden izmed ciljev pri snovanju MASDA je bil dobiti večagentni model, ki omogoča človeško razumevanje. Zato vsakemu vozlišču in povezavi v AAG priredimo človeku razumljiv opis agentne vloge in agentne akcije, kar izboljša razumevanje opisanih akcijskih konceptov. Ker so taki opisi akcij in agentnih vlog zajeti v hierarhijah agentnih vlog H_{role} in akcij H_{action}, vsakemu vozlišču in povezavi v AAG določimo najmanj splošen opis agentne akcije in vloge, ki še odgovarja vsem primerom v vozlišču oz. povezavi.

Definicija 39. Vozlišče $v \in V$ iz AAG(V, E) predstavlja koncept začetka oz. konca agentne akcije $action_v$:

$$action_v = c \in H_{action}:((\forall a,b \in v.actions: c = common\_ancestor_{Haction}(a, b))$$

Definicija 40. Vozlišče $v \in V$ iz AAG(V, E) predstavlja koncept začetka oz. konca agentne akcije, ki jo je opravil agent z vlogo $role_v$:

$$role_v = c \in H_{role}:((\forall a,b \in v.roles: c = common\_ancestor_{Hrole}(a, b))$$

Definicija 41. Povezava $e \in E$ iz AAG(V, E) predstavlja koncept agentne akcije $action_e$:

$$action_e = c \in H_{action}:((\forall a,b \in e.instances: c = common\_ancestor_{Haction}(a, b))$$

Definicija 42. Povezava $e \in E$ iz AAG(V, E) predstavlja koncept akcije, ki jo je opravil agent z agentno vlogo $role_e$:

$$role_e = c \in H_{role}:((\forall a,b \in e.roles: c = common\_ancestor_{Hrole}(a, b))$$

Ker lahko iz vozlišča $v \in V$ iz AAG(V, E) izhaja več povezav, so opisi akcij in vlog v vozlišču v bolj splošni, kot v izhajajočih povezavah $e \in out(v)$. Zato velja sledeče:

$$\forall v \in V: \forall e \in out(v):$$
$$(action_v = common\_ancestor_{Haction}(action_v, action_e) \wedge$$
$$role_v = common\_ancestor_{Hrole}(role_v, role_e))$$

Opis povezav z uporabo človeško razumljivih opisov agentnih vlog in akcij omogoča, da podrobneje opišemo tudi celoten koncept makroakcije. Koncept makroakcije, ki ga predstavlja pot $path \in AAG$, tako opišemo z zaporedjem parov *vloga:akcija*:

$$role_{e1}:action_{e1} \rightarrow role_{e2}:action_{e2} \rightarrow ... \rightarrow role_{eK}:action_{eK}$$
$$\forall e_i \in path.E, K = length(path), role_{ei} \in H_{roles}, action_{ei} \in H_{actions}$$

```
Vhod:
- strateški koncepti makroakcij: strategicPaths

Algoritem:
// opišemo vsako strateško pot
for each (path ∈ strategicPaths ) {
  // vsako vozlišče v poti
  for each (v ∈ path.V) {
    v.action = common_parent(v.actions) // skupen opis akcije
    v.role = common_parent(v.roles) // skupen opis agentne vloge
  }
  // vsaka povezava v poti
  for each (e ∈ path.E) {
    // skupen opis akcije
    e.action = common_parent(e.instances.action)
    // skupen opis agentne vloge
    e.role = common_parent(e.instances.role)
  }
}

Izhod:
- karakterni opisi strateških konceptov makroakcij: strategicPaths
```

Slika 27. Algoritem, ki generira karakterne opise makroakcij.

Opis zaporedja vlog in akcij omogoča boljše človeško razumevanje zaporedja akcijskih konceptov, ki ga predstavlja koncept makroakcije, kot sama podatkovna struktura *path* $\in$ AAG_{abs}. Če združimo opise agentnih vlog in akcij, ter domenske pozicije agentov, ki jih vsebuje *path* $\in$ AAG_{abs}, potem dobimo ***karakterni opis makroakcije***, ki nam omogoča človeško razumevanje, kot tudi avtomatsko uporabo pri zaznavanju makroakcij. Karakterni opis makroakcije predstavlja zaporedje opisov akcijskih konceptov, ki ga predstavlja dana pot. Vsak akcijski koncept opišemo z: vlogo agenta, akcijo in domensko pozicijo agenta. Karakterni opis vključuje tudi opis koncepta zaključka zadnje akcije, kar je zaželeno predvsem zaradi človeške analize. Razumevanje stanja ob zaključku makroakcije je včasih koristno za razumevanje same makroakcije. Postopek, ki generira karakterne opise makroakcij je prikazan na sliki 27. Karakterni opis makroakcije, ki jo predstavlja *path* $\in$ AAG_{abs}, kjer je e_i $\in AAG_{abs}.E$, K = length(path), je predstavljen v preglednici 14.

Vloga agenta	$role_{e1}$	$role_{e2}$	...	$role_{eK}$	$role_{eK}$
Akcijski koncept	začetek akcije $action_{e1}$	začetek akcije $action_{e2}$	...	začetek akcije $action_{eK}$	konec akcije $action_{eK}$
Domenska pozicija agenta	*source*(e_1).pos	*source*(e_2).pos	...	*source*(e_K).pos	*destination*(e_K).pos

Preglednica 14. Karakterni opis makroakcije, ki jo predstavlja pot *path* $\in$ AAG_{abs}, kjer si posamezni akcijski koncepti sledijo od leve proti desni.

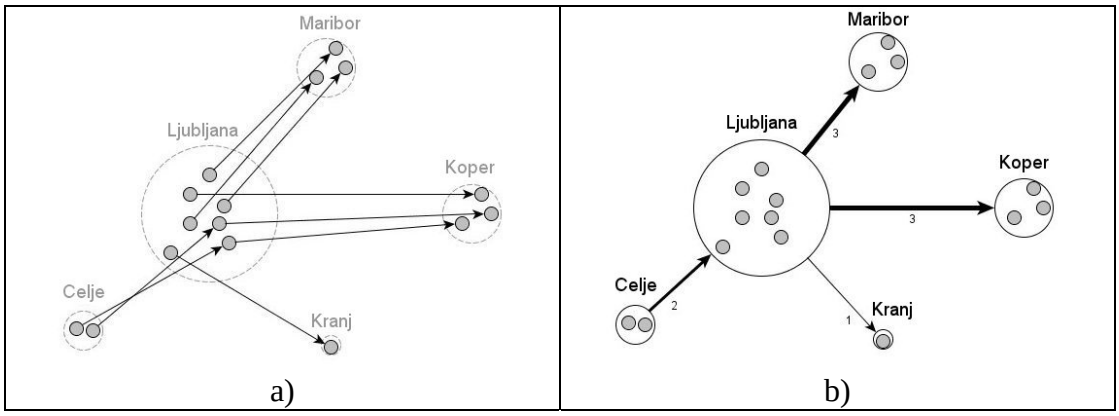
6.2.2 Določitev učnega problema (II.2)

AAG je večagentni model, ki predstavlja akcijsko, časovno in tudi prostorsko komponento večagentnega delovanja, vendar pa ne omogoča poglobljenega razumevanja večagentnega delovanja. Vozlišča in povezave v AAG sicer predstavljajo različne koncepte akcij, vendar pa zgolj poznavanje AAG in vsebovanih primerov ne omogoča razumevanja predstavljenih akcijskih konceptov. Za njihovo razumevanje je potrebno poiskati domenske opise predstavljenih akcijskih konceptov, pri čemer je zaželeno, da opisi obenem omogočajo tudi prepoznavanje in razlikovanje akcijskih konceptov.

Nalogo opisa akcijskih konceptov rešuje MASDA s strojnim učenjem, zato domenski opisi predstavljajo naučena pravila. Vsak akcijski koncept je opisan z enim pravilom, ki je, zaradi lažjega človeškega razumevanja, sestavljen iz konjunktivnih pogojev, ki jih sestavljajo domenske značilke.

Akcijski koncepti sicer omogočajo učenje opisa celotnega poteka akcije, vendar pa bi bila pravila, ki bi opisovala celoten potek akcije, kompleksna in za človeka težje razumljiva, kot pravila, ki bi opisovala zgolj začetek iste akcije. Poleg tega bi bila zaradi predstavitve daljših časovnih intervalov, torej kompleksnejših sprememb stanj, taka pravila pri isti velikosti bolj splošna kot tista, ki predstavljajo zgolj začetek akcije. Poleg tega želimo pri problemu zaznave akcij prepoznati akcijo takoj, ko jo začne agent izvajati, zato bomo s pravili opisovali le koncepte začetkov in zaključkov akcij.

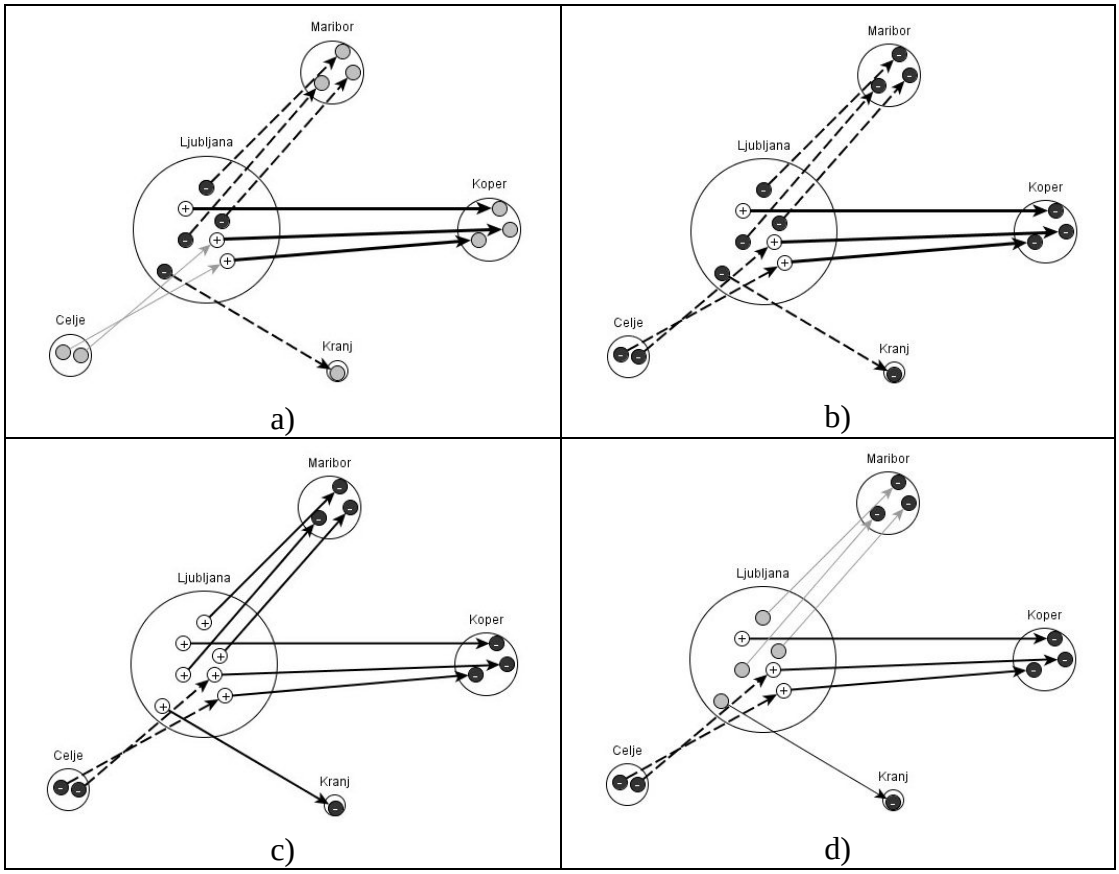
Ker vozlišča in povezave v AAG predstavljajo različne tipe akcijskih konceptov, bomo predstavili učenje obeh. Postopek učenja bomo prikazali na že predstavljenem primeru domene vožnje z avtomobili. Izhodiščni AG je zato ponovno predstavljen na sliki 28a, ustrezni AAG_{abs}, ki bo uporabljen za primer učenja akcijskih konceptov, pa na sliki 28b.



Slika 28. Shematski primer a) izhodiščnega AG in b) AAG_{abs}.

Ker predstavljajo povezave celotni akcijski koncept, vozlišča pa zgolj koncept začetka oz. zaključka akcije, bomo zaradi primerljivost v obeh primerih prikazali učenje koncepta začetka akcije. Torej se bomo tudi v primeru povezave učili zgolj koncepta začetka akcije. Za učna primera smo izbrali koncept začetka akcije, ki ga predstavlja vozlišče *Ljubljana*, in koncept začetka akcije, ki ga predstavlja povezava

Ljubljana→Koper. Učimo se torej dveh različnih konceptov: *začetek vožnje iz Ljubljane* in *začetek vožnje iz Ljubljane v Koper*.



Slika 29. Različni načini izbire pozitivnih in negativnih učnih primerov. Majhni krogi predstavljajo učne primere (+ pozitivne in - negativne). Debelejše polne povezave predstavljajo pozitivni akcijski koncept, debelejše črtkane negativni akcijski koncept, sivih pa v procesu izbire učnih primerov ne upoštevamo.

Na sliki 29 so predstavljeni štiri različni načini izbire pozitivnih in negativnih učnih primerov, ki pojasnjujejo pomenske razlike pri učenju akcijskih konceptov, ki jih predstavljajo vozlišča oz. povezave v AAG_{abs}:

- Za pozitivne učne primere vzamemo le primere, ki ustrezajo ciljni povezavi, za negativne pa vse tiste, ki ustrezajo ostalim izhodnim povezavam istega vozlišča, kot je to predstavljeno na sliki 29a. Učimo se torej koncepta: *začetek vožnje iz Ljubljane v Koper*, kjer učni primeri opisujejo razliko med začetkom vožnje iz Ljubljane v Koper in začetki vožnje iz Ljubljane v ostala mesta.
- Za pozitivne učne primere vzamemo le primere, ki ustrezajo ciljni povezavi, za negativne pa vse ostale, kot je to predstavljeno na sliki 29b. Učimo se torej koncepta: *začetek vožnje iz Ljubljane v Koper*, kjer primeri opisujejo razliko med začetkov vožnje iz Ljubljane v Koper in vsemi ostalimi začetki in zaključki voženj.
- Za pozitivne učne primere vzamemo vse primere, ki jih predstavlja dano vozlišče, za negativne pa vse ostale, kot je to predstavljeno na sliki 29c. Učimo se torej koncepta: *začetek vožnje iz Ljubljane*, kjer učni primeri opisujejo razliko med začetkom vožnje iz Ljubljane in vsemi ostalimi začetki in zaključki voženj.

- Za pozitivne učne primere vzamemo le primere, ki ustrezajo ciljni povezavi, za negativne pa vse ostale, ki niso vsebovani v istem vozlišču, kot je to predstavljeno na sliki 29d. Učimo se torej koncepta: *začetek vožnje iz Ljubljane v Koper*, kjer primeri opisujejo razlike med konceptom začetka vožnje iz Ljubljane v Koper in ostalimi začetki in zaključki voženj, ki niso potekale iz Ljubljane.

Ker je namen MASDA odkriti in opisati koncepte makroakcij, predstavimo tudi učenje koncepta makroakcije, torej učenja opisov akcijskih konceptov, ki jih predstavljajo povezave v poti *path*.

Ker vsebuje pot *path* $\in$ AAG več povezav, razdelimo proces učenja na učenje opisa vsake povezave posebej. Za vsako povezavo $e \in path.E$ definiramo učni problem kot dvorazredni učni problem, kjer:

- pozitivni razred predstavlja akcijski koncept, ki ga predstavlja povezava *e*,
- negativni razred pa akcijske koncepte, ki jih predstavljajo ostale povezave $\forall e_{neg} \in AAG.E; e_{neg} \neq e$.

Pozitivne in negativne učne primere nato izberemo na način, ki je prikazan na sliki 29b. Postopek nato ponovimo za vse povezave v poti. Pri tem velja omeniti, da pojem šibke povezanosti poti določa minimalno število pozitivnih učnih primerov, ki jih vsebuje vsaka povezava v poti. Torej imamo za vsak učni primer vsaj toliko pozitivnih primerov, kot je vrednost šibke povezanosti za dano pot.

V večjih AAG je število učnih primerov, ki predstavljajo negativni razred, običajno veliko večje od števila pozitivnih, zato je porazdelitev učnih primerov izrazito neuravnotežena. Ker v takem primeru običajno delujejo algoritmi za indukcijo pravil slabše [26, 37], iz negativnih primerov izberemo primerno podmnožico negativnih primerov. Ker izbira učnih primerov pomembno vpliva na končna pravila, so tehnike vzorčenja (angl. *sampling*) učnih primerov raznovrstne [28]. V disertaciji predlagamo vzorčenje, ki temelji na oddaljenosti negativnih učnih primerov od pozitivnih:

- Izberemo le tiste, ki predstavljajo ***bližnje primere*** (angl. *near misses*). Tako ohranimo le primere, ki so od vozlišča, ki predstavlja pozitivni razred, oddaljeni manj od podane meje okolice, torej primere, ki so si podobni med seboj. Posledično pravila ločijo podobna razreda med seboj, torej predstavljajo lokalne razlike med podobnimi akcijskimi koncepti.
- Izberemo le najbolj ***oddaljene primere*** (angl. *far misses*). Izbrani primeri so od vozlišča, ki predstavlja pozitivni razred, oddaljeni več od podane meje, torej predstavljajo različne akcijske koncepte. Posledično pravila ločijo zelo različna razreda med seboj, zato predstavljajo globalen opis ciljnega akcijskega koncepta.

Za merjenje oddaljenosti učnih primerov uporabimo funkcijo *dist(a, b)*, ki smo jo uporabili v postopku abstrakcije. Metodi lahko združimo tako, da določimo najmanjšo in največjo sprejemljivo mejo oddaljenosti negativnih učnih primerov. Z spreminjanjem teh mej lahko generiramo učne primere, ki predstavljajo poljuben razpon negativnih akcijskih konceptov. Opisani postopek izbire učnih primerov je predstavljen na sliki 30.

```
Vhod:
- abstrakten akcijski graf: AAG
- pot v abstraktnem akcijskem grafu: path
- najmanjša meja oddaljenosti negativnih učnih primerov: minDist
- največja meja oddaljenosti negativnih učnih primerov: maxDist
- povezava iz referenčne poti: e

Algoritem:

e.examples = {}
for each (inst ∈ e.instances) {
  // pozitivni učni primer
  example.class = classPlus
  example.agent = inst.agent
  example.role = inst.role
  example.action = inst.action
  example.start = inst.start
  e.examples = e.examples ∪ example
}
for each (other ∈ AAG.E) {
  if (other == e) {
    continue  // ignoriramo povezavo iz poti
  }
  if (minDist <= dist(out(e), out(other)) <= maxDist) {
    continue  // ignoriramo primere, ki so preblizu oz. predaleč
  }
  for each (inst ∈ other.instances) {
    // negativni učni primer
    example.class = classMinus
    example.agent = inst.agent
    example.role = inst.role
    example.action = inst.action
    example.start = inst.start
    e.examples = e.examples ∪ example
  }
}

Izhod:
- povezava z generiranimi učnimi primeri: e.examples
```

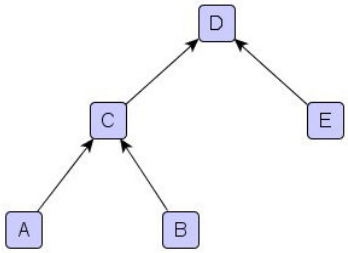
Slika 30. Algoritem, ki za dano povezavo v referenčni poti izbere in označi učne primere v poti, ki predstavlja makroakcijo.

6.2.3 Indukcija pravil (II.3)

Zadnji korak v MASDA je indukcija pravil, ki opisujejo ciljni koncept makroakcije. V prejšnjem poglavju smo predstavili, da za vsak akcijski koncept definiramo nov dvorazredni učni problem, kjer pozitivni učni razred predstavlja ciljni akcijski koncept. Ko določimo pozitivne in negativne učne primere, moramo za vhod v algoritem za indukcijo pravil generirati še ustrezne učne podatke. Le ti v našem primeru predstavljajo značilke, podane v hierarhijah značilk. Učne podatke generiramo tako, da za vsak učni primer generiramo vse značilke, ki so resnične za dani učni primer. Ker želimo agente opisovati z njihovimi vlogami, je naš cilj, da značilke, ki opisujejo agentne lastnosti, zapišemo v obliki ***vloga:značilka***. Vendar pa ima taka predstavitev dve pomanjkljivosti:

- v primeru, ko agenti dinamično spreminjajo vloge, pari *vloga:značilka* niso določeni za vse časovne točke, in
- v primeru, ko ima več agentov isto vlogo, obstaja več vrednosti za par *vloga:značilka*, ki si lahko nasprotujejo.

Za ponazoritev problema vzemimo sledeči primer večagentnega sistema z dvema agentoma (agentX in agentY), kjer zaradi jasnosti uporabimo le dve značilki, ki opisujeta lastnost okolja (zo_1 in zo_2), in dve značilki, ki opisujeta agentni lastnosti (za_1 in za_2). Primer vsebuje 4 pozitivne učne primere, ki opisujejo akcijski koncept (razred AKC), in 2 negativna učna primera (razred NEG). Uporabljena hierarhija agentnih vlog je predstavljena na sliki 31. Interna predstavitev podatkov je prikazana v preglednici 15, vendar taka predstavitev za nas ni primerna iz dveh razlogov: ne opisuje s pari *vloga:značilka* in v našem primeru ne omogoča gradnje popolnoma točnega klasifikatorja, ki bi opisoval razred AKC zgolj s konjunkcijo pogojev. V preglednici 16 je predstavljena odgovarjajoča predstavitev s pari *vloga:značilka*.



Slika 31. Hierarhija agentnih vlog za opisani primer.

		okolje		agentX			agentY		
čas	razred	zo_1	zo_2	vloga	za_1	za_2	vloga	za_1	za_2
1	AKC	T	T	A	T	T	E	F	T
2	AKC	F	T	A	T	F	E	T	T
3	AKC	T	T	E	F	T	B	T	T
4	AKC	F	T	E	F	T	B	T	F
5	NEG	T	F	E	F	F	E	T	F
6	NEG	F	T	A	T	F	B	T	T

Preglednica 15. Interna predstavitev podatkov.

		okolje		vloga A		vloga B		vloga E	
čas	razred	zo_1	zo_2	A:za_1	A:za_2	B:za_1	B:za_2	E:za_1	E:za_2
1	AKC	T	T	T	T	/	/	F	T
2	AKC	F	T	T	F	/	/	T	T
3	AKC	T	T	/	/	T	T	F	T
4	AKC	F	T	/	/	T	F	F	T
5	NEG	T	F	/	/	/	/	T, F	F, F
6	NEG	F	T	T	F	T	T	/	/

Preglednica 16. Predstavitev podatkov s pari vloga:značilka namesto z imeni agentov. Sive celice prikazujejo problematične celice (brez ali več vrednosti).

Ker opisuje MASDA stanja s hierarhijami značilk, je njihovo število linearno odvisno od števila agentov v sistemu. Preveliko število značilk pa običajno slabo vpliva na algoritme za indukcijo pravil [17]. Zaradi velikega števila značilk in zaradi slabe predstavitve v primeru dinamične menjave agentnih vlog, uporablja MASDA transformacijo, ki pretvori značilke v manjše število atributov in obenem odpravi problem predstavitve spremenljivih agentnih vlog.

Transformacija poteka sledeče. Vsak agent je opisan z enakim naborom hierarhij značilk za agentne lastnosti $H_{agentna\text{-}lastnost1}$, ..., $H_{agentna\text{-}lastnostN}$, zato lahko opis združimo tako, da vpeljemo pristop z uporabo atributov, katerih vrednosti so množice (angl. *set-valued attributes*). Za vsako značilko iz hierarhij agentnih lastnosti vpeljemo nadomestni atribut, katerega vrednost je množica agentnih vlog za katere je par *vloga:značilka* resničen. Taka preslikava zmanjša število značilk, ki opisujejo agentne lastnosti, iz možnih n*N na N atributov. Zaradi konsistentnosti pretvorimo tudi vse značilke, ki opisujejo stanje okolja v predstavitev z atributi. Resnična značilka tako predstavlja atribut, ki ima za vrednost množico z enim elementom {okolje}, neresnična pa prazno množico {}. Transformacija torej pretvori opis z n*N+M značilkami v opis z N+M atributi, kjer je n število agentov, N število značilk, ki opisujejo agentne lastnosti, in M število značilk, ki opisujejo lastnosti okolja. Primer pretvorjenih podatkov iz preglednice 16 je predstavljen v preglednici 17.

čas	razred	atribut-zo_1	atribut-zo_2	atribut-za_1	atribut-za_2
1	AKC	{okolje}	{okolje}	{A}	{A, E}
2	AKC	{}	{okolje}	{A, E}	{E}
3	AKC	{okolje}	{okolje}	{B}	{B, E}
4	AKC	{}	{okolje}	{B}	{E}
5	NEG	{okolje}	{}	{E}	{}
6	NEG	{}	{okolje}	{A, B}	{B}

Preglednica 17. Predstavitev podatkov z uporabo atributov, katerih vrednosti so množice.

Prednost take predstavitve podatkov je dvojna, saj učinkovito zmanjšamo število značilk, obenem pa zagotovimo, da ni manjkajočih oz. nasprotujočih vrednosti značilk. Zaradi predstavitve z atributi, katerih vrednosti so množice, bodo pogoji v pravilih oblike ***okolje ∈ atribut-zo*** oz. ***vloga ∈ atribut-za***. Take pogoje skrajšano napišemo ***okolje:zo*** oz. ***vloga:za***, kar pomeni, da je značilka okolja ***zo*** resnična oz. da je za agenta z vlogo ***vloga*** resnična značilka ***za***. Če pogledamo primer podatkov iz preglednice 17, bi se pravilo za razred AKC glasilo:

$$IF\ (okolje \in atribut\text{-}zo_2 \wedge E \in atribut\text{-}za_2) \Rightarrow AKC$$

oz. skrajšano

$$IF\ (okolje:zo_2 \wedge E:za_2) \Rightarrow AKC.$$

Slednje pravilo lahko opišemo sledeče: če je značilka okolja zo_2 resnična in je za agenta, ki opravlja vlogo E, resnična tudi značilka za_2, potem sledi akcija, ki jo predstavlja razred AKC. Zaradi berljivosti bomo odslej vsa pravila pisali v skrajšani obliki.

Zaradi hierarhično urejenih konceptov agentih vlog lahko predstavitev še dodatno izboljšamo. Za vsako vlogo, ki je vsebovana v množici, generiramo vloge, ki predstavljajo njene prednike. Pri tem običajno izpustimo koren hierarhije, ker predstavlja najbolj splošno vlogo, ki opisuje vse agente.

čas	razred	atribut-zo_1	atribut-zo_2	atribut-za_1	atribut-za_2
1	AKC	{okolje}	{okolje}	{A, C}	{A, E, C}
2	AKC	{}	{okolje}	{A, E, C}	{E}
3	AKC	{okolje}	{okolje}	{B, C}	{B, E, C}
4	AKC	{}	{okolje}	{B, C}	{E}
5	NEG	{okolje}	{}	{E}	{}
6	NEG	{}	{okolje}	{A, B, C}	{B, C}

Preglednica 18. Predstavitev podatkov z uporabo atributov, katerih vrednosti so množice, skupaj s skupnimi predniki agentnih vlog (agentna vloga C).

Rezultat postopka na podatkih našega primera je prikazan v preglednici 18. Taka predstavitev omogoča algoritmu za indukcijo pravil, da lahko iskani akcijski koncept opiše tudi z vlogami, ki niso bile neposredno vsebovane v učnih podatkih. Tako bi se pravilo za primer podatkov iz preglednice 18:

$$IF\ (okolje{:}zo_2 \wedge C{:}za_1 \wedge E{:}za_2) \Rightarrow AKC.$$

Slednje pravilo je bolj specifično od prejšnjega, ker vsebuje tri pogoje namesto dveh, obenem pa vsebuje opis agentne vloge, ki je bolj splošna od opisov vlog v osnovnih učnih podatkih (vloga C, namesto vlog A in B).

Opisani postopek transformacije podatkov je predstavljen na sliki 32. Velja opozoriti, da MASDA ne določa uporabe specifičnega algoritma za indukcijo pravil. Edina omejitev pri izbiri algoritma je, da za predstavitev učnih podatkov omogoča uporabo atributov, katerih vrednosti so množice. Primer takega algoritma je SLIPPER [31].

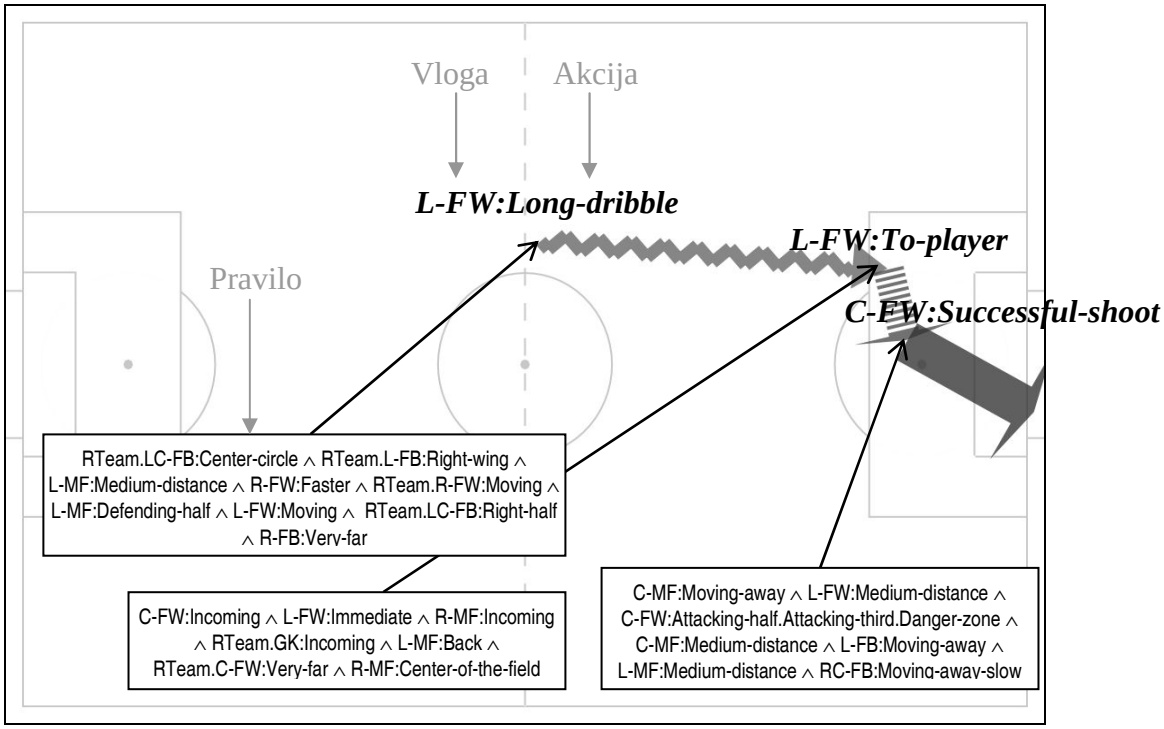
```
Vhod:
- pot z generiranimi učnimi primeri: path

Algoritem:
for each (e ∈ path.E) {
  // generiranje učnih podatkov na osnovi učnih primerov
  bigData = generateLearnData(e.examples)
  // transformacija v zapis z atributi
  compactData = TransformData(bigData)
  // indukcija pravil
  e.rules = induceRules(compactData)
}

Izhod:
- pot skupaj s pripadajočimi pravili: path
```

Slika 32. Algoritem, ki generira pravila, ki opisujejo koncepte makroakcij.

Končni rezultat koraka indukcije pravil so pravila, ki opisujejo celotni koncept makroakcije. Pravila, ki predstavljajo koncept makroakcije, določen s potjo *path* $\in AAG_{abs}$, opisno predstavimo v obliki zaporedja opisov akcijskih konceptov, ki jih sestavlja opis vloge, opis akcije in ustrezno pravilo. Zaporedje, kjer e_i ustreza i-ti povezavi v poti *path*, je sledeče:

$$\begin{array}{c} \text{vloga}_{e1}\text{:akcija}_{e1} \Leftarrow \text{pravilo}_{e1} \\ \downarrow \\ \text{vloga}_{e2}\text{:akcija}_{e2} \Leftarrow \text{pravilo}_{e2} \\ \downarrow \\ \ldots \\ \downarrow \\ \text{vloga}_{eK}\text{:akcija}_{eK} \Leftarrow \text{pravilo}_{eK} \end{array}$$

Končni rezultat MASDA so torej grafično in simbolno opisani strateški koncepti makroakcij, ki omogočajo avtomatsko uporabo ter človeško razumevanje. Grafični del opisa temelji na izrisu ustrezne poti v AAG, simbolni pa predstavlja karakterni opis makroakcije in pripadajoča pravila, ki skupaj opišejo zaporedje akcijskih konceptov. Primer takega opisa makroakcije, ki jo predstavlja uspešen napad na gol v domeni RoboCup, je predstavljen na sliki 33. Primer opisuje prodor (akcija: *Long-dribble*) levega napadalca (vloga: *L-FW*) po levem delu nogometnega igrišča do kazenskega prostora, kjer poda (akcija: *To-player*) srednjemu napadalcu (vloga: *C-FW*), ki uspešno strelja na gol (akcija: *Successful-shot*).



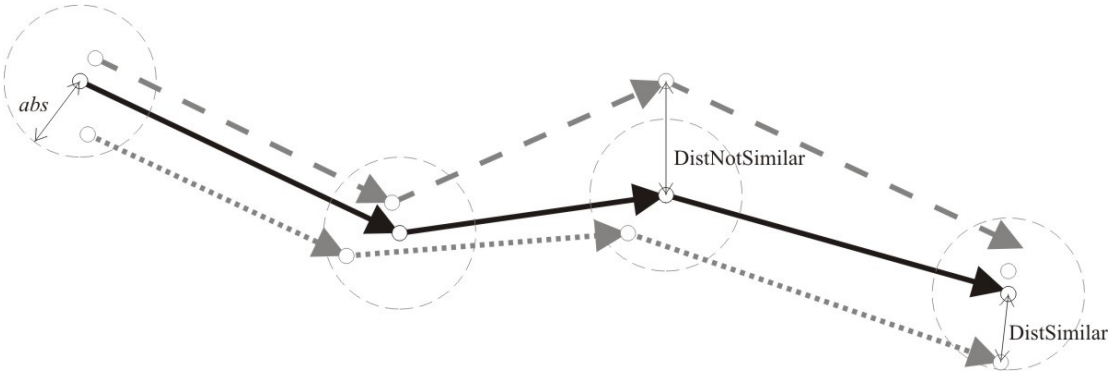
Slika 33. Primer grafičnega in simbolnega opisa koncepta makroakcije v domeni RoboCup, ki predstavlja uspešen napad na gol.

6.3 Uporaba MASDA modelov

Rezultat MASDA so opisi konceptov makroakcij, ki predstavljajo del strategije večagentnega sistema. Ker so sestavljeni iz različnih delov in torej opisujejo različne vidike večagentnega sistema, jih lahko uporabimo na različne načine:

- Grafičen del opisa (korak I.2, stran 48) uporabimo za vizualno predstavitev obnašanja, kar olajša prostorsko analizo večagentnega delovanja.
- Karakterni opis (korak II.1, stran 60) uporabimo za prepoznavanje makroakcij, ki imajo podobno zaporedje akcijskih konceptov.
- Pravila (korak II.3, stran 65), ki opisujejo posamezne akcijske koncepte, uporabimo za ugotavljanje ali lastnosti okolja ustrezajo pogojem za dane akcijske koncepte.
- Skupni opis konceptov makroakcij uporabimo za napovedovanje sledečih akcij, kjer pravila omogočajo zaznavo trenutne akcije, zaporedje akcijskih konceptov, ki ga predstavlja makroakcija, pa odgovarja sledečemu zaporedju akcij.

Postopek, ki z uporabo karakternega opisa makroakcije omogoča iskanje makroakcij s podobnim zaporedjem akcijskih konceptov, je sledeč. Za dani karakterni opis makroakcije iz AAG_{abs} zgradimo referenčno pot, kjer povezave in vozlišča v poti vsebujejo le ustrezne opise agentnih vlog, akcij in domenskih pozicij agentov. Nato vse ostale makroakcije pretvorimo v poti v ustreznem AAG_0. Za dano referenčno pot nato izberemo vse poti v AAG_0 iste dolžine ter paroma primerjamo razdalje med vozlišči referenčne in izbranih poti. Za podobne vzamemo poti, ki imajo največjo razdaljo med vozlišči manjšo ali enako izbrani vrednosti največje sprejemljive razdalje, ki ustreza parametru abstrakcije *abs*. Grafična ponazoritev primerjave je predstavljena na sliki 34, kjer črna pot ustreza referenčni poti, ki smo jo zgradili iz referenčnega karakternega opisa makroakcije. Črtkani krogi predstavljajo največjo razdaljo med vozlišči, ki jo določa parameter abstrakcije *abs*. Zgornja siva črtkana pot ustreza makroakciji, ki ni podobna referenčni, ker je največja oddaljenost med vozlišči *DistNotSimilar* > *abs*. Spodnja siva drobno črtkana pot ustreza makroakciji, ki je podobna referenčni, ker je največja oddaljenost med vozlišči *DistSimilar* ≤ *abs*.



Slika 34. Primer primerjave poti, ki ustrezajo različnim makroakcijam. Črna pot predstavlja referenčno, sivo črtkana predstavlja pot, ki ni podobna, sivo drobno črtkana pa pot, ki je podobna referenčni za dano največjo razdaljo *abs*.

Pravila uporabimo, kadar želimo ugotoviti, ali določena situacija predstavlja začetek neke akcije. Postopek je sledeč. Situacijo najprej opišemo z domenskimi značilkami. Nato pregledamo pravila za vse koncepte makroakcij. Če pravilo določenega

akcijskega koncepta ustreza, potem lahko sklepamo, da bo v danem trenutku izvedena ustrezna akcija. Ker je akcijski koncept le del nekega koncepta makroakcije, lahko sklepamo tudi o nadaljnjem zaporedju akcij.

Postopek uporabe karakternih opisov in pravil lahko tudi združimo. Če neka situacija ustreza pravilom določenega akcijskega koncepta, dodatno preverimo, ali vloga agentov in njihova domenska pozicija ustrezajo karakternemu opisu. Tako pravila uporabimo kot neke vrste predizbor možnih akcijskih konceptov, ki jih nato primerjamo še z opisanim postopkom ugotavljanja podobnosti karakternih opisov makroakcij.

Celoten nabor opisov konceptov makroakcij uporabimo za razumevanje strateškega delovanja analiziranega večagentnega sistema. Z analizo pravil in zaporedij akcij lahko namreč ugotovimo, kakšne so zakonitosti delovanja agentov. To informacijo lahko s pridom uporabimo, na primer v kompetitivnih večagentnih domenah. S poznavanjem delovanja nasprotnikovih večagentnih moštev lahko namreč predvidimo ustrezne proti-strategije, kar izboljša konkurenčno prednost našega moštva.

7. Ovrednotenje pristopa na domeni RoboCup

Not all who wander are lost.

J. R. R. Tolkien

RoboCup je kompleksna, kompetitivna in dinamična večagentna domena. Poznavanje strategije nasprotnikovega moštva je zato zaželeno, saj nudi vpogled v zakonitosti nasprotnikovega obnašanja, kar omogoča pripravo ustreznih protistrategij.

Naš splošni cilj je bil potrditi, ali je MASDA učinkovit pristop za modeliranja večagentnih sistemov – torej ali res zazna in opiše strateške vidike večagentnega delovanja. Poleg tega smo želeli preveriti, če MASDA uspešno uporablja abstrakcijo pri opisu odkritih konceptov makroakcij ter ali so dobljeni opisi makroakcij primerni za človeško in strojno uporabo.

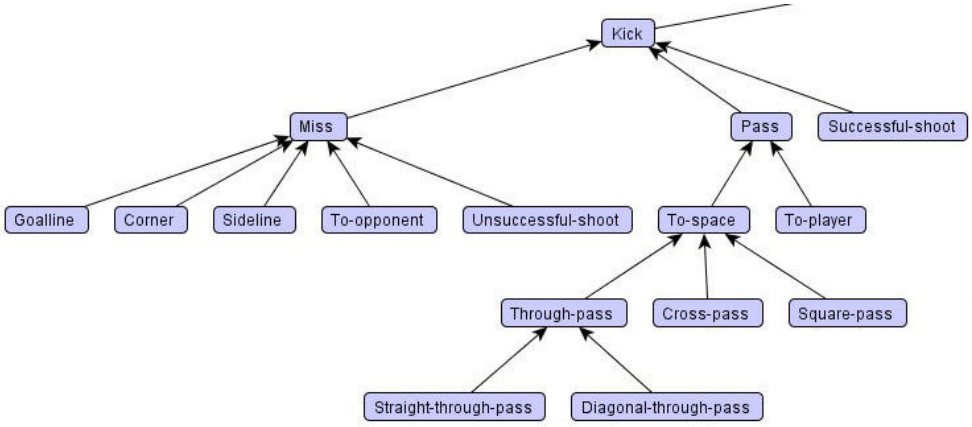
7.1 Opis MASDA modeliranja

V okolju RoboCup predstavlja sled izvajanja posnetek igre, ki opisuje spremembe okolja v teku igre. Ena igra traja 6000 časovnih korakov oz. toliko več, kot znaša izvajanje prekrškov in sodnikovih podaljškov. Vsak korak je opisan s 6 atributi, ki opisujejo stanje okolja (čas, stanje igre, x in y pozicija žoge ter dx in dy komponenta hitrosti žoge) in 21 atributi, ki opisujejo vsakega igralca (x in y pozicija igralca, dx in dy komponenta hitrosti, orientacija telesa, kot pogleda, širina in kvaliteta pogleda, nivo vzdržljivosti, količina vložene moči, moč regeneracije ter statistike, ki opisujejo število opravljenih osnovnih agentnih akcij). Eno igro torej opisuje 468 atributov in več kot 2.808.000 numeričnih vrednosti atributov.

Zaznavanje osnovnih agentnih akcij je v primeru posnetka RoboCup igre enostavno, saj iz sprememb vrednosti atributov, ki opisujejo število opravljenih osnovnih agentnih akcij, sklepamo na izvajane osnovne agentne akcije, ki so predstavljene v poglavju 2.2.1. Osnovne agentne akcije v domeni RoboCup imajo parametre, ki pa jih pri zaznavi ne ugotavljamo, zato jih obravnavamo kot osnovne agentne akcije brez parametrov.

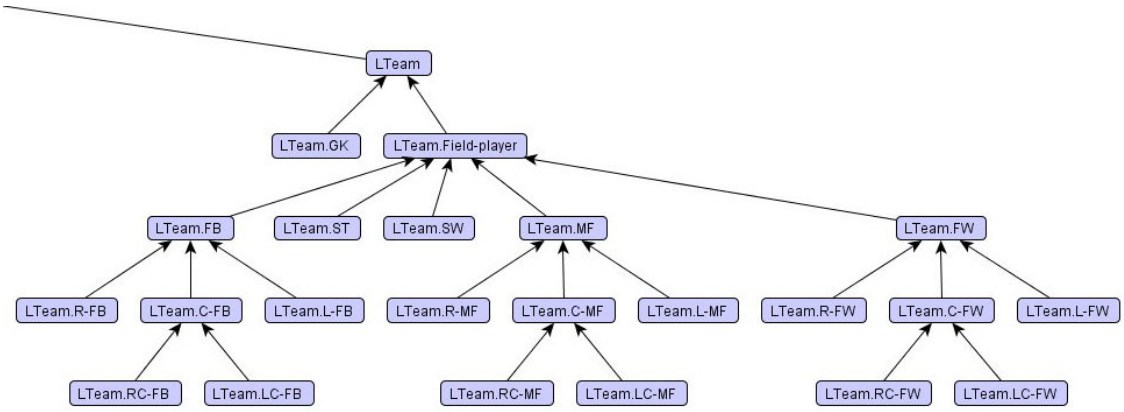
Okolje RoboCup izvaja simulacijo nogometne igre, zato agentne akcije ustrezajo računalniški verziji nogometnih akcij. Ustrezen nabor primernih brezparametričnih akcij smo pridobili iz [35, 52] in je v celoti predstavljen v preglednici 32 v prilogi A. Zaznavanje agentnih akcij smo izvedli z uporabo posebej zgrajenih programskih prepoznavalnikov. Za vsako agentno akcijo smo implementirali ustrezen programski prepoznavalnik, ki iz sprememb domenskega stanja in zaporedja osnovnih agentnih akcij ugotavlja, ali je bila odigrana dana agentna akcija.

Domensko nogometno znanje smo nato modelirali po posvetu z nogometnimi strokovnjaki [79, 84, 115]. Nabor akcij smo uredili v hierarhijo agentnih akcij, ki je predstavljena na sliki 77 v prilogi A. Del hierarhije akcij, ki prikazuje akcije povezane z brcanjem žoge, je prikazan na sliki 35.



Slika 35. Del hierarhije agentnih akcij v domeni RoboCup, ki prikazuje akcije brcanja žoge.

Agentne vloge smo prav tako modelirali po zgledu iz nogometa. Ker pozna sodoben nogomet množico različnih postavitev in posledično mnogo nogometnih vlog, smo opisali zgolj najpomembnejše vloge nogometnih igralcev. Popis vseh vlog je podan v preglednici 33 v prilogi A. Ker nogometne vloge sledijo iz uporabljene formacije, smo vloge določali na podlagi ugotovljenih formacij igralcev. Le-to smo ugotovili z računanjem povprečnih pozicij igralcev v daljšem časovnem obdobju in s primerjanjem povprečnih pozicij s postavitvami različnih formacij. Formacija, ki ima najmanjšo razliko pozicij igralcev, določi njihove vloge. Nabor agentnih vlog smo nato uredili v hierarhijo agentnih vlog, ki je predstavljena na sliki 79 v prilogi A. Hierarhija določa vloge za levo (oznaka LTeam) in desno moštvo (oznaka RTeam). Del hierarhije vlog za levo moštvo je predstavljen na sliki 36.

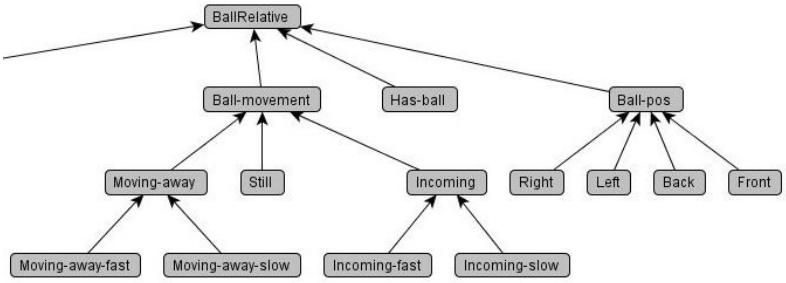


Slika 36. Del hierarhije agentnih vlog, ki prikazuje vloge levega moštva.

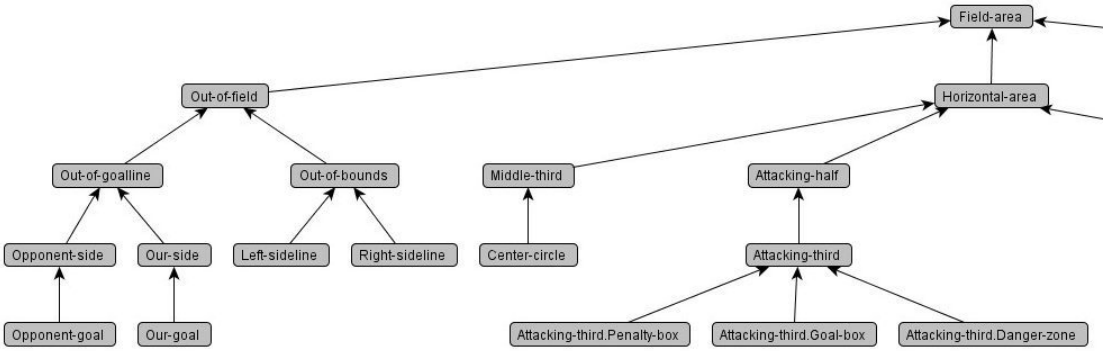
Domenski prostor smo opisali s hierarhijami značilk za tip igre ($H_{Teamplay}$) in nahajanje žoge v posebnih področjih na igrišču ($H_{BallFieldArea}$) ter z naslednjimi hierarhijami značilk za agentne lastnosti:

- posest žoge, relativna pozicija, oddaljenost in hitrost agenta glede na žogo ($H_{BallRelative}$),
- hitrost agenta (H_{Speed}) in
- nahajanje agenta v posebnih področjih na igrišču ($H_{FieldArea}$).

Vse navedene hierarhije značilk so po vrsti predstavljene na slikah 80, 81, 82 in 83 v prilogi A. Del hierarhije značilk $H_{BallRelative}$ je predstavljen na sliki 37, del hierarhije značilk $H_{FieldArea}$ pa na sliki 38. Za vsako značilko iz navedenih hierarhij smo sprogramirali ustrezen programski prepoznavalnik, ki za dano stanje določi, ali je značilka resnična ali ne.



Slika 37. Del hierarhije značilk $H_{BallRelative}$.



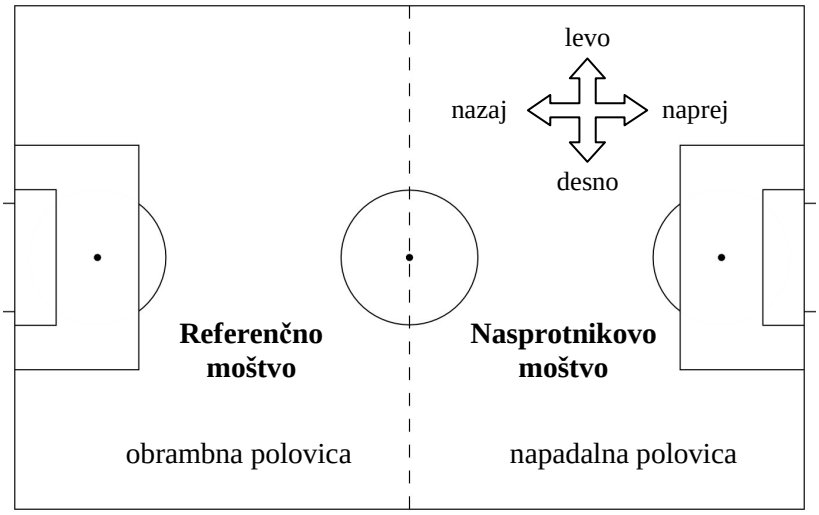
Slika 38. Del hierarhije značilk $H_{FieldArea}$.

Z izjemo hierarhije značilk za tip igre $H_{Teamplay}$, predstavljajo opisane hierarhije značilk nižjenivojski opis stanja na igrišču, saj ne opisujejo strateških lastnosti večagentnega obnašanja. Kljub možnosti višjenivojskega popisa se temu zavedno odrečemo iz naslednjih dveh razlogov:

- določitev primernega nabora strateških lastnosti je težavna naloga tudi za domenskega eksperta. Poleg tega so ekspertne definicije strateških lastnosti subjektivne in pogosto neprecizne, zato je implementacija ustreznih programskih prepoznavalnikov otežena;
- naša osnovna naloga je bila iz sledenja nižjenivojskega obnašanja agentov in zgolj osnovnega domenskega znanja ugotoviti, kakšno skupno strategijo izvajajo agenti. Uporaba značilk, ki opisujejo višjenivojske lastnosti domene, bi zato nalogo preveč poenostavila.

Uporaba kompleksnejšega predznanja, na primer značilk, ki opisujejo daljše ter časovno bolj prepletene strateške elemente igre, bi nedvomno omogočila opis modela, ki bi bil bližje opisu domenskega strokovnjaka in posledično tudi lažji za interpretacijo. Kljub temu pa je uporaba višjenivojskega predznanja nezaželena, saj je naš glavni namen odkriti in opisati višjenivojsko večagentno obnašanje, ki torej ne sme biti že opisano v obliki predznanja.

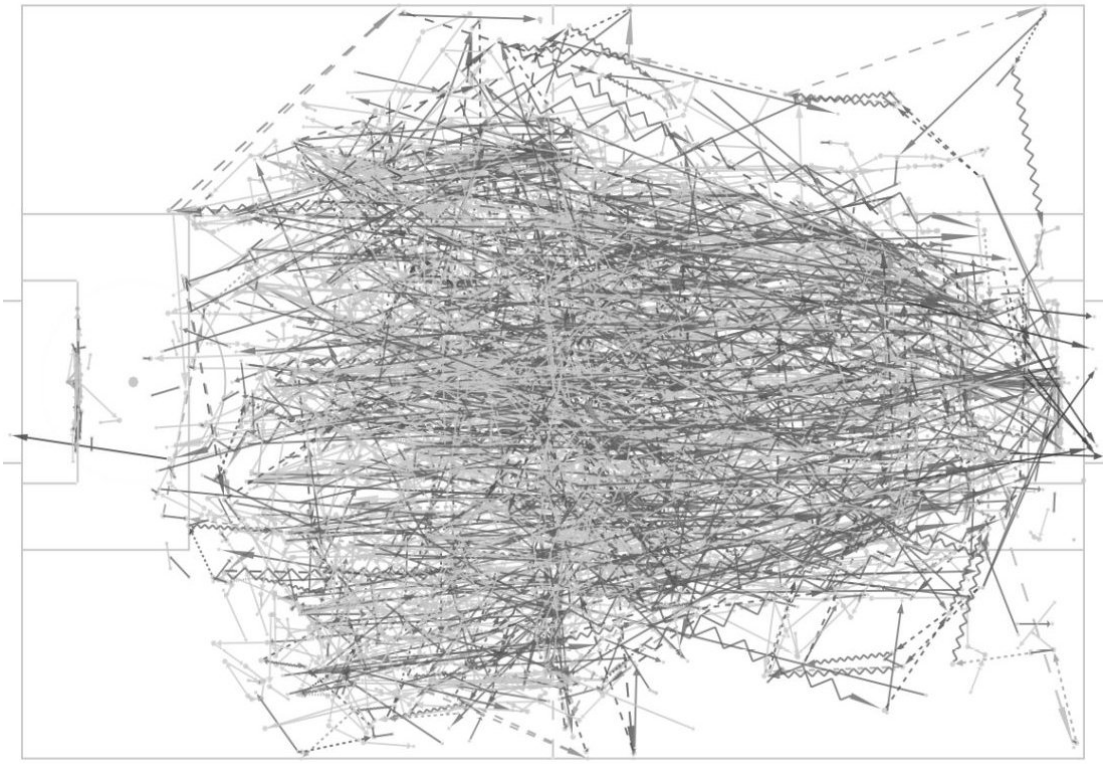
Domenski prostor agentov je simulirano nogometno igrišče, ki je orientirano skladno s sliko 39. Zato domenske pozicije agentov ustrezajo dvodimenzionalnim pozicijam igralcev na igrišču. Moštvo, ki ga analiziramo, vedno zavzame isto polovico igrišča, ki je predstavljena na levem delu slike 39.



Slika 39. Orientacija igrišča v domeni RoboCup.

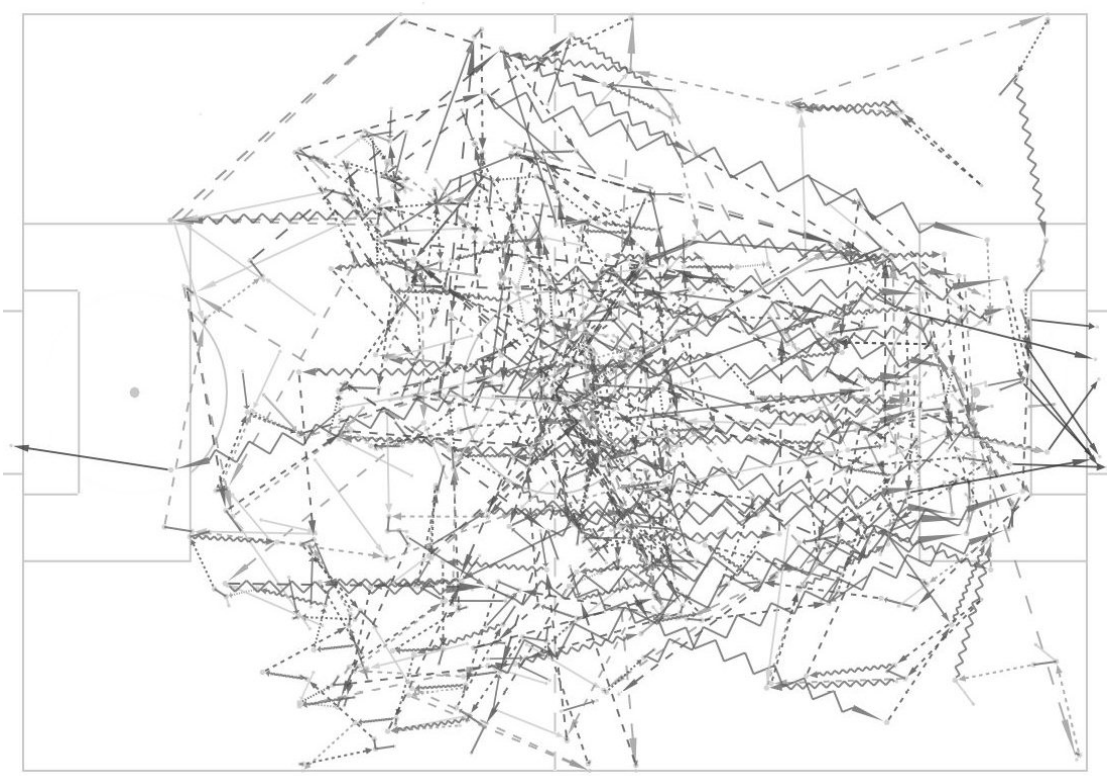
Zaznana zaporedja instanc akcij, ki opisujejo potek RoboCup tekme, služijo kot vhod v MASDA, ki v prvem koraku zgradi ustrezen AG. Pozicije vozlišč v AG ustrezajo domenskim pozicijam agentov na igrišču z naslednjo izjemo. Akcije, ki vključujejo brcanje žoge (vse akcije, ki jih predstavlja poddrevo hierarhije agentnih akcij s korenom v elementu "Kick") dejansko opravijo agenti v eni časovni enoti, njihov potek pa traja več časovnih enot (npr. podaja "To-player" traja, dokler žoge ne ujame drug agent). Zato jih modeliramo tako, kot da bi agent, ki izvaja akcijo, sledil poti žoge do njenega sprejema oz. prestrezanja, čeprav je lahko agent v tem času na drugi poziciji.

Izris AG smo nadalje prilagodili tako, da smo za izris povezav uporabili različne barve in vzorce črt. Tako črtkanim povezavam ustrezajo podaje soigralcem, cikcak črtam ustreza preigravanje, polnim črtam v področje gola pa uspešne strele na gol. Primer AG za eno RoboCup igro je predstavljen na sliki 40.



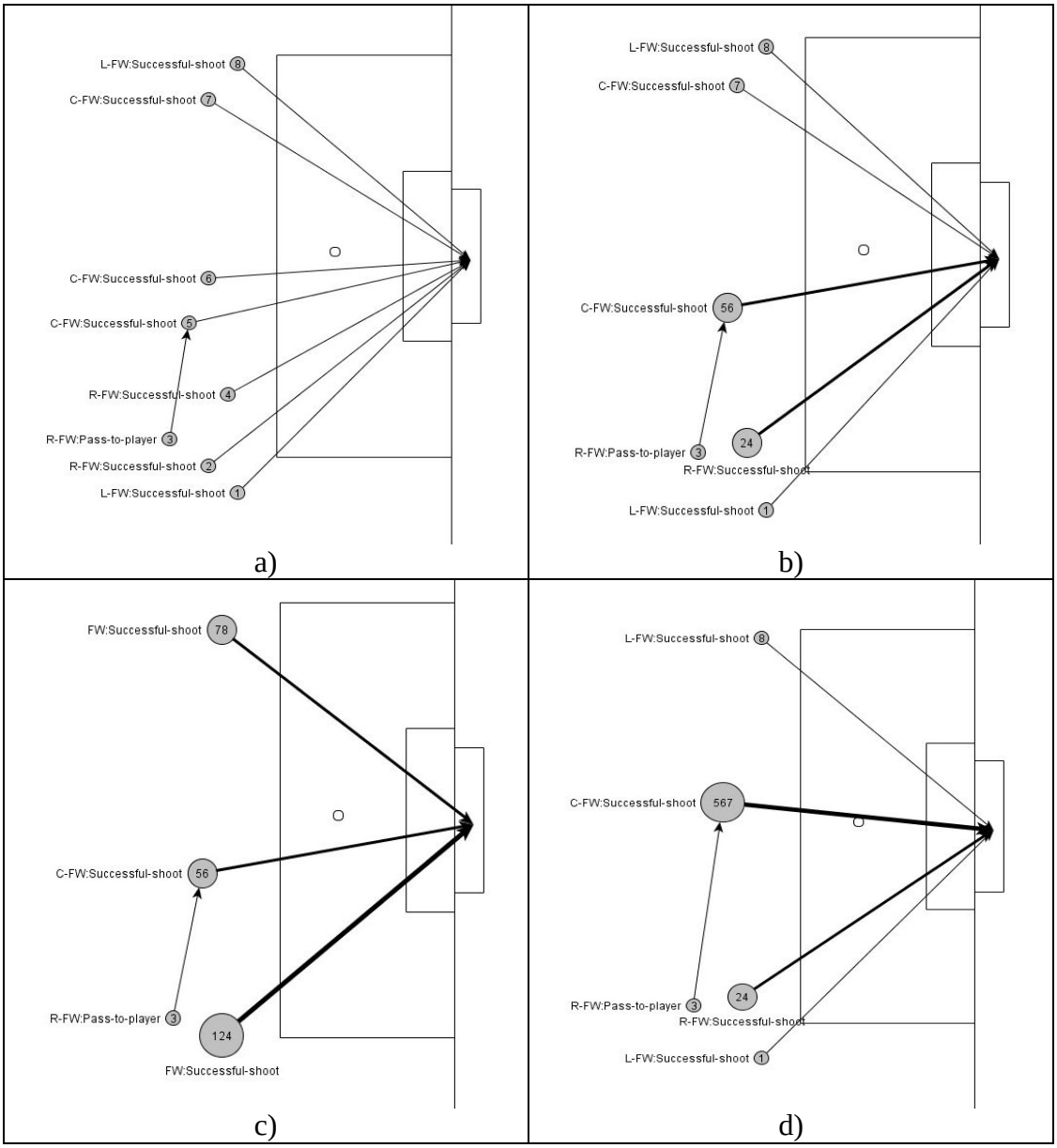
Slika 40. Primer akcijskega grafa, ki ustreza RoboCup igri med moštvoma STEP (levi) in Brainstormers04 (desni). Končni rezultat je bil 8:1 za moštvo STEP.

Pri začetni analizi smo odkrili, da hkratno vključevanje akcij vseh igralcev naredi AG zelo kompleksen, saj prikrije najpomembnejši akcijski vidik v nogometu, to je vidik igre z žogo. Zato smo uvedli poenostavitev in nismo upoštevali večine akcij, ki opisujejo gibanje brez žoge. Poenostavljena hierarhija akcij je prikazana na sliki 78 v prilogi A. AG za eno RoboCup igro, ki uporablja poenostavljeno hierarhijo akcij, je predstavljen na sliki 41. Tak AG je bolj primeren za hitro vizualno analizo, saj bolj pregledno predstavlja gibanje agentov, ki imajo v posesti žogo. Slabost opisane poenostavitve akcij je nezmožnost opisa akcij agentov, ki se neposredno ne navezujejo na igro z žogo. Primer za to je gibanje vratarja, ki je modelirano v AG na sliki 40, v AG na sliki 41 pa ne.



Slika 41. Primer poenostavljenega akcijskega grafa, ki ustreza RoboCup igri med levim moštvom STEP in desnim moštvom Brainstormers04.

Naslednji korak je definicija ustrezne metrike oddaljenosti akcijskih konceptov. Za razumevanje razdalj med akcijskimi koncepti v nogometu si poglejmo primer na sliki 42a. Na primeru vidimo, da sta si vozlišči 5 in 6 najbližje, poleg tega pa predstavljata isti par vloga:akcija (*C-FW:Successful-shoot*). Zato sklepamo, da predstavljata isti akcijski koncept: *srednji napadalec (C-FW) strelja naravnost na gol (Successful-shoot) izven kazenskega prostora*. Podobno velja za vozlišči 2 in 4, ki skupaj predstavljata primer akcijskega koncepta: *desni napadalec (R-FW) strelja na gol (Successful-shoot) iz desne strani ob robu kazenskega polja*. Primer ustrezno združenih vozlišča je predstavljen na sliki 42b. Toda če želimo nadaljevati s procesom združevanja podobnih primerov akcij, naletimo na dilemo, koliko pomeni razlika v vlogah oz. akcijah glede na oddaljenost primerov na igrišču. Slika 42c prikazuje primer, ko je razlika agentnih vlog manj pomembna od oddaljenosti primerov na igrišču. Zato dobljena vozlišča opisujejo bolj lokacijsko opredeljene akcijski koncepte. Vozlišče 124 tako predstavlja akcijski koncept: *napadalec (FW) strelja na gol (Successful-shoot) iz desne strani ob robu kazenskega polja*, vozlišče 78 pa: *napadalec (FW) strelja na gol (Successful-shoot) iz leve strani ob robu kazenskega polja*. Slika 42d prikazuje primer, ko je razlika agentnih vlog bolj pomembna od oddaljenosti agentov na igrišču. Dobljeno vozlišče zato opisuje akcijski koncept, ki je bolj opredeljen s specifično agentno vlogo. Vozlišče 567 torej predstavlja akcijski koncept: *srednji napadalec (C-FW) strelja naravnost na gol (Successful-shoot) izven kazenskega prostora.*



Slika 42. Shematski primeri agentnih akcij, ki pomagajo razumeti razlike med akcijskimi koncepti v domeni RoboCup. Vozlišča, ki imajo oznako iz več cifer, predstavljajo združena vozlišča iz osnovnih vozlišč iz primera a).

Če bi nadaljevali proces abstrakcije, bi z združevanjem vozlišč dobili opis akcijskega koncepta: *napadalec strelja na gol izven kazenskega prostora*. V danem primeru je akcijski koncept, ki ga opisuje vozlišče 3, najbolj oddaljeno od vseh primerov, saj predstavlja akcijo (*Pass-to-player*), ki je pomensko še najbolj različna od ostalih.

Iz tega primera sklepamo, da na razdaljo med akcijskimi koncepti pomembno vpliva domenska oddaljenost primerov agentov, prav tako pa tudi razlika med vlogami in akcijami, ki jih opisujejo akcijski koncepti. Metrika oddaljenosti mora torej upoštevati razdaljo med agenti na igrišču ter razdaljo med opisi akcij in agentnih vlog, kar je skladno z našo definicijo funkcije razdalje med dvema vozliščema v AAG:

$$dist(a, b) = w_{pos} \cdot \overline{dist_{pos}(a,b)} + w_{role} \cdot \overline{dist_{Hrole}(a,b)} + w_{action} \cdot \overline{dist_{Haction}(a,b)},$$

V domeni RoboCup definiramo razdaljo med agenti na igrišču $\overline{dist_{pos}(a,b)}$ kot evklidsko razdaljo med dvema točkama na igrišču:

$$\overline{dist_{pos}(a,b)} = \sqrt{(a.pos.x - b.pos.x)^2 + (a.pos.y - b.pos.y)^2}$$

Določitev hierarhične razdalje pa ni tako enostavna, saj je določitev ustrezne metrike cilj mnogih raziskav [1, 34, 85, 86, 93]. Rada in sodelavci [82] so določili hierarhično razdaljo med dvema elementoma *a* in *b* v hierarhični semantični mreži kot dolžino najkrajše poti $distance_H(a, b)$, ki povezuje dva elementa :

$$dist_H(a, b) = distance_H(a, b)$$

Ker MASDA v procesu abstrakcije združuje opise podobnih elementov v bolj splošne, ima ta definicija dobro razlago. Meri namreč razliko med globino primerjanih elementov in globino skupnega elementa, torej opisuje, za koliko se nov opis razlikuje od starih. Vendar pa upošteva le prvo od štirih zahtev [2] za dobro funkcijo hierarhične razdalje, predstavljenih v poglavju 5.3.1, zato jo imamo za slab približek. Možno definicijo so predstavili tudi Giunchiglia in sodelavci [45]. Hierarhično razdaljo so določili kot ekvivalenčno relacijo, če je dolžina najkrajše poti med dvema elementoma hierarhije manjša ali enaka dani meji $maxdist_H$, sicer je razdalja neskončna:

$$dist_H(a, b, maxdist_H) = \begin{cases} 0; \text{ če } distance_H(a, b) \leq maxdist_H \\ \infty; \text{ sicer} \end{cases}$$

Taka definicija poenoti oddaljenost vseh elementov, ki so znotraj meje $maxdist_H$, zato je neodvisna od velikosti hierarhije, ne odpravi pa neupoštevanja globine primerjanih elementov in gostote hierarhije.

Naša definicija hierarhične razdalje temelji na ugotovitvi, da vsak nivo v hierarhiji določa nov nivo abstrakcije. Zato človek pri ocenjevanju hierarhične razdalje kot najpomembnejši faktor upošteva nivo skupnega prednika primerjanih elementov. Če primerjamo razdalji dveh parov elementov, imata daljšo hierarhično razdaljo elementa, ki imata najbližjega skupnega prednika višje v hierarhiji. Če imata oba para najbližjega skupnega prednika na istem nivoju hierarhije, potem ima večjo hierarhično razdaljo tisti par, ki ima večjo dolžino poti med elementoma. Funkcija, ki ustreza danemu opisu hierarhične razdalje, je sledeča:

$$dist_H(a, b) = \left(\frac{distance_H(a,b)}{2 \cdot depth_{Hmax}} \right) + \left(depth_{Hmax} - depth_H(common\_ancestor(a,b)) \right)$$

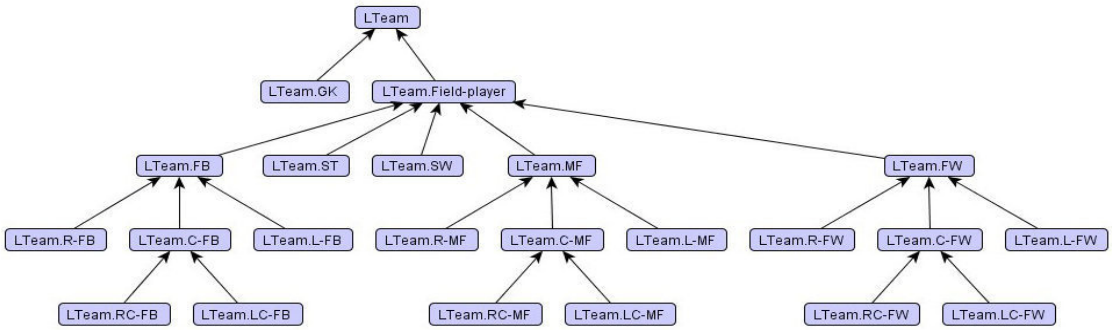
Prvi del je normalizirana dolžina poti med elementoma glede na največjo možno dolžino poti med dvema elementoma v hierarhiji H in ima zalogo vrednosti med 0 in 1. Drugi del je razlika med največjo globino hierarhije in globino najbližjega skupnega prednika v hierarhiji, kar določa absolutno višino najbližjega skupnega prednika v hierarhiji. Zato lahko ta del metrike razumemo kot abstraktnost najbližjega skupnega prednika, saj velja, da je bolj abstrakten element višje v hierarhiji. Zaloga vrednosti drugega dela metrike je med 0 in $depth_{Hmax}$. Skupna zaloga vrednosti je torej med 0 in $depth_{Hmax}$+1. Zaradi lažje primerjave z ostalimi razdaljami jo normiramo, tako da je zaloga vrednosti med 0 in 1. Funkcija, ki določa hierarhično razdaljo med

dvema elementoma v hierarhijah agentnih vlog in akcij v domeni RoboCup, je torej sledeča:

$$dist_H(a, b) = \frac{\left(\frac{distance_H(a,b)}{2 \cdot depth_{H_{max}}}\right) + \left(depth_{H_{max}} - depth_H(common\_ancestor(a,b))\right)}{depth_{H_{max}} + 1}$$

Tako definirana hierarhična razdalja zadosti trem zahtevam od štirih, saj ne upošteva le gostote elementov v hierarhiji. Ker so hierarhije agentnih vlog in akcij relativno enostavne, menimo, da neupoštevanje gostote elementov ni kritična pomanjkljivost, saj ročno grajene hierarhije bolj upoštevajo pomen elementov. Naša definicija funkcija hierarhične razdalje tudi ustreza vsem trem aksiomom razdalje [22]: nenegativnosti, simetriji in trikotniški neenakosti.

Primernost naše definicije hierarhične razdalje lahko utemeljimo na primeru hierarhije agentnih vlog levega moštva, ki je predstavljena na sliki 43. Predstavljena hierarhija je ročno zgrajena in ustrezno upošteva pomen elementov hierarhije, ki je predstavljen v preglednici 33 v prilogi A. Vrednosti hierarhičnih razdalj za nekaj primerov parov elementov iz hierarhije so predstavljene v preglednici 19, kjer so oznake elementov poenostavljene, saj zaradi krajšega izpisa ne vsebujejo niza "LTeam.".



Slika 43. Primer hierarhije agentnih vlog levega moštva v domeni RoboCup.

Človek težko numerično oceni razdaljo med elementoma v hierarhiji, lažje pa oceni, ali je hierarhična razdalja med enim parom daljša oz. krajša od razdalje med drugim parom. Zato bomo vrednotili primernost metrike hierarhične razdalje s primerjavo razdalj med seboj. Analiza urejenosti razdalj parov elementov hierarhije je podana z vrstnim redom parov v preglednici 19, kjer so primerjani pari urejeni glede na naraščajočo vrednost hierarhične razdalje od zgoraj navzdol. Več parov v isti celici predstavlja enako medsebojno oddaljenost. Predstavljeni rezultati potrjujejo sprejemljive lastnosti tako določene hierarhične razdalje, saj je:

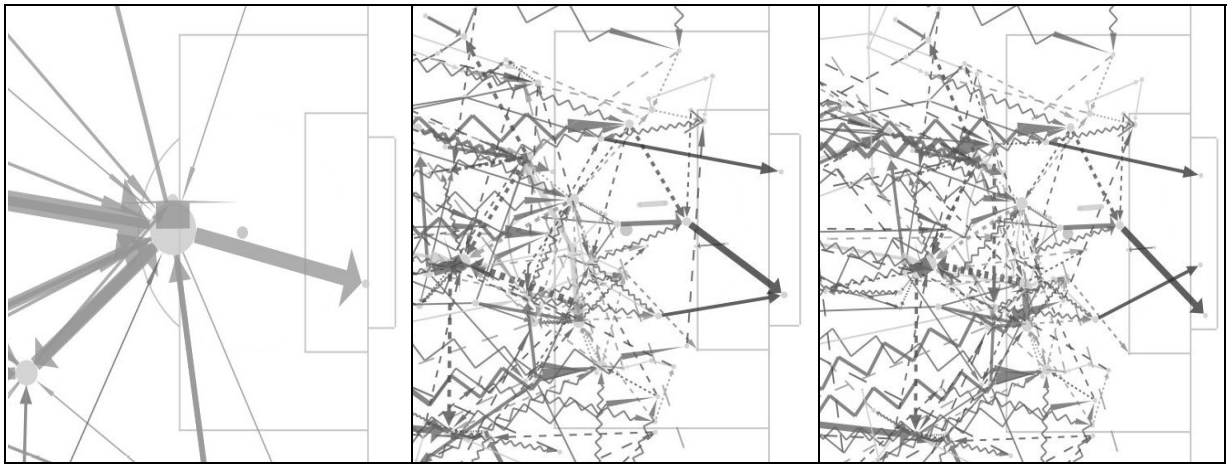
- razdalja med istim elementom je 0. Primer: $dist_H$(RC-FB, RC-FB) = 0.
- razdalja med elementoma je v največji meri odvisna od višine najbližjega skupnega prednika. Primer: $dist_H$(R-FB, L-FB) > $dist_H$(RC-FB, LC-FB).
- razdalji med parom elementov, ki imata najbližjega skupnega prednika enako visoko v hierarhiji, je odvisna od dolžine poti med elementoma. Primer: $dist_H$(R-FB, L-FB) < $dist_H$(RC-FB, L-FB).
- vrednosti razdalj med elementi enakomerno pokrivajo celotno zalogo vrednosti med 0 in 1.

a : b	$dist_H(a, b)$
RC-FB : RC-FB	0
RC-FB : C-FB	0,225
RC-FB : LC-FB	0,25
C-FB : FB	0,425
R-FB : L-FB C-FB : L-FB	0,45
RC-FB : L-FB	0,475
FB : Field-player	0,625
FB : FW FB : ST	0,65
FB : LC-FW C-FB : C-FW	0,7
RC-FB : LC-FW	0,75
GK : LTeam	0,825
LTeam : LC-FW	0,9
GK : LC-FW	0,925

Preglednica 19. Vrednosti hierarhičnih razdalj za različne pare elementov iz hierarhije na sliki 43. Pari so urejeni glede na naraščajočo vrednost hierarhične razdalje od zgoraj navzdol. Več parov v isti celici predstavlja enako medsebojno oddaljenost.

Z določitvijo ustrezne funkcije hierarhične razdalje $dist_H(a, b)$ je za popolno definicijo razdalje med vozlišči potrebno določiti še uteži w_{pos}, w_{role} in w_{action}. Problem določitve uteži je sledeč. Če za uteži w_{role} in w_{action} vzamemo premajhne vrednosti, potem opisi vlog in akcij pri združevanju nimajo velikega vpliva na razdaljo med vozlišči, zato vozlišča v AAG določajo akcijske koncepte zgolj na podlagi agentne pozicije in ne ločujejo različnih konceptov agentnih vlog in akcij. Primer za to je AAG_{15}, ki je predstavljen na sliki 44a, kjer vozlišče v golu predstavlja zaključek najbolj splošnega akcijskega koncepta "Team-roles:Action". Če za uteži w_{role} in w_{action} vzamemo prevelike vrednosti, potem v procesu združevanja preveč upoštevamo opise vlog in akcij, kar ima za posledico AAG, ki striktno ločuje koncepte akcij in vlog. Tak AAG torej ne predstavlja abstrakcije v smislu agentnih vlog in akcij, kot je to prikazano na AAG_{15} na sliki 44c, kjer vozlišča v golu predstavljajo zaključke naslednjih akcijskih konceptov "LTeam.C-FW:Successful-shoot", "LTeam.R-FW:Successful-shoot" in "LTeam.L-FW:Successful-shoot". Naš cilj je določiti uteži tako, da z zvišanjem parametra abstrakcije povezave in vozlišča v AAG_{abs} predstavljajo abstrakcijo akcijskih konceptov v prostoru agentnih vlog, akcij in domenskih pozicij agentov.

Za določitev ustreznih uteži smo generirali AAG_{abs} pri različnih vrednostih uteži in parametra abstrakcije *abs* ter ocenjevali, kateri AAG_{abs} prikazuje po naši oceni najbolj smiselno abstrakcijo agentnega delovanja. Izbran nabor parametrov je sledeč: w_{pos} = 1, w_{role} = 30 in w_{action} = 30, kar je prikazano na AAG_{15} na sliki 44b, kjer vozlišča v golu predstavljajo zaključke akcijskih konceptov LTeam.FW:Successful-shoot" in "LTeam.L-FW:Successful-shoot", kjer prvo vozlišče predstavlja akcijski koncept z delno abstrahiranim opisom agentne vloge (FW). Ker bomo v nadaljevanju vedno opisovali akcije levega moštva, bomo pri opisu agentne vloge izpustili opis moštva "LTeam.", pri čemer se privzame, da opisana vloga pripada levemu moštvu.



a) $w_{role} = 1$, $w_{action} = 1$ b) $w_{role} = 30$, $w_{action} = 30$ c) $w_{role} = 100$, $w_{action} = 100$

Slika 44. Prikaz dela AAG_{15} pri $w_{pos}=1$ in različnih utežeh w_{role} in w_{action}.

Tak način izbire parametrov metrike razdalje med vozlišči ni eksakten, daje pa človeku ustrezne rezultate, kar je skladno z zaključkom, ki jih je podal Aggarwal [1]. Eden izmed njegovih zaključkov je namreč ta, da je izbira metrik razdalje v kompleksnih domenah izrazito uporabniško naravnana, saj je prav uporabnik tisti, ki uporablja dobljene rezultate.

Pri izbiri sistema za indukcijo pravil smo se odločili za sistem za indukcijo pravil SLIPPER [31] (angl. *Simple Learner with Iterative Pruning to Produce Error Reduction*). SLIPPER generira množice pravil s požrešnim algoritmom za generiranje pravil, ki jih nato ocenjuje s *confidence-rated boosting* metodo, ki je različica metode AdaBoost [96]. Implementacija sistema je osnovana na učnemu sistemu RIPPER [30] istega avtorja. Podobno kot RIPPER je SLIPPER hiter, robusten in preprosto za uporabo. Eksperimentalno je pokazano [31], da je časovna kompleksnost algoritma $O(n \log n)$, kjer je n število učnih primerov. Poleg tega so njegove hipoteze kompaktne in lahko razumljive, kar je dobrodošel dejavnik pri človeški analizi pravil. Za opis učnih podatkov lahko uporablja SLIPPER tudi atribute, katerih vrednosti so množice, kar zadosti pogoju, ki ga pri algoritmu za indukcijo pravil postavlja MASDA. Ker SLIPPER generira množico pravil, za opis posameznega akcijskega koncepta uporabimo le prvo (najbolje ocenjeno) pravilo iz množice generiranih.

Prva analiza kaže, da ima domena RoboCup kompleksen domenski prostor, kar ima za posledico majhno število primerov, ki opisujejo posamezne akcijske koncepte. Posledično lahko pričakujemo, da bomo lahko primerno opisali le najbolj pogoste akcijske koncepte.

7.2 Izvedba meritev

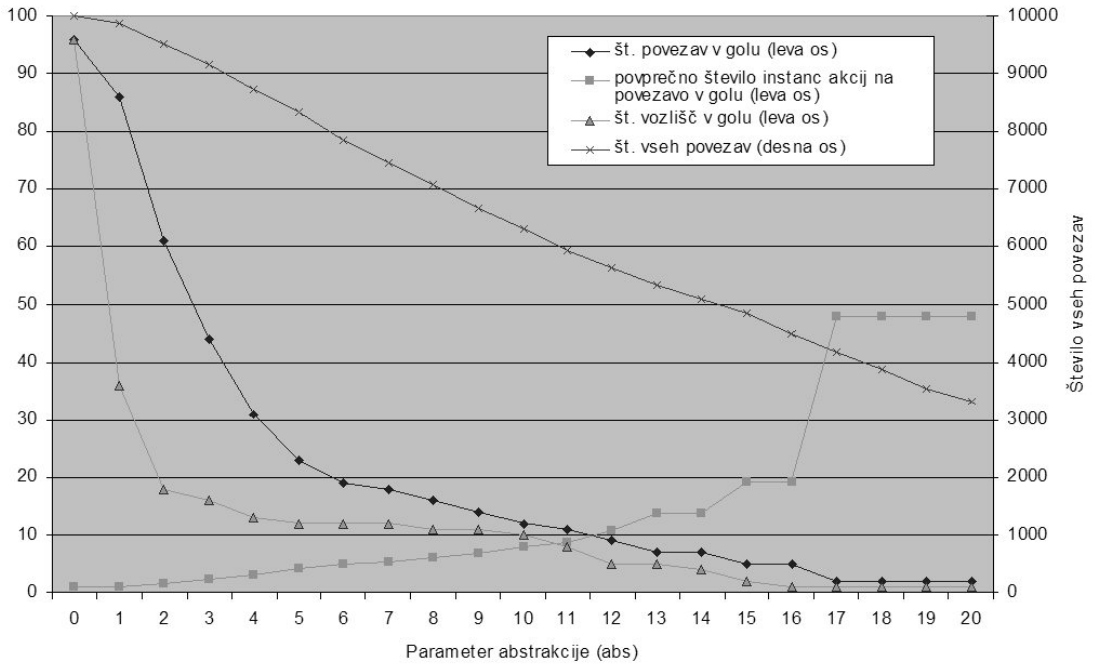
Ker je bil naš cilj pokazati, da modeliranje z MASDA opiše strateške vidike večagentnega delovanja, smo se odločili, da ocenimo modeliranje makroakcije, ki predstavlja uspešen napad na gol. Razlogi za to so sledeči:

- napad na gol predstavlja strateško dejavnost,
- napad na gol zahteva interakcijo večjega števila agentov, torej predstavlja neko zaporedje akcij različnih agentov,

- z gotovostjo lahko ugotovimo ponavljanje podobnih situacij - število golov določa število uspešnih napadov na gol.

Če želimo, da MASDA uspešno odkrije in opiše vzorce strateškega delovanja, moramo poskrbeti, da vhodni podatki opisujejo večkratno ponavljanje podobnih situacij. Zato smo za vhodne podatke pripravili igre z večjim številom golov, ki predstavljajo tekme med referenčnim moštvom in poljubnim nasprotnikom. Za referenčno moštvo smo izbrali ekipo STEP, ki je zmagala na svetovnem prvenstvu RoboCup05. Med vsemi igrami ekipe STEP na RoboCup05 smo izbrali 10 iger z največjim številom zadetkov za moštvo STEP. S tem smo hoteli doseči čim večje ponavljanje podobnih situacij - napadov na gol. Izbrane igre predstavljajo igro moštva STEP proti 8 različnim nasprotnikom, saj le dve igri predstavljata tekmi proti istemu nasprotniku. Skupaj je bilo zadetih 99 golov (STEP 96, nasprotniki 3), v povprečju 9,9 na tekmo (9,6 za STEP, 0,3 za nasprotnike), največja razlika v golih je bila 18 (STEP 18, nasprotnik 0) in najmanjša 3 (STEP 4, nasprotnik 1). Igre smo združili v eno samo igro, imenovano referenčna igra. Referenčna igra ima dolžino 66216 časovnih enot in velikost preko 90 MB. V njej smo zaznali več kot 3.000.000 osnovnih agentnih akcij in natanko 10.000 agentnih akcij.

Iz referenčne igre smo zgradili ustrezen referenčni AG in nato še nabor referenčnih AAG_{abs} za vrednosti parametra abstrakcije abs = 1, 2, ..., 20. Preglednica 20 prikazuje osnovne statistike, ki opisujejo število vseh povezav v referenčnih AAG_{abs} ter število povezav, instanc akcij in končnih vozlišč, ki predstavljajo strele na gol v AAG_{abs}, pri različnih vrednosti parametra abstrakcije. Omenjene vrednosti so prikazane v obliki grafa na sliki 45.



Slika 45. Grafični prikaz števila povezav, ki prikazujejo akcijske koncepte strela na gol, povprečno število instanc akcij v teh povezavah, število končnih vozlišč v golu in število vseh vozlišč v referenčnih AAG_{abs} za različne vrednosti abs.

Iz predstavljenih statistik sledi, da se število povezav, torej število predstavljenih akcijskih konceptov, v referenčnem AAG manjša skoraj linearno z velikostjo parametra abstrakcije, kar je zaželena lastnost. Število povezav, ki predstavljajo uspešne strele na gol, se z večanjem parametra abstrakcije najprej hitro zmanjša, nato pa se pri vrednosti abs=6 ustali v skoraj linearen trend padanja do vrednosti abs=17. Podoben trend kaže tudi število vozlišč v področju gola, ki predstavljajo koncepte zaključkov strelov v gol. Ker število povezav v gol pada do abs=6 veliko hitreje, kot število povezav v celem AAG, lahko sklepamo, da so si instance akcij, ki predstavljajo strele na gol, v povprečju bolj podobne, kot ostale instance akcij.

Povprečno število instanc akcij, ki jih opisujejo posamezne povezave, ki predstavljajo strele na gol, se do vrednosti abs=13 veča skoraj linearno, nato pa se v dveh večjih skokih dvigne na končno vrednost. Linearno manjšanje števila povezav in linearno večanje števila vsebovanih instanc akcij v povezavah je zaželeno in delno potrjuje primernost izbrane metrike razdalje med vozlišči .

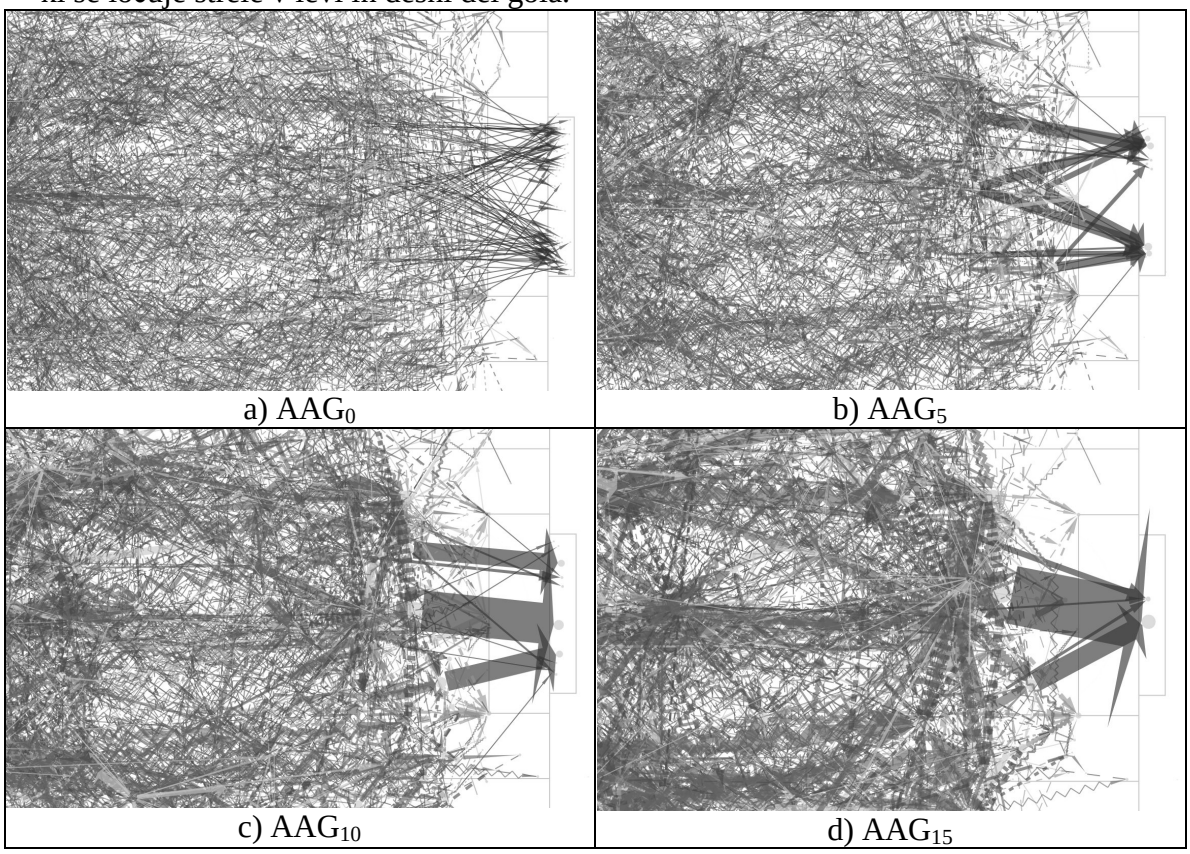
AAG_{abs}	število vseh povezav	število povezav v AAG, ki predstavljajo strele moštva STEP na gol	povprečno število instanc akcij v teh povezavah	število končnih vozlišč v golu
AAG_0	10000	96	1	96
AAG_2	9525	61	1,57	18
AAG_4	8739	31	3,10	13
AAG_6	7866	19	5,05	12
AAG_8	7077	16	6	11
AAG_{10}	6305	12	8	10
AAG_{12}	5647	9	10,66	5
AAG_{14}	5088	7	13,71	4
AAG_{16}	4478	5	19,2	1
AAG_{18}	3869	2	48	1
AAG_{20}	3306	2	48	1

Preglednica 20. Število vseh povezav v AAG_{abs} ter število povezav, instanc akcij in končnih vozlišč, ki predstavljajo strele na gol v AAG_{abs} pri različnih vrednosti parametra abstrakcije.

Na sliki 46 so predstavljeni deli referenčnih AAG_{abs} za različne vrednosti parametra abstrakcije, ki prikazuje del igrišča pred nasprotnikovem golu. Pri analizi strelov na gol se že pri AAG_0 (slika 46a) izkaže, da streli na gol potekajo večinoma v levi ali desni del gola. To še bolj potrdi AAG_5 (slika 46b), kjer se vse povezave, ki predstavljajo strel na gol, zaključijo bodisi v levem ali desnem delu gola. Z večanjem abstrakcije se akcijski koncepti združujejo v bolj abstraktne in ne ločijo več strela v levi oz. desni del gola, kar je vidno pri AAG_{10} (slika 46c), še bolj pa pri AAG_{15} (slika 46d).

Podrobnejša analiza pokaže, da se pri abs=9 začnejo združevati vozlišča, ki opisujejo zaključek strela v levi oz. desni del gola, v vozlišča, ki predstavljajo le splošen zaključek strela v gol. Ker želimo pri analizi napada na gol dobiti opise akcijskih konceptov, ki opisujejo čim večje število akcijskih instanc, ob enem pa še vedno

dobro ločijo ciljno lokacijo strela, smo za analizo izbrali parameter abstrakcije *abs*=8, ki še ločuje strele v levi in desni del gola.

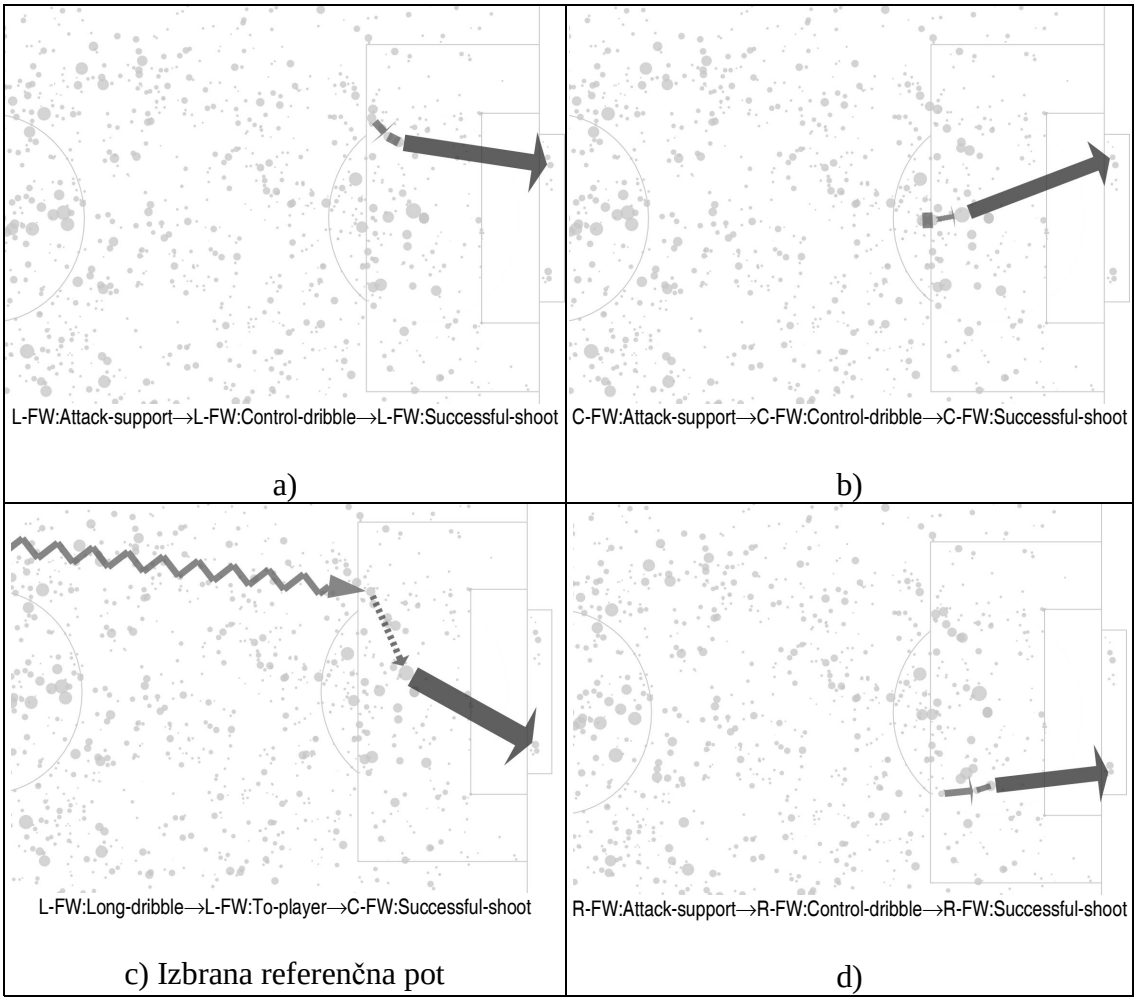


Slika 46. Prikaz delov referenčnih AAG_{abs}, ki predstavljajo strele na gol, za različne vrednosti parametra abstrakcije abs.

Ker uspešen napad na gol običajno zahteva vsaj tri zaporedne akcije, smo za analizo izbrali poti v referenčnem AAG_8 dolžine 3 ali več, ki predstavljajo uspešen napad na gol. Na sliki 47 so predstavljene vse štiri poti v AAG_8, ki predstavljajo uspešen napad na nasprotnikov gol in imajo močno povezanost vsaj 3, kar pomeni, da se je omenjen napad v celoti zgodil vsaj 3x. Poti na sliki 47a in 47b imata močno povezanost 4, medtem ko imata poti na sliki 47c in 47d močno povezanost 3. Ker izmed predstavljenih poti na sliki 47 le pot na sliki 47c predstavlja akcije več kot enega agenta, v našem primeru agenta z vlogo L-FW in agenta z vlogo C-FW, smo to pot izbrali za referenčno. Karakterni opis izbrane referenčne poti iz AAG_8 je predstavljen v preglednici 21.

Vloga agenta	**L-FW**	**L-FW**	**C-FW**	**C-FW**
Akcijski koncept	začetek akcije **Long-dribble**	začetek akcije **To-player**	začetek akcije **Successful-shoot**	konec akcije **Successful-shoot**
Domenska pozicija agenta	**(x: 0,20, y: -13,27)**	**(x: 37,25, y: -8,95)**	**(x: 40,70, y: -1,69)**	**(x: 53,31, y: 4,72)**

Preglednica 21. Karakterni opis referenčne poti iz AAG_8.



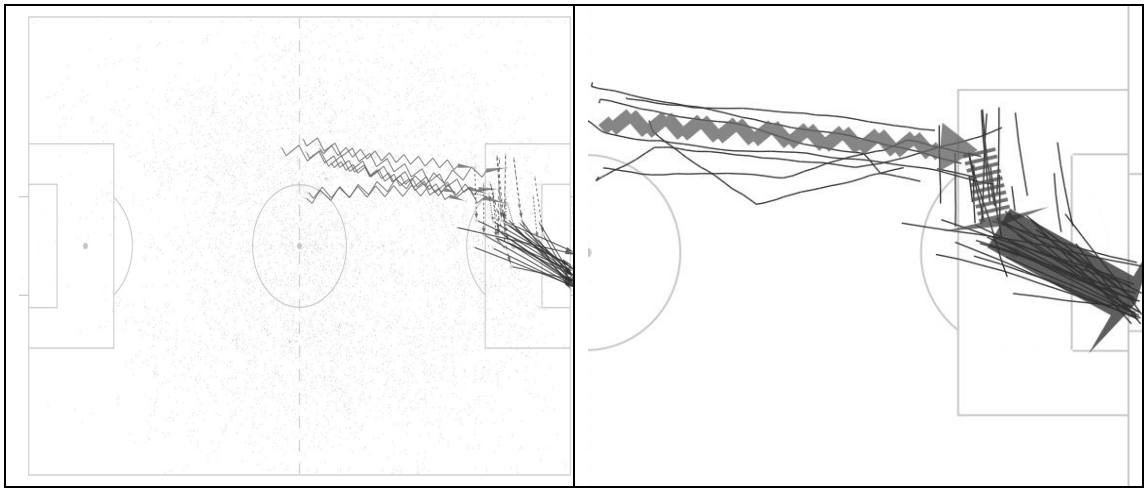
Slika 47. Vse poti v referenčnem AAG8, ki predstavljajo uspešen napad in imajo šibko povezanost vsaj 3. Pod vsako sliko je predstavljen ustrezni karakterni opis poti brez pozicij vozlišč, ki so razvidne iz same poti.

Izbrana referenčna pot, prikazana na sliki 47c, predstavlja strateški večagentni koncept makroakcije, ker:

- Pot predstavlja strateški koncept, ker opisuje strateško aktivnost v nogometu, to je napad na gol.
- Pot predstavlja večagentni koncept, ker opisuje delovanje več agentov. V našem primeru dveh agentov, prvega z vlogo L-FW in drugega z vlogo C-FW.
- Pot predstavlja koncept makroakcije, ker opisuje specifično zaporedje akcij: dolgo preigravanje, podaja soigralcu in strel na gol, ki se dejansko pojavlja v nogometni strategiji.

7.2.1 Človeška analiza

Izbrana referenčna pot opisuje koncept makroakcije, ki predstavlja uspešen napad na gol. Z besedami ga lahko opišemo sledeče: ***Levi napadalec preigrava*** *(L-LF:Long-dribble) po levi polovici igrišča v levi del kazenskega prostora, kjer* ***poda*** *(L-FW:To-player)* srednjemu *napadalcu v sredino kazenskega prostora.* ***Srednji napadalec nato uspešno strelja*** *(C-FW:Successful-shoot) v desni del nasprotnikovega gola.*

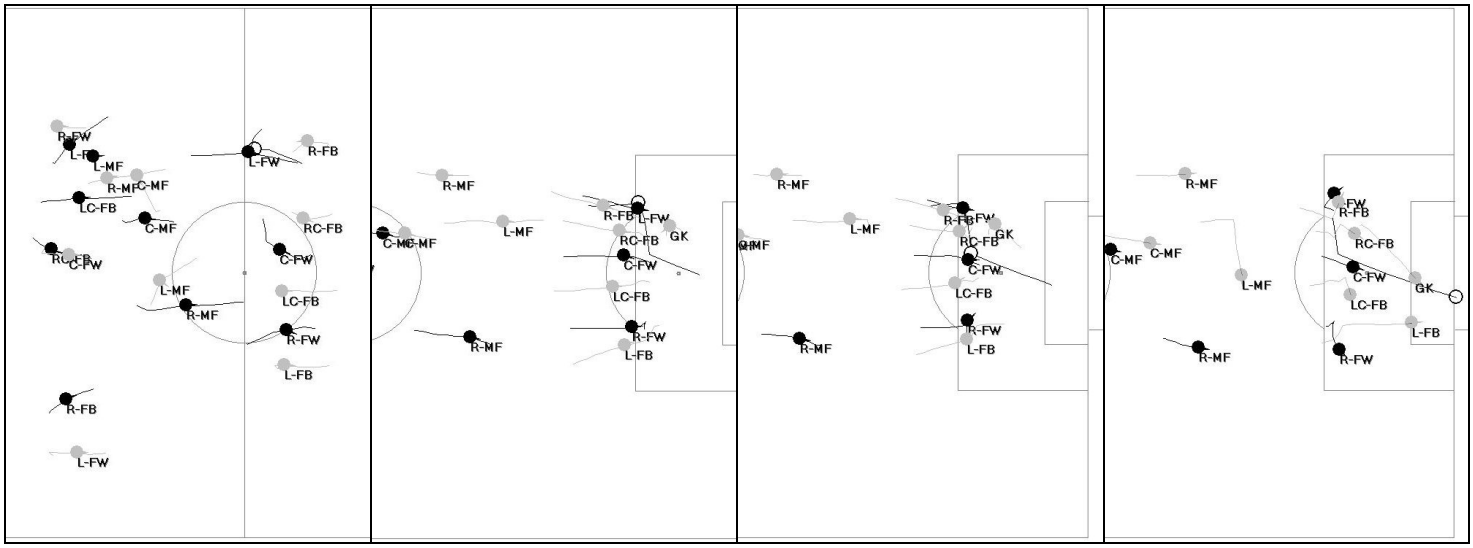


a) Poti v izhodiščnem AG, ki sestavljajo referenčno pot v AAG_8.

b) Referenčna pot brez vozlišč z označenimi potmi sodelujočih agentov.

Slika 48. Dva prikaza instanc akcij, ki sestavljajo referenčno pot.

Referenčna pot ima močno povezanost 3 in šibko povezanost 7. Torej je zgrajena iz poti v izhodiščnem referenčnem AAG, ki predstavljajo natanko 3 nepretrgane napade na gol in vsaj 7 delnih napadov na gol. Dejanske poti iz izhodiščnega referenčnega AG, ki sestavljajo referenčno pot, so predstavljene na sliki 48a. Ker je bil eden izmed ciljev pri snovanju MASDA dobiti opise večagentnega delovanja, ki omogočajo vizualno predstavitev, preverimo tudi prostorsko ujemanje predstavljene poti in opravljenih poti agentov. Na sliki 48b so s tanko temno črto predstavljene poti agentov, ki so sodelovali v makroakciji, ki jo predstavlja referenčna pot. Iz slike lahko sklepamo, da pot, kot grafični del opisa koncepta makroakcije, sledi splošnemu poteku gibanja agentov. Do razhajanj prihaja predvsem pri opisovanju daljšega gibanja, ki ne predstavlja pretežno linearnega gibanja, ki ga sicer dobro ponazarjajo usmerjene povezave v AAG.



a) b) c) d)

Slika 49. Zaporedje slik iz napada, ki ga predstavlja makroakcija. Črte, ki izhajajo iz igralcev in žoge, ustrezajo njihovi dejanski poti.

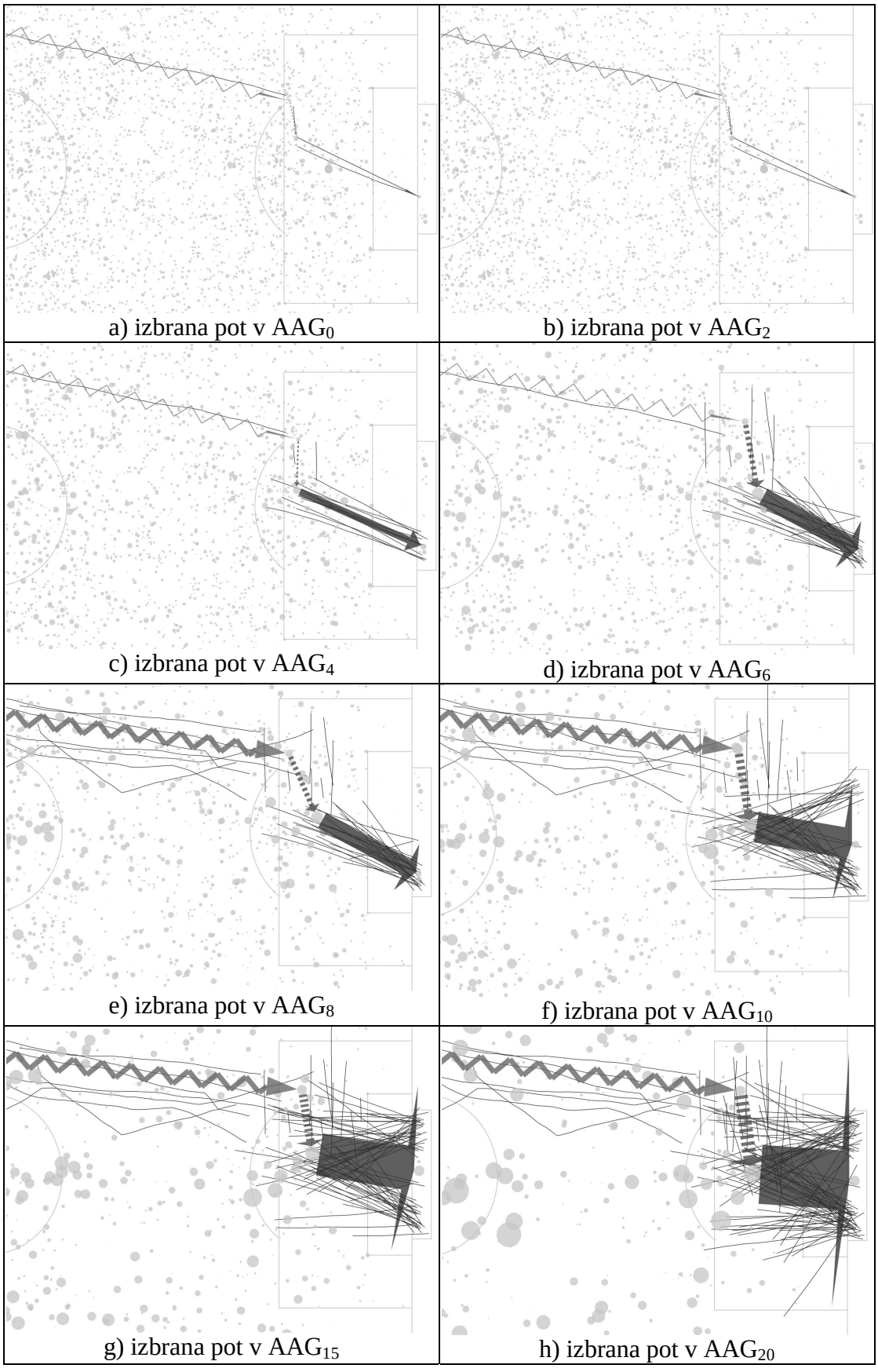
Vsi trije neprekinjeni napadi, ki jih predstavlja referenčna pot, so predstavljeni v obliki videoposneteka, ki je dostopen na sledečem internetnem naslovu: http://dis.ijs.si/andraz/phd/reference-3attacks.avi. Posnetku prvega napada ustreza zaporedje slik, predstavljenem na sliki 49, ki odgovarjajo naslednjim akcijskim konceptom:

- Slika 49a: začetek dolgega preigravanje z žogo levega napadalca (začatek akcijskega koncepta *L-FW:Long-dribble*).
- Slika 49b: začetek podaje levega napadalca soigralcu (začatek akcijskega koncepta *L-FW:To-player*).
- Slika 49c: začetek uspešnega strela na gol srednjega napadalca (začatek akcijskega koncepta *C-FW:Successful-shoot*).
- Slika 49d: konec uspešnega strela na gol srednjega napadalca (konec akcijskega koncepta *C-FW:Successful-shoot*).

Slika 50a predstavlja pot v referenčnem AAG_0, ki ustreza predstavljenemu napadu. Z abstrakcijo AAG_{abs} lahko spremljamo postopno abstrakcijo poti, torej abstrakcijo opisa koncepta makroakcije, ki jo predstavlja dani napad. Na sliki 50 so predstavljeni ustrezni grafični opisi, v preglednici 22 pa karakterni opisi izbrane poti pri različnih vrednostih parametra abstrakcije, kjer so s sivo označeni bolj abstraktni opisi.

Karakterni opis poti v **AAG_0**				
Vloga	L-FW	L-FW	C-FW	C-FW
Akcijski koncept	začetek akcije Long-dribble	začetek akcije To-player	začetek akcije Successful-shoot	konec akcije Successful-shoot
Položaj agenta	(x: -3,60, y: -14,60)	(x: 40,34, y: -6,27)	(x: 41,62, y: -2,78)	(x: 53,60, y: 5,59)
Karakterni opis poti v **AAG_{10}**				
Vloga	L-FW	L-FW	C-FW	C-FW
Akcijski koncept	začetek akcije Long-dribble	začetek akcije To-player	začetek akcije Successful-shoot	konec akcije Successful-shoot
Položaj agenta	(x: 0,20, y: -13,27)	(x: 38,85, y: -9,61)	(x: 40,36, y: -1,00)	(x: 53,33, y: 1,04)
Karakterni opis poti v **AAG_{15}**				
Vloga	L-FW	L-FW	FW	FW
Akcijski koncept	začetek akcije Long-dribble	začetek akcije To-player	začetek akcije Successful-shoot	konec akcije Successful-shoot
Položaj agenta	(x: 0,20, y: -13,27)	(x: 38,85, y: -9,61)	(x: 40,15, y: -2,41)	(x: 53,39, y: -0,41)
Karakterni opis poti v **AAG_{20}**				
Vloga	L-FW	Field-player	Field-player	Field-player
Akcijski koncept	začetek akcije Movement-on-the-ball	začetek akcije To-player	začetek akcije Successful-shoot	konec akcije Successful-shoot
Položaj agenta	(x: 3,25, y: -16,40)	(x: 38.90, y: -10,08)	(x: 40,69, y: -0,20)	(x: 53,37, y: 0,60)

Preglednica 22. Karakterni opisi izbrane poti za različne AAG_{abs}. Sive celice predstavljajo opise, ki imajo bolj abstrakten opis vlog in akcij od prejšnjih.



Slika 50. Prikaz abstrakcije izbrane poti (z vrisanimi potmi agentov).

Predstavljeni karakterni opisi in grafične predstavitve poti prikazujejo naraščajočo abstrakcijo v smislu opisanih:

- **agentnih vlog** (npr. abstrakcija vloge strelca iz ***C-FW*** v ***FW*** ter v ***Field-player***),
- **akcij** (npr. abstrakcija agentne akcije ***Long-dribble*** v ***Movement-on-the-ball***) in
- **domenskega prostora** (npr. abstrakcija iz ***strel v desni del gola*** v ***strel v gol***).

Karakterni opis in grafični opis sicer v določeni meri opišeta predstavljeni koncept makroakcije, vendar opisujeta zgolj delovanje agentov, ki so neposredno vključeni v izvajanje makroakcije. Specifične lastnosti okolja in agentov, ki podrobneje opisujejo izbrano makroakcijo, predstavljajo šele ustrezna pravila.

Pravila opisujejo začetke oz. zaključke predstavljenih akcijskih konceptov. Pri indukciji pravil lahko vplivamo na grajenje pravil s primernim izborom učnih primerov. Ker imamo na voljo relativno majhno število pozitivnih učnih primerov, nadziramo le izbor negativnih učnih primerov, kjer s spreminjanjem spodnje in zgornjo meje sprejemljive razdalje določamo izbor negativnih učnih primerov. Zato se število negativnih učnih primerov spreminja, število pozitivnih učnih primerov pa ostaja enako. Pravila, ki opisujejo dani koncept makroakcije, so predstavljena v naslednji obliki:

Primer št. X: **spodnja meja razdalje - zgornja meja razdalje**	**število učnih primerov**	
	+	**-**
pravila, ki opisujejo **začetek** akcijskega koncepta **L-FW:Long-dribble** ↓ pravila, ki opisujejo **začetek** akcijskega koncepta **L-FW:To-player** ↓ pravila, ki opisujejo **začetek** akcijskega koncepta **C-FW:Successful-shoot** ↓ pravila, ki opisujejo **konec** akcijskega koncepta **C-FW:Successful-shoot**		

Najprej smo pri izboru negativnih učnih primerov želeli izbrati bližnjo okolico. Zato smo spodnjo mejo nastavili na 0, zgornjo mejo sprejemljive razdalje pa na 16. Inducirana pravila so sledeča:

	Primer št. 1: **0 - 16**	**število učnih primerov**	
		+	**-**
Pravilo 1	RTeam.LC-FB:Center-circle ∧ RTeam.L-FB:Right-wing ∧ L-MF:Medium-distance ∧ R-FW:Faster ∧ RTeam.R-FW:Moving ∧ L-MF:Defending-half ∧ L-FW:Moving ∧ RTeam.LC-FB:Right-half ∧ R-FB:Very-far	**7**	**259**
Pravilo 2	↓ C-FW:Incoming ∧ L-FW:Immediate ∧ R-MF:Incoming ∧ RTeam.GK:Incoming ∧ L-MF:Back ∧ RTeam.C-FW:Very-far ∧ R-MF:Center-of-the-field	**11**	**271**
Pravilo 3	↓ C-MF:Moving-away ∧ L-FW:Medium-distance ∧ C-FW:Attacking-half.Attacking-third.Danger-zone ∧ C-MF:Medium-distance ∧ L-FB:Moving-away ∧ L-MF:Medium-distance ∧ RC-FB:Moving-away-slow	**24**	**278**
Pravilo 4	↓ C-FW:Left ∧ Ball:Opponent-goal ∧ RTeam.LC-FB:Attacking-half.Attacking-third.Penalty-box ∧ RTeam.C-FW:Moving ∧ RTeam.L-FW:Far ∧ C-FW:Attacking-half.Attacking-third.Penalty-box	**24**	**56**

Sedaj spremenimo meje sprejemljivih negativnih učnih primerov na daljno okolico, torej med 16 in 64. Inducirana pravila so sledeča:

Primer št. 2: **16 -64**	**število učnih primerov**	
	+	**-**
RTeam.LC-FB:Center-circle ∧ RTeam.R-MF:Back ∧ LC-FB:Right ∧ GK:Moving-away ∧ L-MF:Medium-distance ∧ RTeam.R-FW:Far ∧ C-MF:Medium-distance ∧ C-FW:Moving ∧ RTeam.LC-FB:Far ∧ L-MF:Defending-half ∧ RTeam.R-FW:Moving ∧ R-FW:Faster ∧ RTeam.L-MF:Middle-third	**7**	**5275**
↓		
L-FW:Immediate ∧ RTeam.GK:Short-distance ∧ RTeam.GK:Incoming ∧ R-MF:Incoming ∧ L-MF:Back ∧ C-FW:Incoming ∧ L-FB:Very-far ∧ R-MF:Center-of-the-field	**11**	**3958**
↓		
C-FW:Has-ball ∧ C-FW:Attacking-half.Attacking-third.Danger-zone ∧ L-FB:Moving-away ∧ LC-FB:Left ∧ RTeam.L-FW:Very-far ∧ R-FW:Short-distance	**24**	**3915**
↓		
RTeam.C-MF:Moving-away-fast ∧ C-MF:Attacking-half.Attacking-third ∧ RTeam.RC-FB:Attacking-half.Attacking-third.Danger-zone ∧ R-MF:Far ∧ RTeam.L-FW:Moving ∧ LC-FB:Center-of-the-field ∧ R-FW:Center-of-the-field	**24**	**223**

Zaradi visoke neuravnoteženosti števila negativnih in pozitivnih učnih primerov ponovimo indukcijo pravil, pri tem da obdržimo le naključnih 30% negativnih primerov. Nova pravila so sledeča:

Primer št. 3: **16 -64** naključno izbranih le 30% negativnih primerov	**število učnih primerov**	
	+	**-**
RTeam.LC-FB:Center-circle ∧ RTeam.R-MF:Back ∧ RTeam.LC-FB:Far ∧ GK:Moving-away ∧ C-MF:Medium-distance ∧ RTeam.R-FW:Far ∧ L-FW:Moving	**7**	**1583**
↓		
L-FW:Immediate ∧ RTeam.GK:Short-distance ∧ C-FW:Incoming ∧ L-MF:Back ∧ RTeam.GK:Incoming ∧ R-MF:Incoming ∧ R-MF:Center-of-the-field	**11**	**1188**
↓		
C-FW:Has-ball ∧ R-MF:Attacking-half.Attacking-third ∧ RTeam.GK:Left ∧ L-FB:Moving-away ∧ RTeam.GK:Left-half	**24**	**1175**
↓		
Ball:Opponent-goal ∧ C-FW:Left ∧ C-FW:Attacking-half.Attacking-third.Danger-zone	**24**	**67**

Manjše število učnih primerov povzroči občutno manjše število pogojev v posameznih pravilih.

Razlago pravil bomo podali za prvi primer (str. 90). Analizo pravil smo pripravili v sodelovanju z nogometnima strokovnjakoma [79, 84], ki sta poleg opisne razlage situacij ocenila tudi primernost pogojev glede na dano situaciji. Tako **krepki podčrtani tisk** prikazuje zelo pomembne pogoje, **krepki tisk** smiselne pogoje in ~~**krepki prečrtani tisk**~~ nepomembne pogoje. Razlaga obeh strokovnjakov je predstavljena najprej z opisom situacije in nato z razlago pravil.

Razlaga situacije, ki je predstavljena na sliki 49a.

Situacija, ki jo predstavlja prvi akcijski koncept (*L-FW:Long-dribble*) predstavlja napačno postavitev branilcev desnega moštva RTeam.LC-FB in RTeam.L-FB. Omenjena branilca sta postavljena za napadalcema levega moštva C-FW in R-FW,

namesto da bi bila pred njima, zato tudi ne moreta preprečiti neizogibnega prodora napadalnih igralcev levega moštva. Značilnost pozicije so tudi napačne postavitve sredinskih igralcev levega moštva L-MF, C-MF in R-MF, ki se zadržujejo preveč v obrambnih položajih. Njihova vloga v dani situaciji bi morala biti striktno napadalna.

Pravilo 1:

RTeam.LC-FB:Center-circle ∧ RTeam.L-FB:Right-wing ∧ L-MF:Medium-distance ∧ R-FW:Faster ∧ RTeam.R-FW:Moving ∧ L-MF:Defending-half ∧ L-FW:Moving ∧ RTeam.LC-FB:Right-half ∧ R-FB:Very-far

Razlaga pravila:

V tem kontekstu **RTeam.LC-FB:Center-circle** določa pozicijo je komandanta obrambe desnega moštva, zato je njegova pozicija pomembna. **RTeam.L-FB:Right-wing** pove, da je L-FB predaleč na desnem krilu, čeprav bi moral biti v sredini pred R-FW in C-FW. **L-MF:Medium-distance** pove, da je L-MF preveč oddaljen od žoge, in zato ne prispeva pri napadu. **R-FW:Faster** pove, da se desni napadalec pravilno hitro premika (v smeri napada) ter s tem odpira nove napadalne priložnosti. ~~**RTeam.R-FW:Moving**~~ je pogoj, za katerega strokovnjaki nimajo prave razlage in se jim zdi nepotreben. **L-MF:Defending-half** ponovno pove, da se L-MF zadržuje preveč v obrambni polovici, namesto da bi pomagal pri napadu. **L-FW:Moving** pravilno ugotovi, da se napadalec z žogo premika. **RTeam.LC-FB:Right-half** določa, da RTeam.LC-FB ni pravilno postavljen, saj bi moral biti v sredini pred C-FW in R-FW. ~~**R-FB:Very-far**~~ določa primerno razdaljo desnega branilca do žoge, čeprav sam igralec ne prispeva k akciji, zato je tak pogoj nepomemben.

Razlaga situacije, ki je predstavljena na sliki 49b.

Drugi akcijski koncept (*L-FW:To-player*) predstavlja nevarno situacijo za desno moštvo, saj ima napadalno moštvo kar tri napadalce pred nasprotnikovimi branilci, kar daje igralcu z žogo več možnosti: L-FW lahko sam preigra vratarja, lahko poda C-FW ali pa R-FW. Obrabni igralci desnega moštva so "zamudili" pri obrambi in so glavni krivci za nastalo situacijo

Pravilo 2:

C-FW:Incoming ∧ L-FW:Immediate ∧ R-MF:Incoming ∧ RTeam.GK:Incoming ∧ L-MF:Back ∧ RTeam.C-FW:Very-far ∧ R-MF:Center-of-the-field

Razlaga pravila:

V tem kontekstu **C-FW:Incoming** pravilno določi, da se srednji napadalec približuje žogi oz. jo prestreza. Tudi **L-FW:Immediate** smiselno določi, da je igralec z žogo v njeni neposredni bližini. **R-MF:Incoming** nakazuje, da se sredinski igralec pravilno približuje, čeprav je še vedno preveč oddaljen od akcije. **RTeam.GK:Incoming** pove, da gre vratar desnega moštva na žogo, kar je v dani situaciji edina pravilna akcija vratarja. ~~**L-MF:Back**~~ določa, da je levi sredinski igralec za žogo, kar je pravilno, čeprav nepomembno opisuje situacijo. ~~**RTeam.C-FW:Very-far**~~ določa veliko oddaljenost napadalca desnega moštva, kar je v redu, čeprav njegova pozicija nima vpliva v dani situaciji. **R-MF:Center-of-the-field** pove, da je desni sredinski igralec predaleč zadaj v središču igrišča. Njegova pozicija bi morala biti bližje žogi, v podporo napadalcem.

Razlaga situacije, ki je predstavljena na sliki 49c.

Tretji akcijski koncept (*C-FW:Successful-shoot***)** predstavlja brezizhodno situaciji za desno moštvo, saj ima C-FW prosto za nemoten strel na gol. Krivdo za to nosijo predvsem obrambni igralci desnega moštva, ki se niso uspeli postaviti med napadalce in gol.

Pravilo 3:

C-MF:Moving-away ∧ L-FW:Medium-distance ∧ C-FW:Attacking-half.Attacking-third.Danger-zone ∧
C-MF:Medium-distance ∧ L-FB:Moving-away ∧ L-MF:Medium-distance ∧ RC-FB:Moving-away-slow

Razlaga pravila:

V tej situaciji pomeni ~~**C-MF:Moving-away**~~ in ~~**L-FB:Moving-away**~~, da se pri strelu na gol vsi igralci gibljejo relativno preč od žoge. **L-FW:Medium-distance** pravilno oceni, da je L-FW na srednji razdalji od žogo, saj je prav on podal žogo strelcu. **C-FW:Attacking-half.Attacking-third.Danger-zone** je zelo pomembno pravilo, ki določa, da je strelec v nevarnem področju pred golom. **C-MF:Medium-distance** in **L-MF:Medium-distance** določa da sta sredinska igralca na srednji razdalji, čeprav bi morala biti bližje žogi. ~~**RC-FB:Moving-away-slow**~~ v tej situaciji pove, da se je hitro bližal žogi, zato se ob strelu na gol giblje počasi glede na žogo, vendar na samo akcijo nima vpliva.

Razlaga situacije, ki je predstavljena na sliki 49d.

Zadnji akcijski koncept (zaključek *C-FW:Successful-shoot*) predstavlja zaključek uspešnega strela na gol. Opis te situacije za nogometne strokovnjake nima pravega smisla vendar zaradi popolnosti analize opišemo tudi to situacijo.

Pravilo 4:

C-FW:Left ∧ Ball:Opponent-goal ∧ RTeam.LC-FB:Attacking-half.Attacking-third.Penalty-box ∧
RTeam.C-FW:Moving ∧ RTeam.L-FW:Far ∧ C-FW:Attacking-half.Attacking-third.Penalty-box

Razlaga pravila:

C-FW:Left ustrezno določa, da je strelec levo od žoge, torej je streljal v desni del gola. **Ball:Opponent-goal** določi, da je žoga v golu, **RTeam.LC-FB:Attacking-half.Attacking-third.Penalty-box,** da je levi srednji branilec desnega moštva v kazenskem prostoru, ~~**RTeam.C-FW:Moving**~~ in ~~**RTeam.L-FW:Far**~~ nimata pomena v tej situaciji in **C-FW:Attacking-half.Attacking-third.Penalty-box**, da je strelec v kazenskem prostoru.

Ocena nogometnih strokovnjakov je, da pravila v dokajšnji meri opisujejo del pomembnih značilnosti situacij, saj je 10 pogojev po oceni strokovnjakov zelo pomembnih, 10 jih je smiselnih in le 9 pogojev v pravilih neprimernih. Pri tem je pomembno omeniti, da pogoji nakazujejo na glavne prednosti oz. pomanjkljivosti predstavljenih situacij.

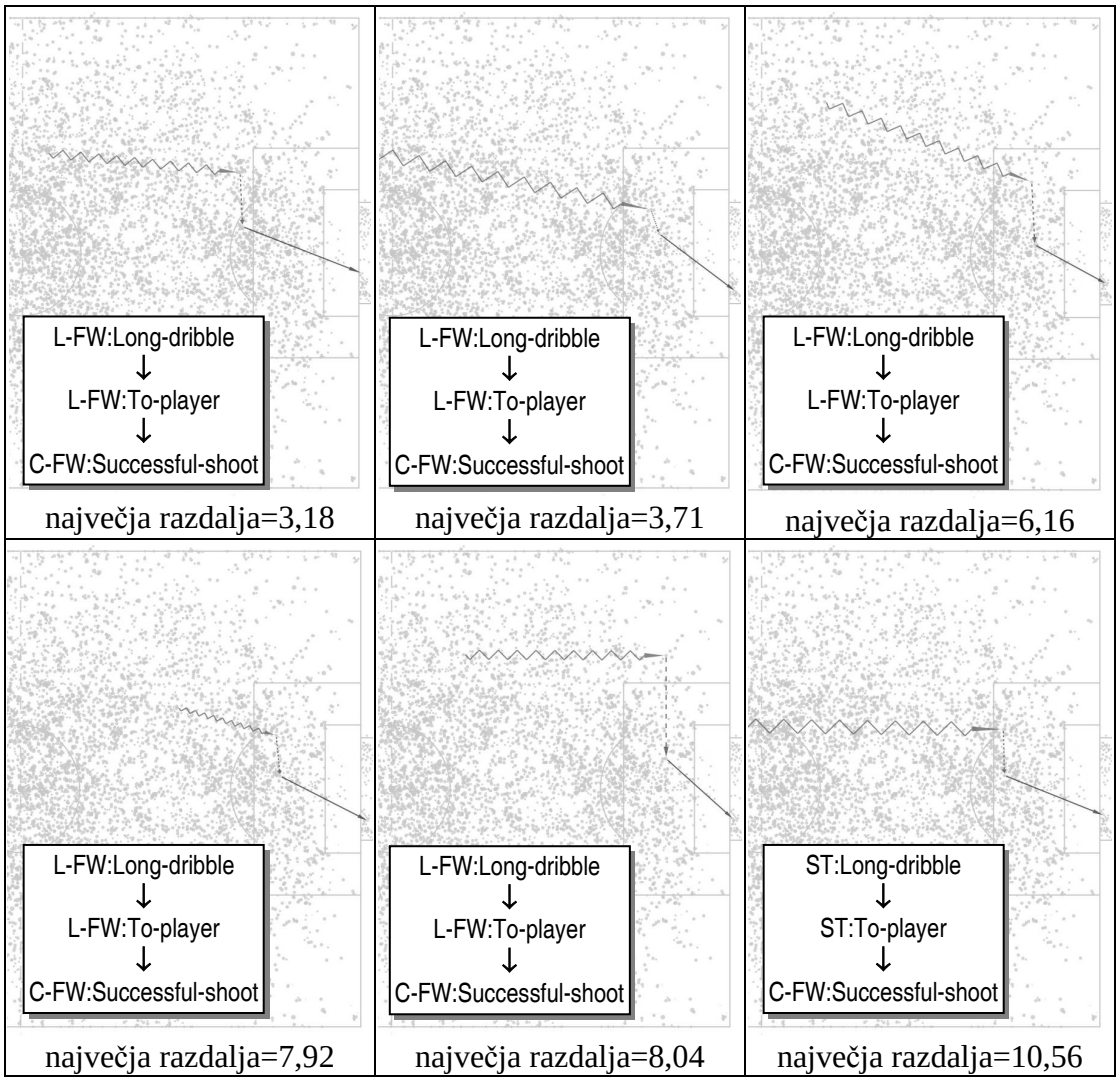
Predstavljena analiza napada na gol pokaže, da pravila opisujejo tudi akcije agentov, ki niso neposredno opisane v karakternem opisu makroakcije. To je v našem primeru vidno na primer v pravilu, ki opisuje akcijski koncept *L-FW:To-player*, kjer pogoj

RTeam.GK:Incoming predstavlja vratarjevo prestrezanje žoge. Zato sklepamo, da tudi simbolni opisi makroakcij predstavljajo interakcijo med agenti.

Opozoriti je potrebno, da strokovnjaki ocenjujejo glede na svoje obširno znanje o človeškem nogometu, MASDA pa se je učil le na osnovi posnetkov desetih iger in osnovnega predznanja.

7.2.2 Strojno ovrednotenje

Za strojno preverjanje uspešnosti klasificiranja s pomočjo modelov, ki jih generira MASDA, smo ponovno izbrali napad levega moštva STEP, ki je predstavljen na videoposnetku, dostopnem na http://dis.ijs.si/andraz/phd/reference-attack.avi in predstavlja koncept makroakcije, ki ga predstavlja referenčna pot iz AAG_0, predstavljena na sliki 50a.



Slika 51. Poti, ki so podobne referenčni poti iz AAG_0.

Najprej smo preverili uporabo karakternega opisa za iskanje podobnih konceptov makroakcij. Na sliki 51 je predstavljenih 6 konceptov makroakcij, ki imajo najbolj podoben karakterni opis glede na referenčno pot iz AAG_0. Slike prikazujejo poti ter

delne karakterne opise makroakcije, pod slikami poti pa so podane vrednosti največje razdalje med soležnimi vozlišči referenčne in primerjane poti.

Slike podobnih konceptov makroakcij kažejo, da je mogoče MASDA modele učinkovito uporabiti tudi za identifikacijo podobnih konceptov makroakcij. Predstavljeno poti so namreč podobne tako v opisih vlog in akcij kot tudi v opisih agentih pozicij.

Za strojno preverjanje smo uporabili 10-kratno prečno preverjanje. Iz vseh 10 iger smo zgradili ustrezni $AG_{reference}$. Pri podanem parametru abstrakcije *abs* smo nato s postopkom abstrakcije generirali $AAG_{abs\text{-}reference}$. Za izbran napad smo v $AAG_{abs\text{-}reference}$ poiskali ustrezno referenčno pot, ki predstavlja učni koncept makroakcije, in jo označili s $path_{reference}$. Referenčna pot pri različnih vrednostih parametra abstrakcije je prikazana na sliki 50.

Nato smo 10 krat ponovili poskus, kjer smo vsakič za testno izbrali novo igro, preostalih 9 pa smo določili za učne. AG, ki predstavlja testno igro, označimo s AG_{test}, AG, ki predstavlja 9 učnih iger, pa s AG_{learn}. Nato smo s procesom abstrakcije generirali ustrezen $AAG_{abs\text{-}learn}$.

V $AAG_{abs\text{-}learn}$ smo nato poiskali pot $path_{learn}$, ki najbolj ustreza referenčni. Ujemanje smo določili s primerjavo vsebovanih primerov med $path_{reference}$ in potmi v $AAG_{abs\text{-}learn}$. Pot, ki je vsebovala največ primerov iz referenčne, je postala $path_{learn}$. Velja omeniti, da je v primeru, ko je referenčna akcija opisana v $AAG_{abs\text{-}test}$ možno, da pri manjših vrednostih abstrakcije pot $path_{learn}$ ne obstaja, ker ni mogoče zgraditi poti, ki bi ustrezala referenčni. Takrat tega poskusa ne upoštevamo in ga preskočimo. Pri večjih vrednostih parametra abstrakcije teh problemov ni, ker so takrat poti sestavljene iz primerov iz več iger.

En primer predstavlja povezavo v izhodiščnem AG, torej predstavlja eno instanco akcije. Ker primeri niso označeni jih moramo najprej označiti z ustreznim razredom. To storimo tako, da za vsako povezavo v poti $path_{learn}$ ponovimo naslednji postopek. Ker predstavljajo povezave v referenčni poti ciljne razrede, označimo vse povezave v testnem AG_{test}, ki so vsebovane kot primeri v istoležni povezavi referenčne poti, z razredom *classPlus*, ostale povezave iz AG_{test} pa označimo s *classMinus*. Sedaj klasificiramo vse primere, ki jih opisujejo povezave v testnem AG_{test}. Če je rezultat klasifikacije razred *classPlus* in je enak označenem razredu povezave, potem štejemo ta primer za resnično pozitivnega (angl. *true positive*, krajše TP), sicer ga štejemo za neresnično pozitivnega (angl. *false positive*, krajše FP). Če je rezultat klasifikacije razred *classMinus* in je enak označenemu razredu povezave, potem štejemo primer za resnično negativnega (angl. *true negative*, krajše TN), sicer je neresnično negativen (angl. *false negative*, krajše FN). Algoritem, ki izvaja predstavljeni postopek strojnega preverjanja, je predstavljen na sliki 52.

V disertaciji smo izvedli klasificiranje na tri različne načine:

- samo z uporabo karakternega opisa (stran 61),
- samo z uporabo pravil (poglavje 6.2.3 na strani 65),
- s skupno uporabo pravil in karakternega opisa,
- z uporabo klasifikatorja, ki vedno klasificira v večinski razred – v našem primeru je bil večinski razred vedno *classMinus*.

V naših podatkih je število negativnih primerov veliko večje od števila pozitivnih primerov. Tako je v primeru učenja strela na gol 96 pozitivnih in kar 9904 negativnih učnih primerov. Če bi ocenjevali klasifikator strela na gol, ki bi vse primere razvrstil v večinski (negativni) razred, bi tak klasifikator dosegel kar 99,04% točnost. Taka ocena pa je zavajajoče visoka, saj je razred večine primerov trivialno določljiv, medtem ko je primere v bližini pozitivnih učnih primerov težje klasificirati.

Zato smo za testne primere vzeli le primere, ki so v določeni okolici referenčne poti. S tem smo zmanjšali množico testnih primerov za tiste primere, ki jih je enostavno klasificirati kot negativne. Pri določitvi okolice smo upoštevali največjo razdaljo med vozlišči referenčne poti in vozlišči naučene poti, ter jo povečali za vrednost parametra abstrakcije. Primeri iz izhodiščnega AG, ki imajo razdaljo do najbližjega vozlišča referenčne poti manjšo ali enako od izračunane največje razdalje, tako predstavljajo testne primere. Ker so testni primeri v relativni bližini učnih primerov, smo pri učenju pravil za izbiro negativnih učnih primerov določili interval oddaljenosti negativnih učnih primerov od pozitivnih med 0 in 16, saj smo želeli doseči čim večjo klasifikacijsko točnost med bližnjimi primeri. Pri testiranju smo spreminjali vrednost parametra abstrakcije, ki je bil nastavljen na vse cele vrednosti med 1 in 20. Merili smo klasifikacijsko točnost (angl. *accuracy*), priklic (angl. *recall*) in natančnost (angl. *precision*) [60]. Z enačbami:

$$klasifikacijska\ točnost = \frac{TP + TN}{TP + FP + TN + FN}$$

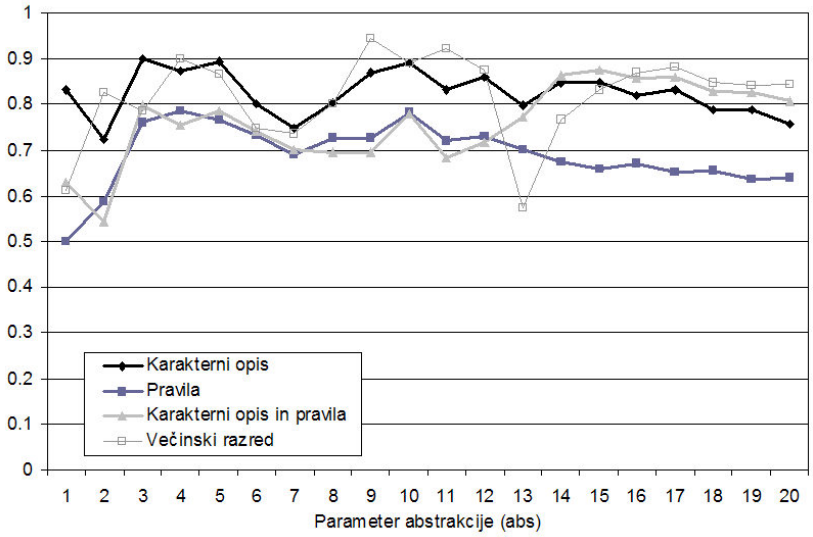
$$priklic = \frac{TP}{TP + FN}$$

$$natančnost = \frac{TP}{TP + FP}$$

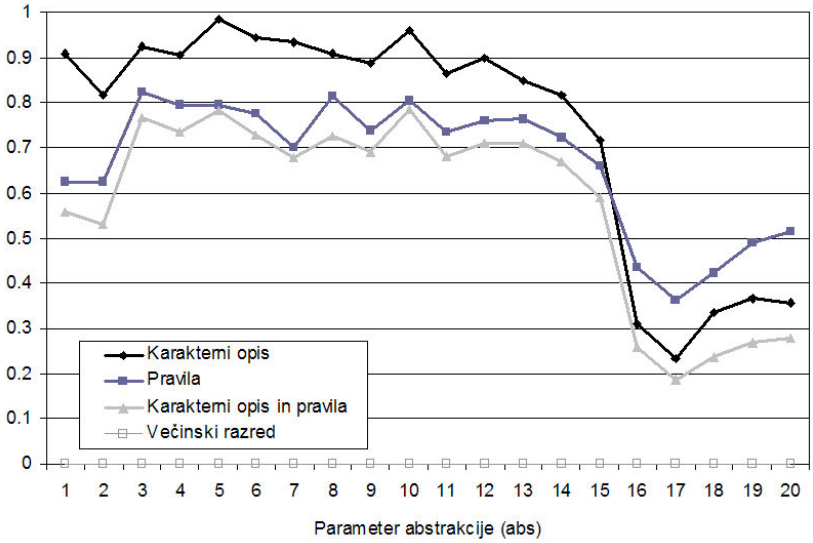
Dobljeni rezultati so podani na sliki 53.

```
Vhod:
- deset iger: AG[10]
- ciljna pot, ki je vsebovana v enem AG[i]: path_target
- spodnja in zgornja meja parametra abstrakcije: abs_min, abs_max

Algoritem:
// referenčni AG vsebuje vse igre
AG_reference = AG[0] ∪ AG[1] ∪...∪ AG[9]
// preverimo za vse vrednosti parametra abstrakcije
for (abs = abs_min; abs <= abs_max; abs++) {
  AAG_abs-reference = AbstractionProces(AG_reference , abs)
  path_reference = findPath(AAG_abs-reference, path_target)
  // 10-kratno prečno preverjanje
  for (test = 0; test<10; test++) {
    // testni AG
    AG_test = AG[test];
    // učni AG
    AG_learn = AG[0] ∪...∪ AG[test-1] ∪ AG[test+1] ∪...∪ AG[9]
    AAG_abs-learn = AbstractionProces(AG_learn, abs)
    path_learn = Find(AAG_abs-learn, path_reference)
    // preverimo vsako povezavo v naučeni poti
    for each e ∈ path_learn {
      // v AG_test označimo vse primere
      for each instance ∈ AG_test.E {
        if (instance ∈ path_reference.E.instances )
          // kot pozitivne, če pripadajo trenutni povezavi
          instance.label = classPlus
        else
          // sicer so negativni
          instance.label = classMinus
      }
      for each instance ∈ AG_test.E {
        result = Classify(instance, e)
        if (result == classPlus) {
          // klasificiran kot pozitiven
          if (instance.label == classPlus)
            tp++
          else
            fp++
        }
        else {
          // klasificiran kot negativen
          if (instance.label == classMinus)
            tn++
          else
            fn++
        }
      }
    }
  }
}

Izhod:
- statistike uspešnosti klasifikacije: tp, fp, tn, fn
```

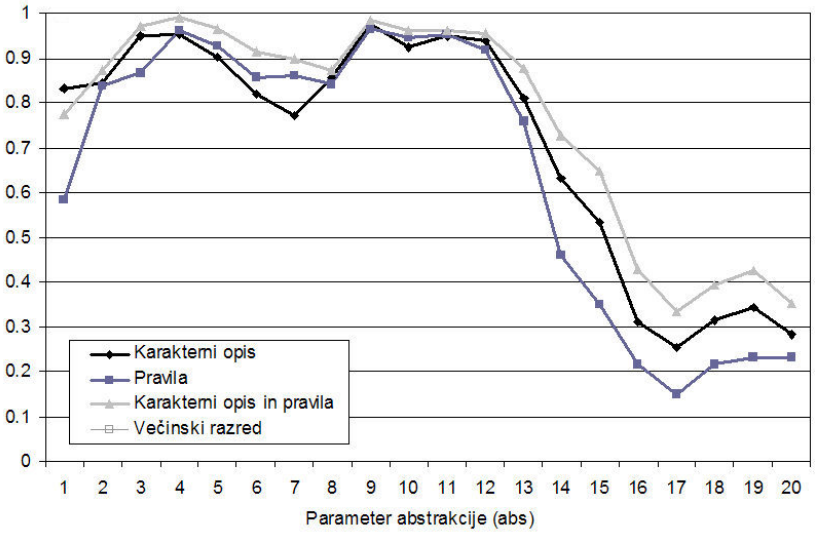
Slika 52. Algoritem, ki prikazuje postopek strojnega merjenja uspešnosti klasificiranja.



a) Klasifikacijska točnost



b) Priklic



c) Natančnost

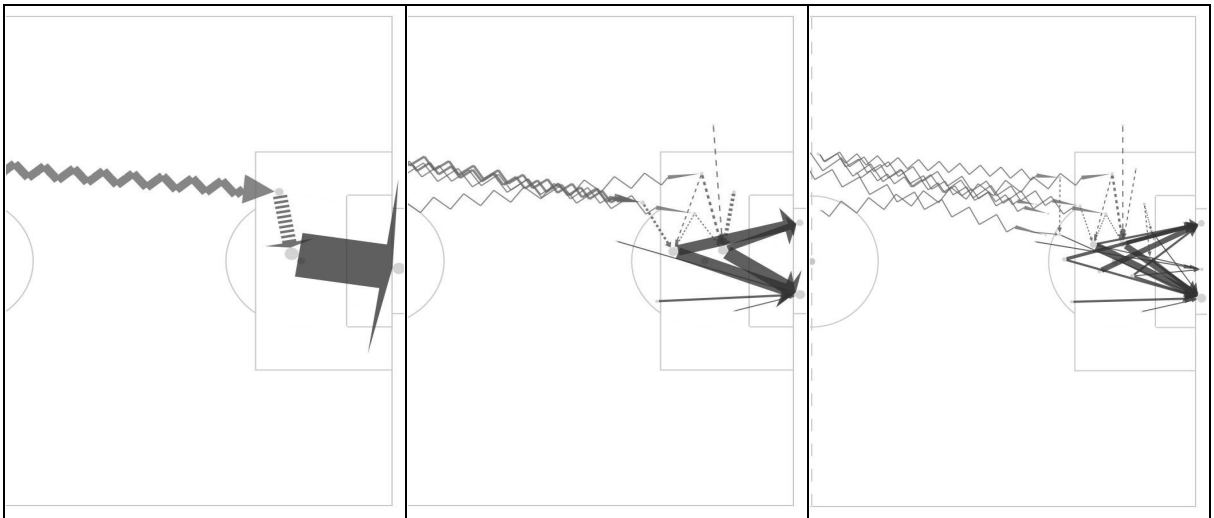
Slika 53. Rezultati meritev strojnega preverjanja.

Predstavljeni rezultati kažejo, da se točnost opisa referenčne makroakcije do abs=3 dvigne, nato pa ostaja relativno konstantna. Zanimivo je, da je za abs<14, najvišjo točnost dosegajo karakterni opisi, nato pa jih prekašajo skupni opisi s pravili. Torej so pravila pri abstrakciji manjši od 14 preveč selektivna. Z višjo abstrakcijo so pravila bolj splošna, kar pri klasifikaciji skupaj s karakternim opisom privede do boljših rezultatov. Točnost pravil se giblje na spodnji meji vseh treh postopkov klasifikacije, točnost klasifikatorja v večinski razred pa se giblje med najboljšo in najslabšo vrednostjo.

Krivulja meritev priklica ponovno narašča do abs=3, nato pa je približno stacionarna do abs=14, kot vse krivulje padejo pod 40%. Ponovno je za klasifikacijo najboljši karakterni opis, ki ima v intervalu parametra abstraktnosti med 2 in 15 priklic nad 80%. Skupna uporaba karakternih opisov in pravila daje najslabše rezultate, torej so pravila ponovno preveč selektivna in spregledajo pozitivne primere. Klasifikator, ki vedno razvršča primere v večinski – negativni – razred, ima vrednost priklica vedno 0.

Meritve natančnosti so za različne načine klasifikacije ponovno podobne. Vse vrednosti rastejo do okoli abs=4, kjer vsi tri načini klasifikacije dosežejo nad 95%, nato do abs=8 padajo in se nato ponovno dvignejo nad 95% pri abs=9. Po abs=12 natančnost klasifikacije strmo pade. Ta meritev ponovno potrdi, da so pravila bolj selektivna od karakternih opisov, torej klasificirajo manj nepravilnih primerov kot pravilnih. Meritve natančnosti klasifikatorja, ki klasificira v večinski razred, na grafu niso prikazane, ker pride v izračunu do deljenja z 0.

Čeprav so rezultati točnosti pri abstrakcijah višjih od 13 še visoki, pa meritve priklica in natančnosti pokažejo strm padec, kar lahko razložimo z velikim številom resnično negativnih primerov. Tako so pravila pri abstrakcij 20 klasificirala 442 resnično pozitivnih primerov, 1188 neresnično pozitivnih, 4194 resnično negativnih in 554 neresnično negativnih primerov.



a) Referenčna pot iz AAG_{10}.

b) Poti v AAG_5, ki so vsebovane v referenčni poti iz AAG_{10}.

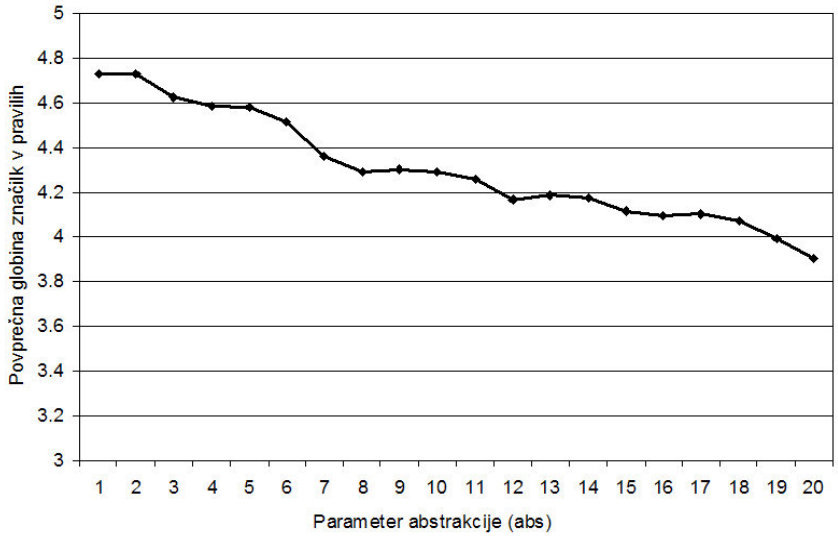
c) Poti v AAG_3, ki so vsebovane v referenčni poti iz AAG_{10}.

Slika 54. Primeri poti različnih abstrakcij, ki prikazujejo strele na gol.

Skupni zaključki predstavljenih meritev so, da so MASDA opisi konceptov makroakcij v domeni RoboCup primerni za klasifikacijo, vendar samo v intervalu abstrakcije med 3 in 13. Višje oz. nižje vrednosti abstrakcije dajejo opise modelov, ki jih ni primerno uporabiti pri klasifikaciji.

Izbira abstrakcije znotraj danega intervala je odvisna od uporabe modelov. Če želimo klasificirati specifične akcije, potem vzamemo model z manjšo abstrakcijo, če pa želimo klasificirati bolj abstraktne akcije oz. želimo uporabiti tudi človeško analizo, izberemo višjo stopnjo abstrakcije. Bolj abstraktni model sicer vsebuje večje število primerov, vendar zato tudi bolj splošno opisuje domenski prostor. Manjši model bolj specifično opisuje domenski prostor, vendar potrebujejo za opis istega prostora večje število konceptov makroakcij. Primer na sliki 54a prikazuje primer referenčne poti iz AAG_{10}. Če bi želeli modelirati vse napade, ki jih opisuje dana referenčna pot, bi v AAG_5 potrebovali opise poti predstavljene na sliki 54b, v AAG_3 pa opise poti, predstavljenih na sliki 54c. Pri modeliranja napada v levi oz. desni del gola, bi vzeli model iz AAG_5 (slika 54b), ki vsebuje manjše število konceptov makroakcij, kot model iz AAG_3, vendar loči smer strela na gol, kar ni razvidno iz modela iz AAG_{10}. Te abstrakcije smo uporabili, ker imajo referenčni koncepti makroakcije pri abstrakcijah 3, 5 in 10 približno enako visoko klasifikacijsko točnost, priklic in natančnost.

Pri merjenju uspešnosti klasificiranja smo želeli tudi ugotoviti, ali bolj abstraktnim makroakcijami t.j. potem iz AAG_{abs} z višjo vrednostjo parametra abstrakcije *abs*, pripadajo tudi bolj abstraktna pravila. Abstraktnost pravil smo izmerili s povprečno globino značilk v hierarhiji, ki so prisotne v generiranih pravilih. Manjša kot je globina značilk v hierarhijah (bližje vrhu so), tem višja je abstraktnost takega opis. Graf na sliki 55 prikazuje izmerjeno povprečno globino značilk, ki so uporabljene v pogojih generiranih pravilih.



Slika 55. Abstraktnost pravil.

Meritve abstraktnosti pravil kažejo, da se povprečna globina uporabljenih značilk v pogojih, z večanjem vrednosti parametra abstrakcije manjša. Če se globina značilk v hierarhijah manjša, se ustrezno veča abstraktnost teh značilk. V našem primeru se čez

celotni interval parametra abstrakcije v povprečju poveča skoraj za en hierarhični nivo, kar potrjuje domnevo, da so bolj abstraktni MASDA modeli opisani tudi z bolj abstraktnimi pravili. Ker rezultati meritev ustrezajo našim predpostavkam, s tem tudi posredno potrjujejo primernost izbrane funkcije razdalje med vozlišči v AAG.

8. Ovrednotenje pristopa na domeni 3vs2 Keepaway

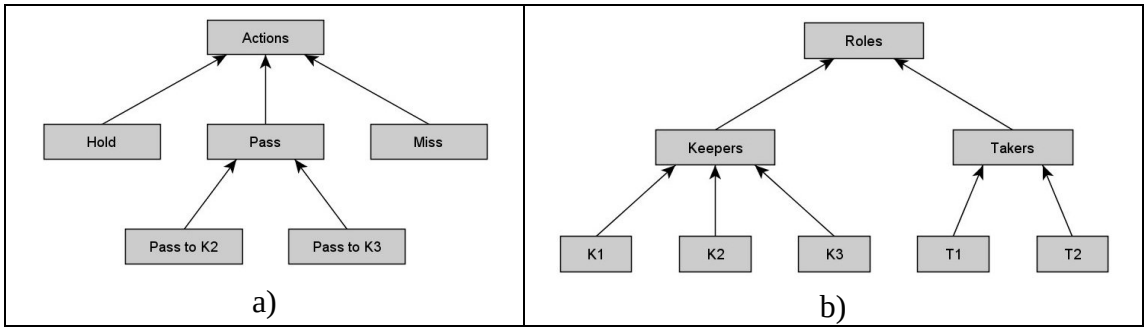
I may not have gone where I intended to go, but I think I have ended up where I intended to be.

Douglas Adams

V primerjavi z domeno RoboCup je domena 3vs2 Keepaway preprostejša, a še vedno zanimiva domena za modeliranje večagentnih sistemov. Prav njena preprostost omogoča relativno enostavno izdelavo večagentnih strategij, kar bomo pri vrednotenju s pridom izkoristili. Naš končni cilj je bil namreč aktivno posnemanje znanih strategij, torej igranje naučenih strategij, ki ustrezajo referenčnim strategijam, kar je v domeni RoboCup izredno zahtevna naloga. Z novo domeno želimo tudi pokazati splošnost in domensko neodvisnost MASDA.

8.1 Opis modeliranja MASDA

Ker deluje okolje Keepaway v okolju RoboCup, je postopek zaznave osnovnih agentnih akcij in postopek zaznave agentnih akcij v obeh primerih enak. Razlika je v podanemu domenskemu znanju, ki je v tem primeru prilagojen novi domeni. Akcije opišemo s hierarhijo akcij, ki je predstavljena na sliki 56a, agentne vloge pa s hierarhijo agentnih vlog, predstavljeno na sliki 56b.



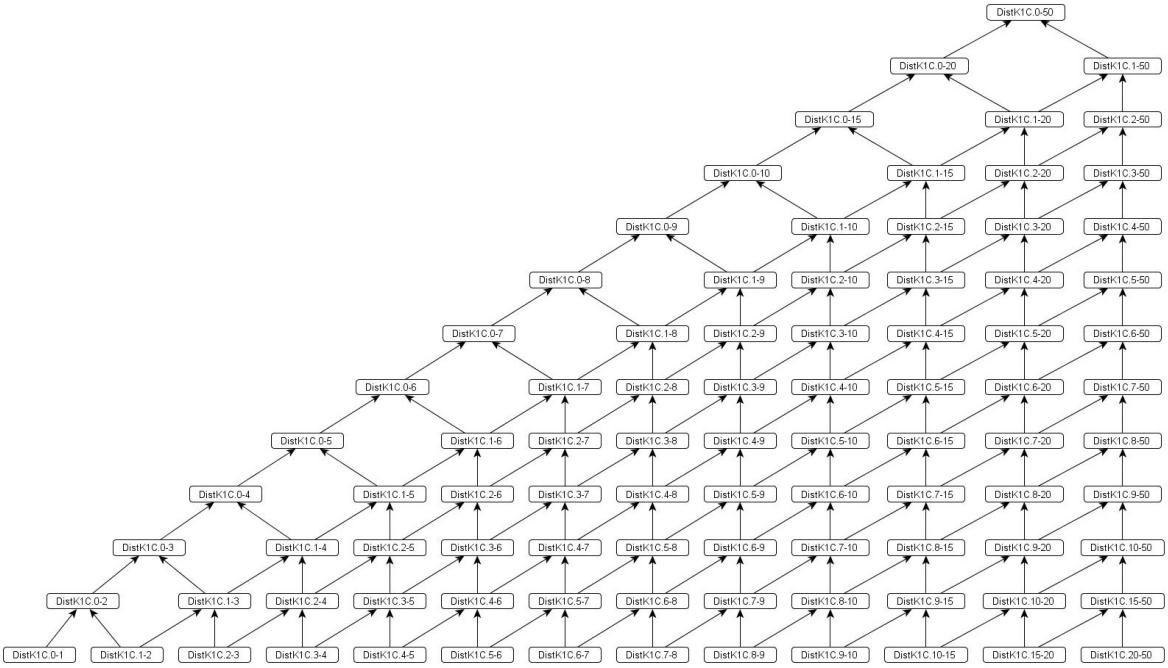
Slika 56. a) Hierarhija akcij in b) hierarhija vlog v domeni 3vs2 Keepaway.

Ker je domenski prostor opisan samo s 13 atributi (predstavljenimi na strani 12), opišemo lastnosti okolja s hierarhijami značilk za vse atribute. Imena hierarhij značilk ustrezajo imenom izhodiščnih atributov:

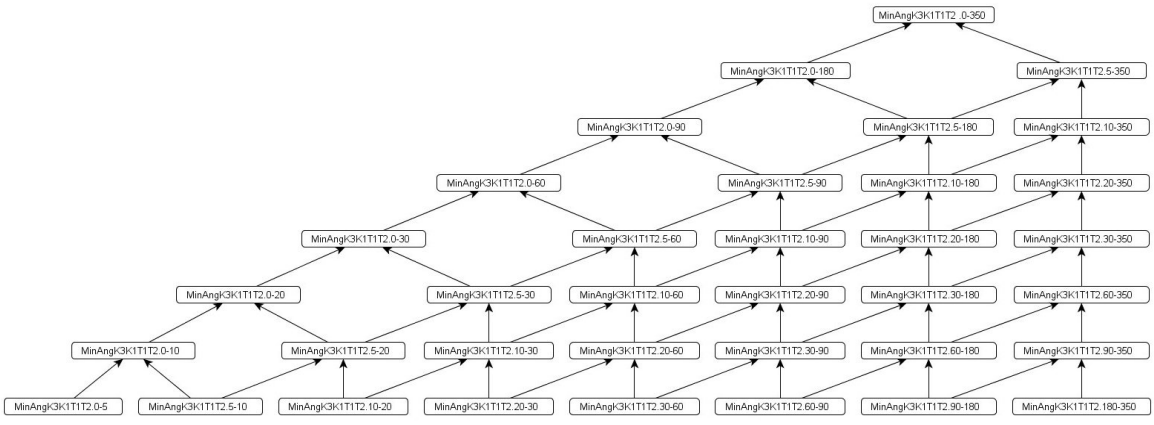
- $H_{DistK1C}$(*dist(K1, C)*, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 50}),
- $H_{DistK2C}$(*dist(K2, C)*, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 50}),
- $H_{DistK3C}$(*dist(K3, C)*, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 50}),
- $H_{DistT1C}$(*dist(T1, C)*, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 50}),
- $H_{DistT2C}$(*dist(T2, C)*, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 50}),
- $H_{DistK1K2}$(*dist(K1, K2)*, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 50}),
- $H_{DistK1K3}$(*dist(K1, K3)*, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 50}),

- $H_{DistK1T1}$(*dist(K1, T1)*, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 50}),
- $H_{DistK1T2}$(*dist(K1, T2)*, {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20, 50}),
- $H_{MinDistK2T1T2}$(*min(dist(K2, T1), dist(K2, T2))*, {0,1,2,3,4,5,6,7,8,9,10,15,20,50}),
- $H_{MinDistK3T1T2}$(*min(dist(K3, T1), dist(K3, T2))*, {0,1,2,3,4,5,6,7,8,9,10,15,20,50}),
- $H_{MinAngK2K1T1T2}$(*min(ang(K2, K1, T1), ang(K2, K1, T2))*, {0, 5, 10, 20, 30, 60, 90, 180, 350}),
- $H_{MinAngK3K1T1T2}$(*min(ang(K3, K1, T1), ang(K3, K1, T2))*, {0, 5, 10, 20, 30, 60, 90, 180, 350}).

Primer hierarhije značilk za atribut *dist(K1, C)* je predstavljen na sliki 57, primer hierarhije značilk za atribut *min(ang(K3, K1, T1), ang(K3, K1, T2))* pa na sliki 58. Značilka DistK1C.4-10 je resnična, če je $4 \leq dist(K1, C) < 10$. Podobno je značilka MinAngK3K1T1T2.10-30 resnična, če je $10 \leq min(ang(K3, K1, T1), ang(K3, K1, T2)) < 30$.

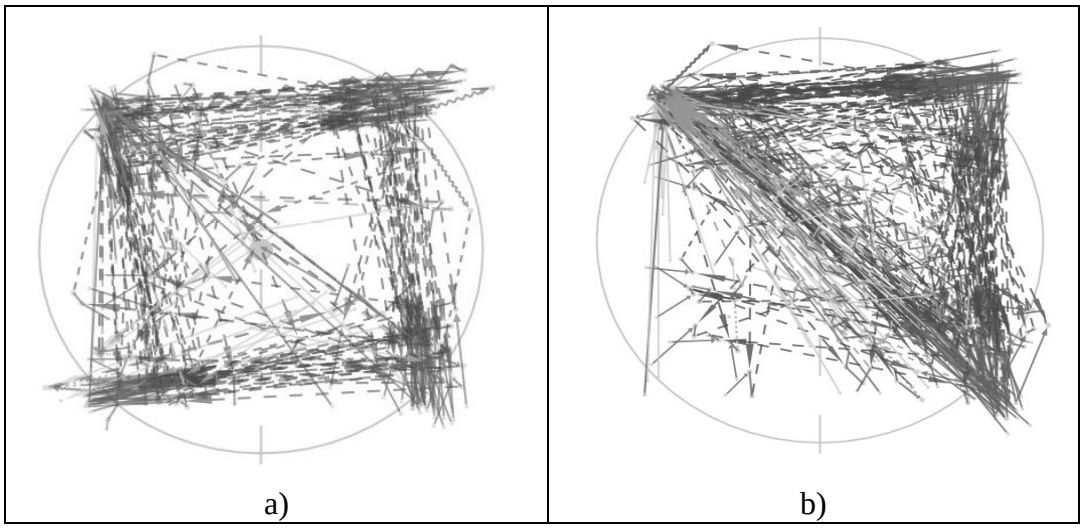


Slika 57. Primer hierarhije značilk $H_{DistK1C}$.



Slika 58. Primer hierarhije značilk $H_{MinAngK3K1T1T2}$.

Ker je vsaka igra v domeni 3vs2 Keepaway razmeroma kratka (manj kot 20s), je vhod v MASDA zaporedje agentnih akcij, ki ustreza večjemu številu iger. Večje število iger tudi zagotavlja ponavljanje podobnih situacij, kar je eden bistvenih pogojev za uspešno modeliranje z MASDA. Pri izrisu AG uporabimo za pozicije vozlišč v AG tako kot v domeni RoboCup kar pozicije agentov na igrišču. Na sliki 59a je prikazan AG za strategijo *hand*, na sliki 59b pa AG za strategijo *rand*.



Slika 59. Primeri AG za a) strategijo *hand* in b) strategijo *rand*.

Ker je domena 3vs2 Keepaway različna od domene RoboCup, moramo določiti tudi novo funkcijo razdalje med vozlišči v AAG. Pri tem želimo ohraniti opise agentnih vlog in akcij, saj jih ne želimo abstrahirati, ker iščemo opise akcijskih konceptov, ki ustrezajo originalnim agentnim akcijam in vlogam. Zato definiramo funkcijo razdalje med vozlišči kot:

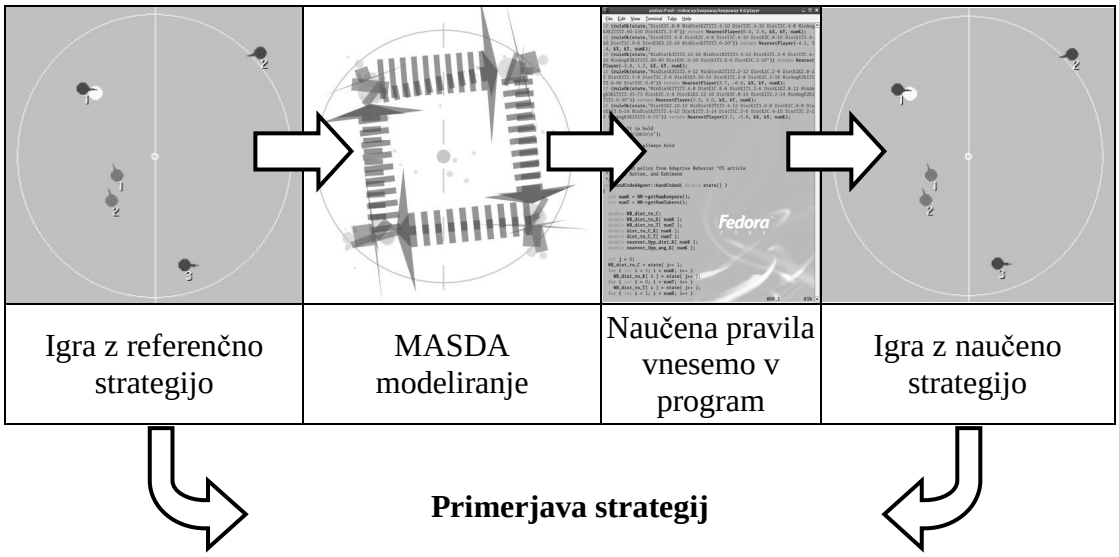
$$dist(a,b) = \begin{cases} \overline{dist_{pos}(a,b)}\text{; če } (\overline{dist_{Hrole}(a,b)} + \overline{dist_{Haction}(a,b)}) = 0 \\ \infty\text{; sicer} \end{cases}$$

Ker se Keepaway okolje izvaja znotraj RoboCup simulatorja, določimo razdaljo med pozicijami agentov enako kot v RoboCup domeni. Domenska razdalja je torej definirana kot evklidska razdalja med pozicijami agentov za primere iz vozlišč a in b:

$$\overline{dist_{pos}(a,b)} = \sqrt{(a.pos.x - b.pos.x)^2 + (a.pos.y - b.pos.y)^2}$$

8.2 Izvedba meritev

Ker je bil naš cilj posnemanje poznanih strategij, smo testiranje izvedli na sledeči način. Za poznano strategijo smo generirali ustrezne igre. Vhod v MASDA je predstavljalo zaporedje akcij iger s poznano strategijo. Z MASDA smo nato generirali opise konceptov makroakcij. Nato smo dobljena pravila in zaporedja akcijskih konceptov ustrezno vgradili v program, ki krmili agente v 3vs2 Keepaway okolju. Nato smo pognali igro z našimi agenti in dobili posnetek igre z naučeno strategijo. Opisani postopek je prikazan na sliki 60.



Slika 60. Postopek generiranja strategije, ki posnema referenčno.

Pri tem smo testirali naslednje strategije:

- dve referenčni strategiji, ki sta priloženi okolju 3vs2 Keepaway (predstavljene na strani 13): *hand* in *rand*,
- strategijo *hand3-6-9*, ki določa različno strategijo za vsakega igralca posebej.

8.2.1 Testiranje referenčnih strategij

Ker imamo v Keepaway okolju na voljo referenčne strategije, smo si za prvi cilj zadali modeliranje teh strategiji. Poznavanje referenčnih strategij nam namreč omogoča, da s primerjavo naučenih in poznanih strategij ocenimo uspešnosti našega pristopa modeliranja večagentnih sistemov.

Primerjavo naučenih in referenčnih strategij smo nato izvedli na štiri načine:

- Z merjenjem povprečnega trajanja epizode, ki omogoča numerično primerjavo uspešnosti referenčnih in naučenih strategij.
- Z merjenjem ujemanja akcij, ki so jih izbrale referenčne in naučene strategije. Tako merjenje pove v kolikšni meri se izvajanje naučene strategije dejansko ujema z referenčno.
- Z analizo dobljenih pravil, ki nam omogoča vpogled v zakonitosti delovanja in omogoča primerjavo z zakonitostmi referenčnih strategij.
- Z vizualno oceno poteka igre kot pomožno oceno, ki upošteva človeško zaznavo pri oceni podobnosti dveh strategij igre.

S postopkom opisanim na sliki 60 smo zgenerirali strategije, ki posnemajo referenčni strategiji *hand* in *rand*. Pri tem smo spreminjali vrednost parametra abstrakcije (1.5, 2, 3, 4 in 5) ter dolžino generiranih konceptov makroakcij (od 1 do 4). Poskuse smo ponovili za vse kombinacije parametra abstrakcije in dolžine konceptov makroakcij. Vsak poskus smo označili z "*abs X, len Y*", kjer X predstavlja vrednost parametra abstrakcije, Y pa dolžino konceptov makroakcij. Število generiranih konceptov makroakcij za naučeno strategijo *hand* je predstavljeno v preglednici 23. Rezultati

kažejo, da se v povprečju število naučenih makroakcij veča z večanjem abstrakcije kot tudi z večanjem dolžine makroakcij.

	dolžina makroakcije				
	len 1	len 2	len 3	len 4	povprečje
abs 1.5	30	28	23	14	23.75
abs 2	18	25	31	28	25.5
abs 3	18	25	30	35	27
abs 4	21	27	31	35	28.5
abs 5	20	28	33	39	30
abs 6	22	29	33	40	31
povprečje	21.5	27	30.22	31.83	27.63

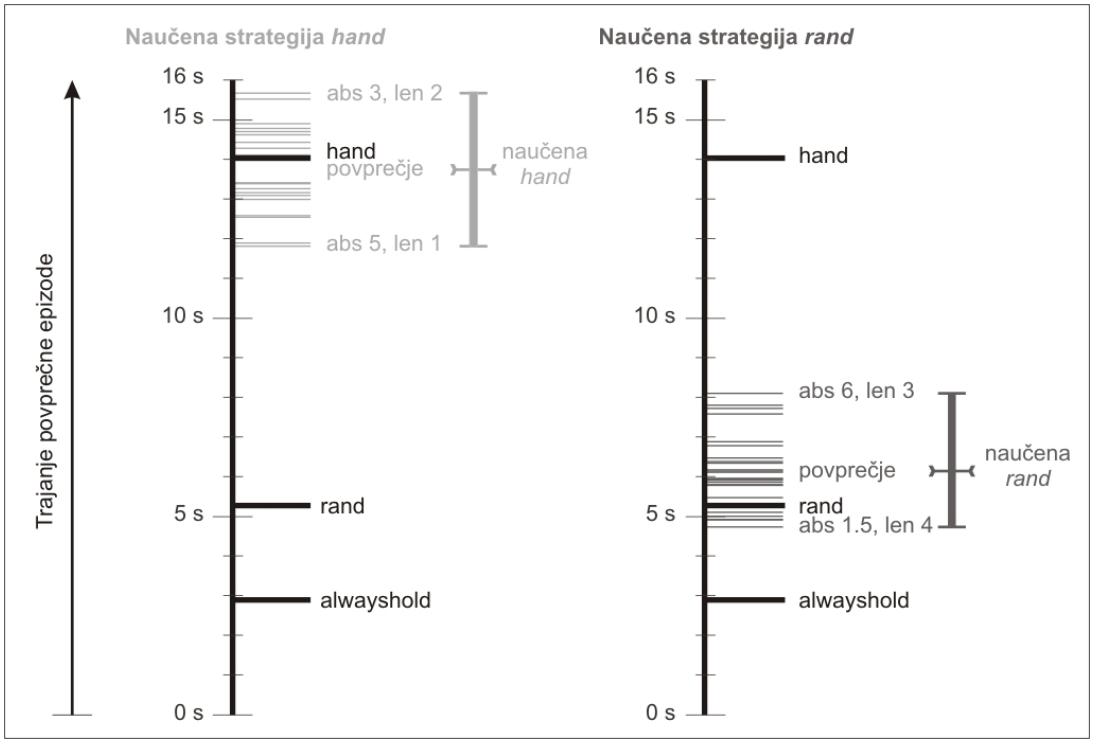
Preglednica 23. Število generiranih makroakcij naučene *hand* strategije za različne vrednosti abstrakcije in dolžine makroakcij.

Ker predstavlja izhod iz MASDA nabor konceptov makroakcij, smo pri implementaciji uporabili le pripadajoča pravila in informacijo o zaporedju akcijskih konceptov. Opis agentne strategije tako sestavlja seznam makroakcij, ki jih sestavljajo zaporedja pravil, ki ustrezajo zaporedju akcijskih konceptov. Seznam je urejen po naraščajočem številu pozitivnih učnih primerov, ki so bili uporabljeni za generiranje posameznih konceptov makroakcij. Izbira primerne agentne akcije poteka po postopku, ki je kot algoritem predstavljen na sliki 61. Postopek najprej poišče primerno makroakcijo. To stori tako, da po vrsti preverja pravila začetnih akcijskih konceptov posameznih makroakcij. Če je pravilo za dano stanje izpolnjeno se začne izvajati izbrana makroakcija, sicer se izvede akcija *hold*. Izbrana makroakcija se izvaja, dokler pravila posameznih akcijskih konceptov te makroakcije ustrezajo stanju. Ko nadaljnje izvajanje makroakcije ni več mogoče, poišče postopek nov primeren začetek makroakcije.

```
Vhod:
- dolžina makroakcij: maxLen
- število makroakcij: numMacro
- seznam makroakcij: macro[numMacro][maxLen]
- trenutno izvajana makroakcija: currMacro
- trenutni akcija v v makroakciji: currAction

Algoritem:
selectedAction = NULL // akcije še ni izbrana
if (currMacro >= 0) {
  if (RuleTrue(macro[currMacro][currAction].rule, state)) {
    // prejšna akcija je izbrana tudi tokrat
    selectedAction = macro[currMacro][currAction].action;
  }
  currAction++; // izberi naslednjo akcijo iz makroakcije
  if (currAction < maxLen) {
    if (RuleTrue(macro[currMacro][currAction].rule, state)) {
      // izbrana je naslednja akcija iz trenutne makroakcije
      selectedAction = macro[currMacro][currAction].action;
    }
  }
  else {
    // v trenutni makroakciji ne ustreza nobeno pravilo
    currMacro = -1;
  }
}
if (selectedAction == NULL) { // akcija še ni izbrana
  for (i=0; i<numMacro; i++) {
    // preveri pravila, ki ustrezajo začetkom makroakcij
    if (RuleTrue(macro[i][0].rule, state)) {
      currMacro = i; // izberi kot trenutno makroakcijo
      currAction = 0;
      selectedAction = macro[i][0].action; // izbrana nova akcija
    }
  }
}
if (selectedAction == NULL) selectedAction = actionHold

Izhod:
- izbrana agentna akcija: selectedAction
```

Slika 61. Algoritem, ki izbira agentne akcije.

Za vsak poskus smo v agentni program vgradili dobljen opis agentne strategije in pognali igro za 30 minut ter izmerili povprečno trajanje epizode. Izmerjeno trajanje povprečnih epizod naučenih in referenčnih strategij je predstavljeno na sliki 62, kjer levi stolpec prikazuje rezultate naučenih strategij, ki posnemajo *hand* strategijo, desni stolpec pa rezultate naučenih strategij, ki posnemajo *rand* strategijo. Višina vodoravnih črt ustreza trajanju povprečne epizode naučenih strategij pri različnih parametrih. Debele črne črte v obeh stolpcih prikazujejo trajanje povprečne epizode referenčnih strategij.



Slika 62. Trajanje povprečne epizode, kjer levi stolpec prikazuje rezultate naučenih *hand* strategij, desni stolpec pa rezultate naučenih *rand* strategij. Različne vodoravne črte ustrezajo časom naučenih strategij pri različnih parametrih. Debele črne črte v obeh stolpcih prikazujejo trajanje povprečne epizode referenčnih strategij.

Rezultati meritev trajanja povprečne epizode kažejo, da različne abstrakcije in različne dolžine konceptov makroakcij izrazito vplivajo na dobljene rezultate. Razpon trajanja povprečne epizode za najboljši in najslabši primer za naučeno strategijo *hand* je 3,86 s, za naučeno strategijo *rand* pa 3,38 s.

Za obe referenčni strategiji velja, da je povprečna vrednost trajanja epizode referenčnih strategij med najboljšo in najslabšo povprečno epizode naučenih strategij. Za naučeno strategijo *hand* je povprečna vrednost poskusov za 0,28 s manjša od povprečnega trajanja epizode za referenčno *hand* strategijo. Za naučeno *rand* strategijo je povprečna vrednost poskusov za 0,87 s večja od povprečnega trajanja epizode za referenčno *rand* strategijo.

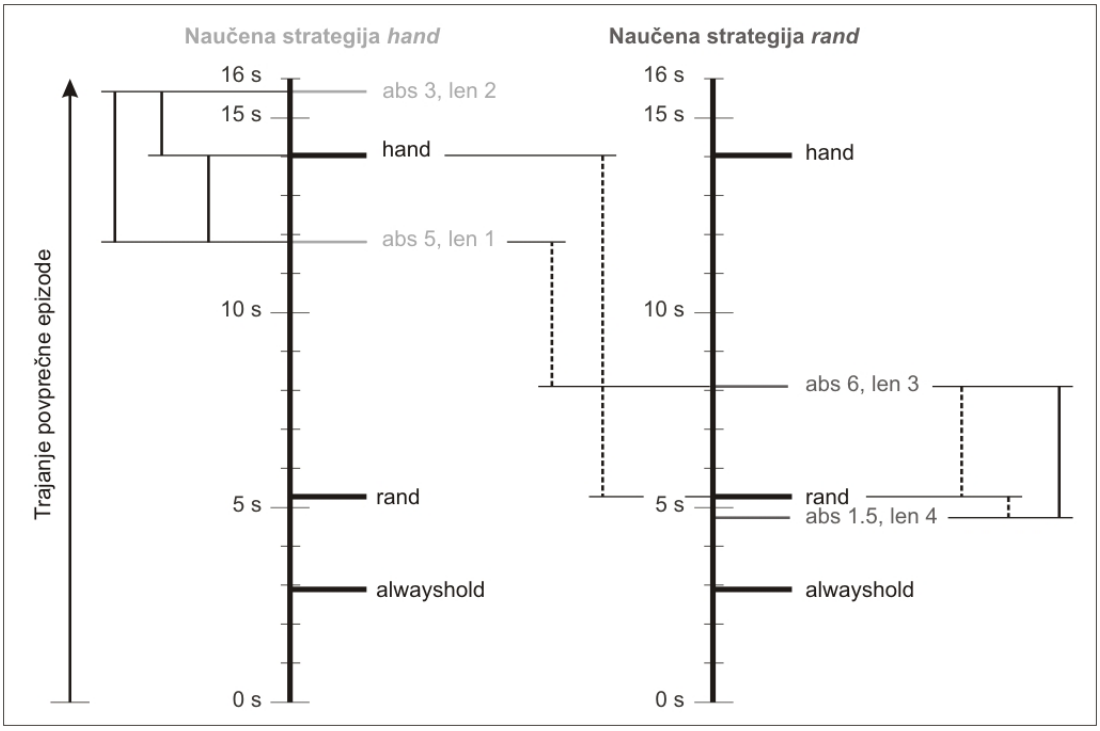
Razlaga, zakaj je povprečna uspešnost naučene strategija *rand* boljša od referenčne, medtem, ko je od povprečna uspešnosti naučene strategije *hand* slabša od referenčne, je posledica dejstva, da se MASDA uči iz zaporedja opravljenih agentnih akcij. Ker je referenčna *rand* strategija relativno slaba, je bil večji del podaj neuspešen. Ker je vhod v MASDA predstavljalo zaporedje akcij, v katerem je bila *rand* strategija vedno uspešna, smo se dejansko učili boljše strategije od referenčne *rand* strategije. Ker je delež neuspešnih podaj v *hand* strategiji manjši, smo se posledično učili strategije, ki je bolj primerljiva s *hand* strategijo.

Primerjava rezultatov med naučeno *hand* in naučeno *rand* strategijo kaže, da se rezultati meritev posameznih strategij ne prekrivajo. To dejstvo nakazuje, da sta naučeni strategiji med seboj bistveno različni, saj je razlika med najslabšim

rezultatom naučene *hand* strategije in najboljšim rezultatom naučene *rand* strategije kar 3,7 s, kar je približno celotni razpon meritev za posamezno strategijo. Za točnejšo primerjavo rezultatov smo opravili meritve statistične signifikantnosti z Wilcoxonovim testom predznačenih rangov po parih (angl. *Wilcoxon Matched-Pairs Signed-Ranks Test*) pri signifikančni meji 95%. V testih smo med seboj paroma primerjali najboljšo (*hand* abs 3, len 2) in najslabšo (*hand* abs 5, len 1) naučeno *hand* strategijo, najboljšo (*rand* abs 6, len 3) in najslabšo (*rand* abs 1, len 4) *rand* strategijo ter referenčni *hand* in *rand* strategiji. V preglednici 24 so predstavljene *p* vrednosti meritev signifikantnosti, grafično pa so rezultati predstavljeni na sliki 63, kjer smo za prikaz (ne)signifikantnosti uporabili notacijo, ki jo predlaga Demšar [32]. Tako so nesignifikantne razlike med testi prikazane kot neprekinjene navpične črte, signifikantne razlike pa kot črtkane navpične črte.

strategija 1	strategija 2	vrednost *p*
hand abs 3, len 2	*hand* abs 5, len 1	0.1419
hand abs 3, len 2	referenčna *hand*	0.8613
hand abs 5, len 1	referenčna *hand*	0.236
referenčna *hand*	referenčna *rand*	0.000000000001708
hand abs 5, len 1	*rand* abs 6, len 3	0.00008636
rand abs 6, len 3	*rand* abs 1, len 4	0.056
rand abs 6, len 3	referenčna *rand*	0.00000004373
rand abs 1, len 4	referenčna *rand*	0.0008286

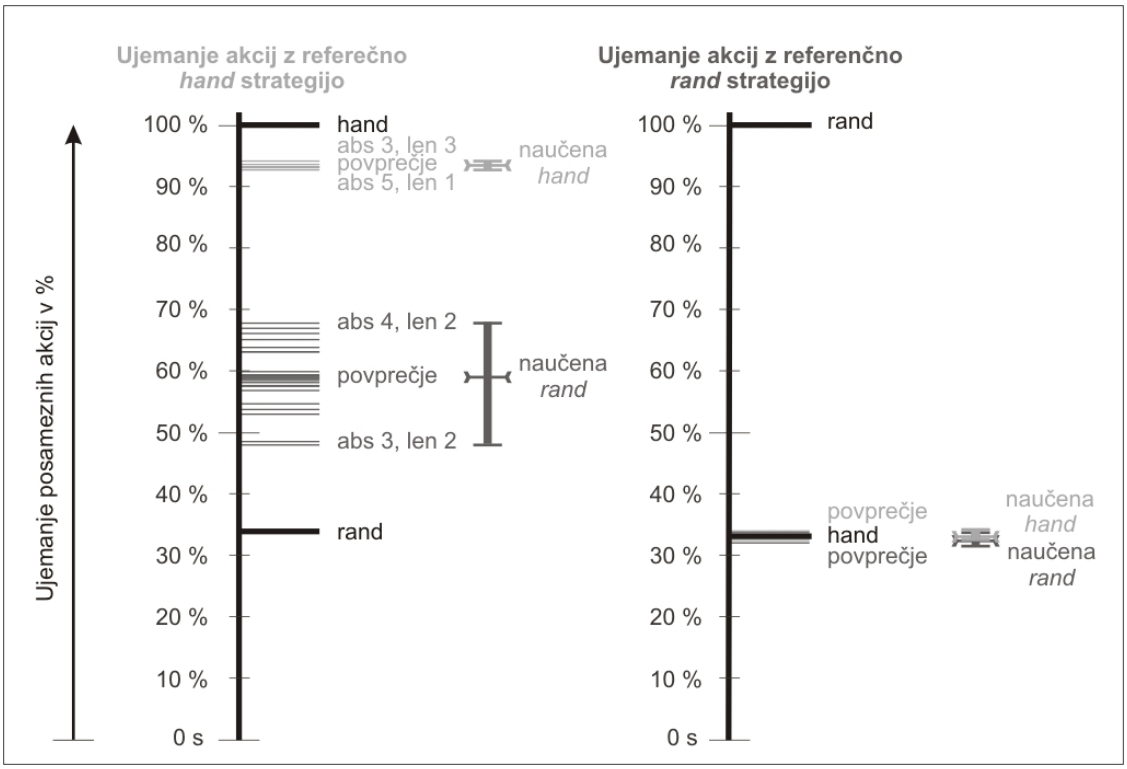
Preglednica 24. Rezultati statistične ocene signifikantnosti z Wilcoxonovim testom predznačenih rangov po parih. Sive vrstice predstavljajo statistično signifikantne razlike med strategijo 1 in strategijo 2.



Slika 63. Grafično predstavljeni testi statistične signifikantnosti. Polne navpične črte, ki povezujejo pare testov, predstavljajo nesignifikantne razlike, črtkane pa signifikantne razlike med testi.

Rezultati kažejo, da se najboljša in najslabša naučena *hand* strategija ter referenčna *hand* strategija med seboj paroma signifikantno ne razlikujejo, kar potrjuje uspešnost učenja *hand* strategij. Signifikantni razliki smo izmerili med najslabšo naučeno *hand* in najboljšo naučeno *rand* ter med referenčno *hand* in referenčno *rand* strategijo, kar kaže na bistveno različnost (naučenih) *hand* in *rand* strategij. Testi kažejo tudi na nesignifikantne razlike med najboljšo in najslabšo naučeno *rand* strategijo. Na prvi pogled pa sta presenetljivi signifikantni razliki med najboljšo naučeno *rand* in referenčno *rand* ter najslabšo naučeno *rand* in referenčno *rand* strategijo. Razlaga za take rezultate leži v implementacije *rand* strategije, ki uporablja generator naključnih števil, ki ga s pravili v naučenih *rand* strategijah ne moremo dobro posnemati. Zaradi nesignifikantnih razlik med najboljšo in najslabšo naučeno *hand* strategijo ter najboljšo in najslabšo naučeno *rand* strategijo, ni mogoče sklepati na optimalno stopnjo abstrakcije, niti na optimalno dolžino makroakcije.

Pri izvajanju poskusov smo izmerili tudi ujemanje odigranih akcij z akcijami, ki bi jih izbrali referenčni strategiji. Vsako akcijo, ki jo je izbral agent na podlagi pravil naučenih strategij, smo preverili, ali ustreza izbrani akciji referenčnih strategij *hand* in *rand*. Rezultati teh meritev so podani v preglednici 25 in na sliki 64, kjer svetlo sive vodoravne črte ustrezajo primerom naučenih *hand* strategij, temno sive pa primerom naučenih *rand* strategij.



Slika 64. Odstotek ujemanja odigranih akcij naučenih strategij z referenčnimi.

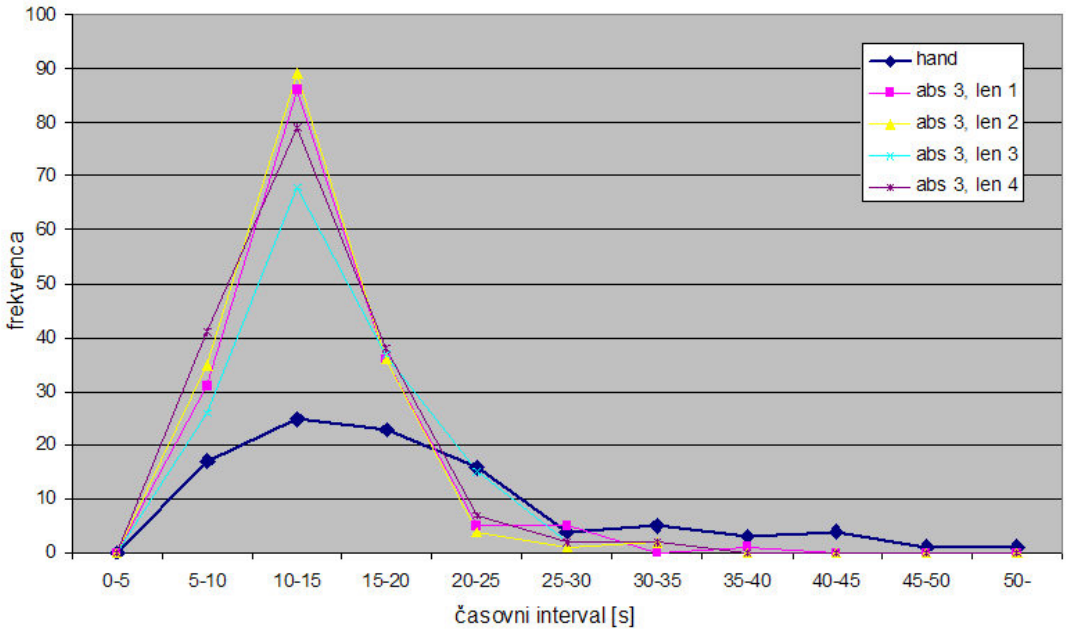
Primerjana strategija	Ujemanje akcij z referenčno strategijo *hand*	Ujemanje akcij z referenčno strategijo *rand*
referenčna *hand*	100 %	33,07 %
najboljša naučena *hand*	94,14 %	33,93 %
povprečna naučena *hand*	93,24 %	33,00 %
najslabša naučena *hand*	92,69 %	32,41 %
referenčna *rand*	33,85 %	100 %
najboljša naučena *rand*	67,76 %	33,72 %
povprečna naučena *rand*	58,90 %	32,83 %
najslabša naučena *rand*	47,93 %	31,33 %

Preglednica 25. Odstotek ujemanja odigranih akcij naučenih strategij z referenčnimi.

Rezultati ujemanja akcij kažejo, da je ujemanje naučenih *hand* strategij z referenčno strategijo *hand* zelo visoko, saj najslabša naučena *hand* strategija doseže kar 92,69 % ujemanje akcij. Ujemanje naučene strategije *rand* z referenčno *hand* je slabše, saj najboljša naučena *rand* strategija doseže 67,76 %, najslabša pa 47,93 % ujemanje. Rezultati kažejo, da naučene *hand* strategije dobro posnemajo obnašanje referenčne *hand* strategije, poleg tega je ponovno razvidno, da se uspešnost *hand* in *rand* strategij jasno loči med seboj.

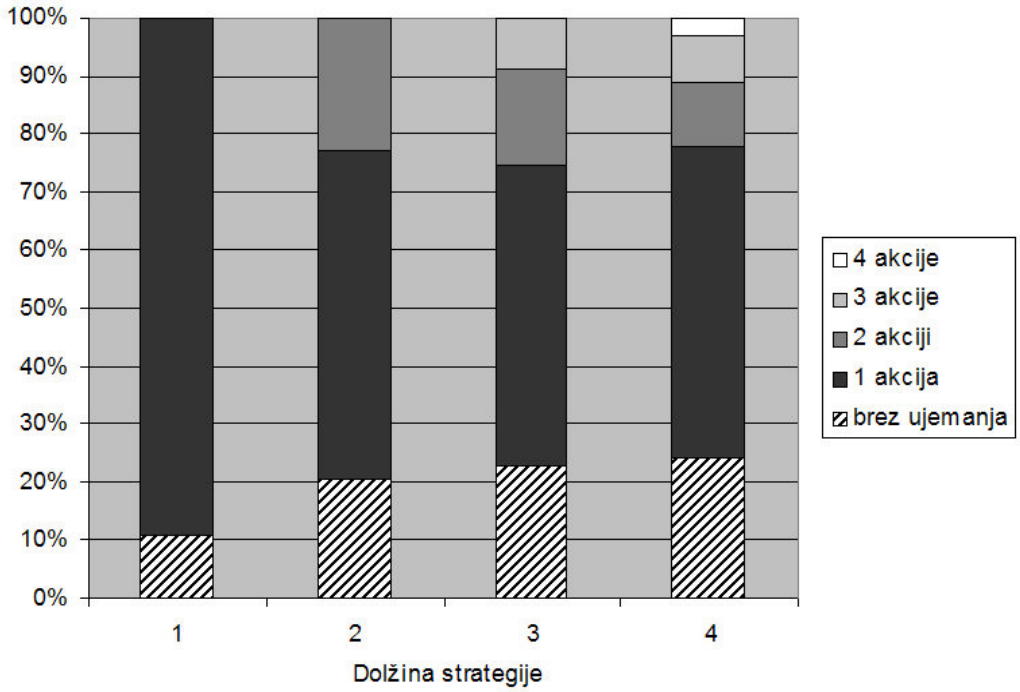
Ujemanje akcij z referenčno strategijo *rand* je v vseh primerih zelo podobno, saj se vsi rezultati gibljejo med 31,33 % in 33,93 %. Taki rezultati so pričakovani, saj referenčna strategija *rand* naključno izbira akcije, zato je njihovo ujemanje z deterministčnim postopkom izbire akcij v naših agentih tudi naključno in se ujema v pričakovanih 1/3 vseh primerov.

Pri analizi rezultatov smo naleteli na zanimivo porazdelitev meritev trajanja epizod. Standardna deviacija trajanja povprečne epizode za referenčno *hand* strategijo je zelo visoka, saj znaša kar 10,60. Standardna deviacija za meritve naučenih *hand* strategij pri različnih abstrakcijah in različnih dolžinah strategij je med 4,31 in 4,97. Porazdelitve meritev referenčne *hand* strategije in naučenih *hand* strategij pri abstrakciji 3 in različnih dolžinah strategij so predstavljene na sliki 65. Ti rezultati kažejo, da naučene strategije opisujejo le najbolj pogosto obnašanje referenčne strategije, ne opisujejo pa ekstremnih primerov. Tako obnašanje je razložljivo s statističnim pristopom, ki ga uporablja MASDA, saj za učenje uporabi le najbolj pogoste akcijske koncepte. Primeri ki ustrezajo ekstremnim časom posameznih epizod statistično niso dovolj pogosti in zato niso upoštevani pri MASDA modeliranju.



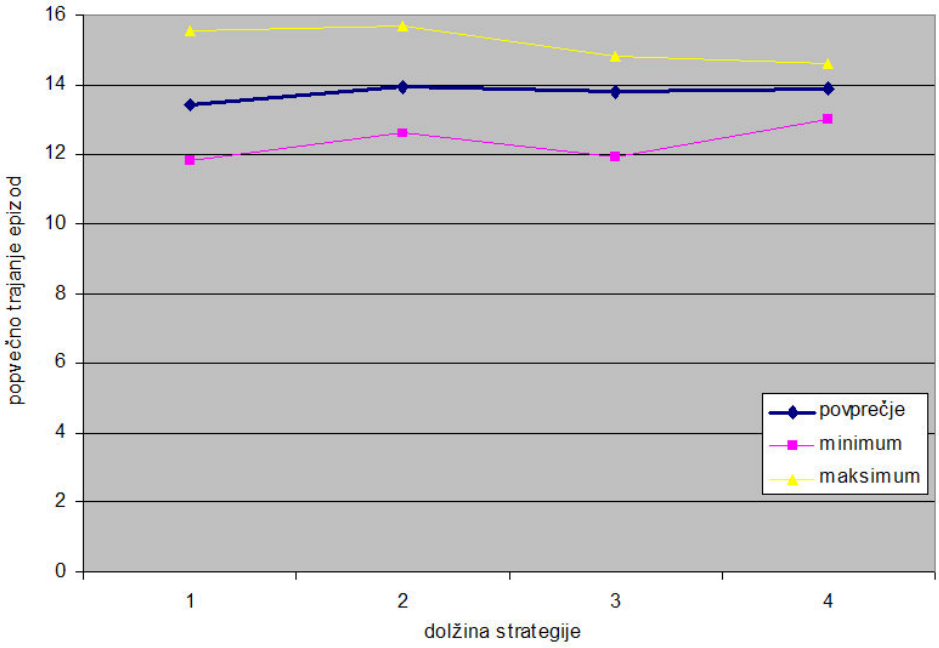
Slika 65. Porazdelitve meritev referenčne *hand* strategije in naučenih strategij.

Merili smo tudi delež dejansko izvajanih dolžin posameznih makroakcij. Meritve so predstavljene na sliki 66, kjer je za vsako dolžino strategije predstavljeno razmerje med dolžinami dejansko izvedenih makroakcij. Pri tem moramo poudarit, da graf prikazuje številčno in ne časovno razmerje med dolžinam izvajanih strategij. Tako je primer s 4 izvedenimi akcijami ustrezno daljši od primera z zgolj eno izvedeno akcijo oz. od primera brez ujemanja, ki traja zgolj en cikel.

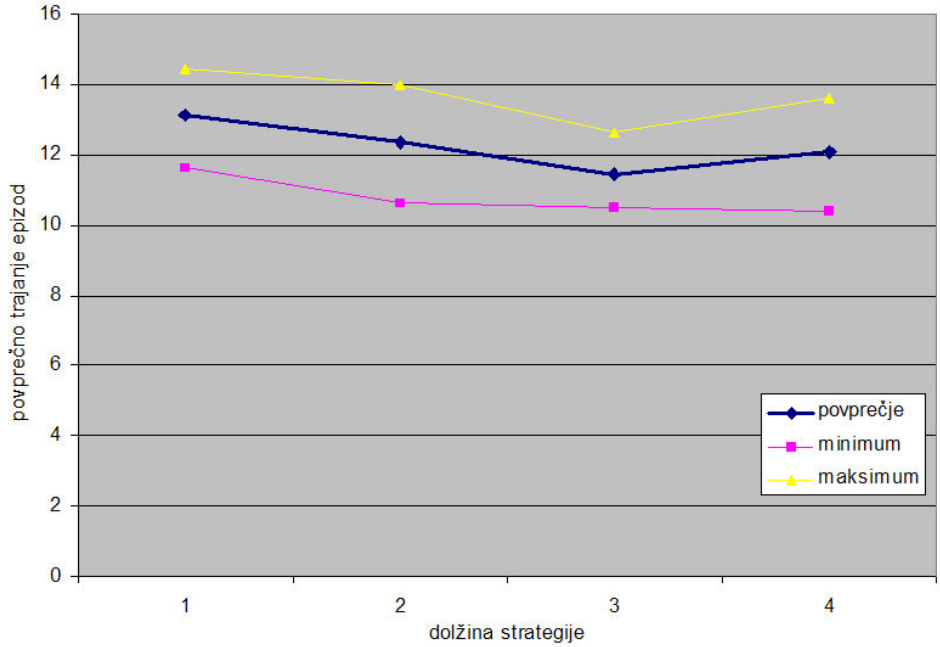


Slika 66. Povprečni delež dolžine dokončanih makroakcij za različne dolžine strategij.

Zanimalo nas je tudi, ali dolžina strategija vpliva na povprečno trajanje epizod. Na sliki 67 so predstavljeni razponi in povprečne vrednosti meritev različnih abstrakcij pri različnih dolžinah strategij. Ker je standardna deviacija meritev pri dolžini strategije 1 enaka 1,29, pri dolžini 2 enaka 1,17, pri dolžini 3 enaka 1,11 in pri dolžini 4 enaka 0,65, meritve ne potrjujejo hipoteze, da dolžina strategije vpliva na povprečno trajanje epizod.



Slika 67. Povprečno trajanje epizod glede na dolžino strategije.



Slika 68. Povprečno trajanje epizod glede na dolžino strategije pri omejenem številu uporabljenih konceptov makroakcij na 20.

Hipotezo, da dolžina strategije vpliva na povprečno trajanje epizod smo nadalje preverili tudi z omejitvijo števila uporabljenih konceptov makroakcij. Vse meritve naučene *hand* strategije smo ponovno pognali, pri čemer smo omejili število uporabljenih konceptov makroakcij na 20. Na sliki 68 so predstavljeni rezultati meritev z omejenih številom uporabljenih konceptov na 20. Rezultati kažejo, da so pri fiksnem številu uporabljenih konceptov makroakcij najbolj uspešne strategije dolžine ena. Taki rezultati so pričakovani, saj se skladajo z dejstvom, da je referenčna *hand* strategija prav tako dolžine ena.

Za človeško analizo pravil smo izbrali pravila, ki opisujejo naučeno *hand* strategijo. Strategija *hand* je preprosta, toda zelo učinkovita strategija. Njeno delovanje je v obliki psevdo kode predstavljeno na sliki 69. Opisno je njeno delovanje sledeče: *Če je T1 od K1 oddaljen več kot 5 m, potem izbere akcijo Hold. Sicer oceni kot in razdaljo med soigralci ter napadalci in poda najbolj odkritemu. Če noben soigralec ni dovolj odkrit, izbere akcijo Hold*.

```
// če so napadalci oddaljeni več kot 5 m, potem zadrži žogo
if (DistK1T1 > 5)
  return Hold

// razmerje med razdaljo in kotom branilca do žoge
dist_weight = 4.0;

// oceni posamezno akcijo
scores[0] = 90; // akcija Hold ima fiksno oceno
scores[1] = dist_weight * MinDistK2T1T2 + MinAngK2K1T1T2
scores[2] = dist_weight * MinDistK3T1T2 + MinAngK3K1T1T2

// Poišči akcijo z najvišjo oceno
best = 0;
for (i=1; i<3; i++ )
  if (scores[i] > scores[best])
    best = i

// izvedi akcijo, ki ima najvišjo oceno
if (best == 0)
  return Hold // noben branilec ni dovolj odkrit
else if (best == 1)
  return PassK2 // branilec K2 je najbolj odkrit
else
  return PassK3 // branilec K3 je najbolj odkrit
```

Slika 69. Psevdo koda, ki predstavlja strategijo *hand*.

Referenčni strategiji *hand* in *rand* se lahko modelirata le s tremi akcijskimi koncepti, kjer vsak koncept predstavlja le posamezno Keepaway akcijo: Hold, PassK2 in PassK3. Ker v procesu abstrakcije ne združujemo vozlišč, ki predstavljajo različne opise akcij in vlog, lahko z višanjem parametra abstrakcije zgradimo AAG_{abs}, ki vsebuje le povezave, ki predstavljajo akcijski koncept za vsako od treh Keepaway akcij.

Z željo, da bi analizirali pravila naučene *hand* strategije, smo iz posnetka iger s *hand* strategijo zgradili ABS_{100}, ki se z večanjem parametra abstrakcije ni več spreminjal. Nato smo inducirali pravila, ki opisujejo posamezni akcijski koncept v ABS_{100}.

Pravila so predstavljena v preglednici 26, kjer so s krepkim tiskom označeni pogoji, ki neposredno sledijo iz definicije strategije. Trajanje povprečne epizode za igre, v kateri uporabljajo agenti predstavljena pravila, je 11,46 s.

Akcija	Število učnih primerov		Pravila
	+	-	
Hold	**827**	**2869**	**DistK1T1.6-50** $\wedge$ DistK1C.2-15
Pass-K2	**342**	**1406**	**MinDistK2T1T2.9-20** $\wedge$ **DistK1T1.1-6** $\wedge$ **MinAngK2K1T1T2.30-350** $\wedge$ **MinDistK3T1T2.3-50** $\wedge$ DistK1T2.2-9
Pass-K3	**459**	**1289**	**MinDistK2T1T2.3-10** $\wedge$ **DistK1T1.1-6** $\wedge$ **MinAngK2K1T1T2.0-30** $\wedge$ DistT2C.2-50 $\wedge$ DistK1K2.7-50

Preglednica 26. Pravila, ki ustrezajo konceptom agentih akcij v AAG_{100}.

Pravilo za akcijo Hold določa, da je branilec K1 od najbližjega napadalca T1 oddaljen 6 ali več metrov ter da je K1 vsaj za 2m oddaljen od središča igrišča. Prvi pogoj neposredno sledi iz definicije strategije, drugi pa je posledica dejstva, da je za branilce bolje, da se gibljejo čimbolj oddaljeni od središča igrišča, saj imajo tako večje možnosti za kasnejše uspešne podaje.

Pravilo za akcijo Pass-K2 določa, da je K2 vsaj za 9m oddaljen od najbližjih napadalcev, da je T1 v 6 metrski okolici od K1, da je kot med žogo, K2 in najbližjim napadalcem vsaj 30 stopinj, da je razdalja med K3 in najbližjim napadalcem skoraj poljubna in da je T2 v bližini K1, vendar za T1. Drugi in tretji pogoj neposredno sledita iz definicije strategije, saj je T1 dovolj blizu (DistK1T1 < 6m), poleg tega pa je prejemnik žoge K2 dovolj odkrit (MinAngK2K1T1T2 > 30 stopinj). Prvi pogoj je posledica dejstva, da so podaje uspešnejše, če je prejemnik žoge bolj oddaljen od napadalcev. Četrti pogoj zgolj pove, da se je K3 gibal v skoraj poljubni razdalji od napadalcev. Zadnji pogoj pa je posledica dejstva, da je strategija napadalcev vedno neposredno napadati branilca z žogo K1. Drugi branilec T2 je zato že zaradi njegove vloge vedno bolj oddaljen od K1 kot prvi branilce T1.

Pravilo za akcijo Pass-K3 določa, da je K2 od najbližjih branilcev oddaljen od 3 do 10 metrov, da je T1 v 6 metrski okolici od K1, da je kot med K2, žogo in najbližjim branilcem manj kot 30 stopinj, da je T2 od središča oddaljen vsaj 2m in da je razdalja med K1 in K2 večja od 7m. Drugi in tretji pogoj ponovno sledita iz definicije strategije, saj je T1 dovolj blizu (DistK1T1 < 6m) vendar pa K2 ni dovolj odkrit (MinAngK2K1T1T2 < 30 stopinj). Ker K2 ni dovolj odkrit je posledično bolj odkrit prejemnik žoge K3. Ostali pogoji so zgolj posledica očitnih dejstev. Prvi pogoj pravi, da je K2 giblje vsaj malo oddaljen od napadalcev, četrti da je T2 od središča oddaljen vsaj 2m, kar je posledica dejstva, da hiti proti K1, in peti, da je razdalja med K1 in K2 velika, kar je skladno s pričakovanjem, da so podaje med bolj oddaljenimi branilci uspešnejše.

Vizualno oceno podobnosti naučenih in referenčnih strategij lahko preverimo z gledanjem posnetih iger. V preglednici 27 so podani internetni naslovi posnetkov iger za različne referenčne in naučene strategije.

Referenčne strategije	Naučene strategije
strategija *Hand* http://dis.ijs.si/andraz/phd/3vs2-hand.avi	naučena strategija *Hand* http://dis.ijs.si/andraz/phd/hand-learn.avi
strategija *Rand* http://dis.ijs.si/andraz/phd/3vs2-rand.avi	naučena strategija *Rand* http://dis.ijs.si/andraz/phd/rand-learn.avi

Preglednica 27. Internetni naslovi posnetih iger z referenčnimi in naučenimi strategijami.

Ogled posnetkov pokaže vizualno podobnost igre med referenčno igro in igro z naučeno strategijo. Ujemanje je mogoče zaznati za igre naučenih strategij obeh referenčnih strategij. Če primerjamo igri naučenih *hand* in *rand* strategij med seboj, se ponovno zazna jasna razlika v načinu igre različnih strategij.

8.2.2 Testiranje strategije *hand3-6-9*

Za potrebe preverjanja pristopa MASDA smo v okolju Keepaway implementirali novo strategijo imenovano *hand3-6-9*, ki določa različno strategijo za vsakega igralca posebej. Ker v osnovi okolje Keepaway naključno premeša agente in ne omogoča enoličnega oštevilčenja igralcev, smo to možnost dodali v okolje. Za namenom, da določimo strategije za vsakega agenta posebej, smo vloge agentov ustrezno spremenili v fiksne vloge, ki ustrezajo številki igralca.

Nova strategija je podobna strategiji *hand*, pri tem da na novo določi najmanjšo bližino napadalcev, pri kateri branilec K1 poda žogo najbolj odkritemu soigralcu. Branilec številka 1 ima najmanjšo bližino napadalcev pred podajo nastavljeno na 3m, branilec številka 2 na 6m in branilec številka 3 na 9m. Psevdo koda, ki opisuje delovanje strategije *hand3-6-9*, je predstavljena na sliki 70.

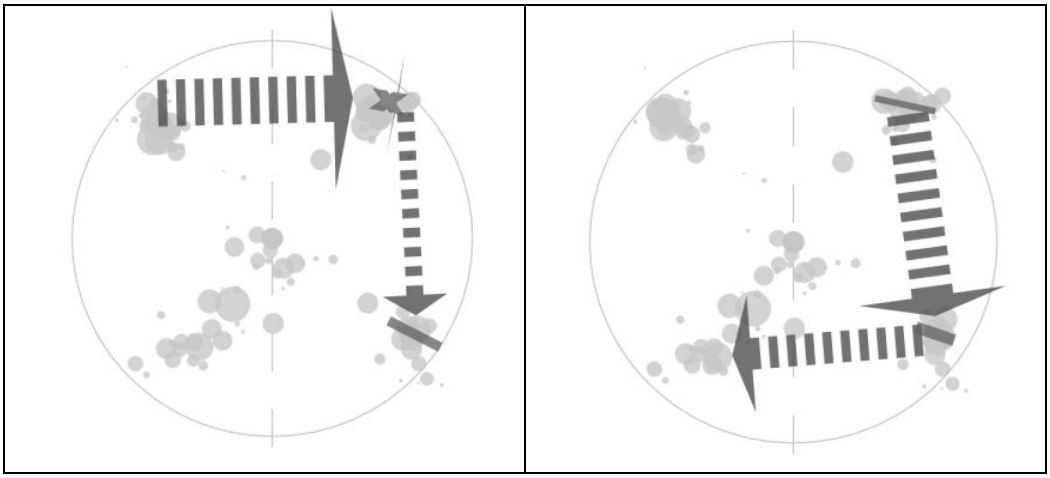
```
if (playerNum == 1) { // igralec številka 1
  if (DistK1T1 > 3) return Hold
}
else if (playerNum == 2) { // igralec številka 2
  if (DistK1T1 > 6) return Hold
}
else { // igralec številka 3
  if (DistK1T1 > 9) return Hold
}
// sicer podaj najbolj odprtemu branilcu
if (MinAngK2K1T1T2 > MinAngK3K1T1T2) {
  return PassK2
}
return PassK3
```

Slika 70. Psevdo koda, ki opisuje delovanje strategije *hand3-6-9*.

Namen testiranja je bil pokazati, da koncepti makroakcij resnično prikazujejo del večagentne strategije. Ker ima v strategiji *hand3-6-9* vsak branilec različno strategijo, bi se to moralo odražati v pravilih, ki opisujejo koncepte makroakcij. Če bi odkriti koncepti makroakcij opisovali več različnih agentnih strategij, potem lahko sklepamo,

da res predstavljajo del večagentne strategije. V ta namen smo v okolju Keepaway implementirali strategijo *hand3-6-9* in iz posnetka igre generirali pravila, ki opisujejo naučeno strategijo *hand3-6-9*.

Na sliki 71 sta prikazana primera dveh poti, ki predstavljata koncepta makroakcij dolžine 4. Ustrezna pravila, ki opisujejo prikazana koncepta makroakcij, so predstavljena v preglednicah 28 in 29, kjer so s krepkim tiskom označeni pogoji, ki neposredno nastopajo v definiciji strategije. Poti in pravila so bila generirana iz AAG_8, ki smo ga zgradili iz iger agentov s strategijo *hand3-6-9*.



a) b)

Slika 71. Primera poti, ki predstavljata dva koncepta makroakcij v strategiji *hand-3-6-9*.

LTeam.1:Pass-K2	**MinAngK2K1T1T2.30-350** ∧ **DistK1T1.0-4** ∧ MinDistK2T1T2.3-20 ∧ DistK3C.4-15 ∧ DistT2C.0-10 ∧ **MinAngK3K1T1T2.0-90** ∧ DistK1T2.1-10
	↓
LTeam.2:Hold	**DistK1T1.7-50** ∧ **MinAngK2K1T1T2.0-20** ∧ **MinAngK3K1T1T2.5-350** ∧ DistK1K2.6-20 ∧ DistK3C.4-15
	↓
LTeam.2:Pass-K3	**DistK1T1.3-6** ∧ **MinAngK3K1T1T2.30-350** ∧ **MinAngK2K1T1T2.0-60** ∧ DistK1T2.0-7 ∧ DistK1C.2-9 ∧ DistT1C.1-20 ∧ DistK3C.4-15 ∧ DistK2C.1-10
	↓
LTeam.3:Hold	**DistK1T1.9-20** ∧ **MinAngK2K1T1T2.0-30** ∧ DistT1C.3-7 ∧ DistK2C.2-15 ∧ DistK1K3.8-50

Preglednica 28. Primer pravil za naučeno strategijo *hand3-6-9*, ki opisujejo koncept makroakcije, ki ga predstavlja pot na sliki 71a.

LTeam.3:Hold	DistK3C.10-15 ∧ MinDistK2T1T2.0-5 ∧ **DistK1T1.8-20** ∧ DistT2C.1-9
	↓
LTeam.3:Pass-K3	MinDistK2T1T2.2-5 ∧ **DistK1T1.6-9** ∧ MinDistK3T1T2.6-15 ∧ DistK1C.7-15 ∧ **MinAngK3K1T1T2.30-90** ∧ DistT2C.0-5 ∧ DistK1T2.7-50
	↓
LTeam.1:Hold	**DistK1T1.4-50** ∧ DistK3C.4-9 ∧ DistT2C.2-5 ∧ DistK1C.5-9 ∧ MinDistK2T1T2.1-10 ∧ DistT1C.1-6 ∧ DistK1K3.6-15 ∧ DistK1K2.5-50 ∧ **MinAngK2K1T1T2.0-60**
	↓
LTeam.1:Pass-K3	**DistK1T1.1-3** ∧ DistK3C.9-10 ∧ MinDistK3T1T2.10-15 ∧ **MinAngK2K1T1T2.0-60** ∧ DistT1C.3-50

Preglednica 29. Primer pravil za naučeno strategijo *hand3-6-9*, ki opisujejo koncept makroakcije, ki ga predstavlja pot na sliki 71b.

Analiza pravil pokaže, da se v posameznih pravilih pojavljaje značilke, uporabljene pri definiciji strategije *hand3-6-9*. V nadaljevanju se zato poglobimo zgolj v analizo pogojev, ki jih sestavljajo DistK1T1, MinAngK2K1T1T2 in MinAngK3K1T1T2.

Koncept makroakcije, ki je s pravili opisan v preglednici 28 se začne z akcijo igralca številka 1 (LTeam.1). Akcijo Pass-K2 izvede, ko je kot med K2, žogo in najbližjim napadalcem večji od 30 stopinj, ko je napadalec T1 bližje od 4 m in ko je kot med K3, žogo in najbližjim napadalcem manjši od 90 stopinj. Razdalja do najbližjega napadalca ustreza naši definiciji, poleg tega pa ustreza tudi večji kot do sprejemnika žoge K2, kot do K3. Žogo sprejme igralce s številko 2 (LTeam.2), ki najprej zadrži žogo z akcijo Hold, dokler je T1 oddaljen od K1 vsaj 7 m, je kot med K2, žogo in najbližjim napadalcem manjši od 20 stopinj in je kot med K3, žogo in najbližjim napadalcem skoraj poljuben. Bližina K1 ponovno ustreza naši definiciji, pri tem da koti do branilcev K2 in K3 nakazujejo naslednjo akcijo. Ker je MinAngK3K1T1T2 > MinAngK2K1T1T2 izbere LTeam.2 za naslednjo akcijo Pass-K3. To stori, ko je K1 od T1 oddaljen manj kot 6 m, ko je K3 primerno odkrit ($30 \leq$ MinAngK3K1T1T2 < 350) in K2 ni dovolj odkrit ($0 \leq$ MinAngK2K1T1T2 < 60). Zadnja akcija v tej makroakciji je sprejem oz. zadrževanje žoge igralca številka 3 (LTeam.3), ki jo zadržuje dokler ni T1 ustrezno blizu ($9 \leq$ DistK1T1 < 20) in K2 ni dovolj odkrit ($0 \leq$ MinAngK2K1T1T2 < 30).

Koncept makroakcije, ki je s pravili opisan v preglednici 29, se začne z akcijo igralca LTeam.3, ki zadržuje žogo dokler je T1 od K1 oddaljen vsaj 8 m. Spodnja meja je tu rahlo premajhna za igralca številka 3, vendar je še vedno višja od mej ostalih igralcev zato menimo da ustreza strategiji. Igralec nato poda žogo Pass-K3, ko je K1 ustrezno blizu ($6 \leq$ DistK1T < 9) in je K3 dovolj odkrit ($30 \leq$ MinAngK3K1T1T2 < 90). Žogo sprejme LTeam.1. ki jo zadržuje z akcijo Hold dokler ni T1 od K1 oddaljen 4 m in K2 ni dovolj odkrit ($0 \leq$ MinAngK2K1T1T2 < 60), kar ponovno ustreza naši strategiji. Ker je kot do K2 ustrezno majhen, pravilo nakazuje na naslednjo akcijo, kjer opravi igralec raje akcijo Pass-K3. To stori, ko je T1 od K1 oddaljen manj kot 3m in K2 ni odkrit ($0 \leq$ MinAngK2K1T1T2 < 60), torej sklepamo, da je odkrit K3.

Primera pokažeta ujemanje strategije vseh treh branilcev, poleg tega pa predstavita tudi pomembno lastnost konceptov makroakcij. Pravila posameznih akcijskih konceptov namreč opisujejo stanje, ki nakazuje na sledeče akcije. Takih opisov zgolj z učenjem posameznih akcijskih konceptov ne bi mogli doseči.

9. Diskusija

Računalniki so neuporabni – ne znajo drugega kot dajati odgovore.

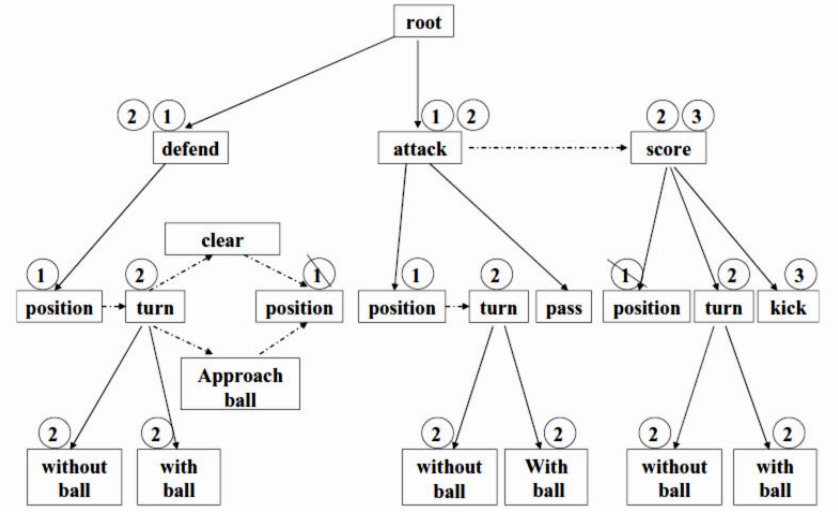
Pablo Picasso

V tem poglavju bomo predstavili izvirnost in pomembnost prispevkov disertacije v primerjavi s sorodnimi deli in jih v diskusiji še enkrat jasno poudarili.

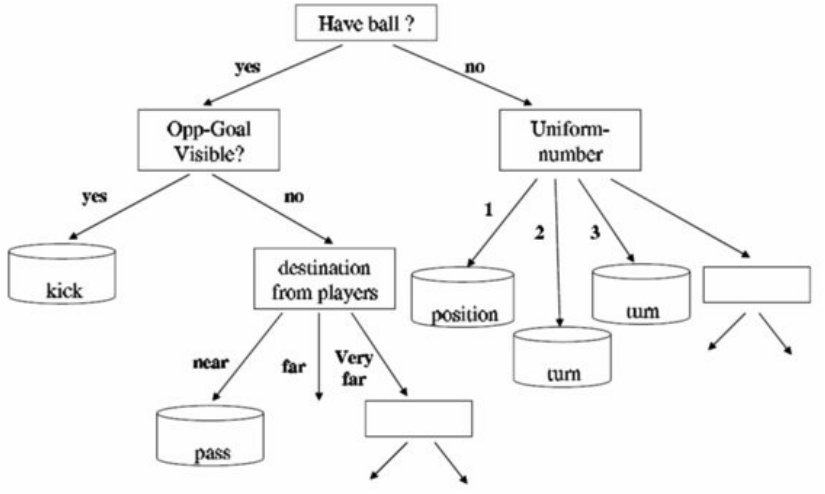
Stone v svojem doktoratu [100] in kasneje v knjigi [101] naslovi problem večnivojskega učenja v večagentnih sistemih. Trdi, da je obnašanje večagentnih sistemov preveč kompleksno, da bi se lahko neposredno naučili preslikave iz vhodnih senzorjev do izhoda v obliki agentnih akcij. Zato predlaga razgraditev problema na več nivojev, od nižjenivojskih do višjenivojskih, ki vsak zase omogočajo učinkovito učenje. Učenje poteka od spodaj navzgor, kjer je naučen model na enem nivoju tudi vhod za učenje na višjem nivoju. Sam pristop večnivojskega učenja ne omogoča avtomatske hierarhične razgradnje nalog, zato Stone prepušča odločitev o uporabi specifičnih metod strojnega učenja uporabniku. Naučene veščine uporabi v uspešnem RoboCup moštvu [101], kjer se je prestrezanja žoge naučil z uporabo nevronske mreže z vzvratnim razširjanjem napake (angl. *back-propagation neural network*), ocenjevanje primernosti podaje z algoritmom za grajenje odločitvenih dreves C4.5 [81] in odločanje o cilju podaje z izvirno razvitim TPOT-RL (angl. *team-partitioned opaque-transition reinforcement learning*) postopkom. V našem primeru učenje z MASDA v veliki meri združi različne postopke učenja iz raznih nivojev, medtem ko jih Stone obravnava ločeno z različnimi učnimi sistemi. Čeprav Stone ne uporablja kompleksnega domenskega predznanja, zahteva njegov pristop domenskega eksperta za izbiro primernih postopkov učenja, kar omeji splošno primernost pristopa. Stone tudi zanemari grafično predstavitev ter možnost človeške interpretacije dobljenega modela. Ugotovitve naše disertacije tudi mečejo novo luč na Stoneovo trditev o prekompleksnosti večagentnih sistemov za učenje neposredne preslikave iz vhodnih senzorjev do agentnih akcij. Rezultati v domeni RoboCup nakazujejo, da bi dobršen del večnivojskega učenja lahko nadomestili z modeliranjem z MASDA. To potrdimo tudi z uspešnim izvajanjem naučenih strategij v domeni 3vs2 Keepaway, saj MASDA zgradi večagentni model, ki omogoča preslikavo iz agentnih senzorjev do agentnih akcij.

Steffens [99] modelira nasprotnike v domeni RoboCup, kjer z uporabo značilk, ki jih definira kot taktične sekvence akcij, in s sklepanjem na podlagi primerov (angl. *case-based reasoning*) klasificira različne nogometne situacije. Njegova ugotovitev, da lahko model neke ekipe dobimo zgolj z opazovanjem iger ene ekipe s poljubnim nasprotnikom, potrjuje način modeliranja v disertaciji, kjer smo uporabili podatke iz desetih iger istega moštva z osmimi različnimi nasprotniki. Za razliko od njegovega uporablja naše delo bolj preproste opise značilk, ki določajo le osnovno domensko znanje in ne celih taktičnih sekvenc akcij. Le-te odkrijemo z uporabo MASDA in tako odpravimo potrebo po kompleksnem predznanju. Ključna razlika med našim in njegovim pristopom je v tem, da ima Steffens za učenje na voljo ročno označene primere, MASDA pa učne primere avtomatsko označi v postopku abstrakcije.

Kaminka s sodelavci [55] obravnava problem nenadzorovanega učenja sekvenčnega agentnega obnašanja. Opazovanja kompleksnega domenskega okolja prevede v časovno serijo prepoznanih osnovnih akcij, jih nato analizira, da odkrije ponavljajoče se in statistično pomembne podsekvence. Njihovo delo je blizu našemu v smislu identificiranja pogostih akcijskih sekvenc, vendar se njihov pristop omeji le na iskanje akcijskih zaporedij in popolnoma zanemari časovni in prostorski vidik odigranih akcij. Kasneje Kaminka in Avrahami [54] uvedeta vedenjski pristop za prepoznavanje planov, kjer v hierarhiji značilk opišeta različna agentna obnašanja. Primer take hierarhije značilk je predstavljen na sliki 72 in v grobem ustreza naši definiciji hierarhije značilk. Rezultat njunega postopka je odločitveno drevo, predstavljeno na sliki 73, ki omogoča omejeno sklepanje o obnašanju moštva. Kljub podobni uporabi hierarhije domenskih značilk je pristop omejen v tem, da hierarhije značilk uporabljata zgolj kot knjižnico za iskanje ujemajočih se opisov in ne za konstruiranje novih akcijskih opisov. Poleg tega njun pristop pri modeliranju agentnega obnašanja upošteva zgolj sekvenčni opis akcij in zato ne omogoča ločevanja enakih sekvenc akcij v različnih situacijah.



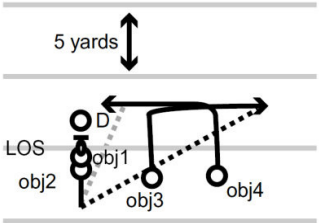
Slika 72. Primer uporabljene hierarhije značilk iz [54].



Slika 73. Primer zgrajenega odločitvenega drevesa iz [54].

Riley v nedavnem doktorskem delu [89] obdela učenje in modeliranje agentov za potrebe dajanja nasvetov. Učenje in uporaba sta eksperimentalno preverjena v različnih scenarijih, med njimi tudi v domeni RoboCup, kjer je razvil tri vrste večagentnih modelov. Prvi model upošteva plan akcij lastne ekipe in verjetnostno napoveduje gibanje nasprotnika. Naučeni model sicer uspešno izbira najbolj verjetne opise nasprotnikovega vedenja, vendar so le-ti ročno zgrajeni. Drugi model je sposoben opisati prostorske relacije med agenti v moštvu in tako določiti nogometno postavitev, kar uspešno uporabi pri igri z znanim nasprotnikom, ko imitira postavitve uspešnih moštev. Tretji model opisuje akcijske vzorce analiziranega agentnega moštva. Z združevanjem v gruče in z uporabo odločitvenih dreves je model sposoben opisati karakteristične vzorce podaj, ki so bili zaznani v odigranih igrah. Modele uporabi za formiranje nasvetov agentom med samo igro, kar pa merljivo ne doprinese k izboljšavi igre. Naš pristop prav tako opisuje prostorske relacije med agenti v moštvu ter opisuje karakteristične vzorce podaj, vendar obenem omogoča tudi odkrivanje nasprotnikovega obnašanja in ne zgolj iskanje posameznih vzorcev obnašanja. Poleg tega MASDA v domeni 3vs2 Keepaway omogoča aktivno posnemanje referenčnih strategij. Ker se v tem primeru učimo zgolj iz uspešnih primerov podaj, lahko s primerno nastavitvijo parametrov dobimo tudi modele strategij, ki so merljivo boljši od referenčnih strategij.

Intille v svojem doktorskem delu [53] obdela nalogo prepoznavanja večagentnih akcij, kjer iz podanih trajektorij igralcev ameriškega nogometa izračuna lokalne dogodke, ki jih nato poveže z zbirko planov s časovnimi relacijami (npr.: pred, potem). Z uporabo domenskega znanja o ameriškem nogometu in propagiranem nezanesljivega sklepanja je njegov sistem nato sposoben določiti, ali je predstavljen del igre primer naučene akcije ali ne. Njegovo delo obdeluje podobno nalogo kot naše – to je prepoznavanje večagentnih akcij. V določeni meri so podobni tudi izračuni lokalnih dogodkov, ki v grobem ustrezajo našemu opisu z značilkami, in opis s časovnimi relacijami, ki jih v našem primeru predstavlja ustrezno zaporedje povezav v abstraktnem akcijskem grafu. Za razliko od našega pristopa pa je pomanjkljivost njegovega pristopa predvsem v tem, da uporablja ročno zgrajeno zbirko planov, obsežno domensko znanje in da ni zmožen odkriti novega obnašanja. Za primer vzame Intille poenostavljeno igro iz ameriškega nogometa, ki je predstavljena v obliki akcijskega diagrama na sliki 74. Rezultat njegovega postopka je opis omenjene igre, predstavljen na sliki 75, ki omogoča prepoznavanje naučene igre. V primerjavi z njegovim ima naš pristop dve bistveni prednosti. Prvič, naš pristop je domensko neodvisen, saj je osnovno domensko znanje podano kot vhod v sistem. Drugič, MASDA ne uporablja zbirke planov in zato omogoča odkrivanje nepoznanega večagentnega obnašanja, ki je predstavljeno grafično z diagramom zaporedja akcij in simbolno s pravili ter opisi akcij in agentnih vlog.

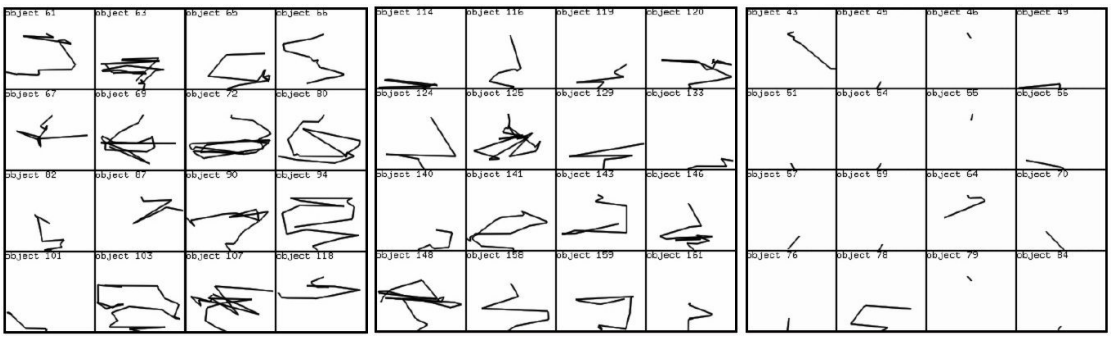


Slika 74. Primer ročno zgrajenega akcijskega diagrama v domeni ameriškega nogometa za igro *simple-p51curl* iz [53].

```
(goalTeam s51 "Team goal for simple-p51curl (s51) play."

  ; Obj1 is always the Center (C)
  (agentGoal obj1 (agent (obj1 (C)))
     (goal obj1_act1 "snapToQB (obj1)")
     (goal obj2_act2 "blockQBPass (obj1)")
     (before obj1_act1 obj1_act2))

  ;Obj2 is always the Quarterback (QB)
  (agentGoal obj2 (agent (obj2 (QB)))
     (goal obj1_act1 "dropback (obj2 5)")
     (goal obj2_act2 "throwPass (obj2)")
     (before obj2_act1 obj2_act2))

  ;The Right Wing Back (RWB)
  (agentGoal obj3
     (agent (obj3 (RWB RTE RHB HB FB TB LWB LSB)))
     (goal obj3_act1 "passPatStreaking (obj3 4 45 defReg
                                        nearRightSidelineReg 0)")
     (goal obj3_act2 "passPatCutting (obj3 70 offSidelineRightReg
                                      freeBlockingZoneReg)")
     (goal obj3_act3 "runBehind (obj3 obj4)")
     (goal obj3_act4 "passPatParaLos (obj3 3 defReg
                                      offSidelineRightReg 4)")
     (goal obj3_act5 "catchPass (obj3)")
     (before obj3_act1 obj3_act2)
     (before obj3_act2 obj3_act4))

  (agentGoal obj4                          ;The Right Flanker (RFL)
     (agent (obj4 (RFL RWB RSB LFL LSB LWB)))
     (goal obj4_act1 "passPatStreaking (obj4 4 50 defReg
                                        offEndZoneReg 0)")
     (goal obj4_act2 "passPatCutting (obj4 70 offSidelineLeftReg
                                      freeBlockingZoneReg)")
     (goal obj4_act3 "passPatParaLos (obj4 3 defReg offCenterLineReg 4)")
     (goal obj4_act4 "catchPass (obj4)") (before obj4_act1 obj4_act2)
     (before obj4_act2 obj4_act3))

  (around obj3_act2 obj4_act2) (xor obj3_act5 obj4_act4))
```

Slika 75. Zgrajeni model igre *simple-p51curl* iz [53].

V smislu vizualne identifikacije podobnih akcij je najbližje delo Hirana in Tsumota [48], ki sta predstavila metodo za iskanje zanimivih vzorcev podaj iz ročno označenih posnetkov nogometnih tekem. Njihovo delo je v osnovi podobno našemu, saj za iskanje prostorsko podobnih sekvenc podaj uporablja hierarhično združevanje v gruče. Na sliki 76 so predstavljeni njihovi rezultati v obliki treh skupin podobnih vzorcev podaj. Za razliko od našega dela pa je omenjeni pristop omejen le na iskanje podobnih sekvenc podaj in ne obravnava ostalih vidikov večagentnega modeliranja. Naše delo presega njihovo tudi v nalogi iskanja podobnih vzorcev podaj, saj MASDA poleg prostorske komponente upošteva tudi konceptualni pomen akcij in vlog agentov. Za predstavitev akcijskih sekvenc uporablja MASDA poti v abstraktnem akcijskem grafu, kar predstavlja abstrakcijo tudi v smislu poenostavitve agentnih trajektorij, kar posledično olajša človeško razumevanje dobljenih modelov. Rezultate iskanja podobnih poti predstavimo na sliki 51 v poglavju 7.2.2.



Slika 76. Primeri treh skupin podobnih vzorcev podaj iz [48].

V poenostavljeni domeni 3vs2 Keepaway pokažeta Stone in Sutton [104], da se je z vzpodbujevalnim učenjem možno naučiti strategij igre, ki je boljša od zelo preprostih ročno kodiranih strategij. Kasneje so z uporabo Keepaway okolja Kuhlmann in sodelavci [62] razširili pristop iz [104], tako da so ponovno uvedli običajno omejitev agentnega zaznavanja ter spreminjali velikosti ekip. Z vzpodbujevalnim učenjem jim uspe naučiti agente, da bistveno izboljšajo kvaliteto igre. Naši najboljši rezultati na domeni 3vs2 Keepaway so v smislu časovne uspešnosti naučenih modelov primerljivi z njihovimi, vendar z MASDA modeliranjem ne moremo bistveno izboljšati opazovanih strategij. Po drugi strani pa zaradi pristopa z vzpodbujevalnim učenjem raziskovalci niso bili sposobni interpretirati naučenih strategij. Avtorji tako sami priznavajo, da nimajo primerne razlage naučenih strategij. Njihovo razlago so okvirno podali zgolj z gledanjem posnetih iger in ne z razumevanjem naučenega modela. Ker gradi MASDA človeku razumljive opise večagentnega delovanja, smo v disertaciji preverili, v kolikšni meri dobljeni simbolni opisi večagentnega delovanja dejansko opisujejo večagentno strategijo. Z analizo pravil naučenih strategij *hand* in *hand3-6-9* in primerjavo z izvorno kodo strategij smo v disertaciji pokazali, da pravila omogočajo človeško interpretacijo ter obenem v veliki meri ustrezajo zakonitostim referenčnih strategij.

S primerjavo s sorodnim delom smo pokazali, da modeliranja večagentnih sistemov z MASDA najbolj celovito naslovi problem modeliranja večagentnih sistemov, obenem pa vse predstavljene pristope modeliranja in učenja večagentnih sistemov preseže v več oz. v vsaj enem od naslednjih vidikov: uporaba zgolj osnovnega domenskega predznanja pri modeliranju nepredvidenega večagentnega obnašanja, avtomatska abstrakcija večagentnih modelov, možnost človeške interpretacije modelov in vizualizacije večagentnega obnašanja ter izkazana uspešnost modeliranja. Obdelani vidiki večagentnega modeliranja pri sorodnih delih so tabelarično predstavljeni v preglednici 30.

	Uporaba osnovnega domenskega predznanja	Avtomatska abstrakcija večagentnih modelov	Vizualizacija in človeška interpretacija modelov	Izkazana uspešnost modeliranja
Stone [100, 101]	✓	×	×	✓
Steffens [99]	×	×	delno (interpretacija)	✓
Kaminka s sodelavci [55]	✓	×	delno (interpretacija)	✓
Kaminka in Avrahami [54]	×	×	delno (interpretacija)	✓
Riley [89]	×	delno (vzorci podaj)	✓	×
Intille [53]	×	×	delno (interpretacija)	✓
Hirano in Tsumoto [48]	✓	delno (vzorci podaj)	delno (vizualizacija)	×
Stone in Sutton [104]	✓	×	×	delno (minimalna)
Kuhlmann in sodelavci [62]	✓	×	×	✓
MASDA	✓	✓	✓	✓

Preglednica 30. Prikaz obdelanih vidikov večagentnega modeliranja za sisteme različnih avtorjev in MASDA.

V povzetku lahko ugotovimo:

- Problem, s katerim se ukvarja disertacija, je značilen problem moderne umetne inteligence, ki se pogosto ukvarja z porazdeljenimi, dinamičnimi in kompleksnimi večagentnimi sistemi. S podobnimi nalogami se ukvarjajo disertacije in publikacije v najodmevnejših revijah, zato je iz literature znano, da ima večagentna naloga bistveno drugačne značilnosti kot klasični problemi umetne inteligence (npr. šah). Posebej je naloga zahtevna zaradi časovno prepletene interakcije avtonomnih agentov z ostalimi agenti ter okoljem.
- MASDA je edini poznani algoritem, ki je sposoben iz nizkonivojskega znanja neodvisno od domene generirati dele oz. celotne večagentne strategije. Sorodni sistemi rešujejo podobne naloge ob neprimerno večji količini domenskega znanja in so običajno specializirani le za en tip domene. MASDA je načeloma sposobna s spreminjanjem domenskega predznanja, ki v bistvu določa le opisni jezik, generirati strateške vzorce v kateremkoli večagentnem sistemu.
- Da je MASDA sposoben reševati tovrstne naloge, je bilo potrebno uvesti več novih mehanizmov, med njimi: poseben postopek abstrakcije, merjenje razdalje med akcijskimi koncepti in prilagojeno strojno učenje. Posamezni sorodni sistemi uporabljajo določene podobne rešitve, vendar so le-te nepovezane in primerne le za specifične naloge, MASDA pa te rešitve učinkovito poveže v celovit algoritem.
- MASDA je svojo učinkovitost in domensko neodvisnost pokazal na dveh domenah: RoboCup in 3vs2 Keepaway. V prvi domeni se je samo iz desetih primerov iger naučil vzorcev napada na gol, ki so bili zanimivi tudi domenskemu ekspertu, njihova uspešnost pa je bila prikazana tudi z merili strojnega učenja. Sposobnost algoritma, da ob osnovnem domenskem predznanju sam odkrije

strateške zakonitosti, so po našem mnenju dokaz izredne uspešnosti. V drugi domeni 3vs2 Keepaway je bilo naučeno znanje tudi operativno uporabljeno za igranje. S smo pokazali, da je MASDA na preprostejših večagentnih domenah sposoben odkriti in reproducirati celotno strategijo in ne samo njenih delov, kar je prav tako izjemen dosežek.

- Domensko neodvisnost smo pokazali s prehodom iz domene RoboCup na 3vs2 Keepaway, kjer se je spremenilo število agentov, tip akcij, opis domene in razdalja med akcijskimi koncepti. Kljub navidezni podobnosti domen je bilo potrebno zamenjati celotno domensko predznanje. Zato utemeljeno domnevamo, da bi bil prehod na drugo domeno podobno zahteven.

10. Zaključek

Beware of the young doctor and the old barber.

Benjamin Franklin

Izpolnili smo cilj disertacije – razviti domensko neodvisen postopek, ki iz zaporedja osnovnih akcij večagentnega sistema in z uporabo osnovnega domenskega znanja zgradi človeku razumljive opise strategije delovanja večagentnega sistema.

Zaključki po poglavjih so sledeči:

V disertaciji najprej predstavimo večagentne sisteme in opišemo njihove značilnosti. Poleg tega podrobno predstavimo izbrani večagentni domeni RoboCup in 3vs2 Keepaway ter pokažemo, da sta primerni kot referenčni domeni za modeliranje. Obsežen pregled sorodnega dela pokaže, da je modeliranje večagentnih agentov pogosto raziskovano, vendar so bili zadovoljivo obdelani le enostavni primeri dvoagentnih sistemov z omejenim diskretnim prostorom. V kompleksnejših večagentnih domenah se pri modeliranju višjenivojskih strategij pogosto uporablja visokonivojsko znanje v obliki zbirke planov, kjer sistemi z izbiranjem oz. sestavljanjem posameznih planov generirajo dele strategij. V domenah RoboCup in 3vs2 Keepaway so raziskave omejene predvsem na iskanje modelov, ki omogočajo zadovoljivo napovedno točnost, zanemarijo pa človeško razumevanje pridobljenih modelov.

Predstavili smo sistem za strateško modeliranje večagentnih sistemov MASDS, ki iz sledi večagentnega sistema ter osnovnega domenskega znanja generira model, ki opisuje delno oz. celotno večagentno strategijo. Ključni del sistema je nov domensko neodvisen algoritem za modeliranje večagentnih sistemov MASDA, ki s postopkom abstrakcije gradi abstraktnejše večagentne modele, ki opisujejo delno oz. celotno strategijo opazovanega sistema. Model sestavlja nabor konceptov makroakcij, ki predstavljajo specifično zaporedje agentnih akcij. Vsak odkrit koncept makroakcije je predstavljen z grafičnim in simbolnim opisom.

Domensko neodvisnost MASDA smo potrdili z implementacijo na dveh večagentnih domenah: RoboCup in 3vs2 Keepaway. Za obe domeni smo pripravili osnovno domensko znanje, ki opisuje akcije, agentne vloge in domenske značilke ter omogoča primerjavo akcijskih konceptov. Pokazali smo, da MASDA omogoča abstrakcijo pridobljenih modelov. Rezultati poskusov na domeni RoboCup so potrdili, da algoritem uporablja abstrakcijo pri opisu akcij, agentnih vlog in domenskega prostora. Zgrajeni modeli človeku razumljivo opisujejo interakcijo med agenti na grafičen in simbolen način. To smo pokazali s človeškim in strojnim ovrednotenjem.

Z ekspertno analizo izbranega koncepta makroakcije na domeni RoboCup smo pokazali, da dobljeni opis koncepta makroakcije predstavlja človeku razumljiv opis večagentnega zaporedja akcij, ki je del nogometne strategije. Pri tem je pomembno omeniti, da odkriti pogoji ekspertu nakazujejo glavne prednosti oz. pomanjkljivosti modeliranega napada na gol. Na domeni 3vs2 Keepaway smo z merjenjem časovne uspešnosti agentov, ki uporabljajo naučeno strategijo, in s primerjavo izvorne kode ter naučenih pravil pokazali, da predstavlja dobljeni model celotno strategijo

večagentnega sistema. Človeško analizo pravil smo opravili tudi za naučeno strategijo *hand*, ki pokaže, da se pravila v veliki meri ujemajo s pogoji, ki jih določa implementacijo referenčne *hand* strategije. Analiza modela strategije *hand3-6-9*, ki določa različno strategijo za vsakega igralca posebej, kaže, da MASDA generira koncepte makroakcije, ki vsebujejo različna pravila za različne agente, ki v veliki meri ustrezajo definiciji strategije *hand3-6-9*. Analiza pokaže, da del pravil neposredno ustreza definiciji strategije, drugi del pravil pa ustreza pogojem, ki sicer neposredno ne izvirajo iz strategije, so pa posledica razmer na igrišču, ki so razložljive z dano strategijo. Analiza tudi pokaže, da pravila posameznih akcijskih konceptov opisujejo stanja, ki nakazuje na sledeče akcije. Takih opisov zgolj z učenjem posameznih akcijskih konceptov ne bi mogli doseči, zato sklepamo, da dobljeni koncepti makroakcij celostno opisujejo zaporedje akcijskih konceptov – torej dejansko predstavljajo večagentni model. Zato sklepamo, da je MASDA pristop bolj primeren za modeliranje večagentnih domen, kot enoagentni postopki modeliranja.

Primernost MASDA modelov za avtomatsko napovedovanje smo v domeni RoboCup potrdili z merjenjem klasifikacijske točnosti, priklica in natančnosti z uporabo treh različnih načinov klasificiranja. Skupni zaključki predstavljenih meritev so, da so modeli MASDA večagentnih akcij primerni za avtomatsko klasifikacijo v domeni RoboCup, vendar pa moramo uporabljati modele, ki so znotraj primernega intervala parametra abstrakcije. V domeni 3vs2 Keepaway smo pokazali, da lahko z uporabo MASDA generiramo strategije, ki dejansko posnemajo delovanje referenčnih strategij. Primerjavo naučenih in referenčnih strategij smo izvedli na štiri načine: z merjenjem povprečnega trajanja epizode, z mejenjem ujemanja posameznih akcij, z analizo dobljenih pravil in z vizualno oceno poteka igre. Z merjenjem in primerjavo povprečnega trajanja epizode smo pokazali, da naučene strategije dosegajo rezultate, ki so primerljivi z referenčnimi. Poleg tega smo s primerjavo rezultatov med naučeno *hand* in naučeno *rand* strategijo pokazali, da se rezultati meritev posameznih strategij ne prekrivajo in da sta naučeni strategiji med seboj signifikantno različni. Analiza meritev ujemanja izbranih akcij med referenčnima in naučenimi strategijami pokaže, da je naučena *hand* strategija v več kot 92 % izbrala identično akcijo kot referenčna *hand* strategija, kar potrjuje uspešnost posnemanja referenčne *hand* strategije. Na tej domeni smo tudi vizualno primerjali posnetke iger referenčnih in naučenih strategij ter potrdili vizualno podobnost iger naučenih strategij z referenčnimi.

V disertaciji uporabljeni načini ovrednotenja modeliranja večagentnih sistemov z uporabo MASDA so strnjeno predstavljeni v preglednici 31.

	Človeška analiza modela		Strojno preverjanje	Deklarativno preverjanje	Proceduralno preverjanje
Domena	Ekspert	Neekspert			
RoboCup	✓	✓	✓	✓	
3vs2 Keepaway		✓	✓	✓	✓

Preglednica 31. Povzetek načinov ovrednotenja modeliranja z MASDA.

Opravljene meritve in analize potrjujejo, da je MASDA domensko neodvisen postopek, ki iz zaporedja osnovnih akcij večagentnega sistema in z uporabo osnovnega domenskega znanja zgradi človeku razumljive opise strategije delovanja večagentnega sistema.

Literatura

1. Aggarwal, C.C. *Towards Systematic Design of Distance Functions for Data Mining Applications*. V zborniku *SIGKDD '03*. 2003: ACM Press.

2. Agirre, E. in G. Rigau. *Word Sense Disambiguation Using Conceptual Density*. V zborniku *Coling'96*. 1996. Copenhagen, Danska.

3. Albus, J.S., *Brains, Behavior, and Robotics*. 1981, New York: McGraw-Hill, Inc.

4. Allen, J.F., *Maintaining knowledge about temporal intervals.* Communications of the ACM, 1983. **26**(11): str. 832–843.

5. Allen, J.F., *Time and time again: The many ways to represent time.* International Journal of Intelligent Systems, 1991. **6**(4): str. 341-356.

6. Andre, D. in S.J. Russell, *Programmable reinforcement learning agents.* Advances in Neural Information Processing Systems, 2001. **13**: str. 1019-1025.

7. Andre, D. in S.J. Russell. *State abstraction for programmable reinforcement learning agents*. V zborniku *The 18th National Conference on Artificial Intelligence*. 2002.

8. Andre, D. in A. Teller. *Evolving team Darwin United*. V zborniku *RoboCup-98: Robot Soccer World Cup II*. 1999: Springer Verlag, Berlin.

9. Balch, T. *Teambots*. 2000. Dostopno na: http://www.teambots.org.

10. Balch, T. *Teambots domain: Soccerbots*. 2000. Dostopno na: http://www-2.cs.cmu.edu/~trb/TeamBots/Domains/SoccerBots.

11. Bernstein, D.S., S. Zilberstein in N. Immerman. *The complexity of decentralized control of markov decision processes*. V zborniku *The Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-2000)*. 2000.

12. Bežek, A. *Modeling Multiagent Games Using Action Graphs*. V zborniku *Modeling Other Agents from Observations (MOO 2004)*. 2004. New York, USA: ACM.

13. Bežek, A. *Discovering Strategic Multi-Agent Behavior in a Robotic Soccer Domain*. V zborniku *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*. 2005. Utrecht, Nizozemska: ACM.

14. Bežek, A. in M. Gams, *From basic agent behavior to strategic patterns in a robotic soccer.* Informatica, 2005. **29**(4): str. 461-468.

15. Bežek, A., M. Gams in I. Bratko. *Multi-Agent Strategic Modeling in a Robotic Soccer Domain*. V zborniku *Fifth International Joint Conference on*

Autonomous Agents and Multiagent Systems (AAMAS'06). 2006. Hakodate, Japan.

16. Billings, D., in ostali. *Opponent modeling in poker*. V zborniku *The Fifteenth National Conference on Artificial Intelligence (AAAI-98)*. 1998. Madison: AAAI Press.

17. Blum, A.L. in P. Langley, *Selection of Relevant Features and Examples in Machine Learning.* Artificial Intelligence, 1997. **97**(1-2): str. 245-271.

18. Boutilier, C. *Sequential optimality and coordination in multiagent systems*. V zborniku *The Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*. 1999.

19. Bowling, M., B. Browning in M. Veloso. *Plays as effective multiagent plans enabling opponent-adaptive play selection*. V zborniku *The 14th International Conference on Automated Planning and Scheduling (ICAPS-04)*. 2004.

20. Bratko, I., *Prolog Programming for Artificial Intelligence*. 3 ed. International computer science series. 2001: Addison-Wesley. 678.

21. Bratko, I. in D. Šuc., *Learning qualitative models.* AI Magazine, 2004. **24**(4): str. 107-119.

22. Calmet, J. in A. Daemi. *From Entropy to Ontology*. V zborniku *Fourth International Symposium From Agent Theory to Agent Implementation*. 2004.

23. Carmel, D. in S. Markovitch. *Incorporating opponent models into adversary search*. V zborniku *The Thirteenth National Conference on Artificial Intelligence (AAAI-96)*. 1996. Portland.

24. Carmel, D. in S. Markovitch. *Learning Models of Intelligent Agents*. V zborniku *The Thirteenth National Conference on Artificial Intelligence*. 1996. Portland, Oregon.

25. Carmel, D. in S. Markovitch, *Model-based learning of interaction strategies in multiagent systems.* Journal of Experimental and Theoretical Artificial Intelligence, 1998. **10**(3): str. 309-332.

26. Chawla, N., N. Japkowicz in A. Kolcz, eds. *SIGKDD Explorations, Special Issue on Class Imbalances*. ACM SIGKDD Explorations Newsletter. Vol. 6 (1). 2004.

27. Chen, M., in ostali, *Users Manual - RoboCup Soccer Server - for Soccer Server version 7.07 and later*. 2001.

28. Cochran, W.G., *Sampling Techniques*. Third ed. 1977: New York: Wiley.

29. Cohen, P.R., *If Not Turing's Test, Then What?* AI Magazine, 2005. **26**(4): str. 61-67.

30. Cohen, W.W. *Fast Effective Rule Induction*. V zborniku *The 12th International Conference on Machine Learning*. 1995: Morgan Kaufmann.

31. Cohen, W.W. in Y. Singer. *A Simple, Fast, and Effectove Rule Learner*. V zborniku *Sixteenth National Conference on Artificial Intelligence and Elevenths Innovative Applications of Artificial Intelligence Conference*. 1999: American Association of Artificial Intelligence.

32. Demšar, J., *Statistical Comparisons of Classifiers over Multiple Data Sets.* Journal of Machine Learning Research, 2006. **7**(1): str. 1-30.

33. Dietterich, T.G., *Hierarchical reinforcement learning with the maxq value function decomposition.* Journal of Artificial Intelligence Research, , 2000. **13**: str. 227-303.

34. Doan, A., in ostali, *Ontology matching: A machine learning approach*. Handbook on Ontologies in Information Systems, ur. S. Staab in R. Studer. 2003: Springer-Velag.

35. Elsner, B., *Nogomet; teorija igre*. 1997, Ljubljana: Fakulteta za šport, Inštitut za šport. 158.

36. Fédération Internationale de Football Association (FIFA). *Laws of the games*. Dostopno na: http://www.fifa.com.

37. Forman, G. in I. Cohen. *Learning from Little: Comparison of Classifiers Given Little Training*. V zborniku *Knowledge Discovery in Databases: PKDD 2004: 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*. 2004. Pisa, Italija: Springer Berlin / Heidelberg.

38. Fortnow, L. in D. Whang. *Optimality and domination in repeated games with bounded players*. V zborniku *The 26th ACM Symposium on the Theory of Computing*. 1994. New York: ACM.

39. Franklin, S. in A. Graesser. *Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents*. V zborniku *The Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*. 1996.

40. Freund, Y., in ostali. *Efficient algorithms for learning to play repeated games against computationally bounded adversaries*. V zborniku *The 36th IEEE Conference on Foundations of Computer Science*. 1995.

41. Freund, Y. in R.E. Schapire, *Adaptive game playing using multiplicative weights.* Games and Economic Behavior, 1999. **29**: str. 79-103.

42. Gams, M., *Weak intelligence : through the principle and paradox of multiple knowledge*. Vol. 6. 2001: Nova Science.

43. Gerevini, A. in L. Schubert, *Efficient algorithms for qualitative reasoning about time.* Artificial Intelligence, 1995. **74**(2): str. 207-248.

44. Gerkey, B.P. in M.J. Matarič. *On role allocation in RoboCup*. V zborniku *RoboCup 2003: Robot Soccer World Cup VII*. 2004: Springer-Verlag, Berlin Heidelberg.

45. Giunchiglia, F. in M. Yatskevich, *Element Level Semantic Matching*. 2004, Technical Report #DIT-04-035.

46. Gmytrasiewicz, P.J. in E.H. Durfee. *A rigorous, operational formalization of recursive modeling*. V zborniku *The First International Conference on Multi-Agent Systems (ICMAS-95)*. 1995.

47. Han, K. in M. Veloso. *Automated robot behavior recognition applied to robotic soccer*. V zborniku *The IJCAI-99 Workshop on Team Behaviors and Plan Recognition*. 1999.

48. Hirano, S. in S. Tsumoto. *Finding Interesting Pass Patterns from Soccer Game Records*. V zborniku *The Eighth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-2004)*. 2004.

49. Holland, O. in C. Melhuish, *Stigmergy, self-organization, and sorting in collective robotics* Artificial Life, 1999. **5**(2): str. 173-202.

50. Howard, A. *Team description: MuCows*. V zborniku *RoboCup-2000: Robot Soccer World Cup IV*. 2001: Springer-Verlag.

51. Hsu, W.H. in S.M. Gustafson. *Genetic programming and multi-agent layered learning by reinforcements*. V zborniku *Genetic and Evolutionary Computation Conference*. 2002. New York.

52. Huddleston, D. in K. Huddleston. *Dictionary Of Soccer Terms, Concepts & Rules*. 1999. Dostopno na: http://www.soccerhelp.com/Soccer_Tips_Dictionary_Terms.shtml.

53. Intille, S.S., Visual Recognition of Multi-Agent Action. Doktorsko delo, MIT, 1999.

54. Kaminka, G. in D. Avrahami. *Symbolic Behavior-Recognition*. V zborniku *Modeling Other Agents from Observations (MOO 2004)*. 2004.

55. Kaminka, G., in ostali. *Learning the sequential coordinated behavior of teams from observations*. V zborniku *Proceedings of the RoboCup-2002 Symposium*. 2002.

56. Kaminka, G., in ostali. *Learning the sequential coordinated behavior of teams from observations*. V zborniku *The RoboCup-2002 Symposium*. 2003.

57. Kendall, E.A. *Agent Roles And Role Models : New Abstractions For Intelligent Agent System Analysis And Design*. V zborniku *ECOOP'99, AOP Workshop*. 1999.

58. Kitano, H., in ostali. *RoboCup: The Robot World Cup Initiative*. V zborniku *Workshop on Entertainment and AI/Alife at IJCAI-95*. 1995.

59. Kitano, H., in ostali. *The RoboCup Synthetic Agent Challenge 97*. V zborniku *IJCAI*. 1997.

60. Kohavi, R. in F. Provost, *Glossary of Terms - Special Issue on Applications of Machine Learning and the Knowledge Discovery Process.* Machine Learning, 1998. **30**: str. 271-274.

61. Koller, D. in B. Milch, *Multi-Agent Influence Diagrams for Representing and Solving Games.* Games and Economic Behavior, 2003. **45**(1): str. 181-221.

62. Kuhlmann, G. in P. Stone. *Progress in Learning 3 vs. 2 Keepaway*. V zborniku *RoboCup-2003: Robot Soccer World Cup VII*. 2004: Springer Verlag, Berlin.

63. Kuhlmann, G., P. Stone in J. Lallinger. *The UT Austin Villa 2003 Champion Simulator Coach: A Machine Learning Approach*. V zborniku *RoboCup-2004: Robot Soccer World Cup VIII*. 2004: Springer Verlag, Berlin, 2005.

64. Laird, J.E. *It knows what you're going to do: Adding anticipation to a quakebot*. V zborniku *The Fifth International Conference on Autonomous Agents (Agents-2001)*. 2001.

65. Learning to Play Keepaway Home Page. Dostopno na: http://www.cs.utexas.edu/~AustinVilla/sim/keepaway/.

66. Luke, S., in ostali. *Co-evolving soccer softbot team coordination with genetic programming*. V zborniku *RoboCup-97: Robot Soccer World Cup I*. 1998: Springer Verlag, Berlin.

67. Mackworth, A., *On Seeing Robots*, V *Computer Vision: System, Theory, and Applications*. 1993, World Scientic Press: Singapore. str. 1-13.

68. Marsella, S., in ostali, *Experiences acquired in the design of RoboCup teams: a comparison of two fielded teams.* Autonomous Agents and Multi-Agent Systems, 2001. **4**(2): str. 115–129.

69. Miene, A., U. Visser in O. Herzog, *Recognition and prediction of motion situations based on a qualitative motion description*, V *RoboCup 2003: Robot Soccer World Cup VII*, D. Polani in ost., uredniki. 2004, Springer Verlag. str. 77-88.

70. Miller, G., *WordNet: An on-line lexical database.* International Journal of Lexicography, 1990. **3**(4).

71. Morris, P. in N. Muscettola. *Execution of temporal plans with uncertainty*. V zborniku *The Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*. 2000: AAAI Press/The MIT Press.

72. Muscettola, N., P. Morris in I. Tsamardinos. *Reformulating temporal plans for efficient execution*. V zborniku *The Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*. 1998.

73. Nair, R., M. Tambe in S. Marsella. *Role Allocation and Reallocation in Multiagent Teams: Towards A Practical Analysis*. V zborniku *Second International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS-03)*. 2003.

74. Noda, I., H. Matsubara in K. Hiraki. *Learning cooperative behavior in multi-agent environment: a case study of choice of play-plans in soccer*. V zborniku *PRICAI'96: Topics in Artificial Intelligence (Proc. of 4th Pacific Rim International Conference on Artificial Intelligence)*. 1996. Cairns, Australia.

75. Noda, I., in ostali. *Overview of RoboCup-97*. V zborniku *RoboCup-97: Robot Soccer World Cup I*. 1997.

76. Papadimitriou, C.H. in J.N. Tsisiklis, *The complexity of Markov decision processes.* Mathematics of Operations Research, 1987. **12**(3): str. 441-450.

77. Peshkin, L., in ostali. *Learning to cooperate via policy search*. V zborniku *The Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-2000)*. 2000.

78. Pietro, A.D., L. While in L. Barone. *Learning in RoboCup keep-away using evolutionary algorithms*. V zborniku *GECCO 2002: The Genetic and Evolutionary Computation Conference*. New York: Morgan Kaufmann Publishers.

79. Pocrnjič, M., *Osebni pogovori*. 2005: Ljubljana. Profesor nogometa na Fakulteti za šport.

80. Pynadath, D. in M. Tambe, *The communicative multiagent team decision problem: Analyzing teamwork theories and models.* Journal of Artificial Intelligence Research,, 2002(16): str. 389-423.

81. Quinlan, J.R., *C4.5: Programs for Machine Learning*. 1993: Morgan Kaufmann, San Mateo, CA.

82. Rada, R., in ostali, *Development an Application of a Metric on Semantic Nets.* IEEE Transactions on Systems, Man and Cybernetics, 1989. **19**(1): str. 17-30.

83. Raines, T., M. Tambe in S. Marsella. *Towards automated team analysis: a machine learning approach*. V zborniku *Third international RoboCup competitions and workshop*. 1999.

84. Resnik, J., *Osebni pogovori*. 2006: Kranj. NK Triglav.

85. Resnik, P. *Using information content to evaluate semantic similarity in a taxonomy*. V zborniku *IJCAI*. 1995.

86. Richter, M.M., *On the notion of similarity in case-based reasoning*, V *Mathematical and Statistical Methods in Articial Intelligence*. 1995, Springer Verlag. str. 171-184.

87. Riedmiller, M., in ostali. *Brainstormers 2002 - team description*. V zborniku *RoboCup-2002: Robot Soccer World Cup VI*. 2003: Springer Verlag, Berlin.

88. Riedmiller, M., in ostali. *Karlsruhe brainstormers—a reinforcement learning approach to robotic soccer*. V zborniku *RoboCup-2000: Robot Soccer World Cup IV*. 2001: Springer Verlag, Berlin.

89. Riley, P., Coaching: Learning and Using Environment and Agent Models for Advice. Doktorsko delo, Carnegie Mellon University, 2005.

90. RoboCup Federation. *RoboCup Objective*. Dostopno na: http://www.robocup.org/overview/22.html.

91. RoboCup Soccer Simulator Maintenance Group. *The RoboCup Soccer Simulator*. Dostopno na: http://sserver.sourceforge.net.

92. Russell, S. in P. Norvig, *Artificial Intelligence: A Modern Approach*. 1995: Prentice Hall. 932.

93. Salton, G. in M.J. McGill, *Introduction to Modern Information Retrieval*. 1983, New York: Mc Graw Hill.

94. Sammut, C., in ostali. *Learning to fly*. V zborniku *The Ninth International Conference on Machine Learning (ICML-92)*. 1992: Morgan Kaufmann.

95. Schaeffer, J., in ostali, *A world championship caliber checkers program.* Artificial Intelligence, 1992. **53**(2–3): str. 273–290.

96. Schapire, R.E. in Y. Singer, *Improved boosting algorithms using confidence-rated predictions.* Machine Learning 1999. **37**(3): str. 297-336.

97. Schwalb, E. in L. Vila, *Temporal constraints: A survey.* Constraints, 1998. **2**(3): str. 129-150.

98. Shoham, Y., *Agent-oriented programming.* Artificial Intelligence,, 1993. **60**(1): str. 51-92.

99. Steffens, T., Feature-based declarative opponent-modelling in multi-agent systems. Magistrsko delo, Universität Osnabrück, 2002.

100. Stone, P., Layered Learning in Multi-Agent Systems. Doktorsko delo, Carnegie Mellon University, 1998.

101. Stone, P., *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. 2000: MIT Press.

102. Stone, P., in ostali. *Keepaway Soccer: From Machine Learning Testbed to Benchmark*. V zborniku *RoboCup-2005: Robot Soccer World Cup IX*. 2006: Springer Verlag, Berlin.

103. Stone, P., P. Riley in M. Veloso. *The CMUnited-99 champion simulator team*. V zborniku *RoboCup-99:Robot Soccer World Cup III*. 2000: Springer Verlag, Berlin.

104. Stone, P. in R.S. Sutton. *Scaling Reinforcement Learning Toward RoboCup soccer*. V zborniku *The Eighteenth International Conference on Machine Learning (ICML)*. 2001: Morgan Kaufmann, San Francisco, CA.

105. Stone, P. in R.S. Sutton. *Keepaway soccer: a machine learning testbed*. V zborniku *RoboCup-2001: Robot Soccer World Cup V*. 2002: Springer Verlag, Berlin.

106. Stone, P., R.S. Sutton in G. Kuhlmann, *Reinforcement Learning for RoboCup-Soccer Keepaway.* Adaptive Behavior, 2005. **13**(3): str. 165–188.

107. Stone, P. in M. Veloso, *Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for Real-Time Strategic Teamwork.* Artificial Intelligence, 1999. **110** str. 241-273.

108. Sukthankar, G. in K. Sycara. *A cost minimization approach to human behavior recognition*. V zborniku *The fourth international joint conference on Autonomous agents and multiagent systems*. 2005. Utrecht, The Netherlands: ACM Press.

109. Suryadi, D. in P.J. Gmytrasiewicz. *Learning models of other agents using influence diagrams*. V zborniku *International Conference on User Modeling*. 1999.

110. Sycara, K., *MultiAgent Systems.* AI Magazine, 1998. **19**(2).

111. Šuc, D., *Machine Reconstruction of Human Control Strategies*. Frontiers in Artificial Intelligence and Applications. Vol. 99. 2003, Amsterdam, The Netherlands: IOS Press.

112. Šuc, D., D. Vladušič in I. Bratko, *Qualitatively faithful quantitative prediction.* Artificial Intelligence, 2004. **158**(2): str. 189-214.

113. Tambe, M. in P.S. Rosenbloom. *Resc: An approach for dynamic, real-time agent tracking*. V zborniku *The Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*. 1995.

114. Taylor, M.E. in P. Stone. *Behavior transfer for value-function-based reinforcement learning*. V zborniku *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05)*. 2005: ACM.

115. Trebec, M., *Osebni pogovori*. 2004. Nogometni trener v 2. SNL.

116. Tsang, E., *Foundations of Constraint Satisfaction*. 1993, New York: Academic Press.

117. Vilain, M., H. Kautz in P. van Beek, *Constraint propagation algorithms for temporal reasoning: a revised report.* Readings in qualitative reasoning about physical systems, 1989: str. 373-381

118. Walker, T., J. Shavlik in R. Maclin. *Relational reinforcement learning via sampling the space of first-order conjunctive features*. V zborniku *The ICML Workshop on Relational Reinforcement Learning*. 2004. Banff, Canada.

119. Wegner, P., *Interactive foundations of computing.* Theoretical Computer Science, 1997. **192**(2): str. 315-351.

120. Weiß, G. *Ecai-96 workshop on learning in distributed artificial intelligence. Call For Papers, 1996.* V.

121. Whiteson, S. in P. Stone. *Concurrent layered learning*. V zborniku *Second International Joint Conference on Autonomous Agents and Multiagent Systems*. 2003.

122. Wilson, D.R. in T.R. Martinez, *Improved Heterogeneous Distance Functions.* Journal of Artificial Intelligence Research,, 1997. **6**: str. 1-34.

123. Wooldridge, M., N.R. Jennings in D. Kinny, *The Gaia Methodology for Agent-Oriented Analysis and Design.* Autonomous Agents and Multi-Agent Systems, 2000. **3**(3): str. 285-312.

124. Wooldridge, M.J. in N.R. Jennings, *Intelligent agents: Theory and practice.* Knowledge Engineering Review, 1995. **10**(2): str. 115-152.

125. Wünstel, M., in ostali, *Behavior classification with self-organizing maps*, V *RoboCup-2000*. 2001, Springer Verlag. str. 108-118.

126. Xuan, P., V. Lesser in S. Zilberstein. *Communication decisions in multiagent markov decision processes: Model and experiments.* V zborniku *The Fifth International Conference on Autonomous Agents (Agents-2001)*. 2001.

Izjava

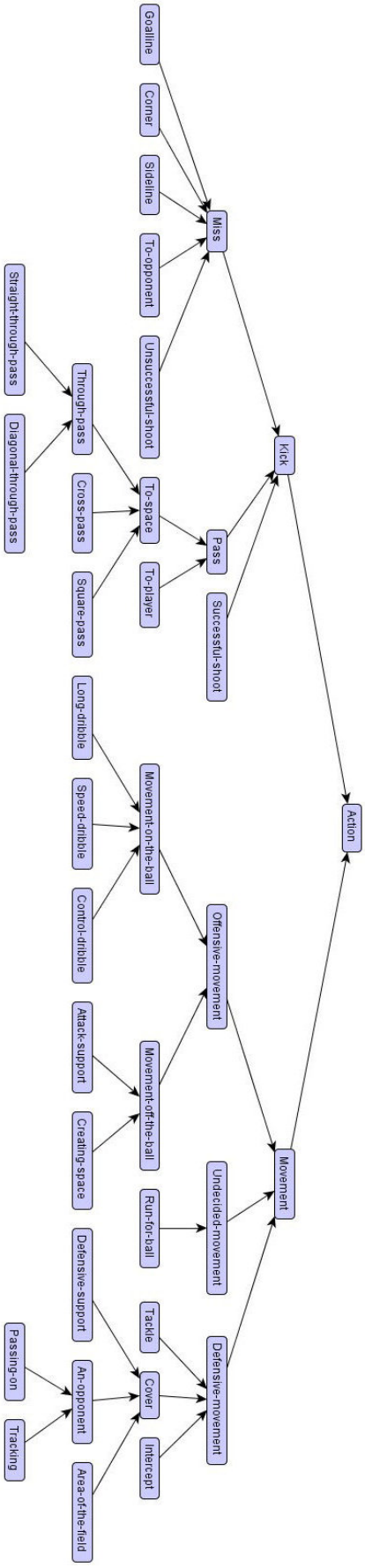
Izjavljam, da sem doktorsko disertacijo izdelal samostojno na Odseku za inteligentne sisteme Instituta Jožef Stefan pod vodstvom somentorja prof. dr. Matjaža Gamsa ter s pomočjo mentorja akad. prof. dr. Ivana Bratka. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Andraž Bežek

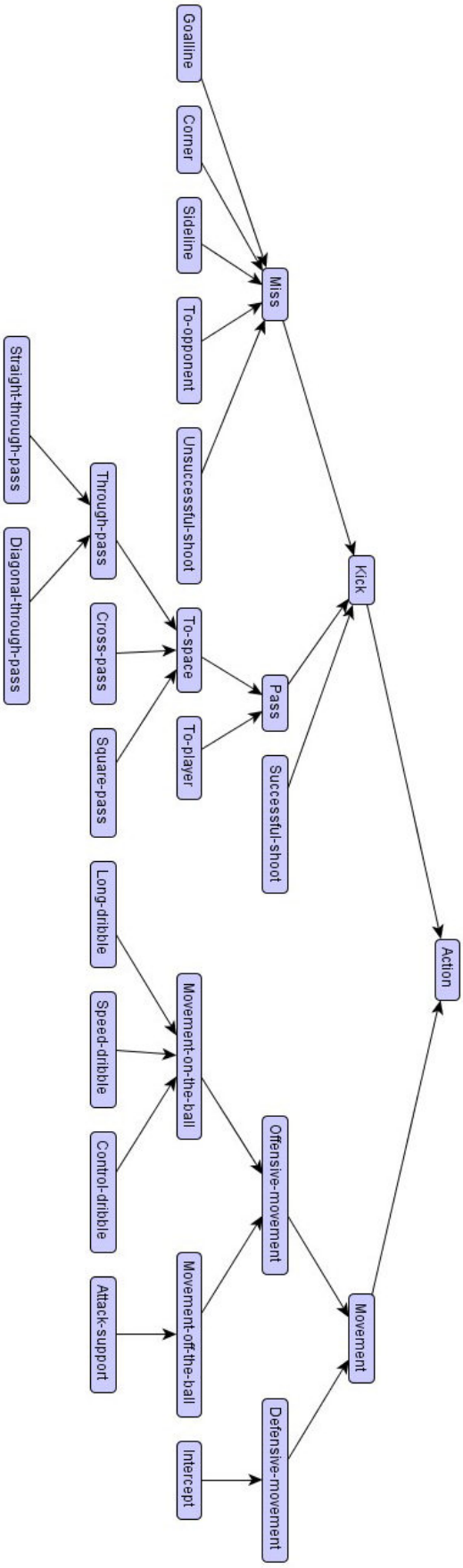
Priloga A

Oznaka akcije	Pomen akcije
Action	Splošen opis agentne akcije
Kick	Brca žoge
Miss	Zgrešena brca
Goal-line	Zgrešena brca, kjer žoga prečka črto gola
Corner	Zgrešena brca, kjer žoga prečka kot
Sideline	Zgrešena brca, kjer žoga prečka stransko črto
To-opponent	Zgrešena brca, kjer žogo prestreže nasprotnik
Unsuccessful-shoot	Zgrešen strel na gol, kjer žogo prestreže nasprotnikov vratar
Pass	Podaja žoge soigralcu
To-space	Podaja v prostor, kjer žogo prestreže soigralec
Through-pass	Dolga podaja v prostor, kjer žogo prestreže soigralec
Straight-through-pass	Dolga vzdolžna podaja v prostor, kjer žogo prestreže soigralec
Diagonal-through-pass	Dolga diagonalna podaja v prostor, kjer žogo prestreže soigralec
Cross-pass	Diagonalna podaja v prostor, kjer žogo prestreže soigralec
Square-pass	Prečna podaja v prostor, kjer žogo prestreže soigralec
To-player	Podaja neposredno pred soigralca
Successful-shoot	Uspešen strel na gol
Movement	Premikanje
Offensive-movement	Napadalno premikanje (moštvo ima žogo)
Movement-on-the-ball	Premikanje z žogo
Long-dribble	Dolgo preigravanje z žogo
Speed-dribble	Hitro preigravanje z žogo
Control-dribble	Taktično preigravanje z žogo
Movement-off-the-ball	Premikanje brez žoge, ko ima moštvo v posesti žogo
Attack-support	Prestrezanje žoge
Creating-space	Premikanje, da ustvarimo primerno prostorsko ureditev
Undecided-movement	Nedoločljivo gibanje brez žoge
Run-for-ball	Tek igralca za mrtvo žogo
Defensive-movement	Obrambno gibanje (moštvo nima žoge)
Tackle	Odvzem žoge
Cover	Kritje
Defensive-support	Obrambna podpora
An-opponent	Kritje nasprotnikovega igralca
Passing-on	Kritje različnih nasprotnikovih igralcev
Tracking	Kritje fiksnega nasprotnikovega igralca
Area-of-the-field	Kritje igralnega področje
Intercept	Prestrezanje

Preglednica 32. Modelirane agentne akcije v domeni RoboCup.



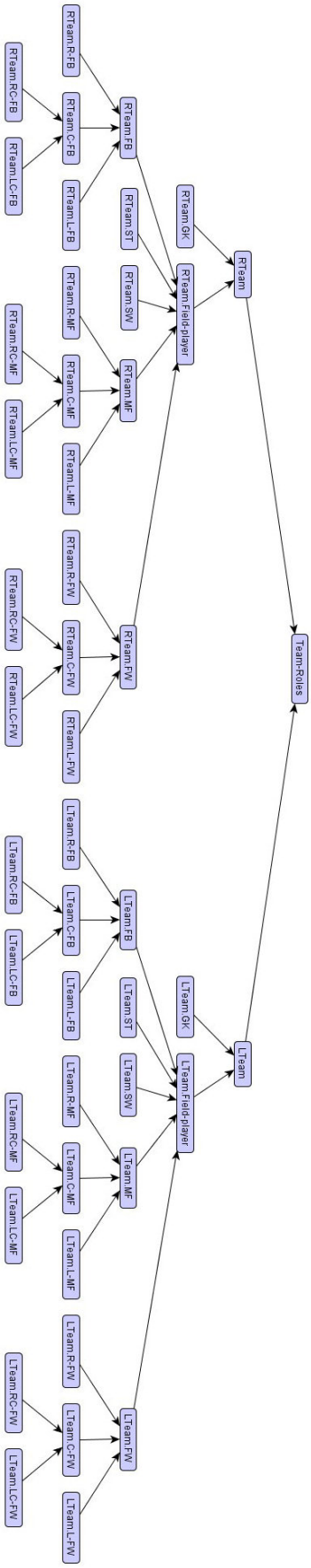
Slika 77. Hierarhija agentnih akcij v domeni RoboCup.



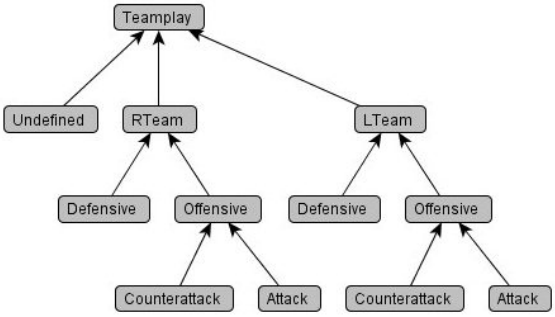
Slika 78. Poenostavljena hierarhija agentnih akcij v domeni RoboCup.

Oznaka (angleško ime)	Slovensko ime
GK (Goalkeeper)	vratar
FB (Fullback)	branilec
R-FB (Right fullback)	desni branilec
C-FB (Center fullback)	srednji branilec
RC-FB (Right-center fullback)	desnosrednji branilec
LC-FB (Left-center fullback)	levosrednji branilec
L-FB (Left fullback)	levi branilec
ST (Stopper)	prosti branilec
SW (Sweeper)	vezni igralec
MF (Midfielder)	sredinski igralec
R-MF (Right midfielder)	desni sredinski igralec
C-MF (Center midfielder)	srednji sredinski igralec
RC-MF (Right-center midfielder)	desnosrednji sredinski igralec
LC-MF (Left-center midfielder)	levosrednji sredinski igralec
L-MF (Left midfielder)	levi sredinski igralec
FW (Forward)	napadalec
C-FW (Center forward)	srednji napadalec
R-FW (Right forward)	desni napadalec
RC-FW (Right-center forward)	desnosrednji napadalec
LC-FW (Left-center forward)	levosrednji napadalec
L-FW (Left forward)	levi napadalec

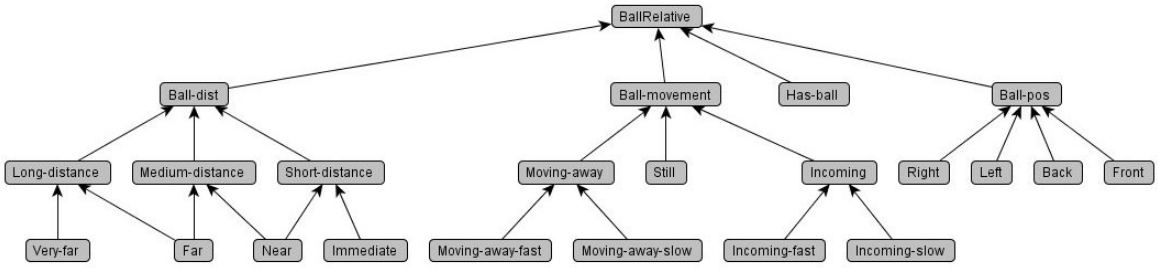
Preglednica 33. Uporabljene vloge igralcev v domeni RoboCup.



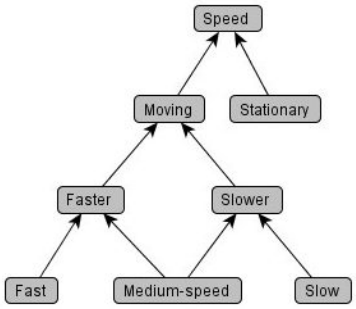
Slika 79. Hierarhija agentnih vlog v domeni RoboCup.



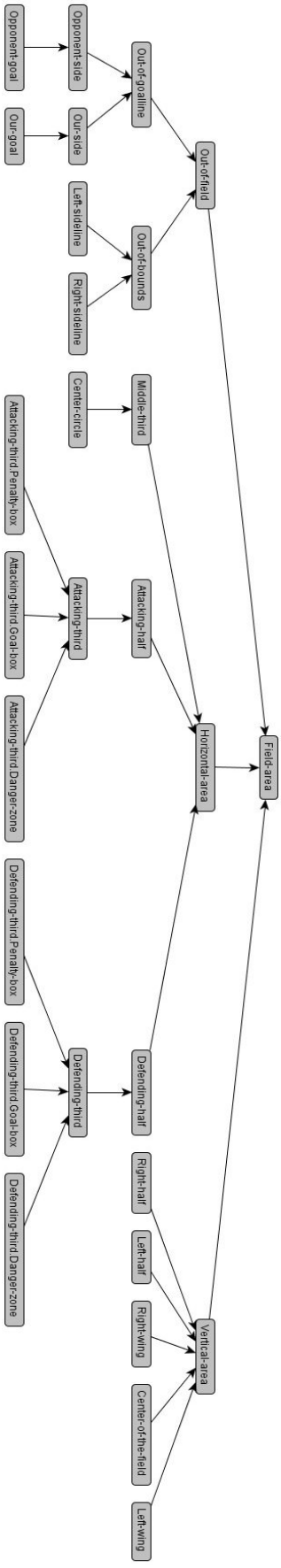
Slika 80. Hierarhija značilk za tip igre $H_{Teamplay}$.



Slika 81. Hierarhija agentne lastnosti $H_{BallRelative}$.



Slika 82. Hierarhija agentne lastnosti H_{Speed}.



Slika 83. Hierarhija agentne lastnosti $H_{FieldArea}$.