Slovenská technická univerzita
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Informatika

Robocup S - Nové stratégie

Analýza a hrubý návrh

verzia 0.3

História zmien

Dátum zmeny	Verzia dokumentu	Popis zmeny	Autor
6.1.2006	0.1	Prepis informácií z Wiki	Juraj Somorovský
14.11.2006	0.2	Finalizácia dokumentu	Juraj Staník
15.11.2006	0.3	Finalizácia dokumentu, prepracovanie architektúry	Martin Kováčik

Tím

Bc. Peter Kohaut	feshi@feshi.com
Bc. Martin Kováčik	mato.kovacik@gmail.com
Bc. Ladislav Lenčucha	ladislav.lencucha@gmail.com
Bc. Tomáš Selnekovič	tomas.selnekovic@mathsite.org
Bc. Juraj Somorovský	somimail@gmail.com
Bc. Juraj Staník	juraj.stanik@gmail.com

Webová stránka tímu:

http://www2.dcs.elf.stuba.sk/TeamProject/2006/team03

Tímový mailinglist

robocup@tequilla.kmit.sk

Anotácia

Tento dokument vznikol za účelom sumarizácie nadobudnutých znalostí o súťaži RoboCup. Informácie zahŕňajú

- Prehľad vybraných tímov, ktoré sa zúčastňujú súťaže RoboCup.
- Pravidlá a opis prostredia, ktoré je potrebné pre beh servera a klienta.
- Opis technológií a metód na riadenie správania hráča.
- Špecifikáciu navrhovaného hráča
- Hrubý návrh hráča, ktorý má byť implementovaný.

Dokument je určený záujemcom o súťaž RoboCup a študentom, ktorý sa zaoberajú multiagentovými systémami a strojovým učením.

Dokument vznikol ako sumarizácia informácií, ktoré sme umiestnili do tímového wiki systému. Informácie z wiki systému sú dostupné prostredníctvom webovej stránky tímu.

Obsah dokumentu

1 Úvod

RoboCup je svetový projekt umelej intelegencie a robotiky. Jeho snahou je odhaľovať nové vylepšenia týchto dvoch odvetví v praxi pri riešení štandardných problémov, pri ktorých je možné vyskúšať širokú škálu technológií. RoboCup si ako reprezentanta problémov vybral práve futbal. Pri futbale je potrebné riešiť mnohé problémy spojené s plánovaním, určovaním stratégie, vnímaním priestoru okolo seba a aj problémy spojené s tímovou spoluprácou a koordináciou akcií.

Dlhodobým cieľom hry RoboCup je vytvorenie tímu robotov do roku 2050, ktorí by dokázali vyhrať nad ľudskými futbalovými majstrami sveta. Na dosiahnutie tohto cieľa je potrebné zdolať množstvo prekážok a vyriešiť mnohé technologické problémy. Sú to predovšetkým dizajn robotov, ich senzory, multiagentové zmýšľanie a modely správania.

V rámci iniciatívy RoboCup vzniklo prostredie, ktoré umožnilo implementovať virtuálnych futbalových hráčov a organizovať zápasy medzi dvoma tímamy a tým simulovať futbalový zápas.

Cieľom nášho projektu je odhaľovanie nových stratégií pri implementácií hráča v prostredí RoboCup.

1.1 Znenie zadania

Téme RoboCup, presnejšie lige simulovaného robotického futbalu sa naši študenti venujú už sedem rokov. Tímy študentov, či už v rámci umelej inteligencie alebo tímového projektu, sa snažia vytvárať a vylepšovať programy, ktoré simulujú správanie sa futbalového hráča. Každý tím sa v rámci obmedzení, určených pravidlami hry futbal a špecifikami simulačného prostredia, snaží vytvoriť čo najlepšieho hráca. Mužstvo, vytvorené z takýchto hráčov, by malo vyhrať nad mužstvom súpera. O súťaži a doterajšej činnosti je dosť popísané aj na stránke STU turnaj v simulovanom robotickom futbale (www.fiit.stuba.sk/robocup).

V rámci fakulty sme realizovali viacero súťaží a posledné ročníky už boli oficiálnymi turnajmi iniciatívy RoboCup. Množstvo pozitívnych ohlasov nás priviedlo k vyhláseniu ďalšieho regionálneho turnaja RoboCup v simulovanej lige, opäť na záver akademického roka. Práve množstvo nových prístupov a riešení, ktoré predviedli nielen študenti tímového projektu, ale aj študenti umelej inteligencie, nám ukázalo, že možnosti na vylepšovanie hráča nie sú zďaleka vyčerpané a dokonca sa stále rodia prekvapujúce úspešné riešenia. V tomto roku sme preto ako podnázov vybrali "Nové stratégie". Znamená to všeobecne hľadanie nových prístupov a stratégií nielen pre hráča, ale aj vo svojej práci, v úpravách zdrojového kódu, podporných aplikáciách, základných aj vyšších schopnostiach hráča, spôsobe učenia a ladenia počas simulácií. Nové stratégie sú komplexnou výzvou do nového kola víťazstiev!

Na spresnenie je vhodné povedať, že v tomto tímovom projekte budeme rozširovať možnosti a vylepšovať správanie sa hráčov, vytvorených vo vlaňajších tímových projektoch. Využije sa existujúci zdrojový kód, dokumentácia a aj vytvorené podporné aplikácie. Musí sa tiež zachovať (a podľa možností aj zlepšiť) modularita a tým aj rozšíriteľnosť hráča. Zimný semester je vyhradený na oboznámenie sa s celým prostredím, najmä existujúcimi hráčmi a návrhu a prototypovej realizácii jeho vylepšení. Očakáva sa najmä návrh nových prístupov a stratégií vo všetkých už spomenutých oblastiach. Vybrané prístupy sa overia vytvorením jedného alebo viacerých prototypových rozšírení existujúceho kódu. Dôležitou súčasťou bude vytvorenie plánu implementácie a overovania nových stratégií v nasledovnom semestri. V letnom semestri nás čaká realizácia navrhnutých prístupov a stratégií a ich overovanie. Produkt by mal byť dohotovený v

deviatom až desiatom týždni semestra, potom je potrebné venovať sa ladeniu a optimalizácii hráča na súťaž, ktorej výsledky idú do celkového hodnotenia tohto projektu.

1.2 Prehľad dokumentu

Druhá kapitola opisuje vybrané existujúce tímy. Cieľom je oboznámenie sa s existujúcimi riešeniami a analýza existujúcich metód potrebných pre implentáciu hráča. Súčasťou druhej kapitoly je opis servera používaného na simuláciu zápasov a opis komunikačného protokolu. Druhá kapitola obsahuje taktiež detailny opis architektúry hráča tímu Uva Trilearn.

Tretia kapitola prezentuje metódy, používané na rozhodovanie hráča. Zdôvodňuje ich vhodnosť a prezentuje aplikáciu na rozhodovanie hráča. Na konci kapitoly je prezentovaná navrhovaná architektúra hráča, v ktorej by mali byť využité prezentované metódy.

1.3 Konvencie používané v dokumente

1.3.1 Komunikácia medzi klientom a serverom

V dokumente používame zápisy správ protokolu servera RoboCup. Zápisy správ sú zapisované pätkovým písmom. Nasleduje ukážka komunikačnej správy so serverom.

```
(hear <Time> <Sender> „<Message>“)
```

Parametre danej správy sú uvádzané v zátvorkách ('<','>'). V prípade, že správa zaberá viac miesta ako jeden riadok, je rozdelená a vhodne odsadená do viacerých riadkov. Server neposiela takto odsadené správy.

V texte sú správy zapisované kurzívou pätkovým neproporciálnym písmom – Courier. Príklad správy v texte: *`hear`*.

1.3.2 Nastavenia a zdrojové kódy

Pri opise správania hráčov sa využívajú referencie na konfiguráciu servera. Referencie v texte sú uvádzané kurzívou, zvýrazneným pätkovým neproporciálnym písmom – Courier. Príklad referencie na konfiguračnú premennú servera: ***`say_msg_size`***.

Zápis odkazov na zdrojový kód – názvy funkcií, premenných – je uvádzaný v texte pomocou neproporcionálneho pätkového písma. Príklad odkazu na zdrojový kód: `printf`.

1.3.3 Definície pojmov

V texte na opis pojmov sú používané definície. Definácie sú formátované podľa nasledujúceho príkladu:

Názov definície

Telo definície, ktoré obsahuje jej opis.

2 Analýza existujúcich tímov, metód a prostredia RoboCup

2.1 Analýza RoboCup Soccer Servera

SoccerServer poskytuje prostredie pre simuláciu futbalového zápasu. Uchováva si informácie o stave hry, rýchlosti a pozícií lopty aj hráčov, ktorým tieto informácie poskytuje. Kontroluje tiež dodržiavanie pravidiel hry. Komunikácia medzi serverom a hráčmi je implementovaná prostredníctvom sieťového rozhrania nad protokolom UDP.

2.1.1 Komunikačné protokoly medzi Soccer Serverom a klientom (hráčom)

Hráč komunikuje so serverom cez nasledovné protokoly[1]:

- Príkazový protokol (*command protocol*) – zabezpečuje pripájanie, odpájanie na server a ovládanie hráča.
- Vnemový protokol (*client sense protocol)* - vykonáva sa tu len komunikácia medzi serverom a hráčom (klient). Server zasiela hráčovi informácie o virtuálnom zápase – rozmiestnenie hráčov, strategických bodov, informácie o polohe lopty a hráčové stavové informácie.

2.1.2 Reprezentácia stavu zápasu a stavu hráča – Vnemový model

Stav je aktualizovaný pomocou zasielaných správ zo Soccer Servera. Kvôli abstrakcií je zavádený pojem vnemový senzor z dôvodu lepšej názornosti. Správy, ktoré sú súčasťou vnemového protokolu, sú vnemy pre špecifický vnemový senzor zo servera. Správy budú v tejto časti v špecifických kontextoch zamieňané za vnemy.

Hráč v simulácii vníma svet cez tri typy senzorov. Sluchový senzor zachytáva správy od rozhodcu, trénera a ostatných hráčov. Vizuálny senzor získava vizuálne informácie o stave na ihrisku, ako sú napríklad vzdialenosti a rýchlosti hráčov, lopty, rozmiestnenie čiar a pod. Telový senzor vníma aktuálny stav hráča, teda sleduje hodnotu *stamina* (ekvivalent k vyčerpanosti), rýchlosť a natočenie hlavy vzhľadom na telo.

2.1.2.1 Sluchový senzor

Zachytáva správy poslané hráčom alebo trénerom po zaslaní príkazu `say`, a tiež správy zasielané rozhodcom. Všetky správy sú prijaté okamžite.

Formát správy sluchového vnemu zasielaného serverom je nasledovný:

```
(hear <Time> <Sender> „<Message>“)
```

Správa sluchového vnemu zasielaného zo servera sa skladá z týchto položiek:

- *Time* indikuje aktuálny čas
- *Sender* je relatívny smer k odosielateľovi správy, ak ním je iný hráč. Inak nadobúda jednu z

1 Formáty správ jednotlivých protokolov na komunikáciu medzi klientom a soccer serverom sú uvedené v používateľskom manuály k serveru. Staršia verzia protokolu je dostupná na [RSSManual]

nasledovných hodnôt

- *self* - vlastná správa. Hráč počuje aj správy, ktoré on sám vysiela
- *referee* – Odosieľateľom správy je rozhodca
- *online_coach_left* a *online_coach_right* – Odosieľateľom je jeden z čiarových trénerov.

- *Message* obsahuje samotnú správu. Maximálna dĺžka správy je ***say_msg_size***[2] bajtov.

2.1.2.2 Zrakový senzor

Zachytáva a poskytuje informácie o objektoch, ktoré sú hráčom aktuálne videné. Vizuálna informácia je hráčovi posielaná zakaždým po uplynutí doby *sense_step*, čo je 150 milisekúnd.

Formát správy vizuálneho vnemu zasielaného serverom je nasledovný:

```
(see <ObjName>
     <Distance>
     <Direction>
     <DistChng>
     <DirChng>
     <BodyDir>
     <HeadDir>)
```

kde

- *ObjName* identifikuje videný objekt (hráč | zástavka | tyčka | …)
- *Distance, Direction, DistChng, DirChng, BodyDir, HeadDir* sú informácie o relatívnej vzdialenosti, smere, natočení hráča a pod.

S rastúcou vzdialenosťou objektu je doručované množstvo informácií obmedzované. Od určitej vzdialenosti objektu správa neobsahuje číslo hráča *UNum*. Ak je objekt príliš vzdialený správa tiež neobsahuje *TeamName* (príslušenstvo k tímu) a *Direction*.

Informácia je posielaná hráčovi periodicky v pevne stanovenom časovom kroku. Kvalitu pohľadu možno z pohľadu hráča zmeniť a to pomocou funkcie

```
(change_view (<AngleWidth>) (<Quality>))
```

AngleWidth musí byť jedna z hodnôt

- *wide* = 180°
- *normal* = 90°
- *narrow* = 45°

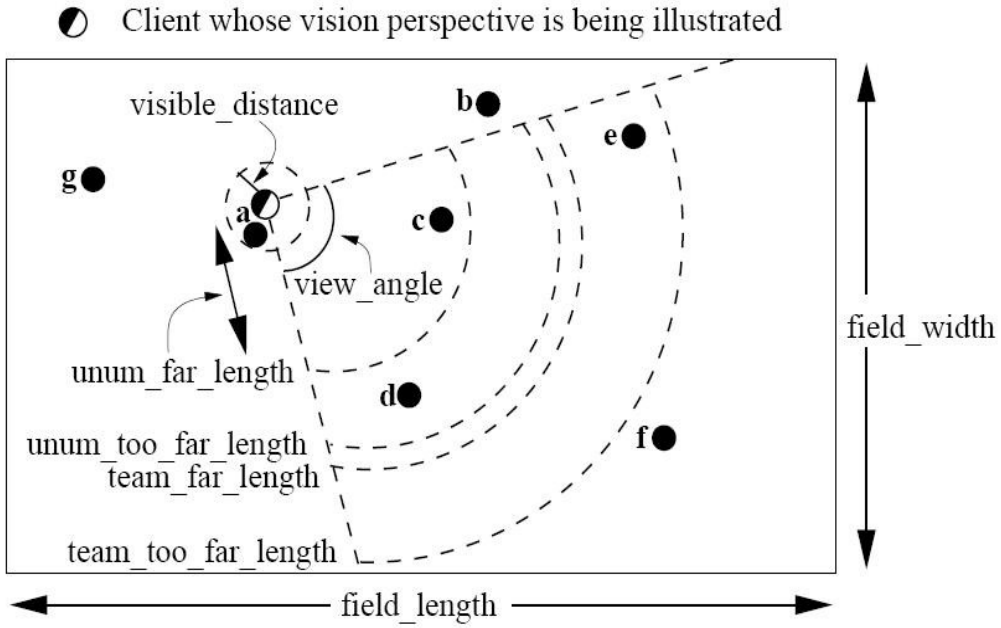
Quality musí byť

- *low* (je posielaná iba informácia o smere)
- *high* (klient dostane informáciu aj o smere aj o vzdialenosti).

Implicitné hodnoty parametrov sú *normal* a *high*. Hráč vníma aj virtuálne značky na hracom poli (bránka, rohy, stred ihriska a ďalšie). Hráč čiastočne vidí aj objekty, ktoré nie sú v jeho zornom uhle

2 Konkrétna hodnota servera je popísaná v dokumentácii ku zvolenej verzii servera. Hodnota je taktiež súčasťou oficiálnych pravidiel zápasov RoboCup.

(a sú blízko) – v tomto prípade vníma len typ objektu, ale nie jeho exaktné meno. Zorné pole hráča je znázornené na obrázku 2.1.



Obrázok 2.1: Pohľad hráča a premenné konfigurácie servera, ktoré ho ovplyvňujú

2.1.2.3 Telový senzor

Telový senzor umožňuje hráčovi získať informácie o svojom vlastnom stave. Na predávanie informácií o stave je využívaná nasledujúca správa

```
(sense_body
     (<TIME>)
     (view_mode <QUALITY> <WIDTH>)
     (stamina <STAMINA_EFFORT>)
     (speed <AMMOUNT_OF_SPEED>)
     (kick <KICK_COUNT>)
     (dash <DASH_COUNT>)
     (turn <TURN_COUNT>)
     (say <SAY_COUNT>))
```

Tu sú určené všetky parametre hráča ako napríklad energia, rýchlosť, natočenie tela, hlavy. Informácia je posielaná periodicky v pevne stanovenom časovom kroku.

2.1.3 Módy hry a riadiace príkazy

Prostredie RoboCup definuje herné módy. Tieto identifikujú jednotlivé časti zápasu – rozohratie, prestávka, samotná hra a i.

Módy hry sú nasledovné:

Play_on

Je "hrací" mód, pri ktorom je lopta v pohybe. Nastane ak sa lopta začne pohybovať vďaka príkazu `kick` z ľubovolného iného hracieho módu (`kick_off`, `throw_in`, `goal_kick`, `corner_kick`).

Kick_off

Rozohranie lopty, pri ktorom musí byť každý hráč na svojej strane. Ak sa na nej nenachádza, simulátor ho umiestni na náhodné miesto na príslušnej polovici hracieho poľa.

Presunutie

Nastáva pri rozohrávaní lopty (`kick_off`, `throw_in`, `corner_kick`, `goal_kick`, `offside`). Simulátor presunie hráčov tak, aby boli minimálne vo vzdialenosti 9.15 od lopty. Hráči sú presunutí na obvod pomyselnej kružnice.

Gól

Nastane v momente, keď sa lopta dostane celým objemom do brány. Simulátor preruší zápas na 5 sekúnd, presunie loptu do stredu ihriska, pošle správu všetkým hráčom. Počas prerušenia sa hráči môžu presunúť na svoju polovicu pomocou príkazu 'move'. V opačnom prípade ich presunie simulátor a umiestni ich na náhodnej pozícii. Simulátor zmení hrací mód na *kick_off* .

Aut

Nastane, keď sa lopta dostane mimo hracieho poľa. Simulátor presunie loptu na príslušnú pozíciu (roh, aut, bránková čiara) a zmení hrací mód na *kick_in, corner_kick, goal_kick.*

Polčas

Simulátor preruší zápas v polovici plánovaného času trvania. Implicitná dĺžka polčasu je 3000 simulačných cyklov (okolo 5. min). Pokiaľ je po skončení zápasu skóre nerozhodné, predlžuje sa až kým sa nedosiahne gól – tzv. rýchla smrť.

Hráč môže v jednotlivých herných módoch vykonávať akcie. Každá akcia je reprezentovaná správou medzi hráčom a serverom. V nasledujúcom texte sú popísané najdôležitejšie riadiace funkcie:

- `move(<X> <Y>)`

  Presun hráča na pozíciu (*X,Y*). Stred súradnicovej sústavy je v centrálnom kruhu, kladný smer osi *X* je určený smerom k súperovej bráne, kladný smer osi *Y* je určený smerom k pravej čiare (ktorá je určená vzhľadom na kladný smer osi *X*). Tento príkaz je k dispozícii iba v móde *before_kick_off.*

- `dash <Power>`

  Zrýchlenie hráča v smere jeho orientácie Z o hodnotu *Power. Power* môže byť z intervalu –30 až 100.

- `kick <Power> <Direction>`

  Kopnutie do lopty silou *Power* a so smerom *Direction*. Možné iba ak je hráč bližšie ako *kickable_margin* + *ball_size* + *player_size*. *Power* môže byť z intervalu -180 až 180 a *Direction* z intervalu -180 až 180.

- `catch <Direction>`

  Príkaz môže použiť iba brankár a slúži na chytenie lopty v smere *Direction*. Loptu môže

chytiť iba vtedy, ak sa nachádza v obdĺžniku s rozmermi *goalie_catchable_area_w x goalie_catchable_area_h* a v smere *Direction*. Ak chytenie nebolo úspešné, určitý čas nie je možné akciu chytenia lopty znovu opakovať.

2.2 Analýza existujúcich tímov

Predpokladom pre realizáciu projektu je využitie už naimplementovanej funkcionality z iných tímov. Nasleduje analýza vybraných domácich a zahraničných tímov. Účelom analýzy je sumarizácia metód používaných na modifikáciu správania hráča.

2.2.1 L.A.S.T. United

Informácie o tíme boli získané z [LASTWWW]. Tento tím bol založený v rámci tímového projektu v školskom roku 2003/2004 a zo 4 súťažných tímov sa umiestnil na konečnom poslednom mieste. Ako základ pre tvorbu slúžil tím Stjupit dox z predchádzajúceho školského roku. Bol implementovaný v prostredí Windows v jazyku LUA. Dokumentácia bola slabá a obsahovala len základné časti.

Tím sa chcel venovať vytvoreniu najvyššej rozhodovacej úrovne ale nakoniec prepísal celého hráča do jazyka LUA a teda samotnému vylepšovaniu hráča sa nevenoval dostatočne.

2.2.1.1 Komunikácia hráčov

Komunikáciu hráčov riešili cez tzv. spojára, ktorý stál v strede ihriska a počúval hráča označeného ako vedúci (automaticky označovaný hráč s loptou) a tieto príkazy vykríkol v ďalšom kroku. Komunikácia mala byť určená na tvorbu formácií a priraďovanie hráčov na rôzne úlohy, ale neskôr bola upravená tak, že po sebe kričali iba kto a kam nahráva.

2.2.1.2 Formácie

Vo svojej práci opisujú 6 základných typov formácií, ktoré je možné ďalej využiť.

2.2.1.3 Zisťovanie situácie

Situáciu na poli chceli určovať pomocou kvadratických stromov a tak zistiť v akej formácii sa momentálne hráči nachádzajú.

2.2.2 FIITMedia

Informácie o tíme boli získané z [FIITWWW]. Tím bol založený v rámci tímového projektu v školskom roku 2005/2006 a z 10 súťažných tímov sa umiestnil na prvom mieste. Ako základ pre tvorbu slúžil tím FC Farmári z predchádzajúceho školského roku. Dokumentácia je prehľadná a dostatočná. Nachádzajú sa tu rôzne grafy na porovnanie jednotlivých možností na základe percentuálnej úspešnosti. Za implementačné prostredie bolo zvolené C++ (Visual studio), ktoré použili i FC Farmári a Stjupit dox.

2.2.2.1 Komunikácia hráčov

Hráči pracujú na základe akcií, ktoré organizuje najbližší hráč k lopte. Hráč pri lopte vždy oznámuje ostatným hráčom aká akcia nasleduje (ak rozozná nejaký vzor).

2.2.2.2 Formácie

Formácie sa realizujú upravenou neurónovou sieťou tímu FC Farmári. Pre každú jednu z hlavných formácií bola realizovaná sieť tak, aby sa lepšie správali proti neznámemu súperovi. Prerobili i systém nahrávania najbližšiemu spoluhráčovi na výber náhodného spoluhráča smerom dopredu.

2.2.2.3 Vyhodnocovanie situácie

Prihrávanie sa realizuje po rozhodnutí hráča (na základe odmeny a trestu) a kouča (po analýze formácií súpera). Vyhodnocovanie situácie je založené na rozoznávaní vzorov a súvisí s akciami (komunikácia hráčov).

2.2.2.4 Učenie sa hráča

Hráč si pomocou pseudonáhodného rozdelenia vyberá jednu neurónovú sieť, ktorú používa. Každej sieti je pridelená pravdepodobnosť použitia. Hráč analyzuje úspešnosť prihrávky a na jej základe zvýši alebo zníži pravdepodobnosť výberu siete. Ďalšej neurónovej sieti, ktorá má rovnaké vstupy ako ostatné, vyberie sieť, ktorú použije. Na základe úspešnosti sa táto "hlavná sieť upravuje".

2.2.2.5 Hráči

Tím sa vo svojej práci venoval hlavne rozvoju útočnej fázy a nahrávok. Navrhol hráča – libera, ktorý ľubovoľne behá po ihrisku medzi obranou a útokom na základe aktuálnej situácie a v prípade potreby robí "nátlak" v útoku.

Tím analyzoval prihrávky, pri ktorých kladie dôraz hlavne na presnosť. Základom na rozhodnutie sa v danej situácii slúžia tieto údaje:

- vzdialenosť hráča, ktorému nahráva
- smer pohybu hráča
- typ prihrávky, ktorá sa má realizovať

2.2.2.6 Percepčná časť

Hráč rozoznáva, či je vedľa neho súper ako okolie, v ktorom súper buď je, alebo nie je.

2.2.3 SKLO

Tím SKLO [SKLOWWW] bol založený v rámci tímového projektu v školskom roku 2003/2004. Zo 6 súťažných tímov sa umiestnil na konečnom druhom mieste. Ako základ mu slúžili zahraničné tímy FC Portugal a CMUnited99. Pri implementácii bol použitý jazyk C++ a vývojové prostredie MS Visual Studio 6.0. Štruktúra súborov, ktoré sú voľne dostupné na internete, je veľmi prehľadná.

2.2.3.1 Architektúra hráčov

Pri architektúre hráča bola použitá trojvrstvová a modulárna architektúra. Využívané sú nasledujúce vrstvy:

- *Stratégia a rozhodovanie* – zabezpečuje ju riadiaci modul. Riadenie je implementované pomocou rozhodovacích stromov, znalosti o formáciách, subformáciách, herných situáciách a manažment vedomostí.
- *Taktika a reprezentácia vnútorného sveta*
- *Časovanie a komunikácia.* Táto vrstva je založená na zdrojových textoch tímu CMUnited99.

V hráčovi boli nízkoúrovňové zručnosti prebrané z tímu CMUnited99 [CMUnited99WWW], pri vedení lopty a strieľaní boli prevedené minimálne modifikácie. Takisto je prebraná reprezentácia vnútorného sveta.

Zameranie tímu bolo hlavne na vylepšenie funkcií najvyššej rozhodovacej úrovne a sociálne zručnosti (koordinácia, komunikácia medzi hráčmi).

2.2.3.2 Metódy využívané tímom SKLO a postup realizácia niektorých akcií

2.2.3.2.1 Simulation Based Strategic Positioning

V situáciách, ktoré nie sú kritické, sa hráči pokúšajú dostať do strategických pozícii generovaných na základe aktuálnej situácie na ihrisku a pravidiel dostupných pre všetkých agentov. Za pomoci detailnejšej analýzy hernej situácie vieme predvídať známe situácie a predpovedať pozície ako našich hráčov tak i hráčov súpera – bez vizuálnej a zvukovej informácie.

2.2.3.2.2 Dynamic Positioning and Role Exchange

Prebraté z tímu CMUnited99, umožnená výmena rolí jednotlivých hráčov, ak je tým umožnený globálny úžitok pre celý tím. Hráči sú homogénni, zmena je oznámená všetkým agentom.

2.2.3.2.3 Inteligentný komunikačný mechanizmus

Inteligentný komunikačný mechanizmus spočíva v tom, že agent sa rozhoduje, či bude komunikovať na základe situácie (ako napríklad zmena rýchlosti lopty a podobne). Agent komunikuje len vtedy, ak verí, že je užitočnejšie, aby komunikoval on a nie jeho spoluhráči.

2.2.3.2.4 Strategický pozorovací mechanizmus

Aby sa dosiahol čo najpresnejší obraz sveta, agent sa snaží určiť čo najlepší uhol natočenia na základe jemu známych pozícií ostatných hráčov a známych pozícií a rýchlosti objekov. Napr. ak hráč kontroluje loptu a je blízko súperovej brány, je preňho najdôležitejšia pozícia a rýchlosť brankára, pričom pozícia hráčov v strede je nezaujímavá.

2.2.3.2.5 Technika pokrývania súperových prihrávok

Ak agent spozoruje, že možná dráha súperovej prihrávky prechádza blízko jeho strategickej pozície, začne sa pohybovať, aby túto hrozbu odvrátil. Pri rozohrávke brankára používa kouča, aby informoval hráčov, kde sa nachádzajú súperovi hráči (táto technika je prebratá z tímu FC Portugal).

2.2.3.2.6 Modelovanie súpera

Zabezpečuje ho samostatný modul. Modelovanie pre trénera je vysokoúrovňové, modelovanie pre hráča je nízkoúrovňové. Po analýze hry sa pomocou strojového učenia a heuristiky určujú parametre súpera – súperova globálna formácia, jeho pozícia, rýchlosť, odhad o jeho parametroch a spôsobe výkopov, atď. Dôsledkom uplatnenia týchto princípov je kvalitná hra. No niekedy sa môže rýchlo zmeniť taktika a má to za následok celkové zhoršenie spôsobu hry.

2.2.3.2.7 Driblovanie

Na určenie cieľu driblovania je vytvorená samostatná dynamická funkcia určujúca cieľ hráča driblujúceho s loptou. Vyberá sa pri tom taký smer, v ktorom je najmenej súperov. Funkcia zistí počet hráčov v dolnom aj hornom kvadrante vo vzdialenosti do 10 m. Ak sa v žiadnom kvadrante nenachádza súper, beží hráč priamo, posúva sa o 10 metrov dopredu. Inak sa presunie do stredu toho kvadrantu, kde je menej hráčov. Toto funguje, keď je hráč s loptou bližšie ako 35 metrov od brány. V opačnom prípade sa hráč orientuje podľa toho, v akej šírke ihriska sa nachádza a snaží sa viesť loptu na pozíciu bližšiu ku stredu ihriska.

2.2.3.2.8 Miesto núdzového kopu

Algoritmus vyberá smer, ktorým má hráč kopnúť v prípade, že je okolo neho nebezpečne veľa súperov. Najprv si zistí, či je voľná úsečka, na ktorej by sa mohol nachádzať spoluhráč. Ak táto nie je voľná, zisťuje sa priestor smerom k súperovým rohom. V najposlednejšom prípade vracia funkcia ako cieľ kopu vnútorný okraj súperovej tyčky.

2.2.3.2.9 Plánovanie výkopu

Keď nastane situácia, v ktorej má hráč prihrať, správa sa hráč nasledovne:

- otočenie okolo osi (preskúmanie výkopu)
- postavenie sa čo najbližšie k lopte
- počkanie 10 simulačných krokov na spoluhráčov
- výber spoluhráča
- zakričanie správy Chytaj, otočenie sa k nemu a prihranie

Správanie sa útočníka s loptou

Pri prebratí lopty sa najprv vždy zakričí správa Mám. Po nej sa zruší beh na pozíciu, ktorá mala byť naplánovaná. Ďalej sa zrealizuje jedna z nižšie uvedených akcií podľa poradia:

- výkop
- ak je na dostrel od brány, tak strieľa
- ak je niekto vpredu, komu môže útočník prihrať, radšej prihráva
- ak je vpredu a môže rýchlo driblovať, volí sa rýchly dribling
- bezpečný dribling
- inak je lopta kopnutá na miesto núdzového výkopu

Stredopoliar s loptou

Priorita akcií:

- streľba na bránu
- rozohranie priameho kopu
- ak môže driblovať, natočí krk a dribluje
- hľadá voľného spoluhráča, ak je taký, nahráva mu
- inak prihrávka do voľného priestoru

Obranca s loptou

Podobne ako pri útočníkovi sa vyberajú tieto úlohy:

- ak je možné kopať na bránu – strieľa na bránu
- prihrávka spoluhráčovi dopredu
- driblovanie
- určenie pozície na prihrávku na základe formácie a následná nahrávka

2.2.4 Stjupit Dox

Tento tím [SDOGWWW] bol založený v rámci tímového projektu v školskom roku 2002/2003 a zo 4 súťažných tímov sa umiestnil na konečnom druhom mieste. Ako základ pre tvorbu slúžil tím Roztoče z predchádzajúceho školského roku. Implementácia je robená pod Windows.

2.2.4.1 Komunikácia hráčov

Riešili problém so zavedením nového pravidla, pri ktorom môže hráč vyslať najviac 10 B pokriky. Za základné si stanovili pokriky: “Za tebou”, “Pusti”, “Prihraj”, “Prenes”(od tretieho hráča), “Vystrel”, “Potiahni”, “Máme” / ”Chyť ju”.

2.2.4.2 Učenie sa hráčov

Bolo použité učenie sa hráčov s odmenou určenej pomocou predchádzajúcej akcie hráča. Na ohodnotenie situácie bola použitá dvojvrstvová neurónová sieť. Vstupom do siete bol vektor opisujúci okolie hráča a akcia, výstupom bolo ohodnotenie akcie.

2.2.4.3 Kouč

Tím, z ktorého vychádzali, vylepšili takisto o kouča, ktorý dokáže sledovať hru (dobu ovládania lopty tímami, počet nepodarených prihrávok, počet prebraných prihrávok, rozostavenie súpera atď) a na jej základe meniť taktiku: zmena formácie, taktiky, typu hráča, parametrov hráča (napr. zníženie čerpania energie).

2.2.4.4 Hráč

Rýchlejší pohyb s loptou na úkor presnosti sa tímu osvedčil. Pri dostatočnej blízkosti od brány hráč zistí polohu brankára, algoritmom vypočíta, či je mu možné dať gól. Ak áno, vystrelí.

2.2.4.5 Brankár

Oproti predchádzajúcemu tímu bránkára značne vylepšili (tím Roztoče totiž používal ako brankára obyčajného hráča, čo viedlo k mnohým chybným riešeniam situácií). Bol implementovaný pohybu dozadu, pri ktorom môže brankár sledovať loptu. Na určenie výhodnej pozície v bráne bol urobený nový algoritmus. V prípade, že má brankár loptu na svojej kopačke, okamžite sa ju snaží prihrať voľnému hráčovi alebo odkopnúť do autu.

2.2.5 Deravá kopačka

Tím [DEKOWWW] pokračoval vo vývoji hráča tímu Roztoče. Architektúru svojho hráča postavili na troch vrstvách:

- *Vrchná vrstva* (stratégia)
- *Stredná vrstva* (taktika)
- *Spodná vrstva* (časovanie a komunikácia).

Spodná vrstva generuje pri každej zmene nový pohľad na svet, pričom staré pohľady, ktoré sa práve používajú, zostávajú. V systéme sú možné maximálne tri pohľady: prvý je využívaný strategickou vrstvou, druhý je posledný a aktuálny, tretí je práve vytváraný pohľad spodnou vrstvou. Stredná vrstva bola doplnená o ďalšiu funkcionalitu. Vrchná vrstva bola vytvorená od základov, keďže predošlý tím túto vrstvu vo svojom hráčovi neuvažoval.

2.2.5.1 Taktická vrstva

Tím označil za hlavný problém ekvivalentné hodnotenie rôznych akcií tak, aby ich hodnotenia boli porovnateľné. Spôsob výberu akcií je riešený sekvenčným systémom – najdôležitejšie akcie dostávajú príležitosť na začiatku. Ak je takáto akcia uskutočniteľná a dosahuje potrebnú užitočnosť, tak sa spustí bez ohľadu na to, že menej dôležitá akcia môže mať vyššiu užitočnosť.

2.2.5.2 Strategická vrstva

Vytváranie jedného plánu pre hráčov a následné určenie rolí pre hráčov, ktoré majú v pláne vykonať.

2.2.5.3 Obrana

Bola prepracovaná od základov. Bola zapracovaná podpora obrancov hráčmi zo stredu poľa. Hlavná úloha obrancu je stáť medzi útočníkom a bránou a brániť útočníkovi v pohybe, prihrávaní a streľbe. Tím však vytvorenú obranu nepovažuje za dokonalú, ale len za postačujúcu.

2.2.5.4 Komunikácia a plánovanie

Snahou je nájsť spôsob, ako účinne môže hráč vytvárať plán. Z toho dôvodu boli analyzované spôsoby, ako môžu hráči medzi sebou čo najúčinnejšie komunikovať. Tiež bol opísaný komunikačný protokol, ktorý zabezpečuje komunikačnú synchronizáciu medzi jednotlivými hráčmi.

2.2.6 UvA Trilearn

Tím [UVATWWW] bol založený v roku 2001. Naprogramovali ho dvaja študenti Amsterdamskej univerzity ako diplomový projekt. Boli nespokojní s vlastnosťami predchádzajúcich tímov, preto sa ho rozhodli preprogramovať všetko (až na komunikačné funkcie) odznova a vytvoriť novú architektúru hráča.

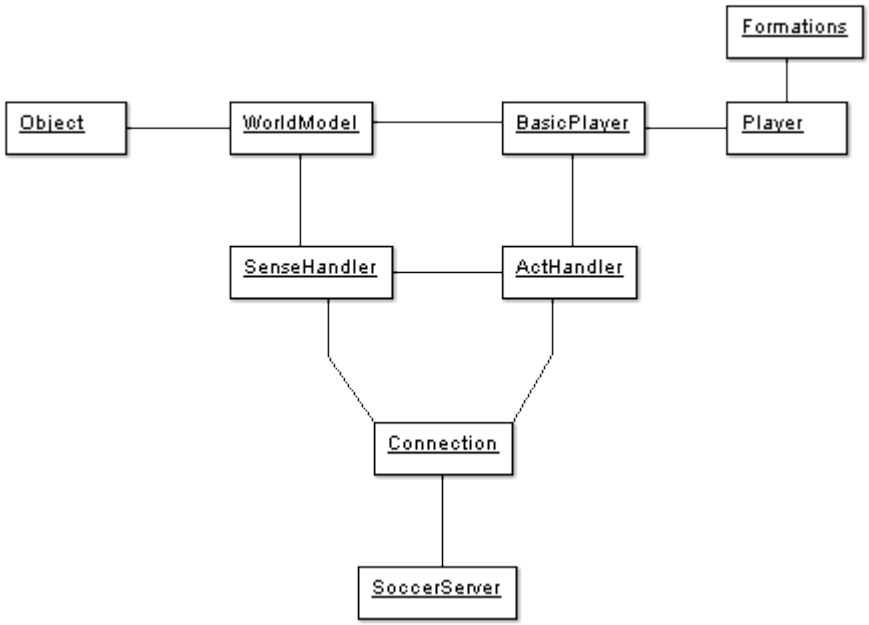
Zo začiatku implementovali základné vlastnosti hráča, neskôr zdokonalovali rozhodovacie funkcie. Z ukážok na videu je tento tím veľmi schopný a poráža väčšinu tímov založených na pôvodných zdrojových textoch. Použitý jazyk je C++, štruktúra súborov je jednoduchá. Zdrojové texty sú veľmi prehľadné, idú pod operačnými systémami Windows aj Linux. Posledne dostupné zdrojové texty sú z roku 2003.

Na základe výsledkov z turnajov a našej analýzy tohto tímu sme sa ho rozhodli použiť ako základ nášho projektu. Jeho analýza je preto obšírnejšia a zachádza do detailov. Keďže zdrojové texty tímu nie sú kompletne dostupné na internete, budeme musieť niektoré časti doplniť z iných tímov alebo nanovo implementovať.

2.2.6.1 Architektúra hráča

Pri návrhu agenta sa rozlíšili 3 hlavné úlohy - vnímanie, rozhodovanie, konanie. Z toho vyplynula viacvláknová architektúra a pre každú úlohu samostatné vlákno.

Aktuálna architektúra hráča tímu vyzerá nasledovne:



Obrázok 2.2: Architektúra hráča tímu UvaTrilearn spolu so základnými komponentami

2.2.6.1.1 Trieda Connection

Táto trieda vytvára spojenie so serverom cez UDP socket. Komunikuje s daným hostname na danom porte. Akonáhle sa vytvorí spojenie je možné správy prijímať a odosielať. Je založená na

vzorovom klientskom programe distribuovanom so Soccer serverom. Prijímanie správ je riešené slučkou, ktorá dookola volá funkciu na prijatie jednej správy. Prijatie jednej správy blokuje program pokým nepríde správa. Preto sa celá slučka ako funkcia volá v inom vlakne.

2.2.6.1.2 Trieda SenseHandler

Úlohou tohto komponentu je získavanie a úvodné spracovávanie informácií, ktoré chodia zo serveru.

SenseHandler je implementovaný ako samostatná trieda a pracuje ako samostatné vlákno. Kód vlákna je implementovaný v metóde handleMessagesFromServer.

2.2.6.1.3 Trieda ActHandler

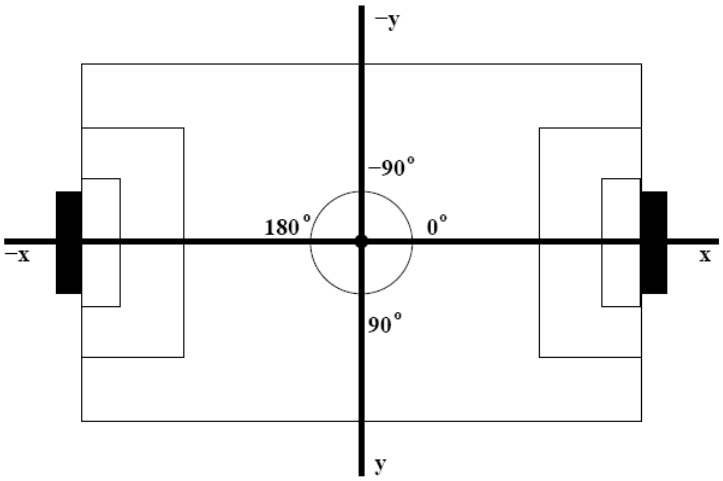
Trieda ActHandler sa používa na posielanie správ Soccer serveru. Obsahuje rad, do ktorého sa vysielané správy postupne zaradzujú. Keď príde signál (v závislosti od objektu SenseHandler) tak správy sa konvertujú na text a odosielajú serveru. Poslané správy sa synchronizujú s objektom WorldModel, tak aby odzrkadloval skutočný stav na základe realizovaných akcií. V každom kroku je možné poslať viac správ naraz, ale niektoré správy môžu byť poslané (podľa pravidiel) iba raz (kopni, bež, pohni sa, chyť, otoč sa). Preto sú interne definované rôzne rady so správami. Prvý rad obsahuje iba jednu správu - posledný zadaný priamy príkaz, ktorý je reprezentovaný príkazom bez použitia poľa. Druhý rad obsahuje správy na zmenu pohľadu (change_view), kedže sa musia synchrónne posielať v daných intervaloch. Tretí rad obsahuje všetky ostatné správy. Druhý a tretí rad sú spoločne ukladané v statickom poli príkazov. Zakaždým ako sa do radu pridá správa, ktorá tam už je, tak sa príkaz nahradí touto novou správou. Správy je možné posielať priamo serveru, využíva sa to v 2 situáciach:

- pri inicializácii
- keď vieme, že správa, ktorú posielame je definitívna (že sa už nebude meniť), pretože žiadna ktorá by sa poslala nebude lepšia

Jednotlivé správy sú reprezentované inštanciami triedy SoccerCommand (opísana nižšie). Okrem toho ActHandler obsahuje odkaz na spojenie so serverom a odkaz na WorldModel.

2.2.6.1.4 Trieda WorldModel

Model sveta opisuje aktuálny stav okolia. Na tento model sa možno tiež pozerať ako na pravdepodobnostnú reprezentáciu vytvorenú na základe predošlých vnemov. Obsahuje rozličné informácie o všetkých objektoch na ihrisku ale aj niekoľko metód na odvodenie vysoko-úrovňových záverov.



Obrázok 2.3: Súradnicový systém

Hráč si musí byť vedomý vlastnej pozície, rýchlosti, rovnako aj pozície a rýchlosti ostatných hráčov a lopty. Uchováva si ich vo svojom modeli sveta. Tieto informácie dostáva cez tri typy senzorov: vizuálne, zvukové a telesné, ktoré sú mu posielané vo forme správ. Okrem senzorických informácií sú spolu s nimi posielané aj časové pečiatky, ktoré určujú čas správy. Uchovávaním tejto časovej pečiatky je potom hráč schopný rozlišovať medzi neaktuálnymi a aktuálnymi informáciami vo svojom modeli sveta.

Dôležitý poznatok je, že soccer server posiela hráčovi informácie o pozíciách a podobne vzhľadom na pozíciu hráča. Teda relatívne pozície. Tieto si musí hráč skonvertovať do globálnej podoby.

Ako už bolo spomenuté, model hráča možno považovať aj za pravdepodobnostný. V tom zmysle, že informácie o viditeľných objektoch sú aktualizované z vnemov, ktoré hráč dostáva. Informácie o objektoch, ktoré hráč nevidí sú tiež aktualizované na základe predošlých vnemov, spolu s *confidence* informáciou, ktorá je odvodená od časovej pečiatky informácie, z ktorej sa vychádza. Confidence hodnota klesá každým prechodom simulačným cyklom.

Aby bolo možné vybrať najvhodnejšiu akciu, poskytuje model sveta rozličné metódy, ktoré umožňujú používať informácie z modelu na vyšších úrovniach. Tieto metódy môžeme zaradiť do štyroch typov:

- *Získavacie/Vyhľadávacie* (retrieval) metódy: pre priame získavanie informácií z modelu sveta agenta.
- *Aktualizačné* (update) metódy: slúžia na aktualizáciu modelu sveta agenta podľa nových vnemových informácií.
- *Predpovedajúce* (prediction) metódy: Slúžia na predpovedanie stavu sveta na základe doteraz získaných vnemov.
- *Vysoko-úrovňové* (high-level) metódy: Slúžia na odvodzovanie úsudkov/záverov zo základných informácií o stave sveta (napr. určenie spoluhráča, ktorý sa najrýchlejšie dostane k lopte).

2.2.6.1.4.1 Atribúty

Atribúty *WorldModelu* tiež možno zadeliť do štyroch kategórií:

- informácie o okolí (environmental information)
- informácie o zápase (match information)
- o objekte (object information)
- informácie o akciách (action information)

Informácie o okolí

Ide o informácie, ktoré sú špecifické pre prostredie poskytované serverom. Tieto atribúty reprezentujú rozličné parametre servera a parametre pre heterogénne typy hráčov.

Informácie o zápase

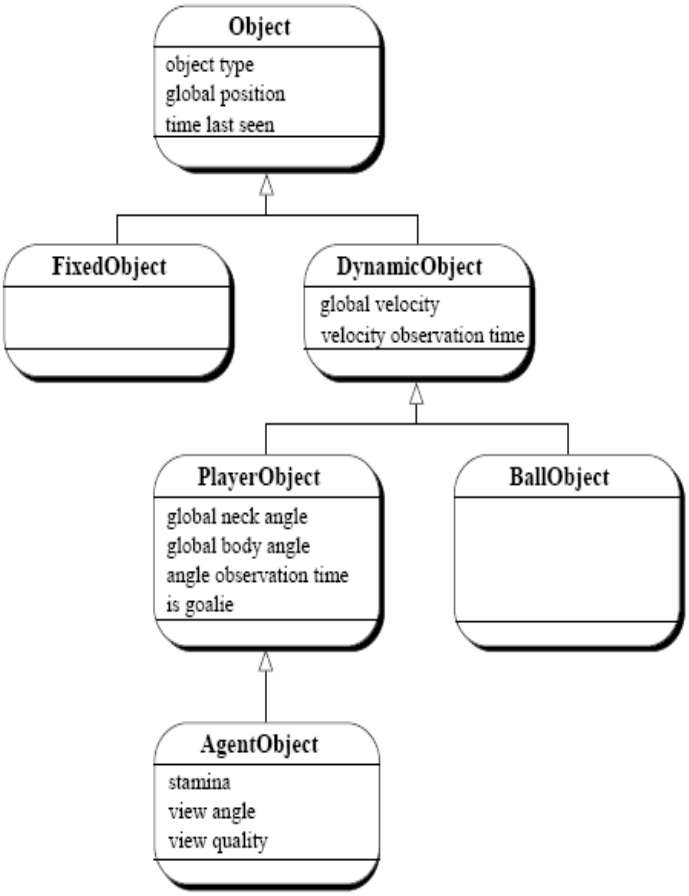
Sú to základné informácie o aktuálnom stave zápasu:

- *Time* – tento atribút reprezentuje serverovský čas ako usporiadanú dvojicu *(t, s)*, kde t je počet simulačných cyklov a s je čas od zastavenia hodín pri situáciach keď sa nehrá. Napr. pri faule a pod.
- *PlayerNumber* – číslo hráča. Je poskytované serverom a nemení sa počas celého zápasu.
- *Side* – reprezentuje stranu, z ktorej tím hrá. Nadobúda hodnoty *left* a *right.*
- *TeamName* – tento atribút reprezentuje meno tímu a ako hodnotu má reťazec. Pri parsovaní správ zo servera slúži na určenie, či je správa určená spoluhráčovi alebo oponentovi.
- *GoalDiff* – tento atribút reprezentuje gólový rozdiel medzi tímami. Jeho hodnota je nulová, ak je skóre vyrovnané, kladná ak tím hráča vedie a záporná ak prehráva.

Informácie o objektoch

Sú to informácie o objektoch na ihrisku. Možno ich rozdeliť na stacionárne (čiary, zástavky a brány) a dynamické, ktoré sa pohybujú naokolo (hráči, lopta). Pre každý objekt sú informácie uložené v triede *Object*, ktorá slúžia ako kontajner pre dáta a neposkytuje žiadnu dodatočnú funkcionalitu okrem metód na nastavovanie (setters) a získavania (getters) informácií. Pozície a rýchlosti sú uložené v globálnej sústave súradníc, ktorá je nezávislá od pozície hráča na ihrisku.

Hráč používa stacionárne objekty (značky) na ihrisku na určenie svojej vlastnej pozície z relatívnych dát získaných o týchto objektoch jeho vnímaním. Na základe vlastnej globálnej pozície potom určí všetky globálne pozície ostatných objektov.



Obrázok 2.4: Hierarchia tried opisujúcich objekty

Object

Abstraktná trieda, ktorá obsahuje základné informácie o objekte. Najdôležitejšie informácie obsahujú typ objektu, globálnu pozíciu a čas, kedy bol objekt naposledy pozorovaný. Treba poznamenať, že atribút „typ objektu“ jednoznačne definuje identitu objektu (napr. `OBJECT_BALL`, `OBJECT_GOAL_R`, `OBJECT_TEAMMATE_9`, `OBJECT_OPPONENT_6` atď.).

Okrem toho atribút „naposledy videný“ môže byť použitý na odvodenie *confidence* hodnoty, ktorá indikuje spoľahlivosť danej informácie. Táto hodnota sa pohybuje v intervale <0; 1> a je definovaná nasledovne:

$$confidence = 1 - \frac{(t_c - t_o)}{100}$$

t_c je aktuálny čas a t_o je čas, kedy bol objekt naposledy pozorovaný. To znamená, že hodnota *confidence* klesá každým simulačným cyklom o hodnotu 0.01 a je obnovená na hodnotu 1.0, keď je objekt znovu spozorovaný. Dôležité je ale uvedomiť si, že hodnota *confidence* sa pre stacionárne objekty nemení a má vždy hodnotu 1.

FixedObject

Je potomkom triedy *Object* a obsahuje informácie o stacionárnych objektoch na ihrisku.

Neobsahuje žiadne ďalšie atribúty okrem tých zdedených od triedy *Object*.

DynamicObject

Je tiež potomkom triedy *Object* a obsahuje informácie o pohyblivých objektoch. Dopĺňa informácie o rýchlosti. Najdôležitešími atribútmi sú tie, ktoré zaznamenávajú globálnu rýchlosť a čas, kedy bola pozorovaná.

PlayerObject

Je to potomok triedy *DynamicObject*. Obsahuje informácie o špecifickom hráčovi na poli (spoluhráč, protihráč). Pridáva atribúty označujúce otočenie krku a otočenie tela (všetko v globálnych súradniciach) a tiež čas posledného zaznamenania týchto informácií. Tiež obsahuje boolovskú hodnotu, či hráč je brankár alebo nie. Samotný agent nie je členom tejto triedy.

BallObject

Je potomkom triedy *DynamicObject*, ktorá obsahuje informácie o lopte. Nepridáva žiadne dodatočné informácie k tým, čo zdedila.

AgentObject

Je potomkom triedy *PlayerObject*, ktorá obsahuje informácie o samotnom hráčovi. Rozširuje rodičovskú triedu o atribúty označujúce staminu, veľkosť uhla pohľadu, a kvalitu pohľadu.

Model sveta teda obsahuje všetky informácie o hracom poli a o objektoch: lopta, 10 spoluhráčov, 11 protihráčov a hráč samotný. Informácie o hráčoch sú uložené na základe ich čísla dresu. Nie je však vždy možné presne vidieť číslo dresu hráča, ak je príliš vzdialený. V takom prípade sa vychádza z predošlej pozície hráča a uvažuje sa možnosť, či na základe minulých údajov a aktuálnych môže ísť o daného hráča. Ak sa nájde zhoda, sú tieto informácie priradené konkrétnemu hráčovi. Ak sa zhoda nenájde, údaje sa priradia hráčovi, ktorého *confidence* hodnota klesla pod prahovú hodnotu.

Informácie o akciách

Model sveta obsahuje tiež informácie o akciách, ktoré agent v minulosti vykonal:

- *Počítadlá akcií* – Slúžia na určenie celkového počtu akcií určitého typu, ktoré hráč vykonal. Využívajú sa tiež na určenie toho, či sa príkaz odoslaný v predošlom cykle vykonal. Ak nebola inkrementovaná hodnota počítadla danej akcie, tak príkaz nestihol doraziť k serveru pred ukončením simulačného cyklu a vykoná sa v nasledovnom.
- *QueuedActions* – Se to zoznam akcií, ktoré agent poslal serveru v posledom simulačnom cykle.
- *PerformedActions* – Zoznam boolovských hodnôt pre každú akciu v zozname *QueuedAction*, ktorý hovorí, či daná akcia bola vykonaná. Tento zoznam teda reprezentuje hráčovu znalosť o tom, ktorá akcia bola vykonaná, na základe čoho môže urobiť zmeny v modeli sveta.

2.2.6.1.4.2 Metódy

Získavacie metódy (retrieval)

Používajú sa na priame získavanie informácií z modelu sveta a majú tvar getMenoAtributu(). Pre atribúty reprezentujúce informácie o prostredí, zápase a akcií môžu byť tieto metódy volané bez parametrov. Na získanie atribútov reprezentujúcich informácie o konkrétnom objekte je nutné ako parameter zadať identifikátor objektu (napr. getGlobalPosition(OBJECT_TEAMMATE_3)).

Aktualizačné metódy (update)

Slúžia na aktualizáciu údajov v modeli sveta na základe získaných informácií z vnemov. Podobne ako na získanie, tak aj na nastavenie majú funkcie tvar setMenoAtributu(). Pre niektoré atribúty môže byť hodnota priamo extrahovaná z vnemu, pre ostatné je nutné ju vypočítať na základe nových informácií. Model sveta je aktualizovaný vždy, keď je obdržaný nový vnem zo servera. Táto aktualizácia prebieha v dvoch fázach. V prvej fáze sú údaje uložené v modeli, v druhej fáze sú aktualizované údaje, ktoré sú kombináciou výpočtu nových údajov a starých.

Predpovedajúce metódy (prediction)

Sú používané na predpovedanie stavu sveta podľa získaných vnemov. Tieto metódy sú dôležité pre proces výberu akcií. Model sveta však obsahuje na predpovedanie stavu iba nízkoúrovňové metódy, ktoré určujú napríklad budúcu pozíciu pohybujúceho sa objektu a pod. Predpovedajúce metódy možno rozdeliť do troch kategórií:

- Metódy predpovedajúce stav hráča po vykonaní určitej akcie. Tie majú ako parameter akciu, ktorú hráč ide vykonať a výstupom je predpoveď stavu agenta (globálna pozícia, rýchlosť,...) po tom, čo túto akciu vykoná.
- Metódy predpovedajúce stav ostatných dynamických objektov na ihrisku. Odhadnúť vlastnosti hráčov je veľmi náročné, nakoľko nie je známe, akú akciu podniknú. Preto sa tieto metódy používajú hlavne na odhad stavu lopty.
- Metóda predpovedajúca čas trvania, ktorý potrebuje hráč, aby sa dostal na určité miesto. Táto metóda má ako parameter cieľ a konkrétny objekt a ako výstup vracia odhadovaný počet simulačných cyklov, kým sa objekt dostane na zvolené miesto.

Vysoko-úrovňové metódy

Model sveta poskytuje tiež niekoľko vysoko-úrovňových metód na odvodenie záverov z informácií o svete. Možno ich zadeliť do piatich kategórií:

- Metódy, ktoré vracajú počet hráčov v určitej oblasti. Tieto metódy majú ako vstup oblasť a výstupom je počet spoluhráčov a/alebo protihráčov v tejto oblasti. Táto oblasť môže byť buď kruh, obdĺžnik alebo kužeľ. Príklady takýchto funkcií sú napr. `getNrTeammatesInCircle()` alebo `getNrOpponentsInCone()`.
- Metódy, ktoré vracajú najbližšieho alebo najrýchlejšieho hráča vzhľadom na určitú pozíciu či loptu. Príklady týchto funkcií sú napr. `getFastestOpponentTo(OBJECT_BALL)` alebo `getClosestTeammateTo(x,y)`.
- Metódy, ktoré indikujú, či daný objekt spĺňa niektoré vysoko-úrovňové podmienky. Tieto

metódy môžu byť použité na určenie príznačných vlastností, kedy je možné použiť najlepšiu akciu. Najdôležitejšie sú napr.:

- `isBallKickable`: vracia *true,* ak sa lopta nachádza v oblasti, kde je hráč schopný do lopty kopnúť.
- `isBallCatchable`: vracia *true,* ak je lopta v dosahu brankára.
- `isBallInOurPossesion`: vracia *true,* ak je najrýchlejším hráčom k lopte spoluhráč.
- `isBallHeadingToGoal`: vracia *true,* ak lopta smeruje na bránu a má dostatočnú rýchlosť, aby ju dosiahla. Toto je relevantná informácia pre brankára.
- Ďalšie príklady sú: `isFreeKickThem, isGoalKickUs, isKickInThem`, atď.

- Metóda určujúca smer najbezpečnejšej trajektórie lopty medzi protihráčmi: `getDirectionOfWidestAngleBetweenOpponents(a1, a2, d).`
- Metódy na výpočet argumentov pre akcie s cieľom dosiahnuť želaný stav. Tieto metódy používajú známe rovnice servera na výpočet stavu po vložení parametrov. Invertujú tieto rovnice na získanie parametrov po vložení želaného stavu. Medzi takéto metódy patria napr.:

  ```
  getAngleForTurn(a), getActualKickPowerRate(),
  getKickPowerForSpeed(s), getKickSpeedToTravel(d, e),
  getPowerForDash(x,y).
  ```

2.2.6.2 Opis funkcionality tried BasicPlayer, Player a Formations

2.2.6.2.1 Trieda BasicPlayer – zručnosti hráča

Zručnosť je schopnosť agenta vykonať nejakú akciu. Zručnosti agenta (hráča) sú hierarchicky usporiadané na 3 katerógie zručností podľa zložitosti:

- *zručnosti nízkej úrovne*
- *zručnosti strednej úrovne*
- *zručnosti vyššej úrovne*

Zručnosti nižnej úrovne sa dajú vyjadriť ako akcie alebo príkazy, ktorým priamo rozumie server. Zručnosti strednej úrovne sú implementované pomocou zručností nižšej úrovne. Zručnosti na najvyššej úrovni používajú zručnosti definované na strednej úrovni ako a vytvárajú požadované a väčšinou zložité správanie sa agenta.

Spôsob vykonania každej z týchto zručností je závislý od aktuálneho stavu sveta WorldModel, nastavenia hráča *PlayerSettings*, nastavenia servera *ServerSettings*. Trieda WorldModel triede *BasicPlayer* poskytuje pomocné funkcie, ktoré sa používajú pri realizácií zručnosti. Jedná sa hlavne o predikčné metódy, ktoré odhadujú premennú uloženú vo WorldModeli v nasledujúcich *n* cykloch alebo o metódy ziskavania týchto premenných.

Spôsob vykonania akcie (aj keď už bola vybraná na vykonanie) môže adaptovať v neskorších fázach, ak to je potrebné. Agent môže realizovať v jednom cykle iba jednu zručnosť (vykonať akciu), bezohľadu nato, koľko zručností bolo rozhodovacím mechanizmom vybraných. Naviac, vykonávaná akcia sa nemusí dokončiť a jej vykonávanie môže byť rozhodovacím mechanizmom prerušené. Výber akcie na vykonanie sa teda prehodnocuje v každom cykle.

2.2.6.2.1.1 Zručnosti nízkej úrovne

Zručnosti nízkej úrovne predstavujú jednoduché akcie, ktoré možno špecifikovať v zmysle povelov, ktorým priamo rozumie server. Tieto zručnosti sú opísané jednoduchými funkciami, ktorých návratovou hodnotou je požiadavka na vykonanie akcie - SoccerCommand na triedu ActHandler. Tieto metódy neobsahujú žiadne zložité procedúry na rozhodovanie. Medzi implementované zručnosti patria napr.:

- vyrovnanie uhlu otočenia krku hráča s uhlom natočenia tela oproti súradnicovej sústave
- otočenie tela k zvolenému bodu na ihrisku
- otočenie sa chrbtom k zvolenému bodu na ihrisku
- hľadanie lopty v blízkosti hráča
- utekanie k zvolenému bodu (nie objektu)
- zastavenie pohybujúcej sa lopty
- zachytenie lopty
- zakričanie
- načúvanie vybranému hráčovi
- kopnutie lopty určitým smerom
- kopnutie lopty blízko tela hráča

2.2.6.2.1.2 Zručnosti strednej úrovne

Zručnosti strednej úrovne na svoju realizáciu využívajú zručnosti nižnej úrovne. Môžu obsahovať aj jednoduché procedúry na rozhodovanie a ich spôsob vykonania je taktiež závislý od parametrov vo WorldModeli, PlayerSettings a ServerSettings. Medzi implementované zručnosti patria napr.:

- otočenie tela hráča k objektu
- otočenie krku hráča k objektu
- presunutie sa na vybranú pozíciu
- účelová zrážka s loptou
- zastavenie lopty, ak sa nachádza blízko hráča
- zastavenie lopty, ak sa nachádza blízko brankára
- odkopnutie lopty do určitého bodu tak, aby v cieľovom bode mala požadovanú rýchlosť
- otočenie sa tela agenta spolu s loptou
- pohybovanie sa po zvolenej čiare

2.2.6.2.1.3 Zručnosti vysokej úrovne

Zručnosti vysokej úrovne zodpovedajú akciám na najvyššej úrovni abstrakcie z toho pohľadu, že tieto zručnosti na svoje modelovanie zvyčajne využívajú zručnosti strednej úrovne. Tieto zručnosti rovnako využívajú premenné a funkcie, ktoré poskytuje WorldModel a trieda PlayerSettings. Medzi implementované zručnosti patria napr.:

- zachytenie lopty, ak sa nachádza vo väčšej vzdialenosti od hráča
- driblovanie
- priama prihrávka lopty objektu
- leading pass - predkop lopty vybranému hráčovi, ktorý ju dostihne v behu
- through pass - výkop lopty do oblasti medzi súperovými obrancami a brankárom tak, aby ju hráč stihol dobehnúť a priamo ohrozil brankára
- obohranie oponenta v situácií 1-1
- značenie súperov
- bránenie bránkovej čiary
- zadržanie lopty

2.2.6.2.2 Trieda PlayerSettings

Obsahuje viacero parametrov, ktoré ovplyvňujú rozhodovací proces agenta a metódy pre získavanie a nastavovanie týchto hodnôt. Najdôležitejšie parametre z PlayerSettings sú prahové parametre na rozhodovanie, či konkrétna akcia má byť vykonaná alebo nie. Tieto parametre prispôsobujú správanie hráča.

2.2.6.2.3 Trieda Player - Mechanizmus na vyhodnocovanie

Účelom tejto triedy je poskytnutie rozhodovacej procedúry na výber ďalšej akcie. Výber akcie je založený na najaktuálnejšom stave sveta WorldModel a aktuálnej role agenta v aktuálnej tímovej formácií. Zároveň sa využívajú univerzálne nastavenia hráča v triede PlayerSettings. Táto trieda dedí triedu BasicPlayer, čiže aj všetky implementované zručnosti hráča.

Vo verejnej verzii je použitá rozhodovacia procedúra tímu Simple FC Portugal 2000. Táto verzia tímu bola vydaná zakladateľmi tímu FC Portugal. Rozhodovacia procedúra je založená na klasickom prístupe if-then-else. Celý prístup na výber najlepšej acie možno algoritmicky opísať nasledovne:

```
ak istota spojená s aktuálnou pozíciou lopty je príliž nízka
potom

     hľadaj loptu

inak ak je možné do lopty kopnúť potom

     kopni do lopty s maximálnou silou do smerom do oponentovej
     bránky do náhodného rohu

inak ak najrýchlejší hráč, ktorý môže získať loptu som ja potom

        zachyť loptu

inak ak vzdialenosť k strategickej pozícií je > 1 potom
```

```
            presuň sa na strategickú pozíciu

    inak

            otoč sa smerom k lopte
```

Procedúry rozhodovacieho mechanizmu sú definované v hlavičkovom súbore triedy Player a implementované v súbore PlayerTeams.cpp. Ďalej trieda obsahuje bežné metódy, slúžiace napríklad na výkop penalty, rozhodovanie, či má hráč niečo povedať alebo nie, zisťovanie stavu lopty a pod.

2.2.6.2.4 Trieda Formations - Tímové stratégie a formácie

Stratégia tímu agentov je definovaná ako kolektívny plán ktorý je nasledovaný s cieľom dosiahnutia spoločného cieľa s dostupnými prostriedkami.

Stratégia podľa UvA Trilearn má spĺňať nasledujúce kritéria:

- musí špecifikovať formáciu tímu a pozíciu agentov v tejto formácií
- definuje, ktoré formácie sú vhodné pre rôzne herné situácie
- stratégia špecifikuje pre každý typ hráča jeho (rolu) a adekvátne správanie súvisiace s touto rolou v tíme
- musí obsahovať informáciu o tom, ako má agent adaptovať jeho správanie v danej situácií
- musí špecifikovať ako koordinovať správanie sa jednotlivých agentov
- musí naznačovať, ako každý hráč má narábať so svojou energiou počas hry tak, aby dosiahol maximálny výkon

Stratégiu tímu ovplyvňuje najmä:

- sila súperiaceho tímu
- štýl súperovej hry
- stav hry
- aktuálna herná situáciou
- dostupnosť zdrojov (stamina, …)

2.2.6.2.4.1 Situation Based Strategic Positioning

Základným princípom stratégie je, že tím UvA Trilearn rozlišuje medzi *aktívnymi* a *pasívnymi situáciami*. Agent je v aktívnej situácii, ak sa nachádza v blízkosti lopty a inak v pasívnej. Ak je agent v pasívnej situácii, presúva sa do strategickej pozície v poli v očakávaní, že sa dostane do aktívnej situácie. Táto pozícia je závislá od aktuálnej tímovej formácie, role agenta vo formácii a pozície lopty v poli.

Agent vo formácii má priradenú rolu, ktorú musí vykonávať (útočník, obranca, ľavé krídlo, záložník, atď). V pasívnych situáciách agent určuje svoju strategickú pozíciu v poli počítaním váhovaných súm jeho polohy stanovenej formáciou a aktuálnou pozíciou lopty, ktorá mu slúži ako bod atrakcie. Hráč sa teda pri pasívnej situácii rozhoduje, či bude dodržiavať formáciu alebo sa vydá za loptou. Ak je už veľmi ďaleko od svojej pozície stanovenej formáciou, potom je

príťažlivosť vrátenia sa späť ku svojej pozícii vo formácii veľká a vráti sa späť. Naopak, ak je lopta veľmi blízko, vydá sa k nej.

V aktuálnej implementácii Uva Trilearn 2003 formácie dodržujú najmä hráči, ktorí sa nachádzajú v pasívnych situáciách. Akonáhle je hráč v aktívnej situácii a loptu stratí a nenachádza sa už v tejto situácii tak sa vráti späť na to miesto vo formácii, ktoré je mu dané rolou. Aktívni a pasívni agenti si vyberajú nové akcie v každom cykle, t.j. rozhodovací mechanizmus vyberá akciu a formácie a môže zmeniť stratégiu v každom cykle a na základe aktuálneho stavu prostredia. Celý mechanizmus tvorby formácií je prebraný z tímu FC Portugal.

2.2.6.2.4.2 Implementácia formácií

Implementácia tímových stratégií je realizovaná v troch triedach PlayerTypeInfo, FormationTypeInfo, Formations. Sú tu informácie o všetkých možných tímových formáciách a metódy na určenie strategickej pozície v poli. Formácie sú načítavané zo súboru formations.conf. Pri prihlasovani na server si každý agent tieto formácie stiahne. Vo verejnej verzii sú definovane iba 2 formácie.

Triedy implementujúce tímové stratégie:

- `PlayerTypeInfo`: Obsahuje informácie o typoch hráča vo formácii, obsahuje parametre na určenie atrakcie hráča k lopte v smere x a y - váhy atraktivity k lopte [0,1], minimálne, maximálne súradnice role vo formácii, do ktorých hráč ešte môže ísť, atď.
- `FormationTypeInfo`: Obsahuje špecifické informácie o každej formácii.
- `Formations`: Obsahuje informácie o všetkých možných formáciách a aktuálne používanej formácii.

Obrázok 2.5: Triedy implementujúce tímové stratégie

2.2.6.2.5 Komunikácia agentov

Individuálne konanie hráčov je ovplyvnené a koordinované informáciami, ktoré sú dostupné pre všetkých agentov. Táto koordinácia neprebieha vo forme komunikácie medzi hráčmi, pretože komunikačný kanál má nízku prenosovú kapacitu a je nespoľahlivý. Agenti preto nespoliehajú na vzájomnú komunikáciu, ak sa má vykonať nejaká strategická akcia. V súčasnej implementácii agenti používajú komunikáciu iba na to, aby zvýšili množstvo aktuálnych informácií vo WorldModeli.

2.2.7 Robolog

Projekt [ROBOWWW] vznikol na nemeckej univerzite Koblenz-Landau v rámci predmetu Umelá inteligencia. Na európskom šampionáte v Amsterdame (2000) získal 5. miesto, v Nemecku 2. miesto. Tento tím bol implementovaný pod operačným systémom Linux, program na kontrolu správania sa hráčov je napísaný v Prologu. Dokumentácia k tomuto tímu je veľmi stručná. V nasledujúcich riadkoch sú teda aspoň stručne popísané základné princípy, ktorých sa tím drží a sú zaujímavé.

2.2.7.1 Stable marriage

Predstavuje taktiku hovoriacu o bránení súperov. Pri bránení sú obsadení hráči tak, aby si každý hráč našiel jedného súpera. Najprv sa hráč rozhliadne a zhodnotí situáciu. Snaží sa pritom zamedziť situácii, pri ktorej by bol jeden súper obsadený dvoma hráčmi.

2.2.7.2 World model

Agenti sú vybavení rôznymi pohľadmi na svet, ktoré im pomáhajú vidieť svet v rôznych situáciach. Sú to tieto pohľady: Global world model (základné akcie agentov), Sensed world model (vnemy získané pomocou senzorov – oči, uši...) Calculated world model (odvodený od globálneho modelu), Fullstame world model (ukladanie správ).

2.2.8 FC Portugal

Tento tím [FCPOWWW] je zameraný na zlepšovanie schopností hráča pomocou kouča aj pred hrou aj počas nej. Stratégia je definovaná v konfiguračnom súbore a počas hry môže byť menená.

2.2.8.1 Tímová stratégia

Stratégia je založená na množine typov (rolí) hráčov a taktík, ktoré obsahujú niekoľko formácií pre rôzne herné situácie (obrana, útok, atď.). Formácie určujú hráčovi strategickú pozíciu podľa svojej role. Typy hráčov sú definované na troch úrovniach: strategické správanie, držanie lopty a získanie lopty.

2.2.8.2 Situation Based Strategic Positioning

Jedným z kľúčových bodov, ktoré priniesli tomuto tímu v minulosti úspech bol mechanizmus rozmiestňovania hráčov založený na rozlišovaní medzi strategickou a aktívnou situáciou. SBSP mechanizmus je použitý iba pri strategickej situácii, keď si hráč myslí, že nebude v krátkom čase v aktívnej situácii. Vtedy je hráč presúvaný na svoju strategickú polohu. Pri nastatí aktívnej situácie je určenie pozície hráča závislé od viacerých faktorov (držanie lopty, získanie, ...). Podľa zvolenej taktiky, formácie a rozmiestnenia vypočíta hráč svoju strategickú pozíciu na ihrisku a vo formácií. Táto pozícia je potom ešte upravená vzhľadom na pozíciu lopty, rýchlosť...

2.2.8.3 Typy hráčov, správanie

V tíme FC Portugal boli implementované tri úrovne správania sa hráčov:

- strategické správanie – určuje pozíciu hráča v poli, ktorý si myslí, že sa čoskoro nedostane do aktívnej situácie

- držanie lopty – rozhoduje sa, aké kroky podnikne hráč s loptou
- získanie lopty – rozhoduje, kedy sa hráč pokúsi získať loptu, priblížiť k hráčovi s loptou

2.2.8.4 DPRE – Dynamic Positioning and Role Exchange

Flexibilné role a výmena rolí medzi hráčmi. Umožňuje výmenu rolí medzi jednotlivými hráčmi. Táto informácia musí byť oznámená všetkým hráčom a uskutoční sa, len ak je to pre tím výhodné.

3 Hrubý návrh riešenia

Ako základ pre implementáciu hráča sme vybrali hráča tímu Uva Trilearn. Dôvodom bola jeho architektúra. Jeho návrh nevychádzal z existujúceho hráča preto je zdrojový kód homogénny. Ďalším dôvodom sú kvalitne spracované nižšie úrovne správania sa a aj model prostredia, ktorí tento tím vytvoril.

Zdrojové kódy tohto hráča neobsahovali vysokoúrovňové riadenie hráča. V dodaných zdrojových kódoch bolo správanie naimplementované podľa staršej verzie tímu FC Portugal. Predpokladom pre pokračovanie je reimplementácia vysokoúrovnového riadenia hráča.

Východiskom pre riadenie hráča je analýza metód, ktoré sú využívané v iných tímoch.

3.1 Metódy využívané na riadenie hráča

3.1.1 Rozhodovanie sa agentov pomocou koordinačných grafov

3.1.1.1 Koordinačný problém

Proces využívajúci teóriu koordinačných grafov zabezpečuje, že individuálne rozhodnutia agentov vyústia do spoločného optimálneho rozhodnutia skupiny.

Základným princípom uplatneným v prístupe pomocou koordinačných grafov je heslo „robme to, čo je najlepšie pre nás ako jednotlivca a čo je čo možno v najväčšom súlade s potrebami kolektívu“.

3.1.1.2 Teória koordinačného problému

Nech

$$A = A_1 \times A_2 ... \times A_n$$

je spoločná akcia n agentov v danom cykle. Táto je aj množina dostupných akcií v danom cykle pre agenta i.

Každý agent má definovanú výnosovú funkciu

$$R_i(A) \rightarrow R$$

Každý agent si zvolí jednu z dostupných akcií a obdrží výnos založený na vybraných akciách ostatných agentov. Všetci agenti používajú rovnakú výnosovú funkciu, teda

$$R_1 = R_2 = ... = R_n$$

Agenti volia také akcie, aby platilo, že ak

$$a^* \in A$$

je spoločná akcia agentov

$$G_1, G_2, ..., G_n$$

Potom

$$\forall i \forall a_i \in A_i$$

platí, že

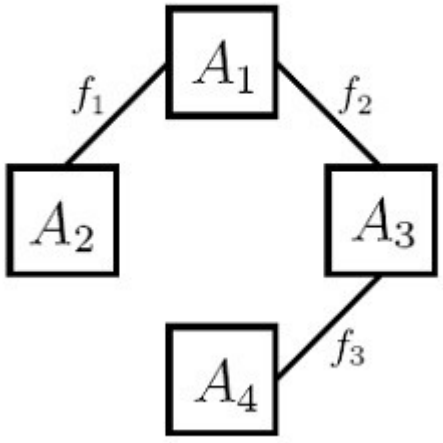
$$R_i=(a_i^*, a_{-i}^*) \geqslant R_i=(a_i^* - a_{-i}^*)$$

kde a_{-i} je spoločná je akcia všetkých ostatných agentov okrem agenta i. Taký stav sa nazýva Nashova rovnováha. Pri výbere akcií agentov v nejakom cykle sa teda snažíme dospieť do bodu Nashovej rovnováhy.

Nashova rovnováha nastáva, ak jeden "hráč" urobí to, čo je pre neho najvýhodnejšie v závislosti na činnosti druhého "hráča", ktorý tak isto robí to, čo je pro neho najvýhodnejšie s ohľadom na činnosť prvného "hráča“.

3.1.1.3 Koordinačný graf

Koordinačný graf sa skladá podobne ako iné grafy z uzlov a hrán. Uzol v grafe predstavuje agenta G_i. Hrana v grafe identifikuje závislosť medzi agentami, t.j. akcia, ktorú v danom cykle vykoná agent G_i môže závisieť od vybraných akcií s tými agentami, s ktorými je agent G_i v grafe spojený. V danom cykle koordinujú svoje akcie iba prepojení agenti.



Obrázok 3.1: Koordinačný graf

A2 musí koordinovať svoje akcie s A1, A4 musí koordinovať s A3. A3 musí koordinovať akcie s A4 a A1. A1 musí koordinovať akcie s A2 a A3.

Funkcia f špecifikuje lokálne koordinačné závislosti medzi akciami agentov. R(a) je globálna výnosová funkcia pre daný graf. R(a) sa dá rozložiť na niekoľko lokálnych výnosových funkcií uvažujúcich iba akcie v grafe závislých agentov.

$$R(a)=f_1(a_1, a_2)+f_2(a_1, a_3)+f_3(a_3, a_4)$$

3.1.1.4 Algoritmus eliminácie premenných

Tento algoritmus umožňuje dospieť k stavu, kedy si všetci agenti v danom cykle zvolia také akcie, ktoré maximalizujú globálnu výnosovú funkciu pre daný graf a daný cyklus. Snažíme sa nájsť takú spoločnú akciu a^* je z A, pre ktorú je R najväčšie. Agenti sú z grafu postupne eliminovaní vykonaním tzv. lokálneho maximalizačného kroku pre každého agenta.

3.1.1.4.1 Princíp algoritmu eliminácie

- Vyberie sa agent na elimináciu.
- Tento agent si zo svojich susedov vezme pravidlá v ktorých figuruje jeho akcia.
- Agent optimizuje svoje rozhodnutie uvážením všetkých možných kombinácií akcií jeho susedov a príslušných ohodnotení výnosovou funkciou.
- Agent pošle svojim susedom upravené pravidlá v ktorých zmenil svoju akciu tak, aby maximalizoval výnosovú funkciu.
- Proces pokračuje dovtedy, pokým eliminujeme všetkých agentov z grafu.
- Posledne eliminovaný agent si jednoducho vyberie individuálnu optimálnu akciu, ktorá maximalizuje jeho finálnu stratégiu.
- Graf sa začne prechádzať naopak aby si každý agent mohol následne vybrať optimálnu akciu založenú na ich stratégii a už fixovaných akcií jeho susedov z grafu.

3.1.1.4.2 Príklad

Eliminujeme agenta A1, t.j. snažíme sa o maximalizáciu hodnôt výnosových funkcií a sumy f1+f2.

$$max_a R(A) = max_{a2,a3,a4}(f_3(a3,a4) + max_{a1} f1(a1,a2) + f_2(a1,a3))$$

Agent A1 vytvorí novú akciovo založenú výnosovú funkciu ktorá vracia takú kombináciu akcií a2 a a3, že výnosová lokálna funkcia tejto kombinácie bude najväčšia, eliminuje premennú a1 (neuvažuje ju a robia sa kombinácie akcií a2 a a3).

$$f_4(a2,a3) = max_{a1} f_1(a1,a2) + f_2(a1,a3)$$

$$max_a R(a) = max_{a1,a2,a3} f_3(a3,a4) + f_4(a2,a3)$$

$$f_5(a3) = max_{a2} f_4(a3,a4)$$

Eliminujeme A2, dostaneme:

$$max_a R(a) = max_{a3,a4} f_3(a3,a4) + f_5(a3)$$

Eliminujeme A3, dostaneme:

$$f_6(a_4) = max_{a_3} f_3(a_3,a_4) + f_5$$

$$max_a R(a) = max_{a_4} f_6(a_4)$$

Agent A4 si teraz vyberie takú a4, že maximalizuje hodnotu svojej lokálnej výnosovej funkcie. Následne túto akciu a4 pošle svojim susedom, oni zohľadnia a4 a akciu ai tak, aby maximalizovali ich lokálnu výnosovú funkciu.

Tento proces ide presne v opačnom poradí eliminácie agentov z grafu.

Týmto dosiahneme to, že každý agent si zvolí takú akciu, ktorá maximalizuje lokálnu výnosovú funkciu s tým dôsledkom, že následne spoločná akcia a, kde

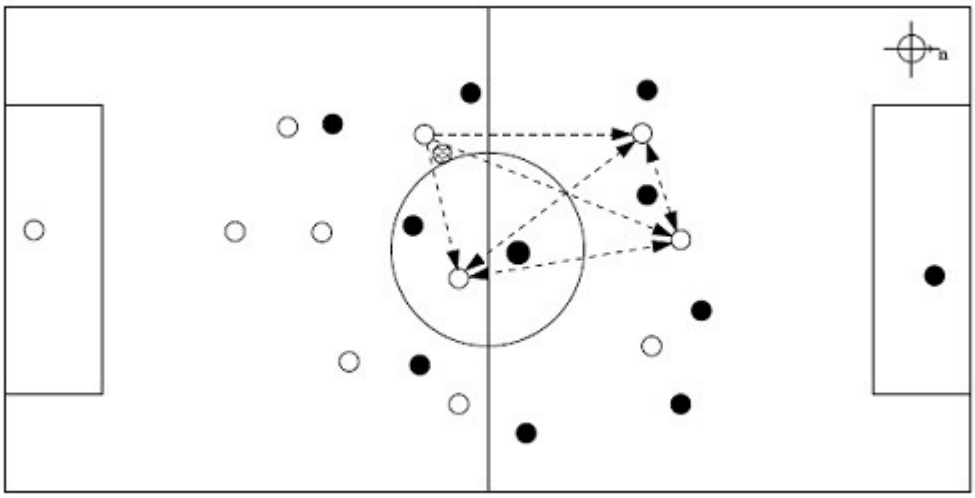
$$a \in A$$

bude taká, pre ktorú globálna výnosová funkcia R(a) bude maximálna.

3.1.1.5 Pravidlovo-založené výnosové funkcie

V každom cykle v aktívnej hernej situácií (t.j. pre agentov, ktorí sa nachádzajú v istej vzdialenosti od lopty) majú agenti dynamicky priradenú rolu – nahrávač, príjemca, atď. Táto rola je vypočítaná a každý agent vie, akú rolu má v momentálnej situácii priradenú.

Medzi týmito agentami je koordinačný graf. Akcie, ktoré v danom cykle vykonajú koordinujeme v závislosti od ich rolí. Snažíme sa, aby ich spoločná akcia v danom cykle mala čím najvyššiu výnosovú funkciu.



Obrázok 3.2: Koordinačný graf medzi agentami

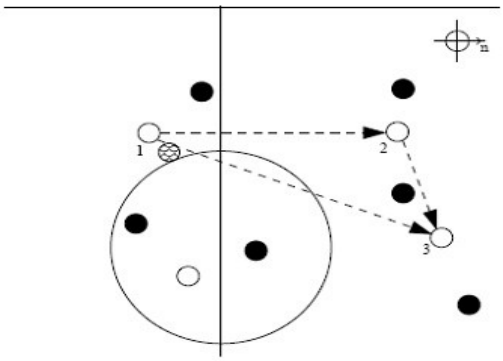
V Robocupe, konkrétny agent môže v danom cykle vykonať napr. tieto akcie:

$$\begin{aligned} \langle p_1^{passer} \quad ; \quad & \text{has-role-receiver}(j) \quad \wedge \\ & \neg\text{isPassBlocked}(i, j, dir) \quad \wedge \\ & a_i = \text{passTo}(j, dir) \quad \wedge \\ & a_j = \text{moveTo}(dir) : u(j, dir)\rangle \ \forall j \neq i \end{aligned}$$

$$
\begin{aligned}
\langle p_2^{passer} \;;\; & \text{is-empty-space}(i, \text{n}) \;\wedge \\
& a_i = \text{dribble(n)} : 30\rangle \\
\langle p_3^{passer} \;;\; & a_i = \text{clearBall} : 10\rangle \\
\langle p_4^{passer} \;;\; & \text{is-in-front-of-goal}(i) \;\wedge \\
& a_i = \text{score} : 100\rangle \\
\langle p_5^{receiver} \;;\; & \text{has-role-interceptor}(j) \;\wedge \\
& \neg\text{isPassBlocked}(j, i, dir) \;\wedge \\
& a_i = \text{moveTo}(dir) : u(i, dir)\rangle \quad \forall j \neq i \\
\langle p_6^{receiver} \;;\; & \text{has-role-passer}(j) \;\wedge \\
& \text{has-role-receiver}(k) \;\wedge \\
& \neg\text{isPassBlocked}(k, i, dir) \;\wedge \\
& a_j = \text{passTo}(k, dir2) \;\wedge \\
& a_k = \text{moveTo}(dir2) \;\wedge \\
& a_i = \text{moveTo}(dir) : u(i, dir)\rangle \; \forall j, k \neq i \\
\langle p_7^{receiver} \;;\; & \text{moveToStratPos} : 10\rangle \\
\langle p_8^{intercep.} \;;\; & \text{intercept} : 100\rangle \\
\langle p_9^{passive} \;;\; & \text{moveToStratPos} : 10\rangle
\end{aligned}
$$

Každý agent G_i, pre každú rolu, ktorú môže zastávať má definované pravidlá a príslušný príspevok k výnosovej funkcií.

Predstavme si, že nastala táto situácia na ihrisku:



Obrázok 3.3: Príklad situácie na ihrisku

Agenti majú priradené role G1 – passer, G2 – receiver, G3 - receiver.

Predstavme si, že platia iba predikáty:

$$\neg isPassBlocked\,(1,2,s)\wedge\neg isPassBlocked\,(2,3,nw)$$

Potom zodpovedajúce pravidlá pre každého hráča budú teraz budú nasledovné:

$$
\begin{aligned}
G_1 : \langle p_1^{passer} \quad ; \quad & a_1 = \text{passTo}(2, \text{s}) \quad \wedge \\
& a_2 = \text{moveTo}(\text{s}) : 50 \rangle \\
\langle p_2^{passer} \quad ; \quad & a_1 = \text{dribble}(\text{n}) : 30 \rangle \\
\langle p_3^{passer} \quad ; \quad & a_1 = \text{clearBall} : 10 \rangle \\
G_2 : \langle p_7^{receiver} \quad ; \quad & a_2 = \text{moveToStratPos} : 10 \rangle \\
G_3 : \langle p_6^{receiver} \quad ; \quad & a_1 = \text{passTo}(2, dir) \quad \wedge \\
& a_2 = \text{moveTo}(dir) \quad \wedge \\
& a_3 = \text{moveTo}(\text{nw}) : 30 \rangle \\
\langle p_7^{receiver} \quad ; \quad & a_3 = \text{moveToStratPos} : 10 \rangle
\end{aligned}
$$

Pre vykonanie eliminačného kroku pre agenta G1 z obrázku 3.3 zoberieme všetky pravidlá zo susedov G1, v ktorých figuruje akcia agenta G1. Túto akciu vylúčime a spravíme všetky možné kombinácie akcií ostatných agentov a vyhodnotíme výnosovou funkciou každú vzniknutú kombináciu.

Dostaneme:

$$
\begin{aligned}
\langle p_1^{passer} \quad ; \quad & a_2 = \text{moveTo}(\text{s}) \quad \wedge \\
& a_3 = \text{moveTo}(\text{nw}) : 80 \rangle \\
\langle p_1^{passer} \quad ; \quad & a_2 = \text{moveTo}(\text{s}) \quad \wedge \\
& a_3 = \neg\text{moveTo}(\text{nw}) : 50 \rangle \\
\langle p_1^{passer} \quad ; \quad & a_2 = \neg\text{moveTo}(\text{s}) : 30 \rangle
\end{aligned}
$$

Vidíme, že vo vzniknutých akciách už nie je akcia agenta G1 a1. Tieto vygenerované stratégie posiela všetkým svojim rodičom. Ak takto vygenerované stratégie obsahujú akciu ai, pričom Gi nie je rodičom G1 ale iba jeho nasledovníkom, vygenerujeme nový spoj medzi G1 a Gi tak, že G1 sa stane rodičom G_i a tak môže obdržať túto správu.

Vo všeobecnosti eliminácia a posielanie stratégií pokračuje dovtedy, pokiaľ sa už nedá eliminovať žiadny agent. Posledný agent už dôsledkom eliminačného procesu má na výber iba jeho akciu a zvolí si takú, ktorá zabezpečí maximálnu hodnotu výnosovej funkcie. Túto akciu si zafixuje, pošle info o nej predchádzajúcemu agentovi z procesu eliminácie (opačné poradie výberu agentov ako v procese eliminácie) a ten si vyberie takú akciu vzhľadom na zafixovanú akciu predchádzajúceho agenta. Na záver máme k dispozícii spoločnú akciu, ktorá maximalizuje globálnu výnosovú funkciu.

3.1.2 Rozhodovanie sa pomocou poradcov

Predpokladom rozhodovania sa pomocou poradcov je existencia externého poradcu. Úloha poradcu je hľadanie analógií medzi realizovanými akciami [KnTransAdT]. Príkladom je napr. priama strela na bránku. Poradca po zhliadnutí akcie vidí analógiu s inou akciou - napr. prihrávka spoluhráčovi.

Analógiu si vytvára na základe analýzy vzdialeností medzi hráčmi, cieľovou pozíciou lopty a tak. Analógia je vytváraná na základe parametrov prostredia. Výsledkom sú pravidlá, ktoré preferujú používanie analogických akcií, ktoré hráč ovláda na dosiahnutie hlavnej úlohy.

Príkladom môže byť situácia, keď hráč chce dať gól - jeho úloha je streliť gól. Hráč má strelenie gólu definované ako sled akcií, ktoré v danej situácii má vykonať. Tento sled je definovaný nejakým algoritmom. Môžme predpokladať že v prípade úlohy "strelenie gólu" je v slede akcií akcia `streľ na bránu`. Poradca zistil, že na vystrelenie nie sú úplne ideálne podmienky a zároveň odhalil na základe vzdialenosti od brány a uhla analógiu s nahrávaním druhému spoluhráčovi. Podmienky pre naučené nahrávanie sú lepšie. Tak isto aj cieľ nahrávania (dostať loptu do určitého miesta) zhodný, alebo podobný s akciou `streľ na bránu`. Preto poradca navrhuje namiesto `streľ na bránu` použitie analógie `prihratie hráčovi`. Takže hráč na to aby strelil gól prihrá virtuálnemu spoluhráčovi. Toto umožňuje rýchlejšie sa naučiť plniť jednotlivé úlohy. Výsledným pravidlom, ktoré zavádza poradca, môže vyzerať takto:

```
IF vzdialenosťKBráne < 30 AND natocenie = 30° THEN

  preferuj prihravkaSpoluhracovi pred strielanimNaBranu

END IF;
```

Tento postup avšak mení len preferencie jednotlivých akcií. Nemení ich aktuálne poradie, ale iba prioritu ich vykonávania, pričom nová priorita sa po zoradení nemusí prejaviť. Ďalší problém je, že takýto spôsob uvádza len preferencie pre akcie a nie parametre akcií. *Z* tohto dôvodu je potrebné definovať preklad parametrov medzi jednotlivými akciami.

3.1.2.1 Implementácia poradcov

Poradca môže byť externý, alebo môže byť súčasťou herného systému. Poradcovia ako súčasť herného systému pôsobia ako:

- Couch
- Modul hráča analyzujúci prostredie

V prípade modulu hráča je možné realizovať poradcu ako samostatné vlákno, ktoré priamo nezasahuje do činnosti hráča, ale len analyzuje situáciu a v prípade požiadavky zverejní svoje preferencie.

3.1.2.2 Učenie sa pomocou poradcov

Predpokladajme, že každá akcia v rámci úlohy má definované pravdepodobnosťou ohodnotené akcie, ktoré môžu nastať. Toto pravdepodobnostné ohodnotenie pochádza z hodnotenia situácie. Naskytajú sa tu tieto otázky:

- Ako môže poradca ovplyvniť túto pravdepodobnosť?
- Kedy a akým spôsobom prebehne učenie?

3.1.2.2.1 Ako môže poradca ovplyvniť túto pravdepodobnosť?

Plnenie úlohy - sled akcií pre plnenie úlohy je možné reprezentovať ako Markovovskú postupnosť. V prípade, že máme 3 akcie, ktoré môžeme vykonať, tak pravdepodobnosť prechodu môžeme zapísať ako maticu 3x3:

$$P = \left(\begin{bmatrix} q_0^0 & q_0^1 & q_0^3 \\ q_1^0 & q_1^1 & q_1^3 \\ q_2^0 & q_2^1 & q_2^3 \end{bmatrix} \right)$$

Prvok v i-tom riadku a j-tom stĺpci obsahuje pravdepodobnosť prechodu medzi i-tou a j-tou akciou. Toto nám definuje podmienku, že suma hodnôt v riadku sa musí rovnať 1. Ovplyvnenie pravdepodobne prebieha tak, že predpokladáme, že akcia sa v nasledujúcom kroku vykoná - tj. ovplyvníme hodnotu v riadku, v ktorom máme aktuálnu akciu - zvýšime preferenciu akcií v prospech akcie, ktorú odporúča poradca. Interná reprezentácia matice nemusí obsahovať pravdepodobnosť prechodu, ale počet prechodov. Následne keď normujeme vektor, tak získame vektor pravdepodobností prechodu do jednotlivých stavov. Táto reprezentácia umožní len pripočítať 1 k akcií preferovanej poradcom a následne normovať maticu.

Na to aby sme získali celkovú pravdepodobnosť akcií, by sme mali podľa **[EMCWWW]** nájsť vlastný vektor q.

$$\begin{aligned} qP &= q \\ &= qI \\ q(P-I) &= \mathbf{0} \\ &= q\left(\begin{bmatrix} q_0^0 & q_0^1 & q_0^3 \\ q_1^0 & q_1^1 & q_1^3 \\ q_2^0 & q_2^1 & q_2^3 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) = 0 \end{aligned}$$

Vlastný vektor q následne obsahuje ustálenú pravdepodobnosť akcií, ktoré by sa mali vykonávať.

3.1.2.2.2 Kedy a akým spôsobom prebehne učenie?

V prípade, že je akcia zvolená a následne prebehne zlepšenie, matica prechodov sa zachová a slúži ako základ pre ďalšiu akciu pri plnení úlohy. V prípade zhoršenia sa zmena matice uvádzaná v predchádzajúcej časti neuplatní. Tj. matica sa vráti do pôvodneho stavu.

3.1.2.2.3 Transformácia parametrov akcie

Cieľom analógie akcie je nahradenie pôvodnej akcie pôvodnou akciou. Problémom ale môžu byť problémy pri určovaní parametrov potrebných pre vykonanie analogickej akcie. Táto časť by mala byť definovaná programátorom. Alternatívna cesta je poskytnúť agentovi možnosť učiť sa. V tomto prípade je potrebné vyriešiť tieto problémy:

- Je potrebné zaviesť nezávislé ohodnotenie akcie ako alternatívy a ohodnotenie úspešnosti akcie z pohľadu vstupných parametrov.
- Je potrebné zaviesť spôsob adaptácie transformácie vstupných parametrov - možno neurónová sieť.

3.1.3 Rozhodovanie s využitím fuzzy množín

Klasické rozhodovanie s využitím rozhodovacích stromov má ten nedostatok, že výber najvhodnejšej akcie je realizovaný vyhodnotením série podmienok na základe hodnôt true / false. Jednotlivé situácie sa však môžu štylizovať aj nepresne, teda nie len podľa logickej hodnoty 0 / 1, ale z intervalu <0, 1>. Inak povedané - niektoré podmienky sú viac a iné zas menej splnené. Vytvorenie pravidiel na výber akcie v takýchto podmienkach je možné realizovať s využitím znalostí s fuzzy logiky. Na tomto prístupe začal v roku 2004 (ak. rok 2003/2004) pracovať Ľubor Ladický, ktorý aplikoval tieto princípy rozhodovania hlavne v situáciách strely na bránu a nahrávanie najlepšiemu hráčovi.

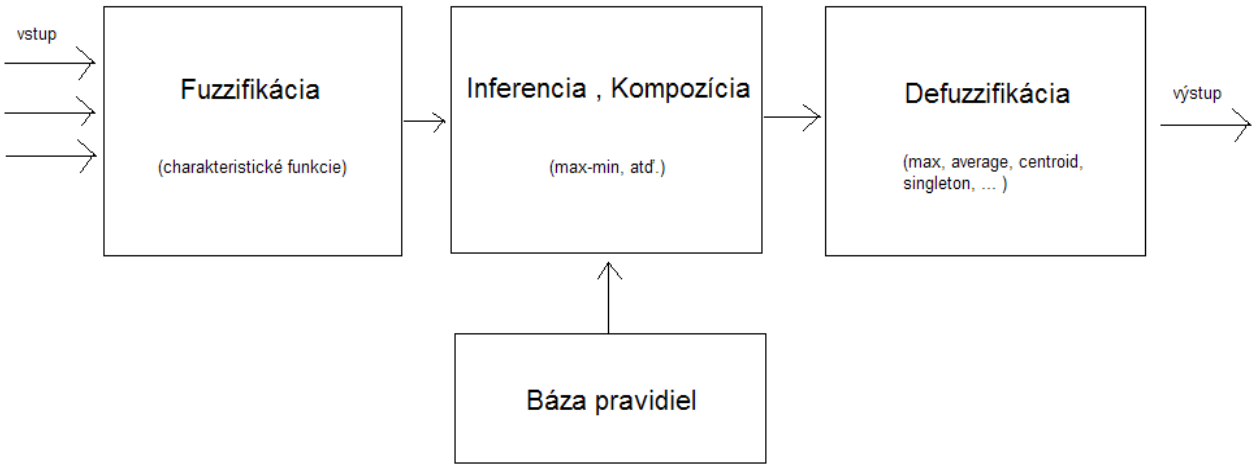
3.1.3.1 Fuzzy regulátor

Aplikovanie znalostí z fuzzy logiky na proces rozhodovania znamená vytvorenie fuzzy regulátora [FUZZYWWW], t.j. regulátora, ktorý na prácu s neurčitosťami využíva fuzzy množiny. Takýto regulátor má produkčné pravidlá v tvare *Ak* predpoklad *Potom* záver, kde predpoklad aj záver sú fuzzy množiny zadefinovaných linguistických premenných.

Príklad:

```
Ak vzdialenost je veľká potom sila_kopnutia je veľká.
```

Schéma fuzzy regulátora je na obr. 3.4.



Obrázok 3.4: Schéma fuzzy regulátora

Časti fuzzy regulátora sú:

Báza pravidiel

Obsahuje všetky pravidlá, ktoré popisujú správanie fuzzy regulátora

Fuzzifikácia

Vykonáva prevod ostrých vstupných hodnôt (crisp) na stupne príslušnosti jednotlivých lingivistických premenných (tie sú zadefinované nad jednotlivými fuzzy premennými v báze

pravidiel).

Inferencia

Určuje výsledný stupeň príslušnosti celej predpokladovej časti pre každé pravidlo. Najčastejšie sa používa metóda *min* (prienik fuzzy množín).

Kompozícia

Určuje výslednú funkciu príslušnosti výstupov pomocou stupňov príslušnosti predpokladov jednotlivých pravidiel a samotných záverov pravidiel. Najčastejšie sa používa metóda **max** (zjednotenie množín).

Defuzzifikácia

Realizuje prevod výslednej funkcie príslušnosti výstupov na ostrú (reálnu) hodnotu. Často používanou metódou defuzzifikácie je metóda ťažiska.

3.1.3.2 Proces inferencie a kompozície

Uvažujme dve pravidlá s použitím týchto skratiek: KM – kladné malé číslo, KS – kladné stredné číslo, $m_A(x)$ je charakteristická funkcia pre stupeň príslušnosti do množiny A, $*m_A(u)$ je charakteristická funkcia pre dôsledky, a stupeň príslušnosti podmienkovej časti (*ancedent*) označovať a_i.

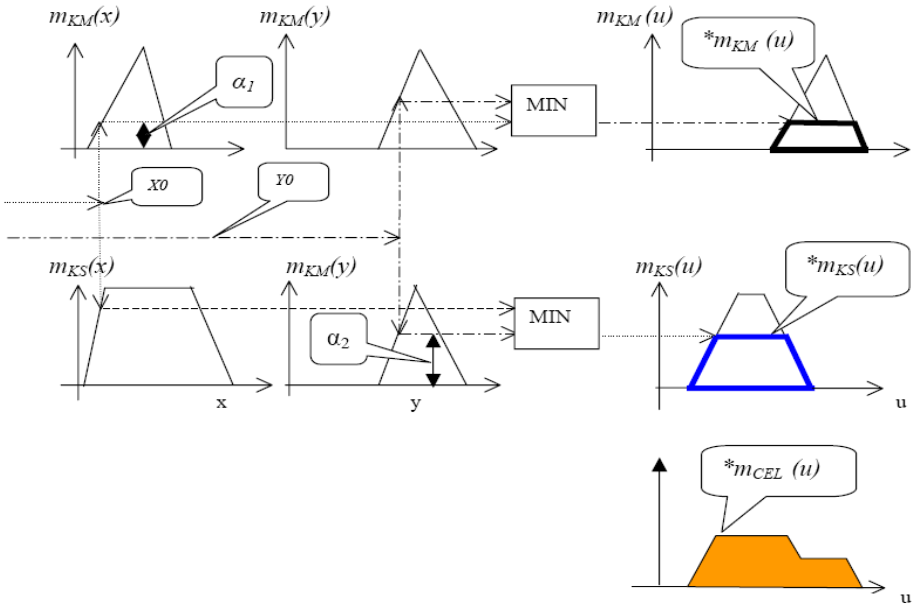
```
IF <x je KM> AND  <y je KM> THEN <u je KM> ELSE

IF <x je KS> AND  <y je KM> THEN <u je KS>
```

Pre podmienkové časti môžeme napísať:

$$a_1 = m_{KM}(x) \wedge m_{KM}(y) = min\{m_{KM}(x), m_{KM}(y)\}$$
$$a_2 = m_{KS}(x) \wedge m_{KM}(y) = min\{m_{KS}(x), m_{KM}(y)\}$$

Pre dôsledky dostaneme:

$$*m_{KM}(u) = a_1 \wedge m_{KM}(u) = min\{a_1, m_{KM}(u)\}$$
$$*m_{KS}(u) = a_1 \wedge m_{KS}(u) = min\{a_1, m_{KS}(u)\}$$



Obrázok 3.5: Nájdenie výstupnej množiny pre dve pravidlá

Dôsledky $*m_{KM}(u)$ a $*m_{KS}(u)$ určujú čiastkové podiely na veľkosti akčnej veličiny. Interpretácia oboch čiastkových výstupov môže byť realizovaná ako ich logický súčet. Pre výstupnú množinu oboch účinkov dostaneme:

$$*m_{CEL} = max\{min\{a_1, m_{KM}(u)\}, min\{a_2, m_{KS}(u)\}\}$$

Rovnaký spôsob možno použiť pre ľubovoľný počet pravidiel.

Výslednú ostrú hodnotu z tohto výsledku získame v procese defuzzifikácie, kde sa môžno použiť napríklad ťažiskovú metódu. Vypočíta sa ťažisko výslednej plochy a jej súradnica (v našom prípade) na osi *u* je ostrá hodnota.

$$u_{výs} = \frac{\int *m(u).udu}{\int *m(u)du}$$

3.1.3.3 Zhodnotenie využitia fuzzy množín

Fuzzy množiny a teda fuzzy regulátor možno s výhodou využiť v procese rozhodovania. Aj keď je možnosť postaviť na tomto princípe i celkové rozhodovanie hráča na výber akcie, celkovo sa však javí optimálnejšie využiť fuzzy regulátor na určenie potrebných hodnôt parametrov na vykonanie už konkrétnej akcie, prípadne jej doladenie.

3.1.4 Využitie kouča pri tvorbe a zmene stratégie

Kouč účinkuje v hre ako 12. hráč. Hoci nemá priamy prístup do hry, dostáva o nej všetky informácie (o všetkých hráčoch, lopte) neskreslené šumom. Do hry môže vstúpiť iba tak, že môže vlastným hráčom predať správu (najčastejšie však každých 500 cyklov). Kouč analyzuje stratégiu protihráčov na základe ich akcií a vytvorí, prípadne upraví, stratégiu hráčov vlastného tímu. Stratégiu protihráča môže analyzovať buď priamo v hre alebo pomocou log súborov z iných hier.

3.1.4.1 Analýza hry

Keď je kouč v online režime, dostáva správy *global see* a v každej tejto správe sú informácie o všetkých hráčoch a lopte. Kouč dostáva ešte jednu správu a to o zmene herného módu. V offline režime berie príkazy miesto servera z log súboru. Tieto správy sa používajú na zistinie vysoko-úrovňových udalostí v hre. Z koučovho pohľadu hra postupuje ako séria priebehov. Zmena priebehu môže nastať pri nasledujúcich situáciach:

- pri zmene hráča, ktorý má loptu
- pri góle
- pri zmene herného módu

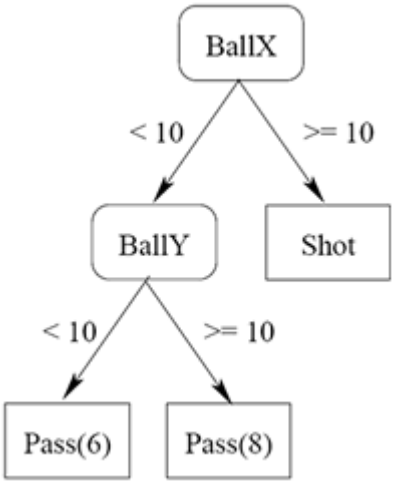
Po každej zmene priebehu kouč analyzuje predošlý priebeh a pokúsi sa ho opísať. Kouč môže napríklad analyzovať priebeh ako driblovanie, ak hráč kopne do lopty, ale tá neopustí jeho vzdialenosť. Alebo môže analyzovať situáciu ako držanie lopty, ak hráč drží loptu niekoľko cyklov.

3.1.4.2 Učenie

Úlohou učiaceho algoritmu je predpovedať daľší krok súpera na základe už naučených akcií a ich postupnosti. Pre každého hráča v tíme je vytvorený rozhodovací strom na základe jeho akcií a pomoucou týchto stromov je možné predpovedať jeho ďalší krok. Učenie sa delí na tieto spôsoby:

3.1.4.2.1 Analýza útoku

Pri učení útoku sa kouč pokúša napodobniť správanie hráča s loptou. Pri analýze berieme do úvahy len prihrávky a strely na bránku. Pomocou parametrov strely a stavu sveta sa vytvorí rozhodovací strom, ktorý je zobrazený na obrázku 3.6.



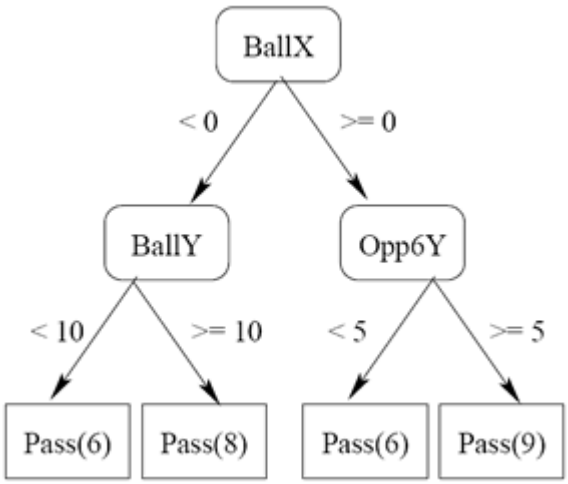
Obrázok 3.6: Rozhodovací strom vytvorený pri analyzovaní útoku

Každý list je akcia a každá cesta skupina podmienok

3.1.4.2.2 Analýza obrany

Na analyzovanie obrany sa kouč snaží modelovať stratégiu nepriateľa a snaží sa rozložiť jeho obranu. Tu je dôležité zisťovať ako hráči získavajú loptu. Uvažujú sa iba prihrávky.

Ukážkový strom pre obrannú radu je na obrázku 3.7.



Obrázok 3.7: Rozhodovací strom vytvorený pri učení sa obrany

3.1.4.3 Učenie formácií

Učenie formácií je podobné ako učenie obrany. Kouč sa snaží zistiť formáciu súpera ktorý porazil protihráča (v inej hre) a na základe toho sa snaží túto formáciu aplikovať na jeho tím.

3.1.4.4 Komunikácia s hráčmi

Na komunikáciu s hráčmi tímu bol vytvorený špeciálny jazyk koučov *Clang[KTWBOOK]*. Tento jazyk umožnuje zapísať pravidlá do textovej podoby tak, aby ich mohli pochopiť hráči, ktorí majú impelentované rozhranie na komunikáciu s koučom. Do tohto jazyka sa zakódujú, akcie ktoré sa vytvorili na základe analýzy rozhodovacích stromov. Keďže jazyk neumožnuje definovanie rolí, tak je dôležité, aby kouč vedel priradiť správneho hráča (identifikovaného číslom uniformy) na danú rolu v strome.

Ukážka CLang:

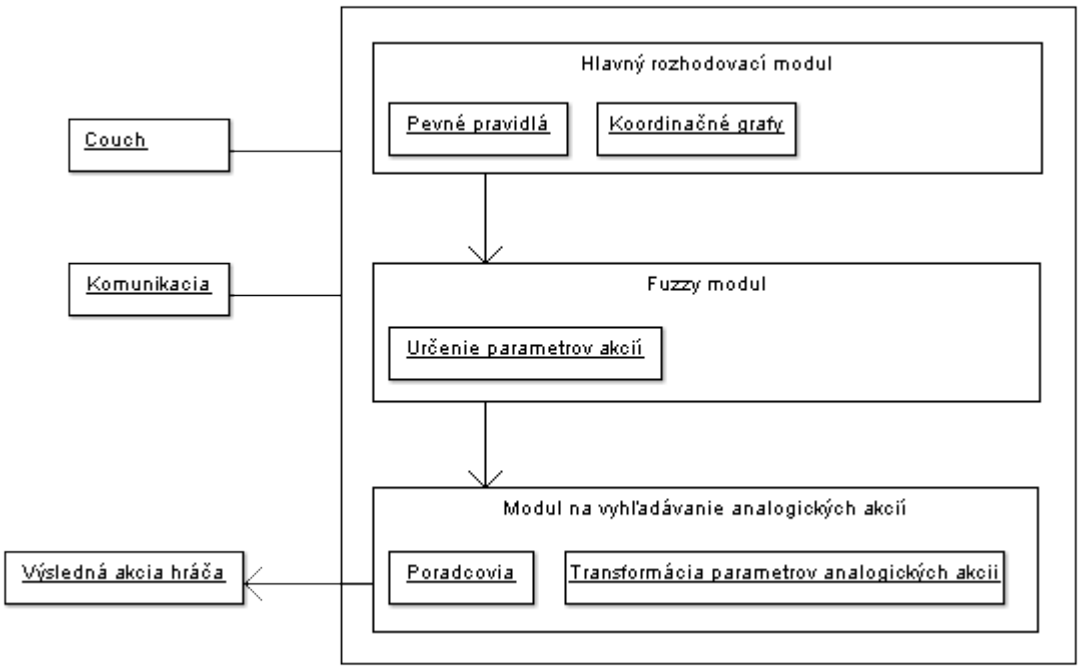
```
(define
 (definerule OffRule1 direc
  ((and (bpos (rec (pt -52.5 -34) (pt 10 34)))
        (bpos (rec (pt -52.5 -34) (pt 52.5 10))))
  (do our {5} (pass {6})))
 )
)
```

Krátky popis ukážky: Definuje sa nové pravidlo OffRule. Musia byť splené 2 podmienky (hovoria o pozícii lopty) a potom sa vykoná prihrávka hráča s číslom 5 hráčovi číslo 6.

3.1.4.5 Implementácia

Kedže vývoj kouča je veľmi komplikovaný proces, rozhodli sme sa, že ho nebudeme implementovať, ale namiesto toho budeme vychádzať z hráča na báze UvaTrilearn s implementovaným modulom na komunikáciu cez CLang. K takto vytvorenému hráčovi sa dá použiť už inde vytvorený kouč.

3.2 Architektúra navrhovaného hráča (rozhodovacia úroveň)



Obrázok 3.8: Navrhovaná architektúra hráča

Východiskom pre navrhovanú architektúru sú horeuvedené metódy používané na rozhodovanie hráča. V obrázku 3.8 je znázornená architektúra hráča spolu s prepojením jednotlivých modulov. Vyhodnocovanie situácie a tvorba reakcie je znázornená prepojeniami šípkami.

3.2.1 Hlavný rozhodovací modul

Hlavný rozhodovací modul na základe formácie a aktuálneho stavu na ihrisku zvolí vhodnú akciu.

3.2.2 Fuzzy modul

Fuzzy logiku sme sa rozhodli použiť na určovanie parametrov akcií. Fuzzy modul odhadne parametre akcií napr. sila kopu, rýchlosť behu, uhol streľby a i.

3.2.3 Modul na vyhľadávanie analogických akcií

Jednotlivé zručnosti môže hráč ovládať na rôznych úrovniach. Cieľom tohto modulu je nahradenie akcií, ktoré hráč ovláda menej za ekvivalentné akcie, ktoré hráč lepšie ovláda. Súčasťou hľadania analógie je transformácia parametrov akcie na parametre cieľovej akcie hráča.

4 Záver

Počas štúdia problematiky RoboCup sme sa oboznámili s rôznymi metódami používanými pri rozhodovaní a správaní hráča. Metódy sú využívané vo viacerých tímoch s rôznou mierou úspešnosti. Preto je pomerne zložité zvoliť vhodnú kombináciu metód, keďže častokrát je zložité odhatnúť efekt danej kombinácie.

Po zvážení viacerých faktov sme sa rozhodli využiť hráča tímu Uva Trilearn. Dôvodom bola jasná a dobre čitateľná architektúra hráča, ako aj kvalitne spracovaný model sveta. Dúfame, že takto hráč zvolený takýmto spôsobom bude jednoducho rozšíriteľný a umožní nám rýchlo sa dostať do problematiky bez toho, aby sme museli opravovať správanie implementované vo viacerých verziách viacerými tvorcami.

Vysokoúrovňové správanie hráča sme sa rozhodli implementovať nanovo použitím prezentovaných metód. Metódy používajúce fuzzy množiny a poradcov neboli vo väčsej miere zastúpené v nami analyzovaných tímoch. Napriek tomu predpokladáme ich úspech.

Prílohy

A. Bibliografia

RSSManual: kol., Robocup Soccer Server Manual, 1999, http://www.dsv.su.se/~johank/RoboCup/manual/download.html

LASTWWW: Členovia tímu L.A.S.T. United, Stránka tímu L.A.S.T. United, 2003, http://www2.dcs.elf.stuba.sk/TeamProject/2003/team05/

FIITWWW: Členovia tímu FIITMedia, Stránka tímu FIITMedia, 2005 , http://www2.dcs.elf.stuba.sk/TeamProject/2005/team02/

SKLOWWW: Členovia tímu SKLO, Stránka tímu SKLO, 2003, http://www2.dcs.elf.stuba.sk/TeamProject/2003/team08/

CMUnited99WWW: kol., Stránka tímu CMUnited 99, , http://www.cs.cmu.edu/~pstone/RoboCup/CMUnited99-sim.html

SDOGWWW: Členovia tímu Stjupit Dox, Stránka tímu Stjupit Dox, 2002, http://www2.dcs.elf.stuba.sk/TeamProject/2002/team01/

DEKOWWW: Členovia tímu Deravá kopačka, Stránka tímu Deravá kopačka, 2002, http://www2.dcs.elf.stuba.sk/TeamProject/2002/team03/index.htm

UVATWWW: Členovia tímu UvA Trilearn, Stránka tímu UvA Trilearn, 2005, http://staff.science.uva.nl/~jellekok/robocup/2005/index_en.html

ROBOWWW: Členovia tímu Robolog, Stránka tímu Robolog, 2002, http://www.uni-koblenz.de/FB4/Institutes/IFI/AGKI/Research/Current/Robolog

FCPOWWW: Členovia tímu FC Portugal, Stránka tímu FC Portugal, 2002, http://www.ieeta.pt/robocup/archive.htm

KnTransAdT: Lisa Torrey, Trevor Walker, Jude Shavlik, Richard Maclin, Knowledge Transfer via Advice Taking, 2005, http://portal.acm.org/citation.cfm?id=1088676

EMCWWW: Wikipedia, http://en.wikipedia.org/wiki/Examples_of_Markov_chains, 2006, http://en.wikipedia.org/wiki/Examples_of_Markov_chains

FUZZYWWW: Jozef Šoltys , Fuzzy regulátor, 1996, http://www.ai-cit.sk/source/publications/thesis/master_thesis/1996/soltys/html/node12.html

KTWBOOK: Lisa Torrey, Trevor Walker, Jude Shavlik, Richard Maclin: Knowledge transfer via advice taking. Banff, Alberta, Canada, 2005, ISBN: 1-59593-163-5

B. Zoznam tabuliek

C. Zoznam obrázkov