VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

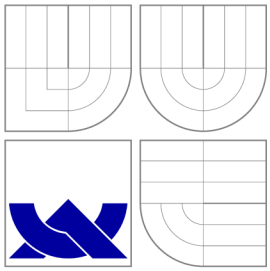
INTELIGENTNÍ ŘÍZENÍ SVĚTELNÉ SIGNALIZACE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

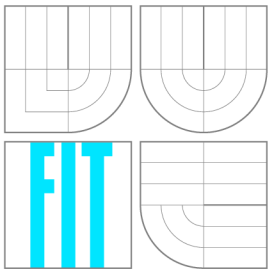
AUTOR PRÁCE
AUTHOR

MILOŠ SKOTÁK

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

INTELIGENTNÍ ŘÍZENÍ SVĚTELNÉ SIGNALIZACE

INTELLIGENT TRAFFIC LIGHTS CONTROL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MILOŠ SKOTÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. ONDŘEJ MALAČKA

BRNO 2012

Abstrakt

V této práci rozebereme multiagentním systémem pro ovládání světelné signalizace. Pro vytvoření reálných situací byl implementován simulátor dopravní situace v jazyce Java, který je založen na celulárním automatu. V multiagentní systému nedochází pouze ke komunikaci mezi agentem a prostředím, ale také i mezi agenty. K implementaci pravidel byl použit rozšířený AgentSpeak s interpretem Jason. Experimenty byly provedeny na reálném silničním úseku ve městě Brně.

Abstract

In this thesis is analyzed Multi-Agent system for intelligent control of traffic lights. For the simulation of a real situation was implemented a simulator of a traffic situation written in Java language, which is based on a cellular automata. In Multi-Agent system can be found not only communication between the agent and environment, but also communication among agents. To implement the rules was used extended AgentSpeak language model based on BSD with Jason interpreter. Experiments were done on a real road section in the city Brno.

Klíčová slova

Inteligentní řízení světelné signalizace, simulace, celulární automat, multiagentní systém, AgentSpeak, Jason

Keywords

Inteligent traffic-light control, simulation, cellular automata, multi-agent system, AgentSpeak, Jason

Citace

Miloš Skoták: Inteligentní řízení světelné signalizace, bakalářská práce, Brno, FIT VUT v Brně, 2012

Inteligentní řízení světelné signalizace

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Ondřeje Malačky. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Miloš Skoták
14. května 2012

Poděkování

Tímto bych velmi rád poděkoval vedoucímu mé bakalářské práce Ing. Ondřeji Malačkovi za poskytnuté odborné vedení a rady během tvorby této práce a všem ostatním, kteří mě podporovali, obzvláště Evě Marcínkové.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Dopravní kongesce, neboli dopravní zácpy, jsou problémem řady větších měst se světelnými křižovatkami. V dnešní době jsou světelné signalizace řízeny dynamicky a využívají časově závislé nastavení programů [1]. Kongesce většinou vznikají neideálním nastavením těchto programů a neschopností flexibilně reagovat na náhlý nárůst či pokles hustoty provozu. Proto se tato bakalářská práce zabývá inteligentním řízením světelné signalizace.

Hlavním cílem bude redukovat fronty vznikající na křižovatkách, snížit dobu čekání ve frontě a zlepšit dynamiku provozu na silnicích, využitím multiagentního přístupu. Výsledky multiagentního přístupu budou porovnávány s výsledky fixního nastavení a agentního přístupu.

Jako způsob řízení byl zvolen multiagentní přístup, který je schopen reagovat na změny hustoty provozu a dokáže komunikovat i s ostatními křižovatkami, a tím docílit lepší synchronizace křižovatek.

V kapitole 2 budou rozebrány možnosti modelování dopravního systému se zaměřením na celulární automat (viz. kap. 2.1.1), u kterého budou zároveň vybrány a navrhnuty pravidla pro pohyb vozidel (viz. kap. 2.1.4). Dále zde budou popsány různé inteligentní způsoby řízení světelné signalizace. V této části se především zaměříme na agentní a multiagentní přístup (viz. kap. 2.2.4). Kapitola 3 se bude zabývat návrhem a implementací dopravního simulátoru. Nastíní se zde i možný způsob realizace získávání dopravních dat a zároveň výběr dat, která jsou při řízení semaforu potřebná. V kapitole 4 bude proveden návrh agenta, který bude moci komunikovat se sousedními křižovatkami. Budou zde definovány informace, které získává od prostředí a pravidla, kterými se bude řídit. Navržení a následné uskutečnění experimentů najdeme v kapitole 5. Budou provedeny dva experimenty. První z nich bude zkoumat dynamiku provozu a druhý se bude zabývat zdržením v křižovatce a délkou front. V závěrečné 6 kapitole budou shrnuty dosažené výsledky a nastíněny možnosti dalšího výzkumu.

[1] http://preference.prazsketramvaje.cz/showpage.php?name=ssz

Kapitola 2

Analýza tématu

2.1 Modelování dopravního systému

V této části se soustředíme na modelování dopravního systému a rozebereme si možnosti, které mohou být použity při simulaci. Nejprve je potřeba si uvědomit, jak dopravní tok budeme modelovat. Jsou dva typy přístupů – mikroskopický a makroskopický model[19].

Makroskopický model je založen na fyzikálních zákonech pohybu plynu v prostoru a použití závislosti hustoty provozu na rychlosti. Tyto rovnice jsou ještě doplněny, abychom mohli modelovat spontánní vznik zácpy a stop & go efekt[1], o další rovnice. Jinými slovy, makroskopický způsob modelování nám nabízí konkrétní chování řidiče, ale neumožňuje jeho flexibilitu, proto tento přístup nebyl pro tuto práci zvolen.

Mikroskopický model je diametrálně odlišný od makroskopického. Tento model nenabízí pouze konkrétní způsob chování řidiče, ale umožňuje simulovat také různé druhy chování. Chování řidiče je ovlivněno okolím, ve kterém se zrovna nachází, ale hlavně dokáže flexibilně reagovat na ostatní řidiče či překážky v provozu. Způsob reakce na vzniklé podněty je ovlivněno pravidly, kterými se každé auto řídí. Na tomto způsobu modelování jsou založeny dva modely. Prvním z nich je celulární automat, druhým je pak kognitivní multiagentní systém (CMAS).

Celulární automat (dále jen CA) je jednoduchý a efektivní způsob simulace prostředí. Využívá pár základních pravidel, které simulují důležité chování řidičů. Například stop & go efekt, zrychlování, zamezení srážky aj. CA bude podrobněji rozebrán v (2.1.1).

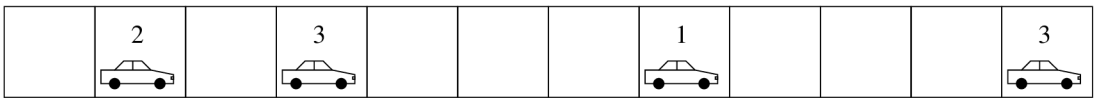
Kognitivní multiagentní systém (dále jen CMAS) je pokročilejší způsob simulace dopravního toku. CMAS využívá tzv. agentů, kteří komunikují s okolím a s dalšími agenty. Komunikace je zde myšlena jako způsob zjišťování polohy překážek a dalších agentů, čehož je následně využíváno pro efektivní plánování trasy. CMAS je založen na reálném chování řidičů, kteří pracují s informacemi, jež získávají v průběhu cesty.

CMAS je velmi efektivní, avšak složitější přístup k simulaci. Naším cílem však není simulace co nejrealističtějšího modelu pohybu vozidla, ale celého dopravního toku. Pro naše zadání se proto více hodí CA.

[1] Stop & go efekt znamená, že jakýkoliv pohyb je po krátké době zastaven. V reálně situaci to vypadá jako popojíždění v zácpě.

2.1.1 Celulární automat

CA je stochastické modelování prostředí, které využívá diskrétního časování. Celý CA se skládá z tzv. buněk. Každá buňka znázorňuje určitou velikost a může být prázdná nebo obsahuje referenci na instanci třídy. Tato třída v sobě uchovává hodnotu, která určuje aktuální rychlost vozidla. Tato hodnota může nabývat stavů: $0, 1, \ldots, v_{max}$. Kde v_{max} je maximální rychlost vozidla. Obsah buněk v CA může vypadat podobně, jako je vyobrazeno na obrázku 2.1.



Obrázek 2.1: Celulární automat – ukázka obsahu buněk

2.1.2 Průměrná délka jedné buňky

Abychom mohli realisticky simulovat dopravní tok je potřeba, aby každá buňka měla určitou velikost. Tato velikost by měla být průměrnou délkou vozidel. Do vozidel zahrnujeme pouze osobní auta, protože se v silničním městském provozu vyskytují nejčastěji. V našem případě použijeme hodnotu 7,5m, která byla použita například i v případě [17]. Tato hodnota v sobě obsahuje průměrnou velikost vozidla a bezpečnou vzdálenost mezi vozidly. Jestliže vezmeme jako průměrnou délku vozidla[2] 4,572m, vyjde nám, že bezpečná vzdálenost je 2,928m. V případě městského provozu, kterým se tato práce zabývá , je toto dostatečná vzdálenost od druhého vozidla, proto jako velikost buňky bude zvolena právě 7,5m. Pokud bychom chtěli brát v úvahu i nákladní vozy, které mohou mít podle zákona maximálně 16,5m, museli bychom zaplnit více buněk a pracovat s nimi jako s množinou, a ne jako se samostatnými prvky.

2.1.3 Stanovení maximální rychlosti v modelu simulace

Další důležitá věc je nastavení maximální rychlosti. Tato hodnota muže nabývat hodnot:

$$v = 0, 1, 2, \ldots, v_{max}$$

Tato hodnota by měla být nastavena tak, aby odpovídala maximální dovolené rychlosti ve městě, např. $50\frac{km}{h}$ nebo 60 $\frac{km}{h}$. Stanovení maximální rychlosti bude tedy záležet na vztahu simulačního času k reálné době. Pokud nastavíme $v_{max} = 4$ a reálná maximální rychlost bude $v_{maxReal} = 60\frac{km}{h}$, pak výsledný vztah $t_{sim} = t_{real}$ bude roven 1krok = 1,8s. Tohoto výsledku docílíme pomocí vzorce:

$$t_{real} = \frac{s}{v_{maxReal}} = \frac{v_{max} * d}{v_{maxReal}}[s]$$

kde:

- t_{real} je reální čas v sekundách,
- v_{max} odpovídá počtu buněk, kterými se projde za 1 krok CA,

[2] Viz. `http://www.hintsandthings.com/garage/stopmph.htm`

- d je velikost jedné buňky v metrech,
- $v_{maxReal}$ je reálná maximální rychlost udávaná v $\frac{m}{s}$.

2.1.4 Pravidla pro CA

Pravidla pro CA jsou velmi důležitým prvkem pro správnou simulaci reálného dopravního provozu. Proto je nezbytné použít vhodná pravidla. V roce 1992 představili Nagel a Schreckenberg (dále jen NaSch) 4 základní pravidla[17], které budou pro CA využity.

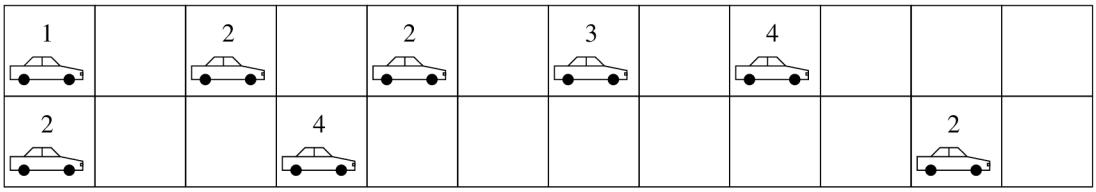
Základními pravidly pro modelování dopravní situace v CA s jedním pruhem podle NaSch modelu jsou:

- **Pravidlo 1:** *Zrychlení* – pokud automobil nedosáhl maximální rychlosti v_{max}, zvětší se jeho rychlost o jedna.
- **Pravidlo 2:** *Bezpečná vzdálenost* – automobil má před sebou d volných buněk. Jestliže je rychlost vozidla v větší, než je počet volných buněk d, sníží se jeho rychlost na d.
- **Pravidlo 3:** *Náhodnost* – každý automobil s pravděpodobností p sníží svojí rychlost o jedna. Toto pravidlo simuluje chování řidičů, kteří ne vždy jedou konstantní rychlostí, ale mohou se zde objevovat určité výkyvy v rychlosti. Společně s pravidlem č.2 simuluje přehnanou reakci na dodržení bezpečné vzdálenosti.
- **Pravidlo 4:** *Pohyb* – každý automobil se posune dopředu o v buněk, kde v je rychlost vozidla, po aplikaci pravidel 1 – 3.

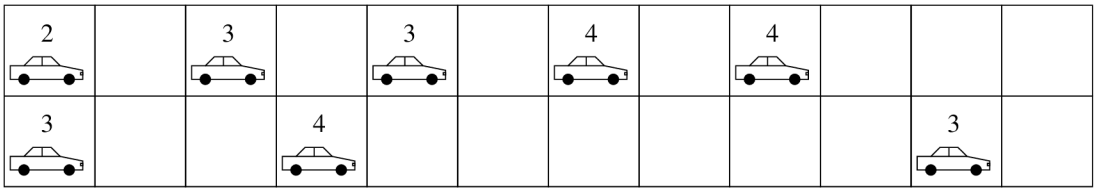
Dále je možné použít tzv. VDR (Velocity - dependent randomization) model, neboli náhodnost závislá na rychlosti. Tento model se nikterak neliší od NaSch modelu, pouze přidává do NaSch pravidel nulté pravidlo, které upravuje pravděpodobnost na snížení rychlosti vozidla v závislosti na rychlosti vozidla. Pro jedoucí vozidla, kde $v > 0$, je pravděpodobnost nastavena na p_1 a pro stojící vozidla, kde $v = 0$, je pravděpodobnost nastavena na p_2.
Bohužel, tyto dva modely nám nestačí, abychom simulovali reálný dopravní provoz. Tyto pravidla je nutné modifikovat na pohyb v n - pruzích, protože existuje jen málo měst s křižovatkami s jedním pruhem. Od druhého základního pravidla proto budeme rozvětvovat a dále přidáme další pravidla, které umožňují změnu pruhů.

Rozšířená pravidla pro modelování dopravní situace v CA s n-pruhy, která využívají modelu NaSch a modelu VDR:

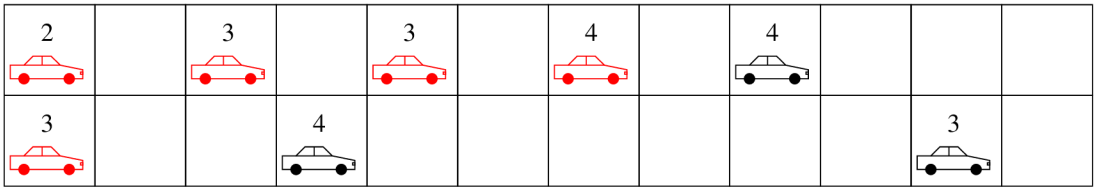
- **Pravidlo 0:** *Náhodnost závislá na rychlosti vozidla* – nastavuje pravděpodobnost náhodného zpomalení. Pro jedoucí vozidla (p_1) a pro stojící vozidla (p_2) je stanovena pravděpodobnost zpomalení.
- **Pravidlo 1:** *Zrychlení* – toto pravidlo je stejné jako v předchozím případě. Aplikace tohoto pravidla je zobrazena na obr. 2.3. Počáteční stav CA je zobrazen na obr. 2.2.
- **Pravidlo 2:** *Změna pruhu* – toto pravidlo se skládá z několika dalších kroků, které je potřeba provést, než bude možné změnit pruh.
  - **Pravidlo 2.1:** *Kontrola bezpečné vzdálenosti* – toto pravidlo je odvozené od pravidla č.3. Pokud vozidlo nemá dostatečný počet volných buněk před sebou a měla by být snížena jeho rychlost, tak se vozidlo pokusí změnit pruh. Aplikace tohoto pravidla je zobrazena na obr. 2.4.



Obrázek 2.2: CA – základní stav

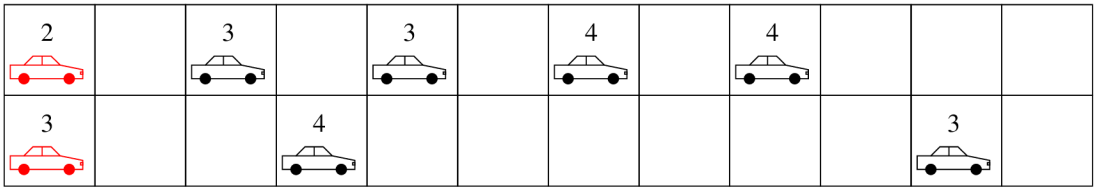


Obrázek 2.3: CA – aplikace 1.pravidla $v_{max} = 4$



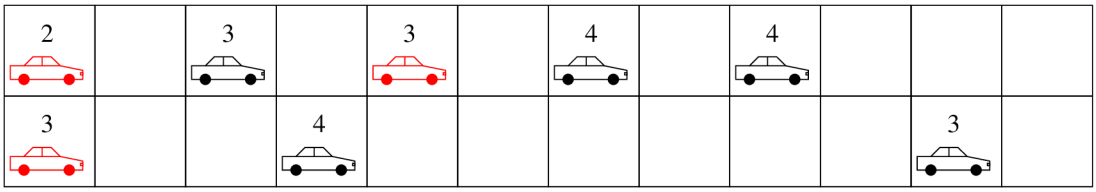
Obrázek 2.4: CA – aplikace 2.1.pravidla

– **Pravidlo 2.2.:** *Kontrola volnosti vedlejšího místa* – než bude možné uvažovat o přesunu vozidla do jiného pruhu, je zapotřebí zjistit, jestli se hned vedle vozidla nenachází jiné vozidlo. Pokud je toto místo volné pokračuje se pravidlem č.2.3, ale pokud je buňka již obsazena, pokračuje se pravidlem č.3. Aplikace tohoto pravidla je zobrazena na obr. 2.5.



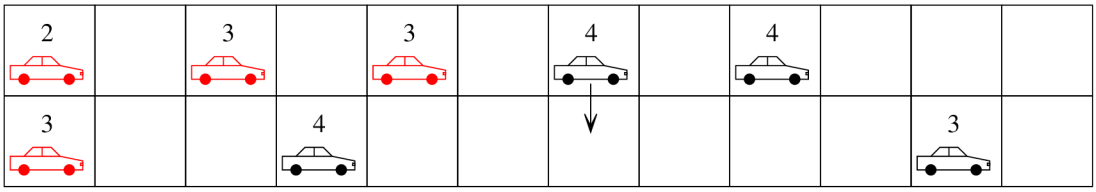
Obrázek 2.5: CA – aplikace 2.2.pravidla

– **Pravidlo 2.3.:** *Kontrola bezpečné vzdálenosti v druhém pruhu* – v tomto pravidle se snažíme předejít nebezpečnému předjíždění do vedlejšího pruhu, a tím ohrožení druhého řidiče. Na začátku simulace si nastavíme bezpečný odstup (d_{odst}), který bude vozidlo držet v bezpečné vzdálenosti od vozidla jedoucího v druhém pruhu za vozidlem prvním. Pokud by nemohl být zachován bezpečný odstup, pokračovalo by se pravidlem č.3. Aplikace tohoto pravidla je zobrazena na obr. 2.6.



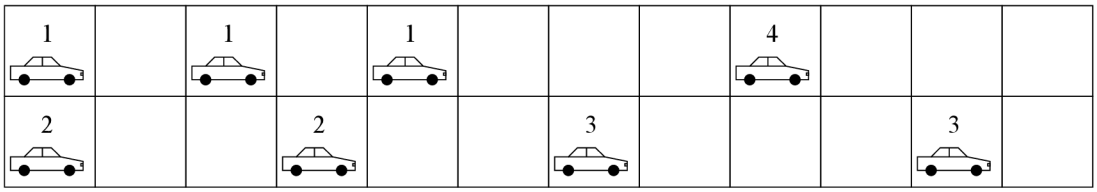
Obrázek 2.6: CA – aplikace 2.3.pravidla $d_{odst} = 1$

– **Pravidlo 2.4.:** *Výhodnost přechodu do jiného pruhu* – bylo by nevýhodné, kdyby se vozidlo přesunulo do druhého pruhu, kde by muselo zpomalit stejně jako v předchozím pruhu anebo by dokonce muselo zpomalit ještě více. V tomto kroku se tedy snažíme najít nejvýhodnější řešení. Pokud najdeme takové řešení, při kterém je výhodnější změna pruhu, vozidlo do tohoto pruhu přesuneme. Aplikace tohoto pravidla je zobrazena na obr. 2.7.



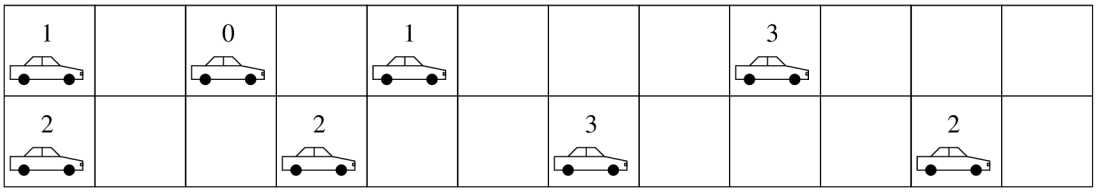
Obrázek 2.7: CA – aplikace 2.4.pravidla

- **Pravidlo 3:** *Bezpečná vzdálenost* – toto pravidlo je stejné jako v předcházejícím případě. Aplikace tohoto pravidla je zobrazena na obr. 2.8.



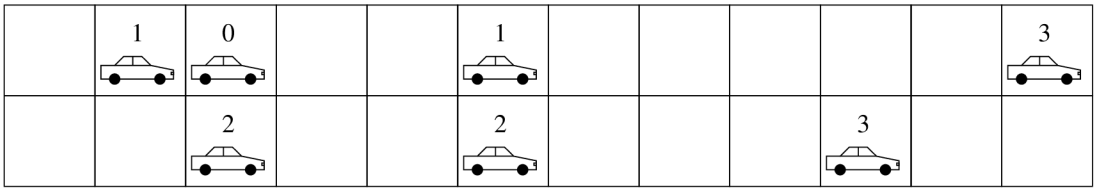
Obrázek 2.8: CA – aplikace 3.pravidla

- **Pravidlo 4:** *Náhodnost* – toto pravidlo je stejné jako v předcházejícím případě. Je zde určitá pravděpodobnost, že vozidlo zpomalí o 1. Aplikace tohoto pravidla je zobrazena na obr. 2.9.



Obrázek 2.9: CA – aplikace 4.pravidla $p_1 = \frac{1}{6}$ a $p_2 = \frac{3}{6}$

- **Pravidlo 5:** *Pohyb* – toto pravidlo je stejné jako v předcházejícím případě. Aplikace tohoto pravidla je zobrazena na obr. 2.10.



Obrázek 2.10: CA – aplikace 5.pravidla

2.2 Způsoby inteligentního řízení dopravy

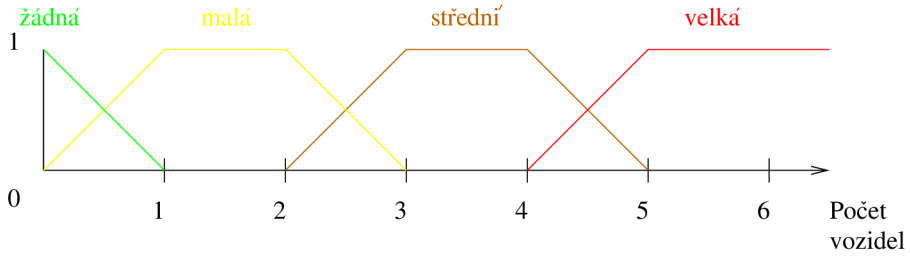
Hlavní problematikou, kterou se budeme v této kapitole zabývat, budou způsoby, jak zmenšit fronty a zlepšit čas průjezdu křižovatkou. Rozebereme si zde možnosti, kterými můžeme těchto zlepšení docílit. Postupně si zde představíme 5 různých přístupů. Budou to:

- fuzzy logika (viz.2.2.1),
- expertní systémy (viz.2.2.2),
- neuronová síť (viz.2.2.3),
- agentní systémy (viz.2.2.4),
- multiagentní systémy (viz.2.2.4).

U každé z těchto metod rozebereme jejich výhody i nevýhody a probereme vhodnost pro naší simulaci.

2.2.1 Fuzzy logika

Fuzzy logika patří do podoboru matematické logiky. Nemá jenom dva stavy (pravda/nepravda), ale dokáže pokrýt realitu v její nepřesnosti a neurčitosti. Fuzzy logika nabízí úplné, částečné a žádné členství. To znamená, že prvek patří do množiny s jistou pravděpodobností (stupeň příslušnosti). Funkce, která každému prvku universa přiřadí stupeň příslušnosti se nazývá funkce příslušnosti[16]. Příklad fuzzy pravidel pro frontu na křižovatce, kde budou brány stavy "prázdná", "malá", "střední", "velká", popisuje obr.2.11.



Obrázek 2.11: Příklad fuzzy pravidel pro frontu na křižovatce.

Mezi základní výhody fuzzy logiky patří[6]:

- snadná implementace,
- jednoduchý a přirozený koncept,
- flexibilita.

Použití fuzzy logiky dokáže zvýšit efektivitu průjezdu křižovatkou. To znamená, že vozidla budou čekat na křižovatce kratší dobu a doba průjezdu křižovatkou se také sníží. Mimo jiné nedochází k prudkým výkyvům těchto hodnot, ale tyto hodnoty jsou ustáleny v určitém rozmezí[18]. U křižovatek, které mají pevně nastavenou dobu zelené, jsou tyto hodnoty větší a dochází u nich k prudkým výkyvům.

Nevýhoda fuzzy logiky je ta, že je vhodná pouze na ovládání jedné křižovatky. S množinou křižovatek by nastaly komplikace. Přesněji, nastali by problémy s definováním rozhodovací logiky, protože existuje příliš mnoho faktorů a závislostí řízení soustavy křižovatek.

2.2.2 Expertní systémy

Expertní systémy (dále jen ES) využívají množinu pravidel pro určení následujícího kroku. K vytváření pravidel se využívají experti z dané oblasti problematiky, aby se předešlo problémům se špatně nadefinovanými pravidly. ES sbírají malé kousky znalostí, ze kterých pak dokážou vytvořit nová pravidla[13], díky čemuž se celý systém může postupně vyvíjet a dosahovat lepších výsledků.

Výhoda ES je ta, že dokáží vytvářet nová pravidla. Další výhodou je, že umí ovládat množinu křižovatek, a proto dosahují lepších výsledků, než metody, které ovládají pouze jednu křižovatku.

Nevýhoda ES je ta, že je velmi důležité nadefinovat správná pravidla. Pokud tyto pravidla nebudou správná, nemuselo by se dostavit zlepšení průjezdu křižovatkou, ale naopak by se situace mohla výrazně zhoršit[7]. Proto jsou k nadefinování těchto pravidel potřeba experti v dané oblasti problematiky.

2.2.3 Neuronové sítě

Neuronové sítě (dále jen NS) jsou prostředkem pro zpracovávání neúplných dat. Základním prvkem NS je neuron, který je inspirován neuronem z lidského mozku. Architektura NS je paralelní, a proto dosahuje při paralelním zpracování, ve srovnání s ostatními metodami, lepších a rychlejších výsledků. Každá NS obsahuje 3 základní vrstvy[8]: vstupní, skrytou a výstupní. Velmi důležitá je část učení. Každou NS je potřeba natrénovat na daný problém, který budeme řešit.

Největší výhodou NS je, že může využívat paralelní zpracování. Díky tomu může velmi efektivně pracovat s velkou množinou dat ze vstupu[4]. Další výhodou je, že po natrénování je NS schopna reagovat velmi rychle a správně.

Pro učení NS je potřeba mnoho relevantních informací, abychom dosáhli dobrých výsledků. Většina těchto informací je použita na trénování a zbytek těchto dat je využit pro testování správnosti natrénování NS[3]. Je tedy velmi důležité, aby se v trénovacích datech nenacházely chyby.

2.2.4 Agentní a multiagentní systémy

Základní vlastnosti

Agentní systém (dále jen AgS) nebo multiagentní systém (dále jen MAS) vždy obsahuje *agenty* a *prostředí*[21]. Agent vnímá prostředí pomocí svých senzorů a samostatně vykonává akce v prostředí pomocí efektorů tak, aby vyhověl stanoveným požadavkům. Základní vlastnosti agentů jsou[20]:

- Autonomie – agent se rozhoduje nezávisle v rámci daného systému.
- Reaktivita – agent reaguje na změny v prostředí tak, aby dosáhl svých cílů.
- Proaktivita – ovlivňování okolí tak, aby agent co nejsnadněji dosáhl svého cíle.

Existují 4 druhy agentů a každý druh agenta má jiné vlastnosti. Tyto druhy jsou[20]:

- Reaktivní agent – tento typ má autonomní a reaktivní vlastnosti. Tento agent reaguje na podněty.
- Deliberativní agent – tento typ má autonomní, reaktivní a proaktivní vlastnosti. Má schopnost plánování svých akcí dopředu.
- Kognitivní agent – tento typ má autonomní a reaktivní vlastnosti, proaktivita v tomto případě není podmínkou, ale možností. Tento agent má schopnost vyvozovat logické závěry z pozorování svého prostředí.
- Racionální agent – tento typ má všechny vlastnosti výše uvedených agentů. Dokáže plánovat akce dopředu a dokáže vytvářet nová pravidla na základě pozorování svého prostředí.

V AgS komunikují agenti pouze s prostředím. Pokud by se v prostředí nacházeli další agenti, agent se o nich dozví díky komunikaci s prostředím. Pokud by se jednalo o MAS, tak by agenti komunikovali nejen s prostředím, ale i mezi sebou. To je zásadní rozdíl mezi AgS a MAS. Jelikož je prostředí důležitým prvkem v AgS i v MAS, je potřeba říci, jaké mohou být jeho vlastnosti z hlediska agentů[20]:

- Plně/částečně pozorovatelné – plně pozorovatelné prostředí znamená, že agent může získat všechny informace z prostředí pomocí svých detektorů.
- Statické/dynamické – pokud se prostředí mění pouze akcemi agenta, je toto prostředí statické.
- Deterministické/nedeterministické – deterministické prostředí je závislé na předcházející akci.
- Diskrétní/spojité – prostředí je diskrétní, pokud má konečné množství nebo spočetně mnoho stavů.

Umělá inteligence agentů využívá bázi znalostí, ale může využívat i neuronové sítě, či fuzzy – expertní systém[15]. Pokud by agenti nevyužívali samotnou bázi znalostí jednalo by se o tzv. hybridní systémy. Znalostní databáze se programuje pomocí funkcionálních jazyků.

Výhody a nevýhody

Výhoda AgS je, že agent se dokáže učit a vyvíjet. Další jeho výhodou je, že dokáže plánovat určité změny dopředu a zároveň dokáže flexibilně reagovat na nečekanou změnu provozu[15]. V našem případě jsou výhody MAS všechny výhody uvedené výše, plus jeho schopnost řídit množinu křižovatek.

Jak již vyplývá z výše uvedeného, nevýhodou AgS je, že nedokáže řídit množinu křižovatek, ovšem tento nedostatek řeší MAS. Implementace AgS a MAS je komplexně složitější.

Řízení světelné signalizace pomocí multiagentních systémů

V článku [5] je použit simulátor dopravní situace, který je vytvořen pomocí NetLogo[3] . Tento simulátor využívá pár předpokladů ohledně dopravních dat, jako například:

- Velikost fronty je definována jako rozdíl dvou senzorů, které jsou umístěny před křižovatkou.
- Velikost místa za křižovatkou je definován jako rozdíl dvou senzorů, které jsou umístěné za křižovatkou.
- Příchozí intenzita udává, kolik aut se dostane do dané fronty za sekundu.
- Odchozí (servisní) intenzita udává, kolik aut se dostane z dané fronty za sekundu.

Díky senzorům, které simulátor využívá, dostaneme kvalitní vyhodnocení provozu. Báze znalostí obsahuje 8 pravidel plus několik podpravidel, které slouží pro práci se semafory a pro spolupráci jednotlivých agentů. Pravidla pro práci se semafory zahrnují například

[3]NetLogo je multiplatformní prostředí napsané v Javě, určené pro multiagentní modelování komplexních systémů [12]

prodloužení času zelené anebo změnu stavu semaforu. Každý agent má pouze informace ohledně své křižovatky, proto je komunikace mezi agenty velmi důležitá. Výsledky experimentu se 4-mi křižovatkami, které byly provedeny v rámci článku, potvrzují, že MAS je lepší než AgS. Průměrná doba zpoždění vozidla pro AgS byla 82,72 s a pro MAS to bylo 66,16 s.

V článku [15], který využívá agenty s proaktivními vlastnostmi, jsou tito agenti ještě rozděleni na 4 druhy:

- agenti úseku silnice (RSA – Road Segment Agent),
- agenti křižovatek (ITSA – Intelligent Traffic Signalling Agent),
- agenti oblasti (Areas Agent), kteří pracují se specificky vymezenou oblastí,
- agenti tras (Route Agents), kteří v sobě mohou zahrnovat několik RSA.

ITSA zahrnuje 4 akce (sběr a distribuci dat pomocí RSA; analýza dopravní situace; optimalizace; rozhodování/spolupráce agentů), kterými se snaží dosáhnout globálního optima, s využitím všech dostupných informací. ITSA v sobě zahrnuje predikční model, který se snaží zohlednit budoucnost, a tím optimalizovat řešení. Zároveň v sobě obsahuje i meta model, který porovnává přesnost predikce a popřípadě upraví parametry pro predikci. S tímto modelem však nebyly provedeny žádné experimenty a tak nelze zjistit, o kolik se mohou výsledky zlepšit.

Kapitola 3

Návrh a implementace dopravního simulátoru

Tato kapitola se bude zabývat návrhem a implementací dopravního simulátoru. V kapitole 3.1 pak rozebereme důležité části, které by měl dopravní simulátor obsahovat a hlavně, z čeho by se měl skládat. V kapitole 3.2 bude popsán způsob implementace jednotlivých prvků.

3.1 Návrh

3.1.1 Základní prvky simulátoru

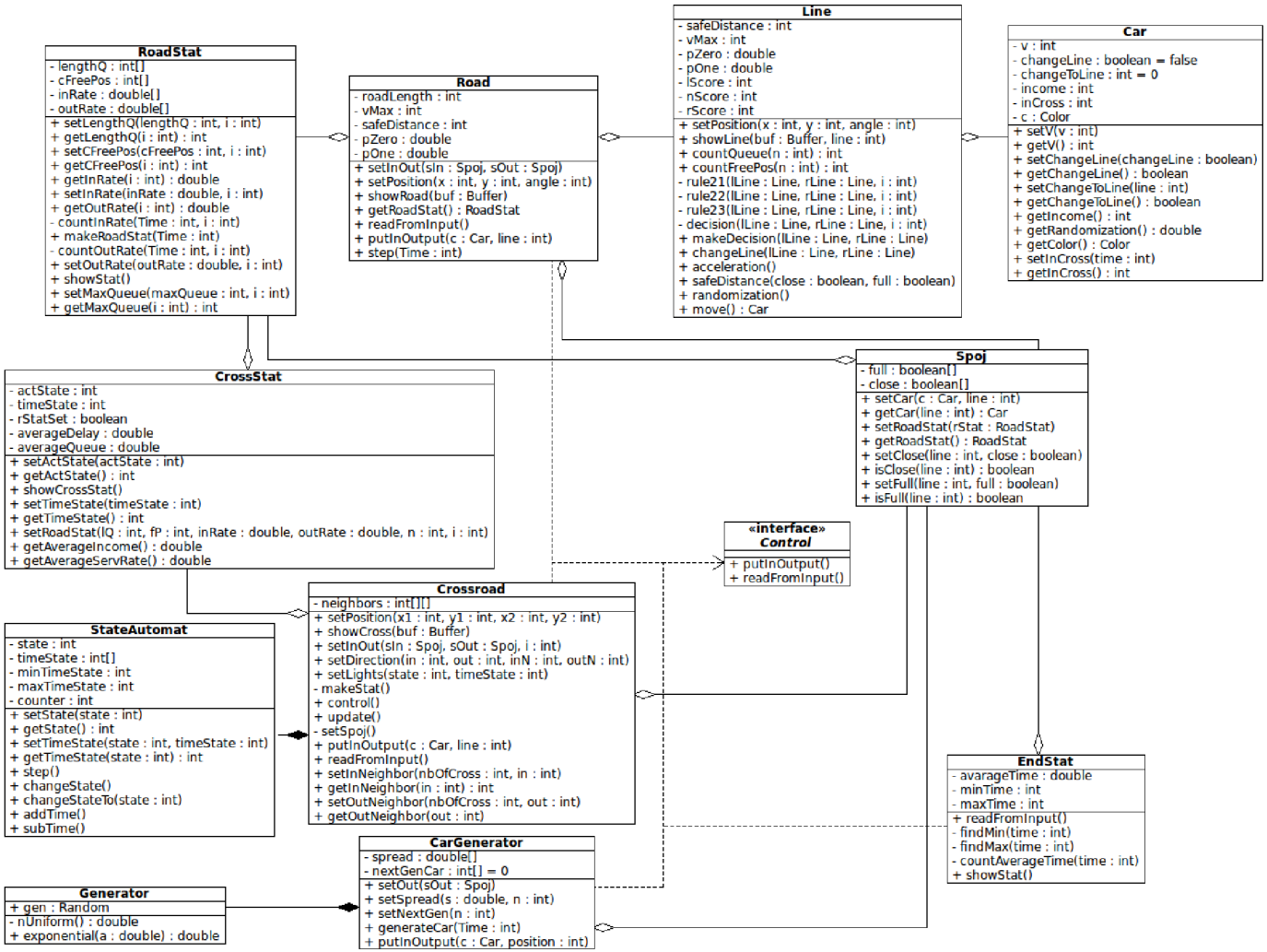
Při dekompozici dopravního úseku byly definovány 5 základních stavebních prvků. Jsou to:

- generátor aut,
- silnice,
- křižovatka,
- výstup,
- spoj.

Každý z těchto prvků zastává důležitou funkci při simulaci. *Generátor aut* používá k nastavení Poissonovo rozložení a převádí ho na exponenciální rozložení. *Silnice* se skládá z jednotlivých pruhů. Každý z těchto pruhů dodržuje pravidla, která byla definována v kapitole 2.1.4. *Křižovatka* v sobě obsahuje světelnou signalizaci a určuje, která *silnice* má aktuálně volný průjezd. Světelnou signalizaci podrobně popisuje kapitola 3.1.2. *Výstup* v sobě zpracovává vozidla, která dokončila svoji trasu. *Spoj* všechny předešlé komponenty propojuje a zajišťuje, aby vozidla byla správně vložena do daných prvků. Prvky *silnice* a *křižovatka* navíc budou uchovávat statistiky dopravních informací. *Křižovatka* bude zpracovávat jednotlivé informace ze *silnic*, které jsou dostupné pomocí *spojů*. Tyto informace jsou detailněji popsány v kapitole 3.1.3.

Dalším prvkem, který je důležitý v dopravě, je *vozidlo*. Tento prvek v sobě nese informace o své rychlosti, příchodu do vozovky a dobu čekaní na křižovatce. Při vizualizaci také uchovává informaci o barvě, aby se odlišil od ostatních *vozidel*. Trasa *vozidla* je závislá na pravidlech *silnice* a na typu pruhu, ve kterém se nachází.

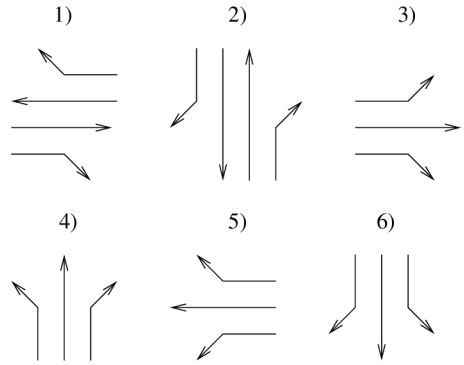
Na základě těchto informací byl sestaven diagram tříd, který popisuje jednotlivé vztahy prvků simulátoru. Tento UML diagram lze vidět na obrázku 3.1.



Obrázek 3.1: Diagram tříd pro prvky simulátoru dopravy

3.1.2 Světelná signalizace

Velmi důležitým prvkem v dopravním simulátoru je světelná signalizace. K jejímu řízení byl využit stavový automat, který obsahuje 6 stavů. Tyto stavy lze vidět na obrázku č. 3.2. Každý z nich má svoji délku trvání, která udává, kolik kroků simulace bude daný stav trvat.



Obrázek 3.2: Stavy světelné signalizace

Jelikož agent reprezentuje světelnou signalizaci, byly definovány akce, se kterými může měnit a upravovat délku stavů. Tyto akce jsou podrobněji rozebrány v kapitole 4. Délka stavů musí splňovat dvě pravidla. Základní délka nesmí klesnout pod minimální délku a nesmí být větší, než je maximální délka stavu. Minimální trvání je nastaveno na 50 % inicializační hodnoty stavu a maximální délka je nastavena na 300 %. Toto omezení bylo stanoveno kvůli zamezení eliminaci stavů (snížení délky na 0) a neúnosného rozsahu (příliš vysoké délce stavu).

3.1.3 Statistiky

Statistiky jsou velmi důležitou částí pro dopravní simulátor. Díky statistikám lze vyhodnotit přínos inteligentního řízení a zároveň slouží k nalezení ideální reakce agenta (světelné signalizace).

K tomu, aby bylo možné nalézt ideální reakci, jsou potřeba znát určité informace. Tyto informace jsou:

- aktuální stav světelné signalizace,
- čas do konce aktuálního stavu (v krocích),
- aktuální délka fronty jednotlivých vstupů,
- maximální délka fronty vstupů,
- počet volných míst na jednotlivých výstupech.

Aktuální stav a čas do konce stavu se používá přímo ze světelné signalizace, proto jsou tyto hodnoty uloženy v *křižovatce*. Každá *silnice* si zpracovává a uchovává záznamy o aktuální a maximální délce fronty a o počtu volných míst na výstupu.

K vyhodnocení přínosu inteligentního řízení bylo potřeba použít takové údaje, které by dokazovaly zlepšení (respektive zhoršení). K tomuto účelu byly použity tyto informace:

- průměrná délka fronty na křižovatce,
- průměrné zdržení na křižovatce,
- intenzita příjezdu vozidel,
- intenzita obsluhy.

Průměrná délka front je zpracovávána ze záznamů, které jsou uchovávány v *křižovatce*. Průměrné zdržení na křižovatce počítáno od doby, kdy auto vstoupilo do fronty (minulo první detektor), až po samotný výjezd z křižovatky (minulo druhý detektor). Čas vstupu do křižovatky je uchováván společně s informacemi o vozidle. Intenzita vypovídá o dynamice provozu dané vozovky. Intenzita příchodu udává, s jakou četností se do vozovky dostávají vozidla [11] a intenzita obsluhy pak říká, s jakou četnosti vozidla opouštějí vozovku. Intenzitu vypočítáme pomocí vzorce:

$$I = \frac{P_{voz}}{t_{act}}$$

kde:

- I je intenzita (příchodů/obsluhy),
- P_{voz} je celkový počet vozidel, které přišli/opustili frontu,
- t_{act} je aktuální čas simulace.

Koncové statistiky slouží k zjištění, kolik aut dokončilo trasu, a také průměrnou, maximální a minimální dobu průjezdu dopravním úsekem.

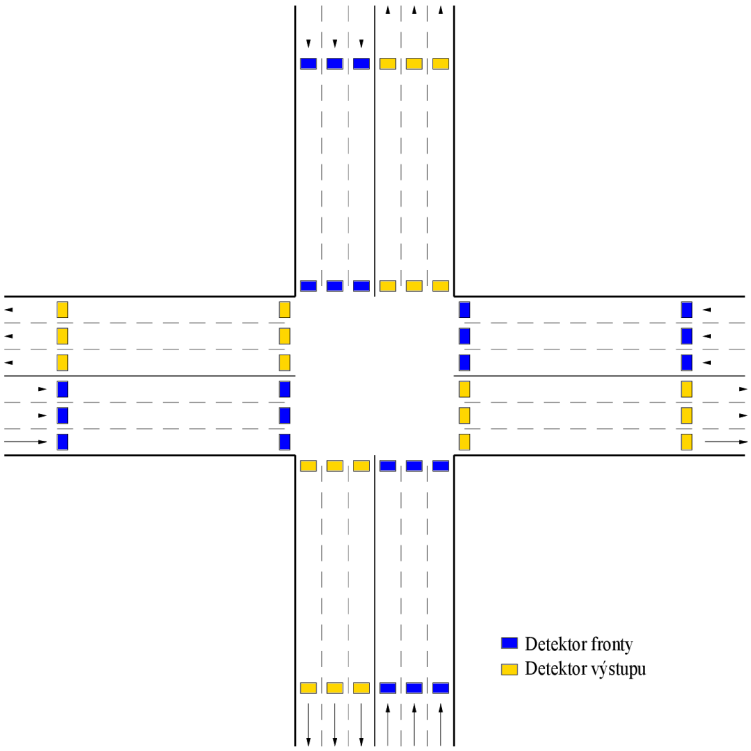
Reálné získávání statistik

Tato kapitola se zabývá možným nasazením MAS v praxi.

Tyto statistiky je potřeba měřit na základě reálného způsobu sběru dat o dopravní situaci. K tomuto sběru slouží různé druhy senzorů. Existují dvě hlavní dělení.
Intrusivní dopravní detektory, které zasahují do vozovky a tzv. neintrusivní dopravní detektory, které nezasahují do vozovky [2]. Do instruktivních detektorů patří indukční detekční smyčky či magnetické detektory. Indukční detekční smyčky jsou přesné a spolehlivé, ale mají omezené možnosti umístění a vyšší pořizovací cenu. Magnetické detektory jsou méně invazivní do vozovky, mají vyšší mechanickou odolnost a menší pořizovací náklady. Nevýhoda těchto magnetických detektorů je ta, že nedokáží snímat vozidla s příliš nízkou rychlosti ($v < 5\frac{km}{h}$).
Do neintrusivních dopravních detektorů patří mikrovlnné detektory (radary), aktivní laser a videodetekce. Neintrusivní detektory mají společnou nevýhodu a ta je, že k získání spolehlivých dat je potřeba dobrého počasí. Sníh, hustý déšť či mlha mohou tyto výsledky značně zkreslit. Jejich společná výhoda je, že nezasahují do vozovky a její oprava proto nemusí zablokovat dopravní pruhy. Vhodnou kombinací těchto dvou přístupů by mohlo být dosaženo eliminace nedostatků, zajištění efektivity a hlavně spolehlivosti.

Způsob rozmístění detektorů musí pokrývat všechny pruhy vozovky a musí být ideálně rozloženy tak, aby se eliminovaly chyby měření. Rozmístění těchto detektorů je vidět na obrázku 3.3. Tento způsob rozmístění však není vhodný pro indukční smyčky [9] a využití

magnetických detektorů by mohlo způsobovat nepřesnosti měření. Proto jsou v tomto případě předpokládány neintrusivní typy detektorů. Jednotlivá čidla jsou od sebe vzdálena 150 m. Pokud by byla silnice kratší než 150 m tato vzdálenost se upraví na nejbližší možnou vzdálenost. Sběr dopravních dat v simulátoru je založen na tomto principu, aby bylo dosaženo stejných podmínek jako při sběru informací v reálné dopravě.



Obrázek 3.3: Rozmístění detektorů na vozovce

3.2 Implementace

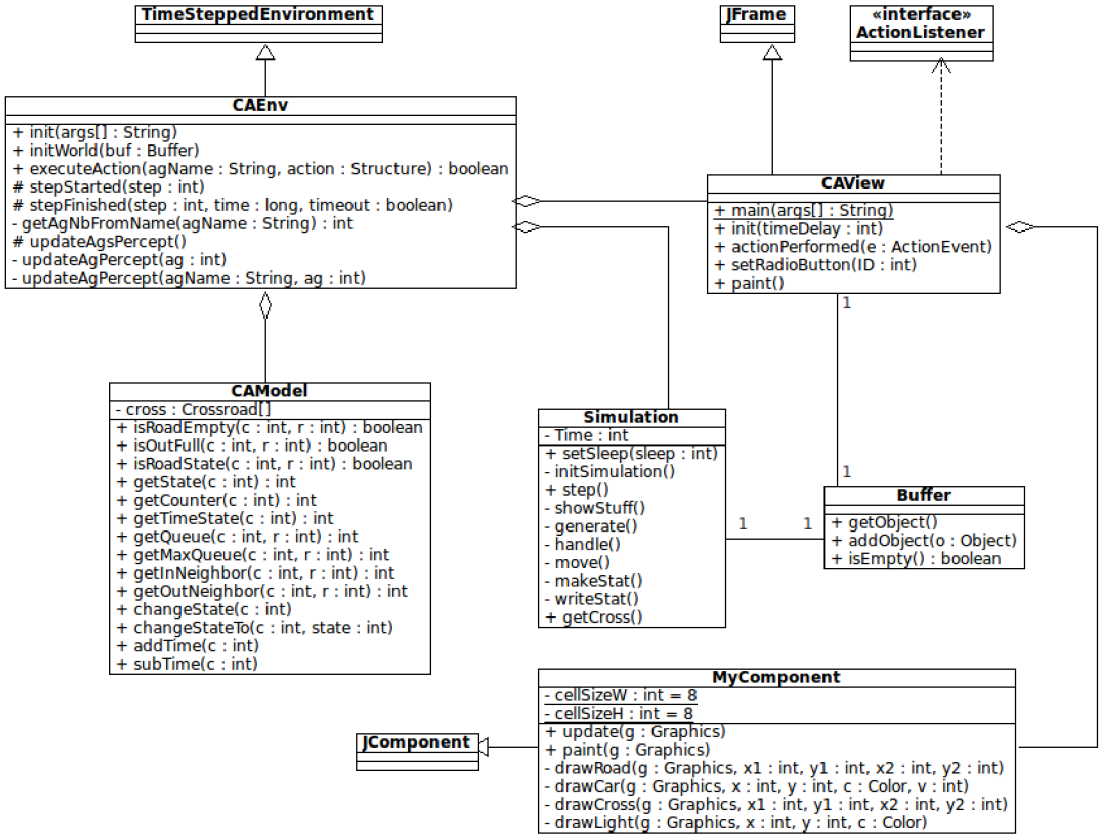
3.2.1 Použité technologie

K naprogramování dopravního simulátoru byl použit interpret Jason jazyka agentspeak a z toho důvodu byl použit programovací objektový jazyk Java, ve kterém se programuje prostředí agentů. Jason poskytuje platformu pro vývoj MAS a je dostupný pod licencí LGPL.

Kompletní simulátor byl implementován podle diagramu tříd, který je vidět na obrázku 3.4. Tento návrh zahrnuje jak komunikaci prostředí s agentem, tak samotnou vizualizaci prvků simulátoru.

3.2.2 Řešení směrování pruhů

Při spojování dvou silnic jsou velmi důležité třídy `Spoj` a `Crossroad`. K propojení se použije výstupní `Spoj` jedné silnice a vstupní `Spoj` druhé silnice a pomocí metody `setInOut(...)` třídy `Crossroad` se nastaví daný směr. Jednotlivé pruhy jsou tak nastaveny na přímý směr.



Obrázek 3.4: Diagram tříd pro simulátor dopravy.

Každý pruh však nemusí vést pouze rovně, ale může být i odbočovací. Z toho vyplývají určité doplňující informace. Je potřeba vědět kam každý pruh směřuje a tuto informaci někam uložit. Právě toto směrování bylo definováno ve třídě `Spoj`. Tato informace v sobě uchovává do jakého výstupu a pruhu vede daný pruh. K nastavení tomuto směrování slouží funkce `setDirection(...)`, která je ve třídě `Crossroad`. Funkce má v sobě kontrolu hodnot, které chceme nastavit. Pokud by hodnoty nebyly ve správném rozsahu, neprovede se žádná změna.

Při předávání vozidla třídě `Spoj`, je výstupní `Spoj` vyhodnocen pomocí metody `readFromInput` třídy `Crossroad`. Pokud by informace o směru nebyla definována, bude vozidlo předáno na svůj přímý výstup.

3.2.3 Propojení simulátoru s agentem

Propojení simulátoru s agentem bylo provedeno pomocí třídy `TimeSteppedEnvironment`[1], která umožňuje synchronizaci více agentů a zároveň dovoluje při začátku kroku provést krok našeho simulátoru. Byla vytvořena třída `CAEnv`, která dědí vlastnosti třídy `TimeSteppedEnvironment`. Politika zpracovávání akcí, které obdrží od agentů byla nastavena na `OverActionsPolicy.ignoreSecond`[2]. To znamená, že pokud by přišla druhá akce, bude se ignorovat a bude pokládána za úspěšně vyřešenou. Je to nastaveno z toho důvodu, že je povolena pouze jedna akce pro daný krok.

Implementace akcí, které přijdou od agentů zajišťuje metoda `executeAction(String agName, Structure action)`. V této metodě bylo potřeba implementovat reakce na specifické akce vyvolané agenty. Tyto akce většinou zasahují do simulátoru, proto třída `CAEnv` obsahuje třídu `CAModel`, která zprostředkovává bezpečný přistup ke křižovatce. Pokud by agenti pracovali s reálným prostředím, třída `CAModel` by nekomunikovala se simulátorem, ale přímo s reálnými prvky.

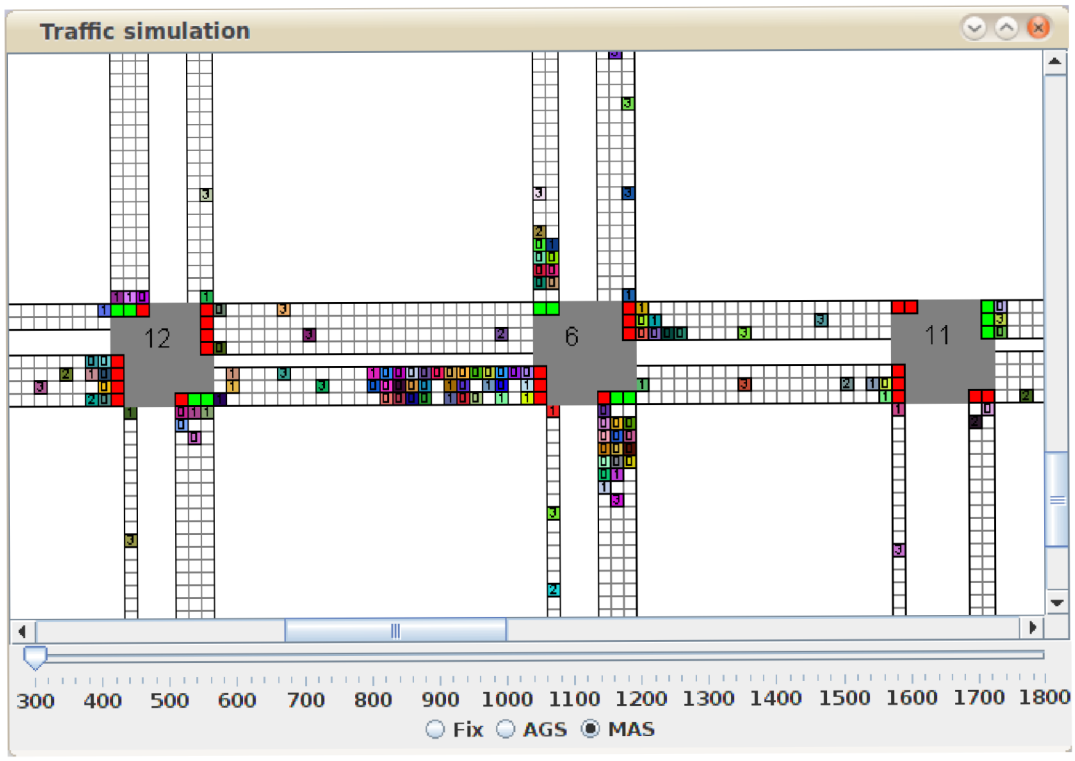
3.2.4 Vizualizace

K vizualizaci byly použity knihovny `Swing` a `AWT`. Z knihovny `Swing` bylo využito třídy `JComponent`, která slouží pro vykreslování aktuálního stavu simulátoru. Tato třída samotná by však nedokázala správně vykreslit všechny prvky, proto byla vytvořena třída `MyComponent`, která dědí metody třídy `JComponent`. Díky tomuto lze jednoduše vykreslovat jednotlivé prvky dopravního úseku. Při nízkém modelovém čase se během vykreslování aktuálního stavu simulace, vyskytl rušivý element. Problém se projevoval jako problikávání obrazu, ale byl vyřešen pomocí tzv. double buffering[3].

Samotné vykreslování těchto prvků by však nestačilo. Bylo potřeba využít možnosti výběru způsobu řízení křižovatky, či možnost ovlivňovat rychlost modelového času. Výběr způsobu řízení křižovatky byl vyřešen pomocí `JRadionButton` kombinovaný se třídou `ButtonGroup`. Na nastavování rychlosti modelového času byla využita třída `JSlider` a metoda `setSleep(...)` třídy `CAEnv`, která nastavovala prodlevu mezi jednotlivými kroky. Čas je možné nastavovat od 300 ms až po 1800 ms.

Po implementaci všech navrhovaných části byl vytvořen simulátor, který lze vidět na obrázku 3.5.

[1] `http://jason.sourceforge.net/api/jason/environment/TimeSteppedEnvironment.html`
[2] `http://jason.sourceforge.net/api/jason/environment/TimeSteppedEnvironment.OverActionsPolicy.html`
[3] `http://www.ecst.csuchico.edu/~amk/classes/csci00P/double-buffering.html`



Obrázek 3.5: Výsledná podoba simulátoru.

3.2.5 Testování

Výsledná aplikace je platformě nezávislá a byla testována na systémech:

- GNU/Linux Ubuntu 10.04 LTS – Lucid Lynx.
- Microsoft Windows XP Professional.

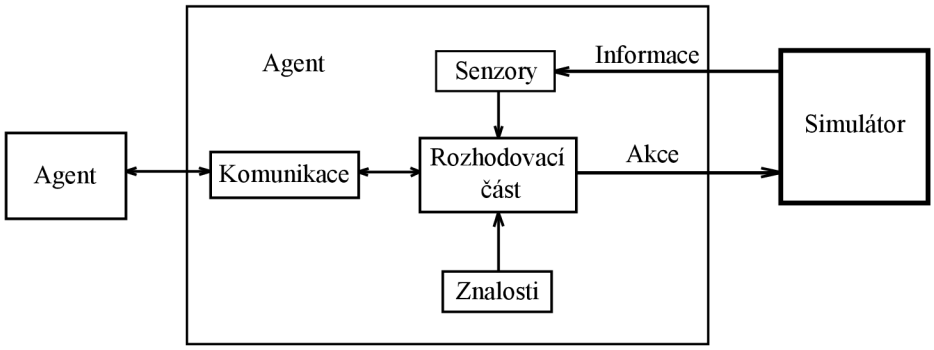
Kapitola 4

Návrh agenta

4.1 Obecně

Agent bude v našem případě představovat světelnou signalizaci. K implementaci pravidel byl využit rozšířený agentně orientovaný programovací jazyk AgentSpeak, který je založen na modelu BDI (Belief-Desire-Intention [10]) a je rozšířen o komunikační protokoly mezi agenty [1]. Pro použití tohoto rozšířeného jazyka, byl využit interpret Jason.

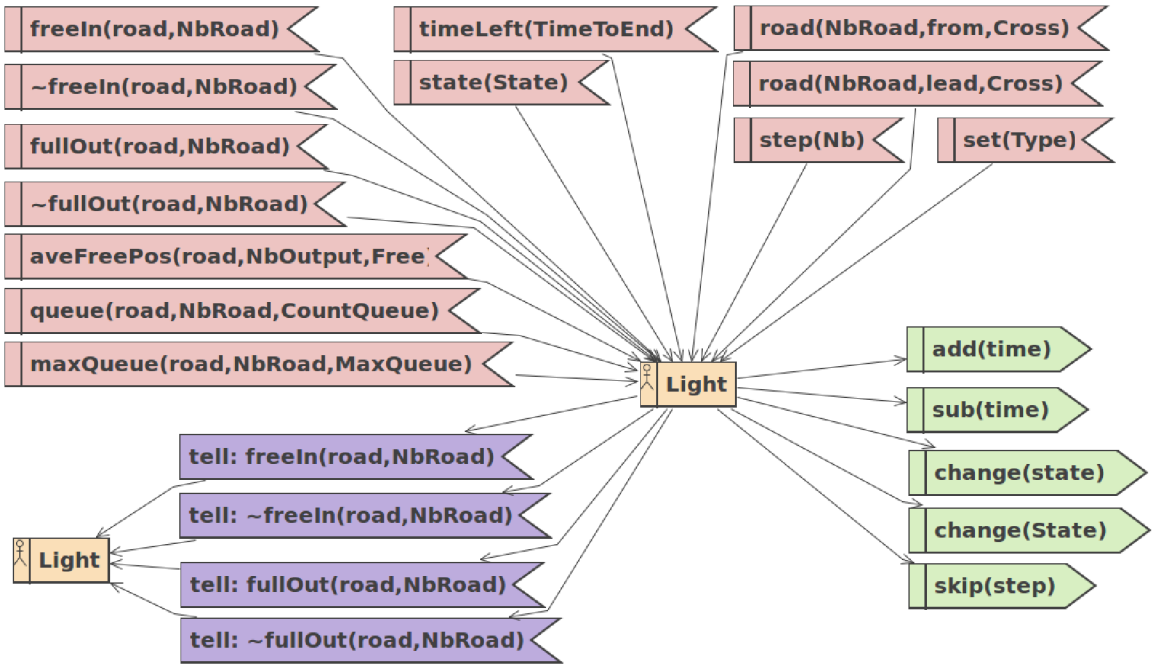
Každý agent se skládá ze senzorů, kterými jsou průběžně získávány informace o stavu na křižovatce. Tyto informace jsou předávány rozhodovací části. Ve znalostech jsou uchovávány plány a cíle agenta, které jsou svěřeny rozhodovací části. Rozhodovací část příjme příchozí data ze senzorů a snaží se najít takové plány, které by vedly k dosažení cílů agenta. To může mít za následek dvě reakce. Buď rozhodovací část odešle zprávu prostředí, nebo odešle zprávu jinému agentu pomocí komunikačního modulu. Tento cyklus se neustále opakuje. Tato architektura je zobrazena na obrázku 4.1.



Obrázek 4.1: Architektura agenta

K popisu systému využívám notaci Prometheus[14], která popíše veškeré aspekty systému. Model byl vytvořen za použiti pluginu Prometheus Design Tool (PDT)[1] pro Eclipse. Výsledný návrh systému lze vidět na obrázku č. 4.2.

[1] http://www.cs.rmit.edu.au/agents/pdt/index.shtml



Obrázek 4.2: Model systému dle notace Prometheus

Systém se skládá jen z jednoho agenta, který představuje světelnou signalizaci. Tento agent může dostávat informace typu:

- `step(`*`Nb`*`)`: informace, která spouští rozhodovací cyklus. *`Nb`* udává aktuální krok simulace.
- `set(`*`Type`*`)`: tento vjem určuje, jaky druh přístupu bude při rozhodování použit. Jsou možné 3 způsoby. První z nich je fixní režim (`fix`), tento způsob neprovádí žádné akce. Agentní režim (`ags`) využívá pravidla, která ovlivňují světelnou signalizaci podle určitých pravidel. Tyto pravidla však v sobě neobsahují komunikaci se sousedy. Multiagentní přístup (`mas`) využívá agentní pravidla a zároveň obsahuje i komunikaci se sousedy.
- `freeIn(road,` *`NbRoad`*`)`: tato informace slouží k detekování prázdné vstupní fronty, kde *`NbRoad`* označuje daný vstup. Tato hodnota má stejný význam i u `queue(road,` *`NbRoad`*`)`, `maxQueue(road,` *`NbRoad`*`,` *`MaxQueue`*`)`, `road(`*`NbRoad`*`,` `from,`*`Cross`*`)` a `road(`*`NbRoad`*`,` `lead,` *`Cross`*`)`. Pokud by nebyla fronta prázdná, objevuje se zde informace, která říká že fronta není prázdná (`~freeIn(road,` *`NbRoad`*`)`).
- `queue(road,` *`NbRoad`*`)`: aktuální počet vozidel v daném vstupu.
- `maxQueue(road,` *`NbRoad`*`,` *`MaxQueue`*`)`: informace o maximální délce fronty na určitém vstupu. Tato hodnota se využívá při zjišťování hustoty provozu.
- `fullOut(road,` *`NbOutput`*`)`: slouží k detekování plného výstupu. Jeho opakem je informace o neplném výstupu `~fullOut(road,` *`NbOutput`*`)`. Hodnota v *`NbOutput`*

určuje o který výstup se jedná. Tato hodnota má stejný význam i u `aveFreePos(road, NbOutput, Free)`.

- `aveFreePos(road, NbOutput,Free)`: průměrný počet volných pozic na výstupu.
- `state(State)`: informuje o aktuální stavu na světelné křižovatce.
- `timeLeft(TimeToEnd)`: informuje o zbývajícím počtu kroků do změny stavu.
- `road(NbRoad, from, Cross)`: definuje, která křižovatka vede do daného vstupu.
- `road(NbRoad, lead, Cross)`: definuje, do které křižovatky vede daný výstup.

Akce, kterými disponuje agent jsou:

- `add(time)`: přidá jeden krok k aktuálnímu stavu. Navýšení je omezené podmínkou, která byla definována v kapitole 3.1.2.
- `sub(time)`: odebere jeden krok od aktuálního stavu. Snížení je omezené podmínkou, která byla definována v kapitole 3.1.2.
- `change(state)`: přeskočení aktuálního stavu na stav s nenulovým trváním.
- `change(State)`: posun stavu na stav definován v `State`.
- `skip(step)`: nastaví agenta na přeskočení kroku.

4.1.1 Komunikace mezi křižovatkami

Způsob komunikace je navrhnut tak, že agent komunikuje pouze se svými nejbližšími sousedy. Toto okolí je definováno pomocí `road(NbRoad, from, Cross)` a `road(NbRoad, lead, Cross)`. Informace, které agenti sdílejí mezi sebou jsou o plných, respektive neplných výstupech. Nebo o prázdných, repektive neprázdných vstupech.

4.2 Pravidla pro řízení světelné signalizace

Tato část bude pojednávat o pravidlech, které byly použity při řízení světelné křižovatky. Pravidla byly inspirovány článkem [5]. Většina z nich však byla upravena a přidána nová. Jelikož je můžeme rozdělit do dvou typů, budou tyto pravidla probrány ve dvou kapitolách. V kapitole 4.2.1 budou rozebrána agentní pravidla, která pracují pouze s jednou křižovatkou. V kapitole 4.2.2 budou rozebrána multiagentní pravidla, která doplňují agentní pravidla o komunikaci se svými sousedy.

4.2.1 Agentní pravidla

V agentních pravidlech jde především o efektivní práci se stavy semaforu a případné prodloužení, či zkrácení času. Agent v tomto případě nezná svoje okolí a nemůže ho tudíž ovlivňovat. Počet těchto pravidel je 4. V následujících odstavcích budou postupně všechny probrány.

Pravidlo č. C.1 reaguje na vstup, kdy je hustota vozidel pod 10 % a aktuální stav semaforu aktivuje tento vstup tím, že agent pošle prostředí akci `change(state)`. Tato akce provede změnu aktuálního stavu na další s nenulovou délkou.

Pro reakci na plný, či na stav kdy průměrný počet volných míst klesne pod 3, je aplikováno pravidlo č. C.2. Reakce je stejná jako v předcházejícím pravidle.

C.3: pokud by hustota provozu v daném vstupu vzrostla nad 70 % a čas by se blížil konci, byl by tomuto stavu přidán jeden krok navíc. Tento čas by nesměl překročit své maximum. Doba, která je brána blízko konce, znamená čas pod 4 kroky (7,2 s) včetně.

Pravidlo č. C.4 je opak C.3. Pokud by hustota provozu klesla pod 25 %, byl by tomuto stavu odebrán jeden krok. Tento čas nesmí klesnout pod své stanovené minimum.

4.2.2 Multiagentní pravidla

V multiagentních pravidlech jde především o synchronizaci křižovatek a nalezení ideálního stavu na světelné signalizaci. Agent má v tomto případě přesně definované okolí (sousedy), se kterými se snaží spolupracovat. První 4 pravidla definují komunikaci mezi sousedy a hlavně, které informace se budou předávat.

Pravidlo č. C.5 informuje sousední křižovatku o prázdném vstupu. Tato informace bude sdělena za předpokladu, že konkrétní vstup je prázdný a nebo jestli jeho hustota je pod 10 %. Pokud nebudou tyto podmínky platit, automaticky je splněno pravidlo č. C.6, které informuje o neprázdném vstupu.

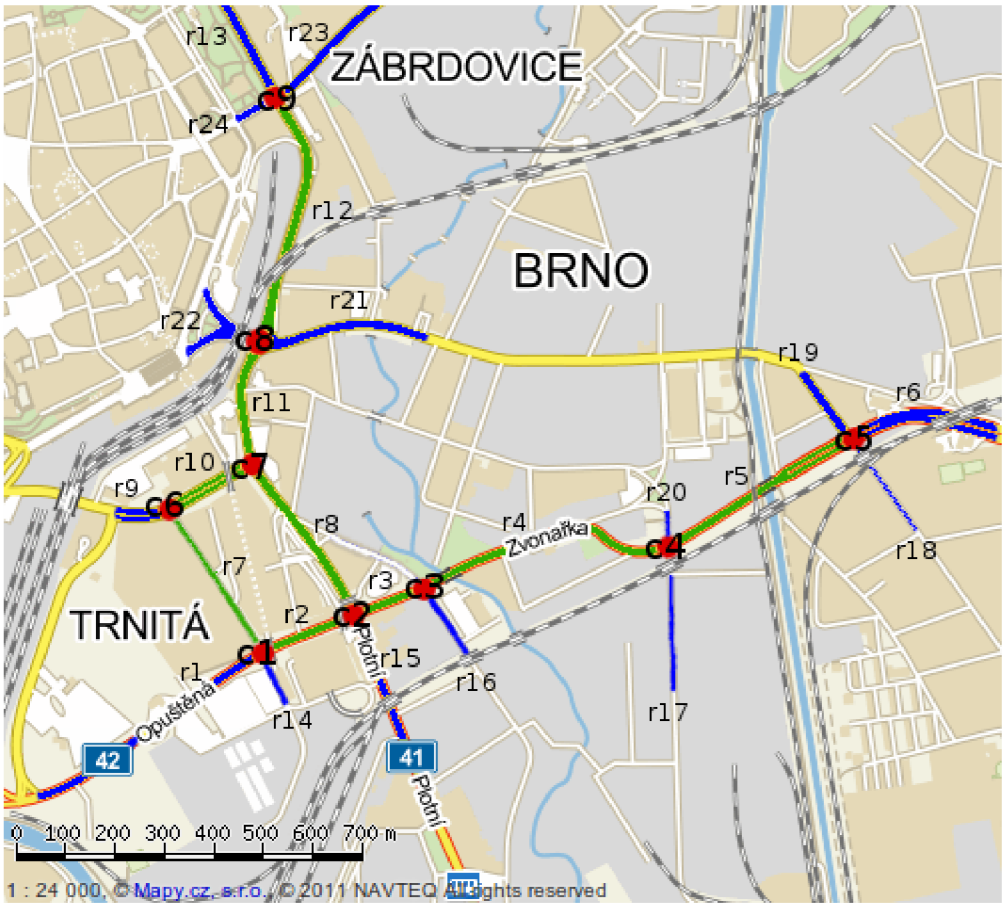
Pravidlo č. C.7 informuje sousední křižovatku o tom, že má plný výstup. Za předpokladu, že průměrný počet volných míst klesne pod 3 včetně. Pokud tyto podmínky nebudou splněny, bude aplikováno pravidlo č. C.8, které informuje o neplném výstupu.

Reakci na informace přijaté od okolních křižovatek zajišťuje pravidlo č. C.9. Jeho cílem je najít nejlepší možný stav, který eliminuje prázdné vstupy, či plné výstupy. Při prázdném vstupu u sousední křižovatky je snaha najít takový stav, který by do daného vstupu poslal vozidla. Pokud by však sám žádné vozidla na svém vstupu neměl, tento stav by nebyl brán jako ideální. Při plném výstupu sousední křižovatky by se našel stav světelné signalizace, který by tento výstup uvolnil, za předpokladu že vlastní výstup není také plný.

Kapitola 5

Experiment s reálným úsekem

Pro vyhodnocení námi navrhovaného řešení byly vytvořeny dva experimenty. Jeden zkoumá dynamiku provozu a její zlepšení (5.2) a druhý se zabývá průměrnou dobou zdržení a průměrnou délkou fronty na křižovatce (5.3). V obou těchto experimentech se pracovalo se simulačním modelem, který byl vytvořen na základě reálného úseku ve městě Brně. Daný úsek lze vidět na obrázku 5.1. Tento model pracuje s reálními hodnotami. Jejich získávání a vyhodnocení je popsáno v kapitole 5.1.



Obrázek 5.1: Mapa simulovaného úseku

5.1 Zdroje dat

K tomu, abychom mohli vyhodnotit přínos inteligentního řízení dopravy, bylo nutné pracovat s reálními daty. To zahrnuje doby trvání stavu na křižovatkách a počet aut vstupujících na dané silnice. Tyto informace byly získávány osobním pozorováním v době, kdy se projevují největší kongesce v dopravě ve městě Brně (tj. 15 – 17 hodina).

Příjezdy aut byly měřeny ve dvou všedních dnech vždy kolem 15. až 17. hodiny. Při měření byl brán ohled na měnící se intenzitu příjezdů aut do vozovky, proto se každý vstup měřil třikrát v jeden den vždy s minimálním odstupem jedné minuty. Následně byly tyto hodnoty zprůměrovány a použity pro nastavení generátorů vozidel. Nastavení jednotlivých generátorů lze vidět v tabulce 5.1. Hodnoty naměřených příchodů se nacházejí v příloze B.

	Počet aut za minutu														
Směr	r_1	r_6	r_9	r_{13}	r_{14}	r_{15}	r_{16}	r_{17}	r_{18}	r_{19}	r_{20}	r_{21}	r_{22}	r_{23}	r_{24}
L	5	4	–	2	1	13	7	3	1	8	7	–	–	4	1
N	70	25	12	30	1	6	–	2	2	5	3	6	5	2	1
R	2	2	10	7	1	2	3	1	5	2	2	22	2	8	2

Tabulka 5.1: Nastavení vstupních generátorů pro dané silnice.

Měření délky jednotlivých stavů bylo prováděno ve všední den, s časovým rozpětím 15. až 17. hodiny. Nebylo zde nutné provádět více měření, protože světelná signalizace v určitém časovém rozmezí je konstatní[1]. Naměřené hodnoty lze vidět v tabulce 5.2. Tyto hodnoty musely být převedeny na modelový čas. Toho jsme docílili pomocí vzorce:

$$t_{modelStavu} = \frac{t_{realStavu}}{t_{real}}$$

kde:

- $t_{modeStavu}$ délka stavu v modelovém času,
- $t_{realStavu}$ délka stavu v reálném čase,
- t_{real} je délka jednoho kroku v reálném čase (výpočtem této hodnoty se zabývá kapitola 2.1.3).

Následně byly tyto hodnoty použity pro nastavení jednotlivých křižovatek.

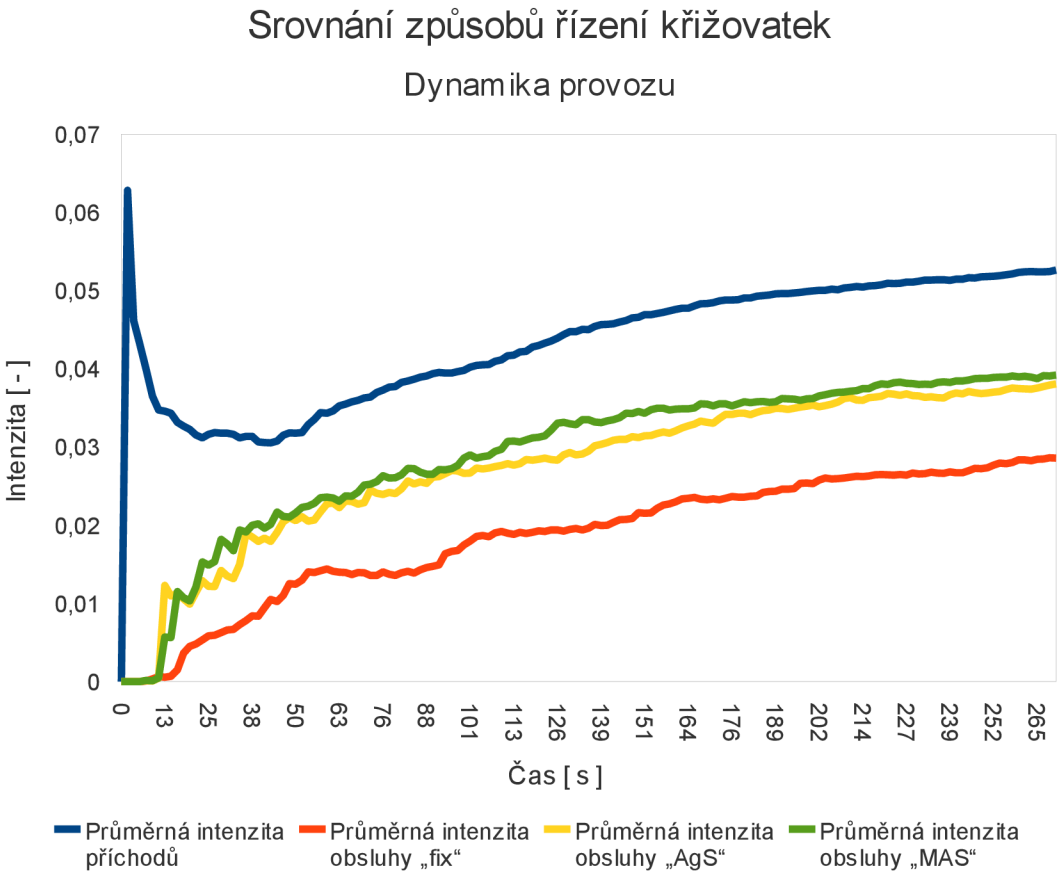
5.2 Dynamika provozu

Cílem tohoto experimentu je vyhodnotit zlepšení, či zhoršení dynamiky provozu na silnici. Tento experiment byl nastaven na délku 4,5 minuty. Na začátku simulace jsou vždy silnice prázdné, proto je tato doba ideální pro vyhodnocení dynamiky. V tomto čase se objevuje rapidní nárůst vstupní intenzity a tím se dá dobře zjistit, jak na tuto vstupní intenzitu dokáže agent reagovat a tím vylepšit obslužnou intenzitu.

Výsledky tohoto experimentu jsou zobrazeny v grafu 5.2. Jelikož tento experiment musel být spuštěn pro každý typ řízení zvlášť, byla vstupní intenzita zprůměrována, aby mohla být porovnávána s obslužnými intenzitami.

	Stav [s]					
Křižovatka	**1**	**2**	**3**	**4**	**5**	**6**
C_1	34	0	13	14	18	14
C_2	43	32	0	16	0	0
C_3	47	0	0	24	16	0
C_4	51	28	8	8	8	8
C_5	39	22	13	10	8	10
C_6	57	0	0	34	0	0
C_7	0	50	26	11	0	0
C_8	21	31	0	11	0	19
C_9	15	42	8	11	8	22

Tabulka 5.2: Naměřené hodnoty trvání stavů na jednotlivých křižovatkách.



Obrázek 5.2: Dynamika provozu

V grafu 5.2 lze pozorovat výrazné zlepšení dynamiky u AgS a MAS. AgS zaznamenal 48,79 % zlepšení oproti fixnímu nastavení a MAS dosáhl 57,82 % zlepšení. Rozdíl mezi AgS a MAS není již tak výrazný. Tento rozdíl je 9,02 %.

5.3 Průměrné zdržení a průměrná délka fronty na křižovatce

Cílem tohoto experimentu je vyhodnotit zlepšení, či zhoršení průměrného zdržení na křižovatce a průměrné délky fronty. K zobrazení výsledků byly využity dva grafy. Jeden, který zobrazoval výše zmíněné hodnoty a druhý, který ukazoval průměrnou, minimální a maximální dobu průjezdu dopravním úsekem. Délka modelového času byla 1 hodina.

Výsledky tohoto experimentu jsou ukázány v grafu č. 5.3 a č. 5.4. Každý typ řízení byl spuštěn 3krát a tyto hodnoty byly zprůměrovány a zobrazeny v grafech. Tento postup byl zvolen z toho důvodu, aby se statisticky snížili odchylky vlivem aplikování pravidel.

Při vyhodnocení grafu 5.3 lze vidět výrazné zlepšení průměrného zdržení, jak u AgS tak i u MAS oproti fixnímu nastavení. AgS dosahuje 37,90 % zlepšení a MAS až 54,11 % oproti fixnímu nastavení. Při porovnávání MAS a AgS je tento rozdíl celkem znatelný. Zlepšení MAS oproti AgS je 26,10 %. U průměrné délky front je toto snížení zhruba o 3 auta na každý vstup. To znamená, že fronty u MAS klesly o 13,72 % oproti fixnímu nastavení. Rozdíl mezi MAS a AgS není již tak znatelný. U grafu 5.4 lze pozorovat téměř stejné chování jako u předchozího grafu. Výjimkou jsou hodnoty MAS, které zaznamenaly nárůst oproti AgS. Z toho vyplívá, že snížením průměrného času zdržení a fronty v křižovatce dojde k zhoršení průměrného, minimálního a maximálního času potřebného k projetí dopravního úseku.

Obrázek 5.3: Srovnání způsobů řízení křižovatek

[1] Viz. `http://preference.prazsketramvaje.cz/showpage.php?name=ssz`

Obrázek 5.4: Statistiky průjezdů

Kapitola 6

Závěr

V rámci této práce byl vytvořen simulátor dopravních situací v jazyce Java a multiagentní systém naprogramovaný pomocí rozšířeného jazyka AgentSpeak s využitím interpretu Jason. Tento způsob inteligentního řízení světelné signalizace přináší značné snížení doby čekání ve frontě oproti světelné signalizaci řízené agentním systémem či řízené pomocí fixního času stavů. Dále bylo dosaženo zkrácení průměrné délky fronty na křižovatkách a také zlepšení dynamiky provozu.

Všechny výsledky jsou porovnávány mezi MAS a fixním nastavením. Dynamika provozu byla zvýšena o 57,82 %. Snížení průměrné délky fronty zaznamenalo 13,72 % pokles a průměrné zdržení na křižovatce dosáhlo snížení o 54,11 %.

Na práci je možné navázat aplikováním MAS na reálné křižovatky a provést patřičné experimenty k potvrzení výsledků získaných simulací. Jako detektory bych navrhoval použít kombinaci videodetektoru a mikrovlného detektoru. Pokud by se výsledky potvrdily, byl by tento přístup rozšířen o preferenci zvláštních vozidel a vozidel MHD. Další možností, jak by bylo možné na tuto práci navázat, je výzkum tzv. dynamických navigací, využívající informace získávané agenty křižovatek. Díky těmto informacím by bylo možné vyhnout se dopravním kongescím a najít tak nejrychlejší trasu k cíli. Dynamickými navigacemi se některé firmy již zabývají, a potřebné technologie se již podařilo vyvinout. Jediné, co tak dělí dynamické navigace od zavedení na trh je, že nejsou k dispozici stabilní a aktuální dopravní data[2].

Literatura

[1] BORDINI, R. H.; HÜBNER, J. F.; WOOLDRIDGE, M.: *Programming Multi-Agent Systems in AgentSpeak using Jason*. Wiley, 2007, ISBN 978-0-470-02900-8.

[2] BRUNA, J.: *Studie provozuschopnosti silniční sítě s využitím telematických přístupů*. Bakalářská práce, Vysoké učení technické v Brně, 2008.

[3] CHANG, E. C.-P.: Estimate Traffic with Combined Neural Network Approach. In *Fusion 1999*, California (USA): International Society of Information Fusion, 1999, [online]. [cit. 2012-05-01].
URL http://isif.org/fusion/proceedings/fusion99CD/C-004.pdf

[4] CHAO, K.-H.; LEE, R.-H.; WANG, M.-H.: An Intelligent Traffic Light Control Based on Extension Neural Network. 2008, [online]. [cit. 2012-05-01].
URL http://www.springerlink.com/content/d34h253q44x71522/

[5] HIRANKITTI, V.; KROHKAEW, J.; HOGGER, C.: A Multi-Agent Approach for Intelligent Traffic-Light Control. In *World Congress on Engineering*, ročník 1, London, 2007.

[6] HOTMAR, P.; PALATOVÁ, M.: Fuzzy řízení dopravy na světelné křižovatce. *Konference MATLAB*, 2002, [online]. [cit. 2012-05-01].
URL http://dsp.vscht.cz/konference_matlab/matlab02/hotmar_palatova.pdf

[7] KAREEM, E. I. A.: An Intelligent Traffic Light Monitor System using an Adaptive Associative Memory. *Information Processing and Management*, ročník 2, č. 2, 2011.

[8] KAY, A.: Co jsou to umělé neuronové sítě? *Science World*, 2001, [online]. [cit. 2012-05-01].
URL http://scienceworld.cz/technologie/co-jsou-to-umele-neuronove-site-4077

[9] KELL, J. H.; FULLERTON, I. J.; MILLS, M. K.: Traffic Detector Handbook. Technická zpráva, U.S Department of Transportation, 1990.
URL www.fhwa.dot.gov/publications/research/safety/ip90002/ip90002.pdf

[10] MAŘÍK, V.; ŠTEPÁNKOVÁ, O.; LAŽANSKÝ, J.: *Umělá inteligence*. 1. vyd., Praha: Academia, 2001, ISBN 80-200-0472-6, str. 328.

[11] NAGY, I.; KRATOCHVÍLOVÁ, J.: Model dopravní mikrooblasti. *Automatizace*, ročník 47, č. 12, 2004: s. 752–758, [online]. [cit. 2012-05-01].
URL
www.automatizace.cz/download.php?d=QXRtX0FydGljbGUscGRmX2FydCw0Mzk=

[12] *NetLogo*. Verze 4.1.3 [online]. [cit. 2012-05-01].
URL
http://ccl.northwestern.edu/netlogo/docs/NetLogo%20User%20Manual.pdf

[13] NWIGBO, S.; CHUKS, A. O.: Expert system: A catalyst in educational development in Nigeria. In *International Technology, Education and Environment Conference*, African Society for Scientific Research, Human Resource Management Academic Research Society, s. 566–571.

[14] PADGHAM, L.; WINIKOFF, M.: *Developing Intelligent Agent Systems*. Australia: Wiley, 2004, ISBN 0-470-86120-7 (HB).

[15] ROOZEMOND, D. A.; ROGIER, J. L. H.: Agent controlled traffic lights. In *ESIT2000*, Aachen (Germany), 2000.

[16] RYDVAL, S.: Základy fuzzy logiky. 2005, [online]. [cit. 2012-05-01].
URL http://www.rydval.cz/phprs/view.php?cisloclanku=2005061701

[17] SCHADSCHNEIDER, A.: Traffic flow modelling. 2000, [online]. [cit. 2012-05-01].
URL http://www.thp.uni-koeln.de/~as/Mypage/traffic.html

[18] TAN, K. K.; KHALID, M.; YUSOF, R.: Intelligent traffic lights control by fuzzy logic. *Computer Science*, ročník 9, č. 2, 1996: s. 29–35.

[19] WIERING, M.; van VEENEN, J.; VREEKEN, J.; aj.: Intelligent Traffic Light Control. Technická zpráva, Information and computing sciences, Netherlands, 2004.

[20] ZBOŘIL, F.: *Plánování a komunikace v multiagentních systémech*. Dizertační práce, Vysoké učení technické, Brno, 2004.

[21] ZBOŘIL, F.: II. Agentní systémy, základní architektury, jejich modely a realizace. *Podklady k přednáškám kurzu AGS*, 2006, [prezentace].

Příloha A

Obsah CD

- env/ – zdrojové kódy v jazyce Java.
- tex/ – zdrojový tvar písemné zprávy.
- CA.mas2j – projekt pro Jason.
- logging.properties – nastavení kontrolních výpisů.
- light.asl – jádro rozhodovací logiky napsané v jazyce agentspeak.
- manual.pdf – manuál pro práci s dopravním simulátorem.
- readme.txt – návod na instalaci všech potřebných komponent.

Příloha B

Naměřené hodnoty

	Vstup pro r_1 ul. Znonařka			**Vstup pro r_{14} ul. Opuštěná**			
Měření	**L**	**N**	**R**	**L**	**N**	**R**	**Čas**
1	10	66	1	0	0	3	15:39
2	8	62	2	1	2	1	15:42
3	3	69	1	1	0	1	15:46
4	5	73	3	1	1	0	16:20
5	1	72	2	0	1	0	16:22
6	2	78	2	1	0	3	16:24
Průměr:	**4,83**	**70,00**	**1,83**	**0,67**	**0,67**	**1,33**	–

Tabulka B.1: Naměřené hodnoty vstupů pro silnice r_1 a r_{14}

	Vstup pro r_{15} ul. Plotní			
Měření	**L**	**N**	**R**	**Čas**
1	13	5	1	15:50
2	7	1	4	15:55
3	14	6	2	15:58
4	17	10	0	16:13
5	19	11	2	16:15
6	10	4	3	16:17
Průměr:	**13,33**	**6,17**	**2,00**	–

Tabulka B.2: Naměřené hodnoty vstupu pro silnici r_{16}

	Vstup pro r_{16} ul. Dornych			
Měření	**L**	**N**	**R**	**Čas**
1	9	–	3	16:02
2	4	–	2	16:04
3	5	–	3	16:06
4	10	–	1	16:04
5	7	–	7	16:06
6	9	–	1	16:08
Průměr:	**7,33**	**–**	**2,83**	–

Tabulka B.3: Naměřené hodnoty vstupu pro silnici r_{16}

	Vstup pro r_{17} ul. Hladíkova			**Vstup pro r_{20} ul. Masná**			
Měření	**L**	**N**	**R**	**L**	**N**	**R**	**Čas**
1	3	2	1	8	3	1	16:16
2	4	0	2	7	2	2	16:19
3	5	4	0	6	4	0	16:25
4	3	1	1	9	5	2	15:52
5	2	2	0	5	4	3	15:55
6	3	3	1	3	0	2	15:58
Průměr:	**3,33**	**2,00**	**0,83**	**6,33**	**3,00**	**1,67**	–

Tabulka B.4: Naměřené hodnoty vstupů pro silnice r_{17} a r_{20}

	Vstup pro r_{18} ul. Charbulova			**Vstup pro r_{19} ul. Tržní**			**Vstup pro r_{6} ul. Olomoucká**			
Měření	**L**	**N**	**R**	**L**	**N**	**R**	**L**	**N**	**R**	**Čas**
1	0	0	6	8	2	2	4	34	2	16:33
2	0	3	7	6	5	1	7	29	2	16:38
3	2	1	8	8	7	2	4	28	2	16:43
4	1	0	4	9	4	2	4	24	2	15:35
5	2	3	5	8	6	2	2	19	1	15:40
6	0	3	2	9	5	2	3	18	3	15:45
Průměr:	**0,83**	**1,67**	**5,33**	**8,00**	**4,83**	**1,83**	**4,00**	**25,33**	**2,00**	–

Tabulka B.5: Naměřené hodnoty vstupů pro silnice r_{18}, r_{29} a r_{6}

	Vstup pro r_9 ul. Úzká			
Měření	**L**	**N**	**R**	**Čas**
1	–	15	12	15:24
2	–	9	9	15:26
3	–	18	12	15:28
4	–	12	14	16:31
5	–	10	8	16:33
6	–	9	7	16:35
Průměr:	–	**12,17**	**10,33**	–

Tabulka B.6: Naměřené hodnoty vstupu pro silnici r_9

	Vstup pro r_{21} ul. Křenova			Vstup pro r_{22} Hl. nádraží			
Měření	**L**	**N**	**R**	**L**	**N**	**R**	**Čas**
1	–	8	26	–	6	2	15:04
2	–	7	23	–	4	1	15:06
3	–	8	28	–	9	3	15:09
4	–	6	15	–	4	2	16:40
5	–	3	17	–	4	2	16:44
6	–	5	21	–	2	1	16:46
Průměr:	–	**6,17**	**21,67**	–	**4,83**	**1,83**	–

Tabulka B.7: Naměřené hodnoty vstupů pro silnice r_{21} a r_{22}

	Vstup pro r_{24} Malinovské nám.			Vstup pro r_{13} ul. Koliště			Vstup pro r_{23} ul. Cejl			
Měření	**L**	**N**	**R**	**L**	**N**	**R**	**L**	**N**	**R**	**Čas**
1	0	0	2	2	39	9	4	2	7	14:17
2	1	1	3	4	40	7	5	2	10	14:30
3	2	0	1	0	32	8	4	1	8	14:42
4	0	1	3	3	18	3	3	3	7	16:53
5	1	3	2	2	23	5	2	1	5	16:57
6	1	1	1	1	30	7	6	2	10	17:00
Průměr:	**0,83**	**1,00**	**2,00**	**2,00**	**30,33**	**6,50**	**4,00**	**1,83**	**7,83**	–

Tabulka B.8: Naměřené hodnoty vstupů pro silnice r_{24}, r_{13} a r_{23}

Příloha C

Agentní a multiagentní pravidla

Pravidlo C.1.

```
+!checkRules
   :  (state(State) & (freeIn(road, NbRoad)[source(percept)] | (queue(road,
      NbRoad, CountQueue) & maxQueue(road, NbRoad, MaxQueue) &
      too_low_queue(CountQueue, MaxQueue))) & compare_state_road(State,
      NbRoad))
   <- change(state).
```

Pravidlo C.2.

```
+!checkRules
   :  state(State) & (fullOut(road, NbOutput)[source(percept)] |
      (aveFreePos(road, NbOutput, Free) & near_to_full(Free))) &
      compare_state_road(State, NbOutput)
   <- change(state).
```

Pravidlo C.3.

```
+!checkRules
   :  state(State) & timeLeft(TimeToEnd) & end_is_near(TimeToEnd) &
      queue(road, NbRoad, CountQueue) & maxQueue(road, NbRoad, MaxQueue)
      & too_high_queue(CountQueue, MaxQueue) & compare_state_road(State,
      NbRoad)
   <- add(time).
```

Pravidlo C.4.

```
+!checkRules
   :  state(State) & queue(road, NbRoad, CountQueue) & maxQueue(road,
      NbRoad, MaxQueue) & low_queue(CountQueue, MaxQueue) &
      not(too_low_queue(CountQueue, MaxQueue)) & compare_state_road(State,
      NbRoad)
   <- sub(time).
```

Pravidlo C.5.

```
+!shareInStat(NbRoad)
   :  (freeIn(road, NbRoad)[source(percept)] | (queue(road,
      NbRoad, CountQueue) & maxQueue(road, NbRoad, MaxQueue) &
      too_low_queue(CountQueue, MaxQueue))) & road(NbRoad, from, Cross)
   <- .send(Cross, tell, freeIn(road, NbRoad)).
```

Pravidlo C.6.

```
+!shareInStat(NbRoad)
   :  ~freeIn(road, NbRoad)[source(percept)] & road(NbRoad, from, Cross)
   <- .send(Cross, tell, ~freeIn(road, NbRoad)).
```

Pravidlo C.7.

```
+!shareOutStat(NbRoad)
   :  (fullOut(road, NbRoad)[source(percept)] | (aveFreePos(road, NbRoad,
      Free) & near_to_full(Free))) & road(NbRoad, lead, Cross)
   <- .send(Cross, tell, fullOut(road, NbRoad)).
```

Pravidlo C.8.

```
+!shareOutStat(NbRoad)
   :  ~fullOut(road, NbRoad)[source(percept)] & road(NbRoad, lead, Cross)
   <- .send(Cross, tell, ~fullOut(road, NbRoad)).
```

Pravidlo C.9.

```
+!checkRules
   :  findBestState(State) & state(ActState) & not(State == ActState) &
      timeLeft(TimeToEnd) & next_is_end(TimeToEnd)
   <- change(State).
```