
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ**
**DEPARTMENT OF COMPUTER SYSTEMS**


# SAMOČINNÉ ŘÍZENÍ VOZIDEL

**AUTONOMOUS CONTROL OF CARS**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** **VOJTĚCH ČURDA**
**AUTHOR**

**VEDOUCÍ PRÁCE** **JOSEF STRNADEL, Ing., Ph.D.**
**SUPERVISOR**



**BRNO 2020**

**Vysoké učení technické v Brně**
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

# Zadání bakalářské práce


Student: **Čurda Vojtěch**
Program: Informační technologie
Název: **Samočinné řízení vozidel**
**Autonomous Control of Cars**
Kategorie: Modelování a simulace
Zadání:

1. Zdokumentujte požadavky, problémy, pojmy a principy související s činností samočinně řízeného vozidla a jeho vývojem; proveďte detailní rešerši v této oblasti. Navrhněte vhodnou abstrakci samočinně řízeného vozidla, jeho okolí a aspektů klíčových z hlediska jeho izolovaného i kooperativního řízení a chování.
2. Proveďte detailní rešerši v oblasti prostředků výpočetního modelování systémů a analýzy jejich vlastností a zvolte prostředky vhodné k řešení zadané práce.
3. Pomocí zvolených prostředků vytvořte výpočetní model řízení a chování samočinně řízeného vozidla a výpočetní model jeho okolí s cílem analyzovat vliv způsobu řízení vozidla na chování vozidla v daných podmínkách.
4. Vlastnosti modelu ověřte v několika vhodně zvolených situacích (např. způsoby řízení, typy okolí, podmínky).
5. Diskutujte a zhodnoťte možnosti vytvořeného modelu z hlediska analýzy dopadu vlivů zmíněných v bodu 3 a navrhněte možné směry pokračování v projektu.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1, 2 zadání, představení variant řešení, vytvoření základního modelu samočinně řízeného vozidla.

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

Vedoucí práce: **Strnadel Josef, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 25. října 2019

# Abstrakt

Cílem této bakalářské práce je vytvoření modelu samočinně řízeného vozidla a jeho okolí. Práce nejprve nastiňuje téma samočinně řízených vozidel a problematiku modelování systémů. V práci je poté předveden návrh zmíněného modelu a popsána jeho realizace. K realizaci modelu byl zvolen nástroj UPPAAL Stratego, který slouží k modelování, validaci a verifikaci systémů reálného času. S pomocí tohoto nástroje je model implementován v podobě sítí časovaných automatů. Výsledkem je model samočinně řízeného vozidla, které se pohybuje na dálnici s libovolným počtem jízdních pruhů a dokáže reagovat jak na pohyb ostatních vozidel provozu, tak na dopravní značení určující maximální povolenou rychlost. Statistickou analýzou pomocí dotazů verifikačního jazyka zmíněného nástroje jsou poté ověřeny vlastnosti vytvořeného modelu. Model je testován jak nad konkrétními scénáři, tak nad náhodnými, kde je samočinně řízené vozidlo zasazeno do běžného provozu.



# Abstract

The purpose of this bachelor's thesis is to create a model of a self-driving vehicle, along with its surroundings. At first, the thesis outlines the vehicle automation and the system modeling problematics. The thesis then shows the design of said model, as well as its implementation. The model was implemented in UPPAAL Stratego – a tool that is used for modeling, validation and verification of real-time systems, in which the model is implemented as networks of timed automata. The result is a model of a self-driving vehicle, that can drive itself on a motorway with an arbitrary amount of traffic lanes. The self-driving vehicle reacts appropriately to the movement of other vehicles on the motorway and can change its speed according to speed limiting traffic signs. The behaviour of the model is then validated with the help of statistical analysis, which is done using the verification language embedded in the mentioned tool. The model is tested in specific scenarios, as well as random scenarios, where the self-driving vehicle is set into a regular traffic.



# Klíčová slova

samočinně řízené vozidlo, model, modelování systémů, simulace, UPPAAL Stratego, časované automaty, statistická analýza

# Keywords

self-driving vehicle, model, system modeling, simulation, UPPAAL Stratego, timed automata, statistical analysis

# Citace

ČURDA, Vojtěch. *Samočinné řízení vozidel.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Josef Strnadel, Ing., Ph.D.

# Samočinné řízení vozidel

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Josefa Strnadela, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Vojtěch Čurda
28. května 2020


## Poděkování

Tímto bych rád poděkoval svému vedoucímu práce Ing. Josefu Strnadelovi, Ph.D. za ochotu a cenné rady, které mi poskytoval v celém průběhu tvorby této bakalářské práce.

# Obsah

# Kapitola 1

# Úvod

Lze se objektivně shodnout na tom, že lidé nejsou moc dobří řidiči – mají tendenci nerespektovat pravidla silničního provozu, usínat za volantem, věnovat se jiným činnostem při řízení vozidla, nebo řídit pod vlivem alkoholu. Za velkou část dopravních nehod může chyba člověka, podle některých studií až za 94 % [34]. V roce 2018 proběhlo v USA téměř sedm milionů dopravních nehod, které zapříčinily 36560 smrtí [8]. Je to tedy právě bezpečnost, která je jednou z hlavních motivací pro vývoj plně autonomních vozidel.

Plně autonomní vozidla byla dlouhá léta pouze snem futuristů. Naděje po plně autonomních vozidlech byla ale dlouho nedosažitelná. Nedávné technologické průlomy v oblasti elektroniky a senzorových technologií umožnily vznik samočinně řízených vozidel, což je první krok k dosažení plné autonomie.

Pojmem „samočinně řízená vozidla“ se myslí vozidla, která se dokáží sama řídit v určitých situacích, přičemž pojem „plně autonomní vozidla“, označuje vozidla, která se dokáží sama řídit ve všech situacích a nepotřebují tak řidiče. Samočinně řízená vozidla ve většině případů sbírají data o jejich pohybu a „zažitých“ situacích, která posílají na centrálu. Na základě těchto dat se pomocí strojového učení vylepšuje algoritmus zajišťující umělou inteligenci vozidla. V roce 2020 zatím neexistuje společnost, která by disponovala plně autonomním vozidlem. K dosažení tohoto cíle musí ještě samočinně řízená vozidla projít dlouhým stádiem vývoje. Některé společnosti předpokládají příchod prvních plně autonomních vozidel již koncem roku 2020 [29], externí odborníci se spíše ale přiklání k roku 2030 [28].

Tato bakalářská práce se zaměřuje právě na samočinně řízená vozidla a možnosti modelování systémů s cílem vytvoření modelu samočinně řízeného vozidla a jeho okolí.

Kapitola 2 obsahuje rešerši problému samočinně řízených vozidel a modelování systémů. V kapitole 3 lze najít zvolené prostředky pro tvorbu modelu a jeho návrh. Popis vytvořeného modelu obsahuje kapitola 4. Nakonec je vytvořený model podroben testům za účelem ověření jeho vlastností v kapitole 5.

# Kapitola 2

# Rešerše k zadanému problému

Tato kapitola obsahuje rešerši tématu samočinně řízených vozidel (sekce 2.1) a modelování systémů (sekce 2.2).

## 2.1 Samočinně řízená vozidla

Tato sekce přibližuje pojem samočinně řízeného vozidla, popisuje technologie využívané těmito vozidly, shrnuje aktuální stav vývoje, a nakonec obsahuje předpoklad vlivu samočinně řízených vozidel na každodenní život.

### 2.1.1 Klasifikace vozidel

Pojem „samočinně řízené vozidlo“ se často chybně zaměňuje za pojem „autonomní vozidlo“. Společnost SAE International[1] definuje „samočinně řízené vozidlo“ jako vozidlo, které se dokáže samo řídit v určitých situacích, přičemž uvnitř vozidla musí být přítomný řidič, který je připraven převzít kontrolu. Oproti tomu „plně autonomní vozidla“ definuje jako vozidla, která jsou ve všech situacích nezávislá na řidiči.

K dalšímu rozlišení vozidel mohou posloužit úrovně automatizace, které společnost SAE také stanovuje. Standard J3016 [7] začleňuje vozidla do šesti úrovní v rozmezí od „bez automatizace“ až po „plná automatizace“ na základě nutnosti zásahu řidiče do řízení a dovednostech řídícího systému vozidla.

- **0. úroveň („bez automatizace“)** – Přestože vozidlo může obsahovat varovné systémy, nebo systémy zasahující do řízení, vozidlo je plně řízeno řidičem.
- **1. úroveň („asistence řidiče“)** – Vozidlo může být částečně ovládáno základními asistenčními systémy, tím je myšleno dočasné převzetí kontroly nad zatáčením, nebo akcelerací/decelerací vozidla, ale ne obojí zároveň. Řidič musí mít stále kontrolu nad ostatními aspekty řízení.
- **2. úroveň („částečná automatizace“)** – Vozidlo může být částečně ovládáno několika asistenčními systémy zároveň, které mohou současně ovládat zatáčení a akceleraci/deceleraci vozidla. Od řidiče je stále vyžadováno sledování provozu a kontrola činnosti řídícího systému.

[1] SAE International - https://www.sae.org/

- **3. úroveň („podmíněná automatizace“)** – Systém v určitých podmínkách přebírá plnou kontrolu nad ovládáním vozidla včetně plného monitorování okolního prostředí. Řidič se nemusí věnovat řízení, ale je od něj očekáváno co nejdříve reagovat, pokud systém požádá o převzetí kontroly.
- **4. úroveň („vysoká automatizace“)** – Systém v určitých podmínkách přebírá plnou kontrolu nad ovládáním vozidla včetně plného monitorování okolního prostředí a zachovává plnou kontrolu nad řízením, přestože řidič nereaguje na požadavek o převzetí kontroly.
- **5. úroveň („plná automatizace“)** – Systém má plnou kontrolu nad vozidlem ve všech podmínkách. Toto vozidlo lze označit za „plně autonomní“.

### 2.1.2 Senzory a pomocné systémy

Aby mělo samočinně řízené vozidlo přehled o svém okolí a umístění v prostoru, využívá několika senzorů a pomocných systémů. Radar slouží pro určení vzdálenosti a rychlosti objektů. Lidar dokáže také zjistit vzdálenost a dále tvar objektu. Ultrazvukové senzory jsou pro detekci objektů v blízkosti vozidla a jsou tak využívány parkovacími asistenty. Kamery jsou v samočinně řízených vozidlech nutné ke klasifikaci objektů. Dále se v samočinně řízených vozidlech využívají systémy pro určení polohy a pohybu vozidla, což zajišťuje satelitní navigace, resp. inerciální měřící jednotka.

Výstup z těchto senzorů a systémů zpracovává řídící jednotka vozidla a tvoří z nich za pomocí procesu zvaného „senzorová fúze“ digitální 3D reprezentaci okolí, na základě čeho poté ovládá vozidlo pomocí akčních členů, které jsou napojené na mechanické části vozu. Mechanickými částmi vozu se rozumí například brzdná soustava, motor, řízení, popř. převodovka.

Následující sekce popisují zmíněné senzory a systémy, které samočinně řízená vozidla mohou obsahovat.

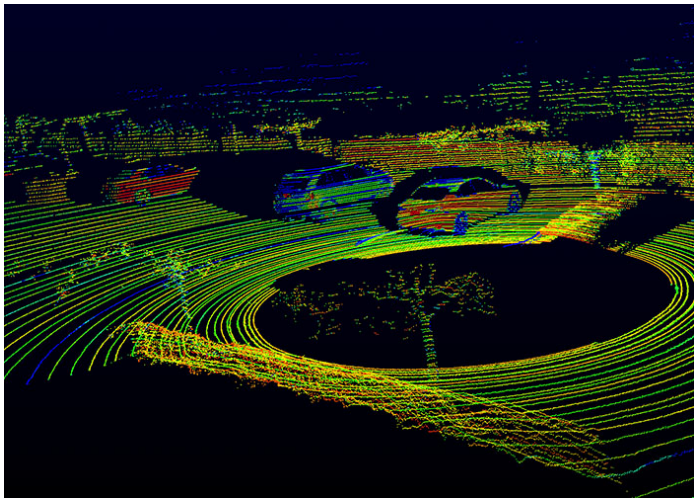
#### Lidar

Lidar („Light detection and ranging“) je technologie určená k detekci objektů a jejich vzdáleností. Je to systém, který vysílá a přijímá miliony pulzujících světelných paprsků každou vteřinou. Paprsky jsou odráženy od povrchu a zachycovány senzorem. Na základě časových rozdílů mezi vysláním a zachycením a vlnových délkách tento systém tvoří digitální 3D reprezentaci bodů, které reprezentují bod odrazu paprsku, tedy nějaký povrch [36], viz obrázek 2.1.

Paprsky vysílané systémem jsou často o vlnové délce 905, nebo 1550 nm, kde hlavní rozdíly mezi zmíněnými vlnovými délkami jsou bezpečnost, absorpce vodou a cena technologie [22]. Díky takto nízkým vlnovým délkám dokáže Lidar detekovat objekty ve vysokém rozlišení, na maximální vzdálenosti 100-300 metrů s 2-3 centimetrovou přesností [38].

Laserová technologie je ale velmi drahá. Velmi oblíbený Lidar Velodyne HDL-64E[3], používaný např. Stanfordskou univerzitou pro jejich projekty samočinně řízených vozidel [27] vyjde na 75000 dolarů [10]. Je také velice náchylná na počasí, kdy za silného deště se může maximální dosah Lidaru snížit z řádů stovek metrů na několik desítek [17].

[2]Převzato z: https://velodynelidar.com/hdl-64e.html
[3]Velodyne HDL-64E –https://velodynelidar.com/hdl-64e.html



Obrázek 2.1: Digitální 3D reprezentace okolního prostředí Lidaru Velodyne HDL-64E[2]

Zmíněné nevýhody zapříčinily, proč Tesla, jedna z hlavních společností v oblasti samočinně řízených vozidel, Lidar vůbec ve svých vozidlech nepoužívá – podle CEO Tesly, Elona Muska je to „nadbytečná“ technologie [18].

### Radar

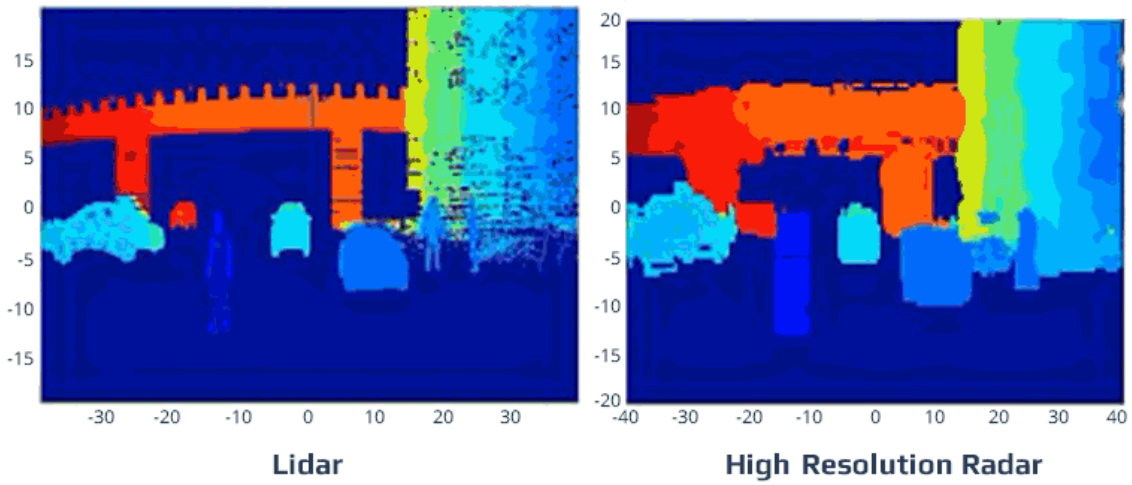
Radarová technologie má v samotném automobilovém průmyslu už dlouhou dobu uplatnění. Radary napomáhají k určení vzdáleností, relativní rychlosti a směru pohybu snímaných objektů [37]. Díky nim mohou spolehlivě pracovat systémy detekce nebezpečí, především systém automatického nouzového brzdění a adaptivní tempomat. Radarový systém spočívá na principu vysílání elektromagnetických vln. Tyto vlny objekty odráží a na základě odrazů je možné určit zmíněné vlastnosti objektu. Zatím v automobilovém průmyslu převládaly radary s vlnovou délkou 24 GHz. V poslední době se ale už často používají radary s vlnovou délkou 77 GHz, které jsou jednak přesnější, ale také mají menší rozměry, tudíž je pro automobilky jednodušší je zakomponovat do designu automobilu [21]. Takové radary se často označují zkratkou MMW (*MiliMeterWave*).

Jedno vozidlo často obsahuje více radarových systémů, s rozdílnými efektivními maximálními vzdálenostmi:

- Blízká vzdálenost (<5 m) – Detekuje překážky na vzdálenost několika metrů od vozidla. Jsou stavěny pro nízko rychlostní scénáře (Parkovací asistence)
- Krátká vzdálenost (0.5 - 30 m) – Využíváno systémem detekce slepého úhlu.
- Střední vzdálenost (80 - 160 m) – Napomáhá systémům pro vyvarování se kolizi.
- Dlouhá vzdálenost (100 - 250 m) – Stavěný pro vyšší rychlosti, využíván adaptivním tempomatem.

Jelikož radarové systémy využívají elektromagnetické vlny, dokáží přesně pracovat i v nepříznivých podmínkách.

[4]Převzato a upraveno z: https://medium.com/@intellias/the-ultimate-sensor-battle-lidar-vs-radar-2ee0fb9de5da

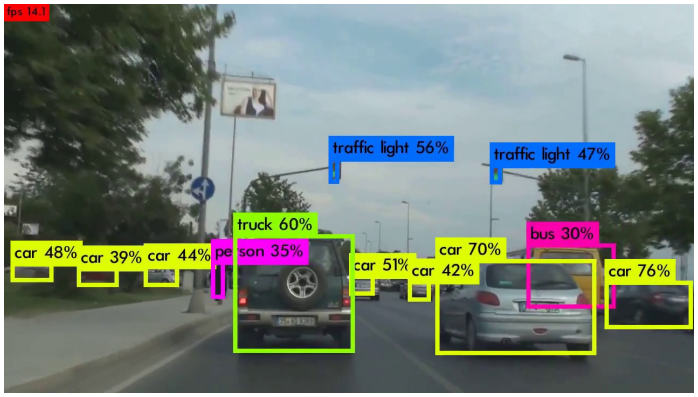



Obrázek 2.2: Porovnání radarového vidění a Lidaru[4]

Jak je vidno z obrázku 2.2, radar nedisponuje stejně vysokým rozlišením jako Lidar, proto vozidlo čistě podle vstupu z radaru těžko rozezná, zdali je snímaný objekt auto, nebo jen kontejner na odpadky.

### Kamery

Kamery se v automobilech používají už nějakou dobu, kdy mohou pomáhat řidiči při parkování, eliminování slepých úhlů, nebo třeba k lepšímu pohledu do křižovatek. Kamery v samočinně řízených vozidlech ale především slouží k digitálnímu mapování okolního prostředí. Představují totiž nejlepší a nejpřesnější způsob vytvoření vizuální 360° reprezentace okolního světa. Tomu se dá docílit buď připevněním kamer na každou stranu vozidla a spojením jednotlivých obrazů softwarově, nebo připevněním 360° kamery na střechu vozidla.




Obrázek 2.3: Rozpoznávání objektů[5]

Pomocí klasifikačních algoritmů vozidlo díky nim dokáže rozpoznat typy okolních objektů,

[5]Převzato z: https://becominghuman.ai/what-exactly-does-cnn-see-4d436d8e6e52

barvy na semaforu, dopravní značky, nebo označení pruhů na silnici. Obrázek 2.3 ukazuje zmíněnou klasifikaci objektů.

Ačkoliv kamery dokáží rozpoznat okolní objekty, neumí na rozdíl od radaru nebo Lidaru určit jejich vzdálenost. Jejich efektivita také klesá s nižší viditelností – při dešti, mlze, nebo v noci. Rychlá jízda také snižuje úspěchy rozpoznání z důvodu rozmazání obrazu. Je doporučeno, aby kamery měly co největší dynamický rozsah (>130 dB), pro zajištění čistého obrazu, i když do čočky svítí přímé sluneční světlo [16].

### Ultrazvukové senzory

Jak jméno napovídá, ultrazvukové senzory používají akustické vlny k získání informace o vzdálenosti objektů. Princip měření vzdálenosti je podobný jako u senzorů radar a Lidar – je měřena doba mezi odesláním a přijetím vyslané zvukové vlny, podle které se poté odvodí vzdálenost překážky, od které se tato vlna odrazí zpět k přijímači. Vlny jsou vysílány na frekvenci 20 až 40 kHz pomocí magnetorezistivní membrány, umístěné ve vysílači vln [33].

Tento typ senzoru dokáže detekovat objekty v blízké vzdálenosti vozidla nezávisle na materiálu a barvě objektu či počasí. Jejich výhodou jsou také nízké náklady, díky čemuž se používají v parkovacích systémech, kde je jich potřeba umístit několik kolem celého vozu k dosažení požadovaných výsledků. Maximální efektivní vzdálenost těchto senzorů bývá obvykle 5 až 10 metrů.

Obrázek 2.4 zobrazuje grafické rozhraní parkovacího asistenta vozidla značky AUDI, využívající ultrazvukové senzory.


Obrázek 2.4: Grafické rozhraní parkovacího asistenta ve vozidlu značky AUDI, které využívá ultrazvukové vlny k detekci blízkých objektů[6]

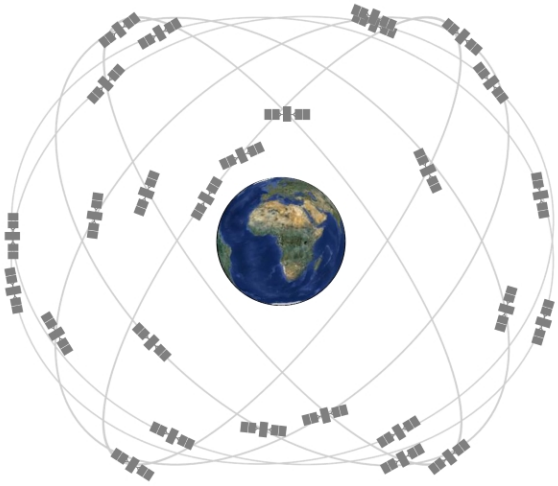
### Satelitní navigace

Satelitní navigace je systém využívající družice obíhající zeměkouli k zajištění samosprávného určení polohy na Zemi. Díky satelitní navigaci má vozidlo představu o své pozici na zeměkouli a dokáže se díky tomu dále navigovat prostředím.

[6]Převzato z: https://www.audiworld.com/forums/a8-s8-d3-platform-discussion-60/mmi-parking-sensor-screen-2845197

V roce 2020 jsou jediné plně funkční systémy satelitní navigace s globálním pokrytím (GNSS) americký *Global Positioning System* (GPS) [5], ruský *Global Navigation Satellite System* (GLONASS) [20] a čínský *BeiDou Navigation Satellite System* (BDS) [1].

**GPS** Nejznámější systém satelitní navigace s globálním pokrytím je americké GPS, který disponuje 27 družicemi obíhajícími zeměkouli ve výšce 20200 km po šesti různých osách (viz obrázek 2.5), což zajišťuje, že na jakémkoli místě na zemi jsou alespoň 4 družice viditelné. GPS slibuje v 95 procentech případů přesnost $\leq 7,8$ m [4].



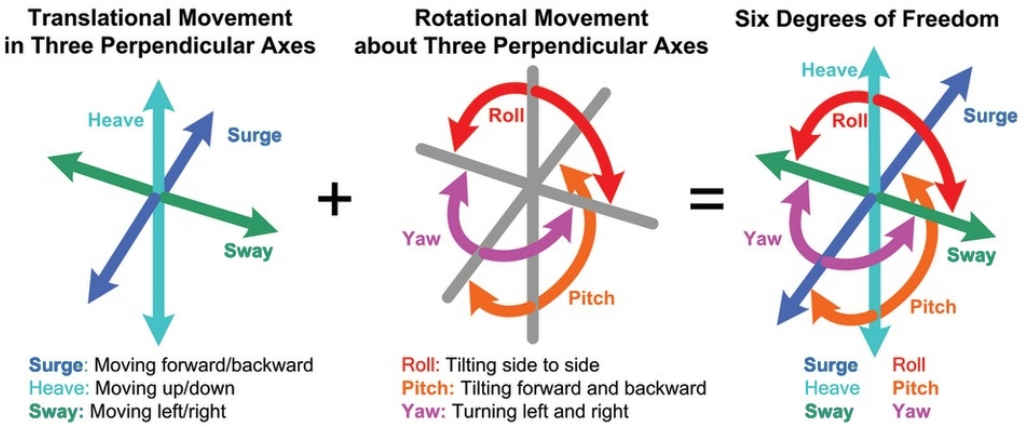
Obrázek 2.5: Vizualizace pohybu družic systému GPS kolem Země[7]

### Inerciální měřící jednotka

Inerciální měřící jednotka (IMU) je další systém pro určení polohy, který ale na rozdíl od satelitní navigace nekomunikuje s jinými venkovními zařízeními. IMU používané v samočinně řízených vozidlech je zařízení, jež obsahuje tříosý akcelerometr pro měření změny rychlosti vozidla a tříosý gyroskop pro získání informace o úhlovém natočení vozidla ve třech osách. Spojení těchto dvou komponent umožní zachytit jakýkoliv pohyb vozidla pomocí tzv. šesti stupňů volnosti [26], viz obrázek 2.6. Dále je také součástí IMU magnetometr, který měří sílu magnetického pole a podle toho určí, jakým směrem je vozidlo iniciálně natočeno. Lze říci, že pokud satelitní navigace určuje absolutní pozici vozidla, IMU určuje relativní.

IMU dokáže pomáhat vozidlu držet se v pruhu, detekovat ztrátu trakce vozidla, zpomalit a zastavit vozidlo v nečekané situaci, nebo při ztrátě GPS signálu aktualizovat pozici vozidla na mapě [19].

[7] Převzato z: https://www.gps.gov/systems/gps/space/




Obrázek 2.6: Šest stupňů volnosti[8]

### 2.1.3 Dosavadní pokroky ve vývoji

V dnešní době již téměř každá automobilka investuje část svého kapitálu do výzkumu a vývoje technologií, které zajistí vyšší automatizaci jejich vozidel. Kromě automobilek pracují na vývoji samočinně řízených vozidel i jiné společnosti, které zabývají čistě vývojem těchto technologií a ke svému vývoji využívají cizí vozy.

Přestože byl zaznamenán obrovský pokrok v oblasti samočinně řízených vozidel za posledních 10 let, neexistuje zatím žádná společnost, která by disponovala vozidly splňující 5. úroveň automatizace[9].

Následující sekce popisují vedoucí společnosti v oblasti automatizace vozidel.

#### Waymo

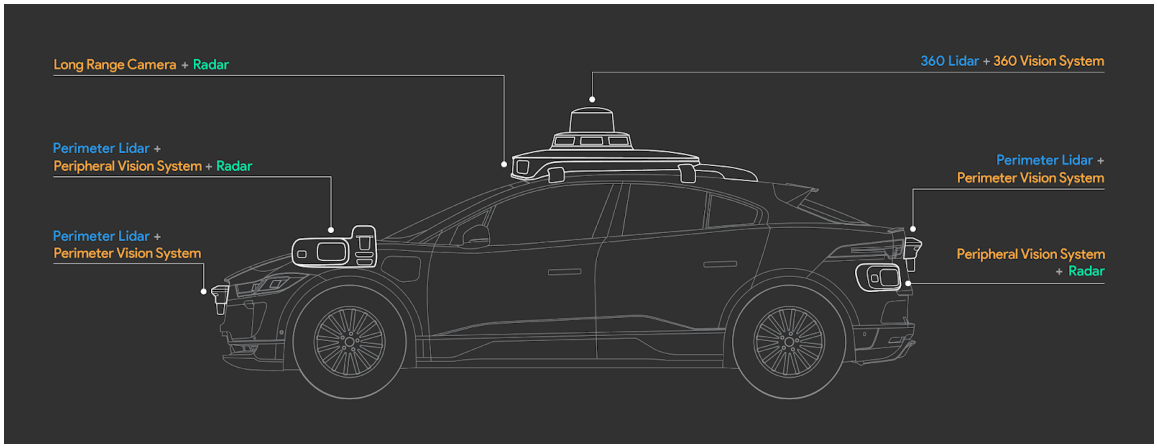
Společnost Waymo [9] začala jako projekt samočinně řízeného vozidla od společnosti Google. Jejich cílem je nahradit klasickou taxi-službu plně autonomními vozidly, která si mohou lidé skrze chytrý telefon přivolat a vozidlo je poté doveze na zadanou destinaci.

Co se týče roku 2020, vozidla společnosti Waymo splňují 4. úroveň automatizace, což je plná automatizace, ale pouze v předem zpracovaných oblastech (tzv. „geofencing“). Od roku 2009, co tento projekt začal, ujela jejich auta přes 32 milionů kilometrů na reálných silnicích a přes 16 miliard kilometrů v simulaci [23]. Hlavní testovací oblastí je část metropolitní oblasti města Phoenix v americkém státě Arizona, kde díky místnímu klimatu jsou vozidla podrobena adekvátní zkouškou. Členové speciálního zákaznického programu mohou dokonce využívat službu vozidel bez řidiče. Pro ostatní je ve vozidle bezpečnostní řidič, který kontroluje, zdali vozidlo jede bezpečně, popřípadě může zamezit nebezpečným situacím.

Aktuálně je nejnovější iterací vozidel jejich pátá generace, která disponuje senzory v kombinaci Lidaru, kamer a radaru (viz obrázek 2.7).

[8] Převzato z: http://www.leadingones.com/articles/intro-to-vr-4.html

[9] Úrovně automatizace – viz sekce 2.1.1




Obrázek 2.7: Kombinace senzorů páté generace vozidel společnosti Waymo[10]

### Tesla

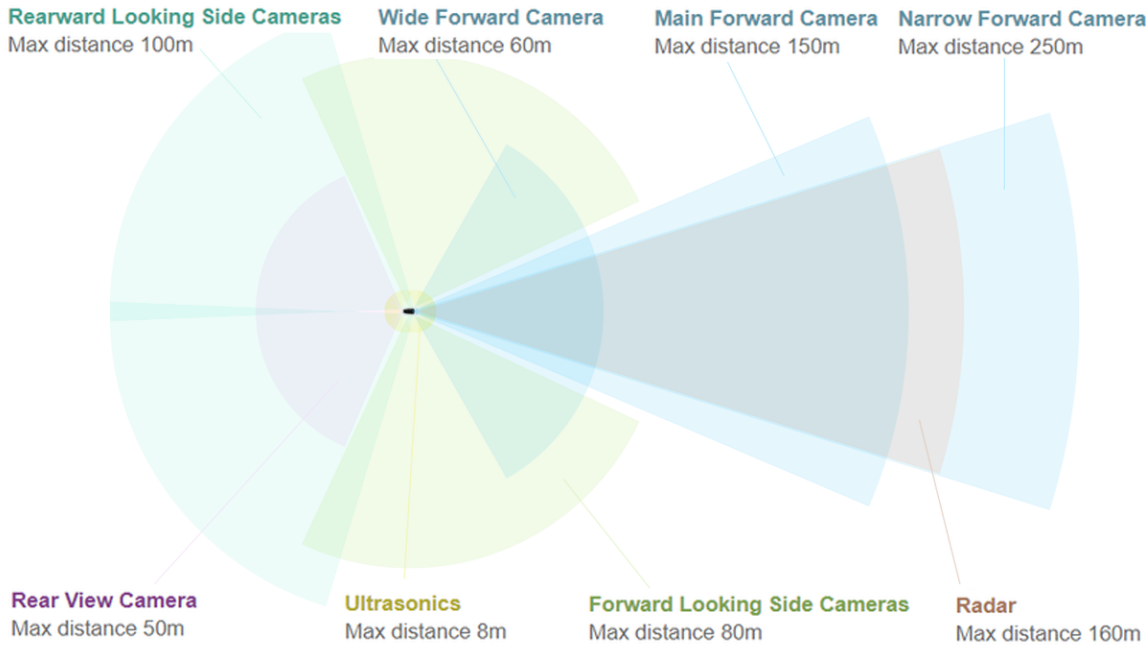
Ačkoliv byla společnost Tesla v předchozích letech hlavní lídr v oblasti samočinně řízených vozidel, nechala se předběhnout jinými společnostmi. Navzdory popularitě jejich vozidel, kterých již bylo vyrobeno více než milion [32], tato vozidla stále splňují pouze druhou úroveň automatizace. Stále se ale dá tato společnost považovat za jednu z předních, a to právě díky zmíněnému počtu vozidel, která denně jezdí po silnicích.

Jejich samočinně řízená vozidla jsou taková velká sběrna dat, která jsou odesílána na centrálu a díky nim se vylepšuje jejich software zvaný Autopilot. Až bude tento software natolik schopný, aby zajišťoval plnou automatizaci, všechny vozy disponující hardwarem verze 3 a více, dostanou možnost se pomocí softwarové aktualizace stát plně autonomní. Kdy tento moment nastane ale není jisté. Navzdory tomu, že ostatní společnosti pomalu odstupují od svých předchozích předpokladů dosáhnutí páté úrovně automatizace, CEO společnosti Tesla Elon Musk slibuje dosažení plné automatizace koncem roku 2020 [29].

Hardware verze 2.5 a 3 obsahuje 8 kamer umístěných tak, aby pokrývaly všech 360° kolem vozidla, s tím, že přední dalekonosná kamera „vidí“ až na vzdálenost 250 metrů. Dále je na palubě vozidla do předu umístěný radar s maximální vzdáleností 160 metrů a 12 ultrazvukových senzorů, které pracují na maximální vzdálenost 8 metrů a slouží na detekci překážek v blízkosti vozidla. Rozmístění jednotlivých senzorů zachycuje obrázek 2.8.

Jelikož vozy Tesla neobsahují Lidar, pro detekci objektů spoléhají pouze na obraz z kamer a na kvalitní zpracování obrazu. Z toho důvodu je nutné mít čip, který dokáže obraz efektivně zpracovávat. Proto je součástí hardware verze 3 procesor, který dokáže zpracovávat 2000 snímků za sekundu a je tak 10krát výkonnější něž procesor hardwaru verze 2.5, který zvládal pouze 200 snímků za sekundu [24].

[10]Převzato z: https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html



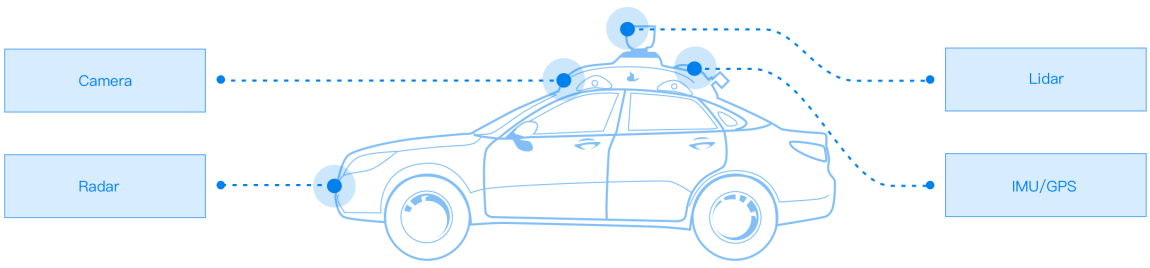

Obrázek 2.8: Vizualizace pokrytí okolí vozidla pomocí senzorů vozidel Tesla s hardwarem verze 2.5, nebo 3[11]

### Baidu

Čínská společnost Baidu nejprve začala jako internetový vyhledávač a brzy se rozvětvila do mnoha jiných oblastí – jednou z nich je právě vývoj plně autonomních vozidel.

Jejich 300 vozidel bylo zatím otestováno na třech milionech kilometrů a stejně jako Waymo, společnost Baidu začala testovat své vozy jako alternativu taxi služby. Vozidla splňují čtvrtou úroveň automatizace, přičemž je uvnitř vždy přítomný bezpečnostní řidič. Vozidla s technologií Apollo mají kromě deseti kamer, ještě Lidar a Radar [35].

Je předpokládáno, že trh samočinně řízených vozidel v Číně bude mít rokem 2030 hodnotu 500 miliard dolarů, a to jistě napomohlo společnosti Baidu zahájit spolupráci s mnoha čínskými automobilovými výrobci [25]. Baidu očekává příchod prvních plně autonomních vozidel kolem roku 2025. Obrázek 2.9 ukazuje umístění senzorů ve vozidlech společnosti Baidu.




Obrázek 2.9: Vizualizace umístění senzorů vozidel Baidu Apollo[12]

[11] Převzato a upraveno z: https://www.tesla.com/autopilot
[12] Převzato z: http://apollo.auto/platform/perception.html

### 2.1.4 Očekávaný vliv

Bezpečnost na cestách je hlavním „tahákem“ zájmu o samočinně řízená vozidla. Méně dopravních nehod ale také znamená méně potenciálních dopravních zácp, což zajistí plynulejší dopravní tok a kratší čas strávený na cestách. Se snižujícím se počtem dopravních nehod se můžeme také těšit z nižších cen za pojištění vozidla. Lze také očekávat, že většina samočinně řízených vozidel bude poháněna elektřinou, nebo jinou alternativní energií, která není pro naše prostředí takovou přítěží, jako vozidla se spalovacím motorem. Pokud ne, dobře promyšlený algoritmus dokáže jistě efektivněji využívat palivo než člověk. Plně autonomní vozidla dále slibují vyšší samostatnost pro lidi s omezenou mobilitou, vyšší pohodlí při jízdě, vylepšenou navigaci prostředím, nebo menší počty vozidel na silnicích skrze potenciální rozšíření služeb sdílených jízd.

Na druhou stranu, v moment, kdy se stanou plně autonomní vozidla dostupná, plno podniků je bude vnímat jako náhradu lidské síly. Tato změna bude nejvíce znát v oblastech transportu a služeb. Mnoho lidí z těchto oborů si bude muset hledat nové zaměstnání a se zvyšující se automatizací v ostatních oborech, je otázkou, zda toto nebude tvořit problém. Není taky jisté, zdali výrobci neuspěchají vydání prvních verzí plně autonomních vozidel, a výsledkem bude zbytečná ztráta na životech – věc které by se dalo lehce vyvarovat delší dobou vývoje.

## 2.2 Modelování systémů

Tato sekce obsahuje vysvětlení základních pojmů v oblasti modelování a simulace systémů a přehled modelovacích/simulačních nástrojů.

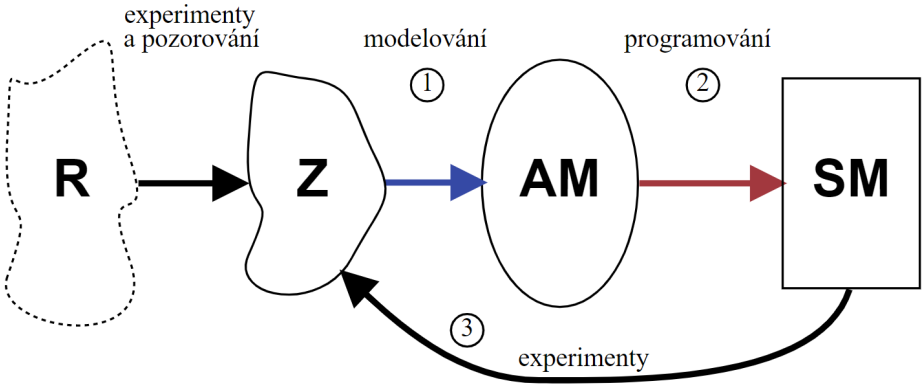
### 2.2.1 Základní pojmy

Tato sekce byla přejata z [31].

- **Systém** – Soubor elementárních částí (prvků systému), které mezi sebou mají určité vazby. Systémy lze dělit na reálné, nereálné, statické a dynamické. Pro simulaci se používají především ty dynamické.
- **Modelování** – Proces vytváření modelu na základě zjištěných znalostí o napodobovaném systému.
- **Model** – Napodobenina systému jiným systémem, v našem případě počítačovým programem. Musí modelovat všechny pro nás podstatné části modelovaného systému. Během modelování je často nejprve vytvořen tzv. „abstraktní model“, který neobsahuje všechny vlastnosti systému, pouze ty důležité pro náš experiment. Z abstraktního modelu poté vychází model simulační. Ten obsahuje všechny vlastnosti abstraktního modelu a je navíc spustitelný – lze na něm provádět simulaci.
- **Validace modelu** – Proces ověřování adekvátnosti abstraktního modelu vůči napodobovanému systému. Validitu lze chápat jako míru přesnosti a použitelnosti získaných výsledků.
- **Verifikace modelu** – Ověření izomorfního (1:1) vztahu mezi abstraktním a simulačním modelem.

- **Simulace** – Provádění experimentů nad modelovaným systémem za účelem získání nových znalostí o systému – obvykle je nutno opakovat vícekrát s různými parametry. Výsledkem jsou informace o chování systému za námi zadanými podmínkami. Po simulaci je nutné provést analýzu a interpretaci výsledků.

Jednotlivé fáze procesu modelování a simulace zobrazuje následující obrázek:




Obrázek 2.10: Jednotlivé fáze procesu modelování a simulace. (Realita → Znalosti → Abstraktní Model → Simulační Model) [31]

### 2.2.2 Nástroje pro modelování systémů

Tato část obsahuje popis nástrojů a prostředí, používaných pro modelování a simulaci systémů.

#### SIMLIB

SIMLIB je simulační knihovna pro jazyk C++ vyvíjená na Fakultě informačních technologií Vysokého učení technického v Brně. SIMLIB nabízí možnosti pro tvorbu spojitých, diskrétních i kombinovaných modelů a řízení simulace. Integrace do jazyka C++ přináší výhody i nevýhody. Mezi výhody patří možnost využití jakýchkoli knihoven napsaných v C++ (např. grafické knihovny a uživatelská rozhraní) a případné rozšíření prostředků knihovny. Nevýhodou je nemožnost dodatečných syntaktických a sémantických kontrol, kterých překladače ostatních simulačních jazyků využívají. Je také očekávána základní znalost jazyka C++.

Model v SIMLIB je založen na principu objektově orientovaného programování – obsahuje množinu prvků (entit), které jsou propojeny vazbami, přes které komunikují. Vazby spolu s chováním prvků určují chování systému jako takového. Objekty modelu současně provádějí akce na základě přijímaných zpráv a zároveň akce nezávislé na přijímaných zprávách. Tyto akce mění stav daných objektů.

Funkce `main` slouží jako popis experimentu. V případě rozsáhlých modelů lze program strukturovat do několika souborů.

Pro diskrétní simulaci obsahuje knihovna standardní třídy pro popis diskrétního chování objektů, pro modelování standardních obslužných zařízení, skladů a sběr statistických údajů. Pro popis jednorázových dějů, které se periodicky opakují lze použít třídu `Event`. Pro popis třídy objektů s vlastním dynamickým chováním slouží abstraktní třída `Process`. U obou těchto tříd se definuje jejich chování v metodě `Behavior`.

K popisu spojitých modelů se využije propojení objektů, které reprezentují integrátory, stavové bloky a různé nelinearity. Bloky se propojí hned při jejich tvorbě, kdy konstruktor bloku dostane jako první parametr odkaz na vstupní objekt. Každý objekt má metodu `Value`, která vrací hodnotu objektu. Tento popis byl přejat z [30].

### Simulink

Simulink funguje jako nadstavba MATLABu. MATLAB rozšiřuje o možnosti modelování, simulování a analýzy dynamických systémů. Modely je možné tvořit pomocí blokových schémat nebo popsat rovnicemi. Simulink se často používá v oblasti automatizace, pro zpracování digitálních signálů, modelování řídících systémů, nebo zpracování obrazu.

Grafický editor umožňuje tvořit hierarchická bloková schémata a obsahuje knihovny předdefinovaných bloků jak pro diskrétní, tak pro spojité systémy.

Simulink také nabízí simulaci v reálném čase, propojení s hardware, nebo automatickou generaci C/C++/HDL kódu pro nasazení na vestavěné platformy [6].

### Dymola

Dymola je modelovací a simulační prostředí pro modelovací objektově orientovaný jazyk Modelica. Používá se pro simulace dynamického chování komplexních fyzikálních systémů. Díky rozsáhlému množství knihoven je možno simulovat chování a interakce mezi systémy z mnoha inženýrských domén, jako jsou např. mechanické, elektrické, termodynamické, hydraulické, a další. Uživatel může jednoduše tvořit nové komponenty, nebo upravovat stávající podle jeho potřeb. Model je možno tvořit v grafickém rozhraní způsobem „drag-and-drop“, nebo popisovat rovnicemi. Interně je model zapsán podle syntaxe jazyka Modelica formou diferenciálních a algebraických rovnic.

Dymola nabízí mnoho možností interoperability. Modely lze importovat/exportovat podle standardu FMI[13], díky kterému může být k modelu přistoupeno na jiných simulačních platformách. Dále lze k modelům přistupovat skrze rozhraní Pythonu, či Simulinku. Lze také generovat zdrojový kód, který může být použitelný na jakékoliv platformě bez potřeby Dymola licence na dané platformě [3]. Dymola používá symbolické zpracování – díky tomu obzvlášť exceluje v rychlosti řešení algebro-diferenciálních rovnice a je možné provádět simulace v reálném čase.

### UPPAAL

UPPAAL je nástroj sloužící k modelování, simulaci a verifikaci systémů reálného času. Systémy mohou být modelovány jako množina nedeterministických procesů, které představují časované automaty rozšířené o celočíselné proměnné, strukturované datové typy, možnosti definovat funkce a synchronizaci procesů pomocí kanálů. UPPAAL je často používán pro ovládací prvky pracující v reálném čase, nebo simulaci komunikačních protokolů – obecně se dá použít ve více odvětvích – především tam, kde jsou důležité aspekty načasování. Nástroj je rozdělen do tří částí – editor, simulátor a verifikátor.

[13] FMI – Functional Mock-up Interface – https://fmi-standard.org/

# Kapitola 3

# Příprava řešení

Tato kapitola popisuje zvolené realizační prostředky k modelování cílového systému (sekce 3.1) a v sekci 3.2 návrh modelu.

## 3.1 Zvolené realizační prostředky

Následující sekce popisuje nástroj UPPAAL, který byl zvolen na implementaci.

### 3.1.1 Sada nástrojů UPPAAL

UPPAAL – nástroj sloužící k modelování, simulaci a verifikaci systémů reálného času, jenž byl uveden v sekci 2.2.2, používá specifický modelovací jazyk, který je založen na časovaných automatech – konečných automatech rozšířených o hodinové proměnné. Čas je zaznamenáván spojitě a běží stejným tempem pro celý systém. Lze zjistit hodnotu časové proměnné, nebo ji vynulovat.

#### Formalismus časovaných automatů

Časované automaty lze formálně zapsat jako šestici $A = (L, l_0, C, A, E, I)$, kde:

- $L$ je množina stavů
- $l_0 \in L$ je počáteční stav
- $C$ je množina hodinových proměnných
- $A$ je množina akcí
- $E \subseteq L \times A \times B(C) \times 2^c \times L$ je množina přechodů
- $I : L \to B(C)$ přiřazuje invarianty ke stavům

Časované automaty jsou v nástroji UPPAAL složeny do sítě paralelně pracujících časovaných automatů se společnou množinou hodinových proměnných a akcí. Formálně lze tedy jazyk nástroje UPPAAL zapsat jako $n$ časovaných automatů $A_i = (L_i, l_i^0, C, A, E_i, I_i), 1 \leq i \leq n$ [12].

### Časované automaty v nástroji UPPAAL

Jednotlivé automaty se v editoru nachází uvnitř parametrizovaných šablon, které kromě automatu obsahují deklarační část. Instancí těchto šablon vzniká proces. Jednotlivé stavy automatu jsou v nástroji označeny jako místa (angl. location) a jednotlivé přechody jako hrany (angl. edge).

**Místa** Typy míst v nástroji UPPAAL shrnuje obrázek 3.1. Počáteční a normální místo mají dostupnou možnost přiřazení `invariant`, což je podmínka, která musí platit, když se proces nachází v daném místě. Každý proces musí mít přesně jedno počáteční místo. Kromě počátečních a normálních míst existují další dva typy:

**Urgent místa** jsou místa, ve kterých je zastaven čas.

**Committed místa** zastavují čas stejně jako `urgent` místa s jednou další podmínkou: pokud se systém nachází v tomto místě, je nutné, aby další přechod proběhl z jednoho z `committed` míst.


Obrázek 3.1: Typy míst nástroje UPPAAL.

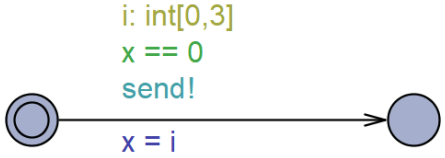
**Hrany** Propojují jednotlivá místa procesu a lze skrze ně později provádět přechody. Hrany lze větvit a přiřadit jim pravděpodobnostní váhy. K hraně jdou přiřadit čtyři různé vlastnosti (obrázek 3.2).

**Select** Nedeterministické přiřazení hodnoty z daného rozsahu do zvolené proměnné. Proměnná je dostupná pouze na této hraně.

**Guard** Podmínka, jejíž pravdivost určuje, zda může být přechod proveden.

**Synchronization** umožňuje přiřadit k hraně výraz v podobě synchronizačního kanálu (typ `chan`). Uvažujme synchronizační kanál deklarovaný jako `chan send`. Přiřazení synchronizačního výrazu `send!` způsobí, že se předá řízení jinému procesu, který má v synchronizační vlastnosti hrany výraz `send?`.

**Update** obsahuje seznam příkazů oddělených čárkami, které se provedou při uskutečnění přechodu.




Obrázek 3.2: Typy vlastností hrany. Žlutá – `select`, zelená – `guard`, tyrkysová – `synchronization`, modrá – `update`.

### Rozhraní a funkce nástroje

**Editor** Editor je rozdělen do dvou částí: postranní panel pro přístup k jednotlivým šablonám, či deklaracím a grafické plátno. Postranní panel obsahuje následující položky:

**Globální deklarace** obsahuje globální proměnné, hodiny, synchronizační kanály a konstanty.

**Šablony** Jeden parametrizovaný časovaný automat je vždy obsahem jedné šablony. K šabloně vždy náleží textový soubor pro lokální deklarace.

**Systémové deklarace** Procesy se zde instancují ze šablon a volí se, které procesy budou součástí systému.

**Simulátor** Systém lze simulovat třemi způsoby: uživatel si může systém spustit manuálně a volit si, které přechody se provedou. Další možností je provést všechny přechody náhodně. Poslední možností je provedení přechodů podle předem dané trasy, kterou lze načíst ze souboru, nebo jako výstup verifikátoru. Simulátor je rozdělen do čtyř částí:

**Ovládací panel** obsahující ovládací prvky, pomocí kterých je možno řídit simulaci.

**Panel proměnných** zobrazující hodnoty proměnných v daném stavu umožňuje sledovat hodnoty proměnných typu integer, bool a obsah prvků pole. Hodnoty hodinových proměnných jsou zde vyjádřeny pouze omezující podmínkou.

**Přehled o systému** umožňuje vidět všechny běžící procesy a jejich aktivní stavy.

**Sekvenční diagram** znázorňující synchronizace mezi jednotlivými procesy.

**Verifikátor** V záložce verifikátoru může uživatel pomocí dotazů ověřit dostupnosti stavů. Verifikátor dokáže průchodem celého stavového prostoru ověřit uživatelem zadané dotazy a v případě, že klauzule dotazu není pravdivá, lze načíst do simulátoru trasu, která vedla jeho nesplnění.

### Dotazovací verifikační jazyk

UPPAAL používá pro verifikaci modelů podmnožinu TCTL logiky. Přehled dostupných dotazů:

- `E<>` `p`: Existuje cesta, kde podmínka `p` někdy platí.
- `E[]` `p`: Existuje cesta, kde podmínka `p` vždy platí.
- `A<>` `p`: Pro všechny cesty podmínka `p` někdy platí.
- `A[]` `p`: Pro všechny cesty podmínka `p` vždy platí.
- `p` $\rightarrow$ `q`: pokud je někdy splněna podmínka `p`, poté bude `q` také splněna.

### UPPAAL SMC

Nástroj UPPAAL SMC je samostatný nástroj, který obsahuje stejné funkce jako základní UPPAAL a navíc přidává funkci statistického model checkingu (*SMC*). Statistický model checking je technika formální verifikace, která kombinuje simulaci a statistické metody pro analýzu stochastických systémů. UPPAAL SMC tedy umožňuje důkladnější monitorování

simulací v podobě statistického zpracování výsledků, které nám dokáží přiblížit míru důvěry určitého jevu [13]. Dále je přidána podpora pro stochastické hybridní systémy – systémy, kde čas může v jednotlivých procesech běžet jiným tempem. Tempo hodinových proměnných lze určit obyčejnými diferenciálními rovnicemi (*ODE*). Přidána byla také podpora pro proměnné s plovoucí desetinnou čárkou (typ `double`) a možnost vytvářet nové instance procesů za běhu simulace. Byly také přidány následující dotazy verifikátoru pro statistické ověření vlastností modelů:

- `simulate N [<=t] { E1,..,Ek }`: monitorování a následná vizualizace hodnot proměnných `E1,...,Ek` za `N` simulací. Časová délka simulace je omezena parametrem `t`.
- `Pr[t](<> expr)`: určení pravděpodobnosti, že se výraz `expr` vyhodnotí pravdivě.
- `E[t;N]([min,max]:expr)`: odhad maximální/minimální hodnoty výrazu `expr`.

Upravením hodnoty parametru *Probability uncertainty* ($\varepsilon$), který lze najít v menu *Options* → *Statistical parameters*, lze zpřesnit výsledky pravděpodobnostních dotazů výměnou za delší simulační čas.

### UPPAAL Stratego

UPPAAL Stratego spojuje funkce nástrojů UPPAAL SMC (*Statistical Model Checking*) a UPPAAL TIGA (*Timed Game Automata*) a přidává také nové strategicky založené dotazy verifikátoru. Stejně jako v posledním zmíněném nástroji, je zde podpora pro tzv. „Timed Game Automata“, což jsou časované automaty, kde je množina akcí $A$ rozdělena na „ovladatelné“($A_c$) a „neovladatelné“($A_u$) akce.

Problematiku těchto automatů lze vysvětlit způsobem dvou hráčů hrajících hru. V našem případě je jeden hráč tzv. „ovladač“, který ovládá ovladatelné (základní) hrany, druhý je „prostředí“, ovládající neovladatelné hrany. Podmínky výhry jsou specifikované pomocí TCTL dotazu. Výhry se dosáhne pouze pokud je přechod na cílový stav skrze ovladatelnou hranu [15].

**Strategie** Strategie je předpis rozkazů, které má hráč typu „ovladač“ provést během hry. Strategie může hráči pouze říci, zdali má v daný moment provést ovladatelnou akci, nebo neprovádět žádnou akci. Strategii lze označit jako výherní, pokud hráč typu „ovladač“ na základě dané strategie dokáže vždy vyhrát, nehledě na akcích zvolených hráčem typu „prostředí“.

Strategii lze ve verifikátoru deklarovat klíčovým slovem `strategy S`, kde `S` je proměnná uchovávající strategii. Poté lze pomocí ní řídit simulaci použitím klíčového slova `under S` na konci dotazu, který spouští simulaci.

V nástroji UPPAAL Stratego lze pracovat s třemi typy strategií. Nedeterministické strategie nabízí množinu akcí pro každý stav, deterministické pouze jednu akci pro každý stav a stochastické distribuci množiny akcí v každém stavu [11]. Možnosti deklarace strategií a jejich použití lze vidět v tabulce 3.1.

| | |
|---|---|
| Safety | `A[] prop under NS` |
| Liveness | `A<> prop under NS` |
| Guarantee objective | `strategy NS = control: A<> prop` |
| Guarantee objective | `strategy NS = control: A[] prop` |
| Evaluation | `Pr[bound](<> prop) under SS` |
| Expected value | `E[bound;int](min: prop) under SS` |
| Simulations | `simulate int [bound]expr1,expr2 under SS` |
| Minimize objective | `strategy DS = minE (expr) [bound]: <> prop under NS` |
| Maximize objective | `strategy DS = maxE (expr) [bound]: <> prop under NS` |

Tabulka 3.1: Tabulka zobrazující možné deklarace strategií, převzato z [11].

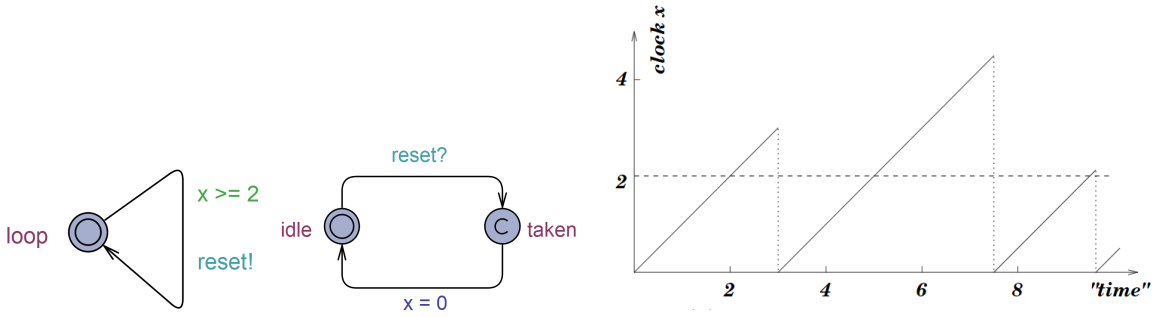
### 3.1.2 Praktická ukázka

První část ukázky obsahuje tři příklady [12], za účelem přiblížení modelování a verifikace systému složeného z paralelních procesů. Druhá část se zaměřuje na jejich statistickou analýzu.

#### Modelování a verifikace

Mějme dva procesy – `Test` a `Observer`, kde proces `Observer` slouží k detekci událostí na zkoumaném systému (`Test`), aniž by musel být zkoumaný systém zásadně upraven. Proces `Test` tvoří cyklus s podmínkou `x<=2` mezi jednotlivými iteracemi, přičemž `x` je globální hodinová proměnná a `reset` synchronizační kanál, po kterém každou iterací cyklu vyšle signál, který proces `Observer` zachytí a vynuluje hodinovou proměnnou `x`. Stav `taken` procesu `Observer` je *committed*, aby jeho navštívení nezasahovalo do měřeného času. Obrázek 3.3 zobrazuje automaty obou procesů a graf chování času. Chování dále ověřují následující dotazy:

- `A[] Observer.taken imply x>=2` : Ověření, že vynulování hodinové proměnné `x` nastane vždy, kdy tato proměnná nabývá hodnoty alespoň dvě.
- `E<> Observer.idle and x>3` : Zjištění, zdali může nastat situace, kdy `x` nabude hodnoty alespoň 3 ještě před přijetím signálu k resetování `x`.

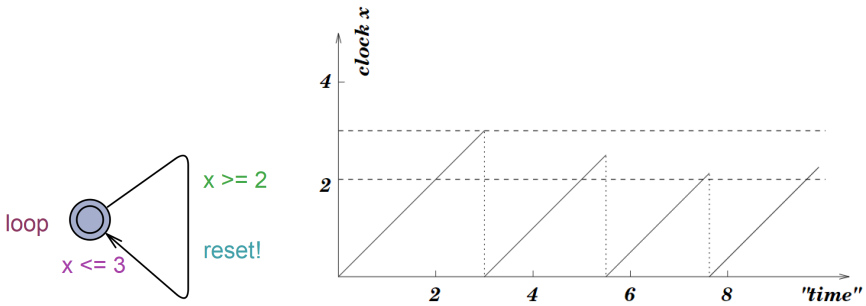



(a) Test. (b) Observer. (c) Graf chování času

Obrázek 3.3: Příklad č.1.

Nyní přidáme výraz `x <= 3` jako vlastnost *invariant* místu `Test.loop` (obr. 3.4). Tímto vznikne omezení na dobu, po kterou může systém čekat v daném stavu. Výsledky je možné vidět již při porovnání grafu minulého příkladu (3.3c) a aktuálního (3.4b). Dotaz `E<> Observer.idle and x>3` z minulého příkladu již pro tuto modifikaci neplatí. Následující dotazy ověřují nové chování systému:

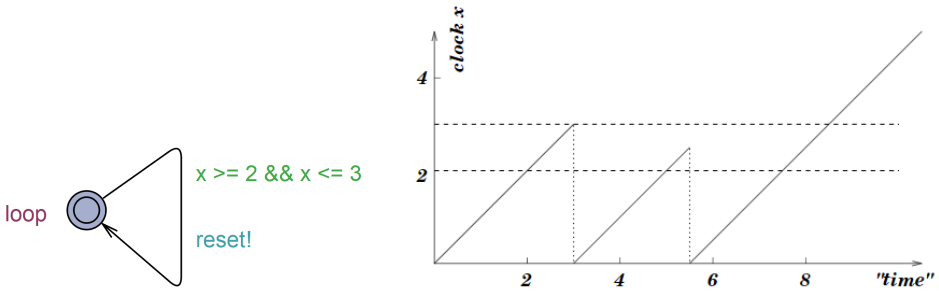
- `A[] Observer.taken imply (x>=2 and x<=3)` ověřuje, že přechod je vždy proveden, když je `x` mezi hodnotou 2 a 3.
- `E<> Observer.idle and x>2` : Je možné, že se přechod provede, když `x` je větší než 2.
- `A[] Observer.idle imply x<=3` : Kontrola horní hranice.




(a) Test. (b) Graf chování času

Obrázek 3.4: Příklad č.2 s přidanou vlastností *invariant* u místa `Test.loop`.

Nyní pokud opět upravíme automat procesu `Test` podle obrázku 3.5a, může vzniknout podezření, že se proces bude chovat stejně jako v příkladu na obrázku 3.4. Co, tato změna ale ve skutečnosti přináší zachycuje graf 3.5b. Proces bude respektovat podmínku pro přechod (`x>=2 and x<=3`), což ale neznamená že se přechod v tomto případě pokaždé provede. Pokud se přechod neprovede mezi 2 a 3 časovými jednotkami, vznikne uváznutí systému. Stane se tak jelikož již neexistuje žádná vlastnost, která by nutila přechodu po dosažení určité hodnoty času tak, jako to dělá vlastnost *invariant*. Toto lze ověřit dotazem `A[] not deadlock`, který se nesplní.




(a) Test. (b) Graf chování času

Obrázek 3.5: Příklad č.3. Automat procesu `Test` oproti příkladu na obrázku 3.3 nyní obsahuje upravenou podmínku na přechodu

#### Statistická analýza

Grafy v předchozích příkladech nebyly generovány nástrojem UPPAAL, ale pokud používáme verzi SMC, či Stratego, které jsou rozšířené o statistický model checking, lze téměř identický graf získat pomocí dotazu 3.1.

```
simulate 1 [<=10] {x}
```
(3.1)

Dále pokud se vrátíme zpět k příkladu č.2 na obrázku 3.4, můžeme pomocí dotazu 3.2 zjistit pravděpodobnost, že proces `Test` vyšle signál `reset`, když hodnota časové proměnné `x` bude menší než 2,1, což by za normálních okolností mělo být přesně 10 %.

```
Pr [<=3] (<> (Observer.taken) && (x < 2.1))
```
(3.2)

## 3.2 Návrh abstraktního modelu

Tato sekce popisuje požadavky a návrh prvků a vlastností abstraktního modelu. K modelovanému vozidlu je referováno jako „samočinně řízené vozidlo“, jelikož model není připraven na všechny možné situace – jeho funkčnost je omezena na rovný úsek dálnice (kategorizace samočinně řízených vozidel podle jejich samostatnosti a funkčnosti byla popsána v sekci 2.1.1).

### 3.2.1 Požadavky

Cílem je vytvoření abstraktního modelu samočinně řízeného vozidla, abstrakci několika proudového úseku vozovky a ostatní vozidla, které se po vozovce spolu se samočinně řízeným vozidlem pohybují. Samočinně řízené vozidlo reaguje jednak na pohyb ostatních vozidel a také na dopravní značení určující maximální povolenou rychlost.

Hlavními prvky abstraktního modelu, které budou popsány v následujících sekcích jsou tedy vozovka, samočinně řízené vozidlo, ostatní vozidla a značení upravující maximální povolenou rychlost.

### 3.2.2 Abstrakce vozovky

V abstraktním modelu se vozovkou rozumí libovolně dlouhý úsek dálnice s libovolným počtem pruhů, po kterém se mohou jak samočinně řízené vozidlo, tak ostatní vozidla pohybovat maximální rychlostí 130 km/h. Pokud není upravena značením, je rychlost vozidel, pod kterou neklesnou, automaticky 80 km/h.

Abstraktní model zanedbává druhou část vozovky, která je v protisměru, jelikož v reálném světě nemají ve většině případů vozidla v protisměru dálnice za normálních podmínek žádný vliv na protijedoucí automobily. U vozovky se dále také zanedbávají zatáčky, či převýšení.

### 3.2.3 Abstrakce samočinně řízeného vozidla

Samočinně řízené vozidlo (dále jen „vozidlo“) pomocí Lidaru dokáže detekovat vozidla a pomocí kamery dopravní značení upravující maximální povolenou rychlost a na základě získaných informací z těchto senzorů adekvátně reagovat v jistých situacích.

Vozidlo se automaticky drží v nejpravějším (prvním) pruhu vozovky, pokud nepředjíždí ostatní vozidla. Pokud je rychlost nejbližšího vozidla v pruhu před vozidlem minimálně

o 10 km/h nižší, než je maximální povolená rychlost v daném místě, pokusí se o předjetí. Jinak zůstane v bezpečné vzdálenosti dvou sekund a upraví svoji rychlost tak, aby se rovnala rychlosti tohoto vozidla (následuje toto vozidlo). Dále následuje nejbližší vozidlo ještě v případě, že jsou všechny pruhy vozovky obsazeny tak, že není možné tímto „shlukem“ vozidel projet bez toho, aby se dostalo do nebezpečné vzdálenosti k některému z vozidel shluku. Pokud se tak stane, snižuje rychlost tak, aby se dostalo zpět do bezpečné vzdálenosti.

Vozidlo přejíždí z pruhu do pruhu skokově a nemůže provést další změnu pruhu, pokud poslední proběhla během poslední 0,5 sekundy.

Vozidlo reaguje na dopravní značení upravující maximální rychlost až v momentě, kdy se dostane do časové vzdálenosti[1] 2,4 sekund od dané značky.

### Volba jízdních pruhů pro předjetí vozidel (koncept cílového pruhu)

Jednoduchou logikou, která by se dala aplikovat pro přejezdy samočinně řízeného vozidla mezi jízdními pruhy by bylo: Přejet do pruhu nalevo, pokud se vyskytuje nějaké vozidlo před samočinně řízeným vozidlem, jinak přejet do pravého. Tato varianta má ale příliš nevýhod, jedna z nich je například, že mohou nastat situace, kdy by vozidlo stále dokola přejíždělo mezi dvěma stejnými pruhy, což není vhodné, a proto model implementuje jinou variantu logiky přejíždění mezi jízdními pruhy.

Použitá varianta na rozdíl od zmíněné nepočítá jenom s jedním (nejbližším) vozidlem, ale s vozidlem v každém pruhu – nejbližším shlukem vozidel, která se nachází před samočinně řízeným vozidlem. Podle analýzy vzdáleností vozidel ve shluku se určí pruhy, kterými lze projet (dále jen „průjezdné pruhy“). Podmínkou nutnou pro splnění, aby byl pruh průjezdný je, aby se v pruhu nenacházelo žádné vozidlo – tento pruh se označí jako průjezdný. Další možností je, aby vzdálenost mezi vozidly jejichž pruhy spolu sousedí, byla minimálně 2,1násobek doporučené[2] časové vzdálenosti mezi vozidly. V tomto případě je pruh, ve kterém je vozidlo vzdáleno dále od samočinně řízeného vozidla, klasifikován jako průjezdný. Tyto pruhy jsou poté označeny jako tzv. „cílové pruhy“.

Z kolekce obsahující cílové pruhy se pomocí iterátoru vybere vždy jeden cílový pruh, do kterého se vozidlo snaží přejet. Tato kolekce je seřazena prioritně, s tím, že nejvyšší prioritu mají prázdné pruhy oproti průjezdným a druhou nejvyšší prioritou jsou čísla pruhu – čísla pruhu jsou seřazena vzestupně (číslo pruhu vpravo je nižší než číslo pruhu vlevo). Díky tomu model upřednostňuje jízdu v pruhu nejvíce vpravo. V případě, že do cílového pruhu nelze přejet, iteruje se mezi jednotlivými cílovými pruhy v kolekci cílových pruhů. Iterátor se inkrementuje vždy pokud vozidlo není v cílovém pruhu a nemůže do něho přejet. Iterátor se resetuje v případě, že se kolekce cílových pruhů změnila, nebo pokud vozidlo v jednom z nejbližších pruhů opustilo kritickou vzdálenost[3] samočinně řízeného vozidla.

### Lidar

Lidar slouží k detekci vozidel v blízkosti samočinně řízeného vozidla. Součástí vozidla jsou 4 senzory typu Lidar, které jsou umístěny na střeše a dohromady pokrývají všech 360° okolí vozidla. Parametry jednotlivých Lidar senzorů zachycuje tabulka 3.2.

[1] Časová vzdálenost – čas, za který vozidlo vzadu ujede při zachování rychlosti aktuální vzdálenost mezi entitami, viz 3.2.7

[2] Doporučená vzdálenost – časová vzdálenost mezi 1,9 a 2,1 sekundami, viz 3.2.7

[3] Kritická vzdálenost – časová vzdálenost menší než 1,9 sekundy, viz 3.2.7

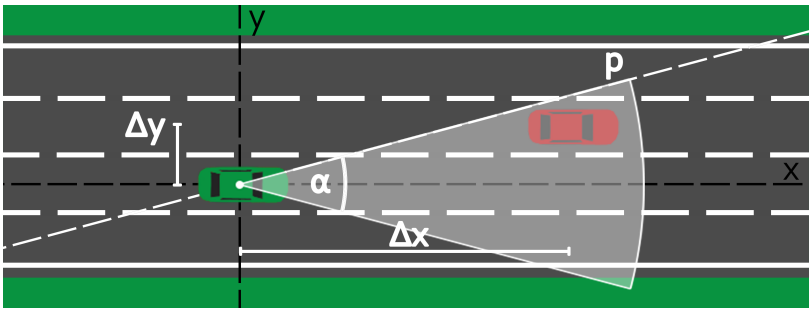
| Nasměrování senzoru | Dosah [m] | Zorný úhel [°] | Latence [ms] |
|---|---|---|---|
| vpřed | 220 | 45 | 70 |
| vlevo/vpravo | 60 | 135 | 90 |
| dozadu | 80 | 45 | 50 |

Tabulka 3.2: Vlastnosti Lidar senzorů

**Ověření, zdali vozidlo spadá do plochy, kterou Lidar pokrývá** Pro ověření, že vozidlo opravdu spadá do plochy, kterou daný senzor pokrývá, se prochází kolekce všech vozidel a testuje se každé vozidlo zvlášť. Pokud se rozhodne, že vozidlo je v dosahu senzoru samočinně řízeného vozidla, je kopírováno do kolekce vozidel v dosahu samočinně řízeného vozidla. Při rozhodování je nutno znát souřadnice samočinně řízeného vozidla, které si označíme jako $(x_1, y_1)$, kde $x_1$ je ujetá vzdálenost a $y_1$ je číslo jízdního pruhu vynásobené šířkou pruhu. Také je potřeba znát souřadnice vozidla, které se testuje $(x_2, y_2)$. Výraz 3.3 využívá Pythagorovu větu pro výpočet vzdálenosti mezi samočinně řízeným vozidlem a testovaným vozidlem na základě jejich souřadnic.

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2} \tag{3.3}$$

Dále je podle zorného úhlu senzoru (tento úhel si označíme jako $\alpha$) nutno určit parametr $a$ přímky $p$, kterou zorný úhel senzoru svírá s osou $x$ podle rovnice $y = ax + c$. Osu $x$ si můžeme představit jako přímku procházející středem samočinně řízeného vozidla do směru, kterým je samočinně řízené vozidlo natočeno. Pro lepší ilustraci může napomoci obrázek 3.6.




Obrázek 3.6: Vizualizace vztahů mezi vozidly při detekci Lidarem.

Zmíněný parametr $a$ lze zjistit pomocí následujícího výrazu:

$$a = tan(\frac{\alpha}{2}) \tag{3.4}$$

Pokud bereme v potaz senzor umístěný směrem dopředu, pozice testovaného vozidla musí splňovat tři podmínky, aby bylo detekováno:

1. Vzdálenost mezi vozidly $d$ musí být menší než hodnota maximálního dosahu senzoru.
2. Musí platit nerovnice $\Delta y < a\Delta x$, která zjišťuje, zdali je vozidlo v zorném poli senzoru.
3. Vozidlo musí být před samočinně řízeným vozidlem ($x_2 > x_1$).

Senzory umístěné do ostatních směrů používají lehce modifikované podmínky. Pro senzory do boku je první modifikací otočení nerovnosti v druhé podmínce a druhou modifikací nahrazení třetí podmínky podmínkou:

- Vozidlo musí být nalevo/napravo od samočinně řízeného vozidla ($y_2 > y_1$), resp. ($y_2 < y_1$).

Pro případ senzoru umístěného dozadu, se pouze otočí nerovnost v podmínce č.3.

#### Kamera

Kamera detekuje značky upravující rychlost na maximální vzdálenost 130 metrů s latencí 80 milisekund. Tato vzdálenost je ovlivněna rychlostí vozidla a v případě nejvyšší povolené rychlosti je reálná maximální detekční vzdálenost pouze 80 metrů. Model tohoto senzoru zanedbává specifické umístění značky, pouze vyhodnocuje její relativní vzdálenost od samočinně řízeného vozidla a porovná ji s dosahem senzoru. Nechť $x_1$ je pozice samočinně řízeného vozidla, $x_2$ je pozice značky a $r$ je dosah senzoru, pak musí platit tyto dvě podmínky:

1. Značka je před vozidlem ($x_2 > x_1$).
2. Značka je v dosahu senzoru vozidla ($\Delta x < r$).

Pokud tyto podmínky platí, je značka zkopírována do kolekce značek v dosahu samočinně řízeného vozidla.

### 3.2.4 Abstrakce ostatních vozidel

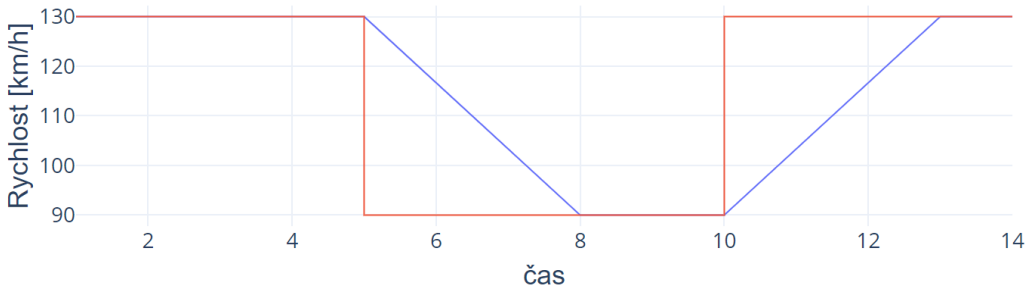
Ostatní vozidla disponují nižší inteligencí než samočinně řízené - nepřejíždí mezi pruhy a mohou se pohybovat buď konstantní nebo proměnnou rychlostí – v obou případech zohledňují dopravní značení a dokáží podle toho přizpůsobit svou rychlost. Rychlost také dokáží snížit, pokud se dostanou do blízkosti jiného vozidla, pro zamezení kolizí mezi vozidly.

Vozidla lze umístit tak, aby simulaci začínala na specifických pozicích, nebo na náhodných. Algoritmus pro umístění vozidel náhodně používá generátor normálního rozdělení pravděpodobnosti pro umístění vozidel do různých pruhů, jelikož pruhy vlevo obvykle obsahují méně vozidel než pruhy, co jsou více vpravo. Stejně tak je rozhodnuto o počáteční rychlosti vozidel.

Pokud je povolena náhodná změna rychlosti ostatních vozidel, šance na zvýšení rychlosti je ovlivněna číslem pruhu, ve kterém se vozidlo nachází. Takto budou pruhy s vyšším označením stále obsahovat rychleji jedoucí vozy.

### 3.2.5 Koncept cílové rychlosti

Cílová rychlost je jednou z vlastností každého vozidla modelu. Její princip je jednoduchý – pokud hodnota aktuální rychlosti není rovna hodnotě cílové rychlosti, snaží se hodnotu rychlosti vyrovnat hodnotě cílové rychlosti. V reálné situaci by se tento koncept dal přirovnat nastavení tempomatu vozidla na určitou rychlost. Automobil se bude snažit držet se této hodnoty. Stejnou funkci má tato cílová rychlost. O vyrovnání rychlosti s cílovou rychlostí se stará algoritmus popsaný v sekci 3.2.6. Graf na obrázku 3.7 blíže vysvětluje vztah mezi aktuální rychlostí a cílovou rychlostí vozidla.




Obrázek 3.7: Graf znázorňující vztah mezi aktuální rychlostí (modrá) a cílovou rychlostí vozidla (červená).

Jako cílová rychlost se vždy zvolí jedna z dvou pomocných hodnot – cílová rychlost daná vozidlem a cílová rychlost daná značkou určující maximální povolenou rychlost. Volí se vždy nižší hodnota. Ačkoliv v modelu za normálních podmínek všechna vozidla dodržují pravidla silničního provozu, ostatní vozidla vždy nemusí jet maximální dovolenou rychlostí, proto je nutné dělit cílovou rychlost na dvě hodnoty. Díky tomu je model připraven i na situace, kdy by ostatní vozidla nedodržovala stanovené rychlostní limity.

### 3.2.6 Algoritmus pro výpočet změny aktuální rychlosti

Změna aktuální rychlosti vozidel se provádí každou iterací hlavního cyklu. Na základě cílové rychlosti a aktuální rychlosti je vypočítána navrhovaná změna rychlosti, daná výrazem $(\Delta v/15) + 1$, kde $\Delta v$ je rozdíl cílové a aktuální rychlosti vozidla. Hodnota, o kterou se rychlost změní, se tedy s každými 15 km/h rozdílu cílové a aktuální rychlosti zvyšuje. Navrhovaná je proto, že je zaprvé limitována maximální akcelerací/decelerací vozidla (limit pro deceleraci byl zvolen 4, což dává teoretickou minimální brzdnou dráhu ze 100 km/h do kompletního zastavení 35 metrů během 2,5 sekund a akcelerace nesmí být větší než 2 – teoreticky tedy vozidlo dokáže z 0 na 100 km/h zrychlit za 5 sekund). Zadruhé hodnota změny rychlosti se může lišit maximálně o 1 oproti předchozí změně rychlosti. Uložení poslední hodnoty změny rychlosti a porovnávání s navrhovanou změnou rychlosti je jedním krokem k dosažení správné změny rychlosti. Pouze tento krok ale nestačí, jelikož např. v situaci, kdy je nastavena hodnota cílové rychlosti pod nulu k zajištění maximální decelerace a rychlost vozidla se blížila nule (10 – 0 km/h), rychlost by se snížila příliš prudce – hodnota změny rychlosti by byla stále 4, přičemž očekávané chování by bylo postupné klesání hodnoty změny rychlosti ze 4 na 0. K tomuto pomáhá vzorec trojúhelníkového čísla (3.5), kdy po dosazení $\Delta v$ za $T_n$ vznikne kvadratická rovnice jejíž diskriminant bude vždy kladný. Kladný kořen této kvadratické rovnice je poté požadovaná hodnota změny rychlosti.

$$T_n = \frac{n^2 + n}{2} \tag{3.5}$$

### 3.2.7 Počítání vzdáleností a jejich typy

Jelikož vzdálenost 20 metrů mezi vozidly jedoucími 50 km/h není z hlediska bezpečnosti to stejné jako při rychlosti 130 km/h, pracuje model s tzv. „časovou vzdáleností“, což je čas, za který se vozidlo vzadu dostane při zachování stejné rychlosti na aktuální pozici vozidla vepředu.

Jako doporučená vzdálenost se v modelu považuje časová vzdálenost mezi 1,9 a 2,1 sekundami, na základě doporučené vzdálenosti 2 sekund dané oddělením ministerstva do-

pravy České republiky BESIP [2]. Jakákoliv nižší hodnota je považována za kritickou vzdálenost a vozidlo je nuceno zpomalit, to stejné v případě, pokud je vzdálenost mezi vozidly menší než 3 metry. Časová vzdálenost mezi 1,9 a 2,6 sekundami je považována za blízkou vzdálenost. V této vzdálenosti samočinně řízené vozidlo provádí změny pruhu, nebo přizpůsobuje svoji vozidlu ve předu. Poté se vždy snaží držet v intervalu doporučené vzdálenosti. Časová vzdálenost vyšší 2,6 sekund je považována jako daleká a v této vzdálenosti samočinně řízené vozidlo neprovádí žádné akce. Posledním typem vzdálenosti, která se v modelu používá je tzv. „průjezdná“ vzdálenost, což je časová vzdálenost větší než 4,2 sekundy a referuje se k ní v případě, kdy se rozhoduje, zda je jízdní pruh průjezdný, viz 3.2.3. Následující tabulka shrnuje jednotlivé typy časových vzdáleností použitých v modelu.

| Označení vzdálenosti | Časová vzdálenost [s] |
|---|---|
| Kritická vzdálenost | $< 1,9$ |
| Doporučená vzdálenost | $1,9 - 2,1$ |
| Blízká vzdálenost | $1,9 - 2,6$ |
| Daleká vzdálenost | $> 2,6$ |
| Průjezdná vzdálenost | $> 4,2$ |

Tabulka 3.3: Typy časových vzdáleností použitých v modelu

Zmíněné typy časových vzdáleností používají k jejich výpočtu vzdálenost mezi vozidly „od nárazníku k nárazníku“. V modelu existuje ještě další typ vzdálenosti, tzv. „poziční vzdálenost“, což je rozdíl jejich pozice (vzdálenost mezi středy vozidel).

### 3.2.8 Konec simulace

Simulace končí po splnění jedné z následujících podmínek:

- Pozice samočinně řízeného vozidla překročí délku úseku.
- Samočinně řízené vozidlo je v kolizi s některým z ostatních vozidel.
- Rychlost samočinně řízeného vozidla je rovna nule alespoň dvě sekundy, což značí, že se vozidlo vyskytlo v koloně stojících vozidel.

# Kapitola 4

# Popis vytvořeného modelu

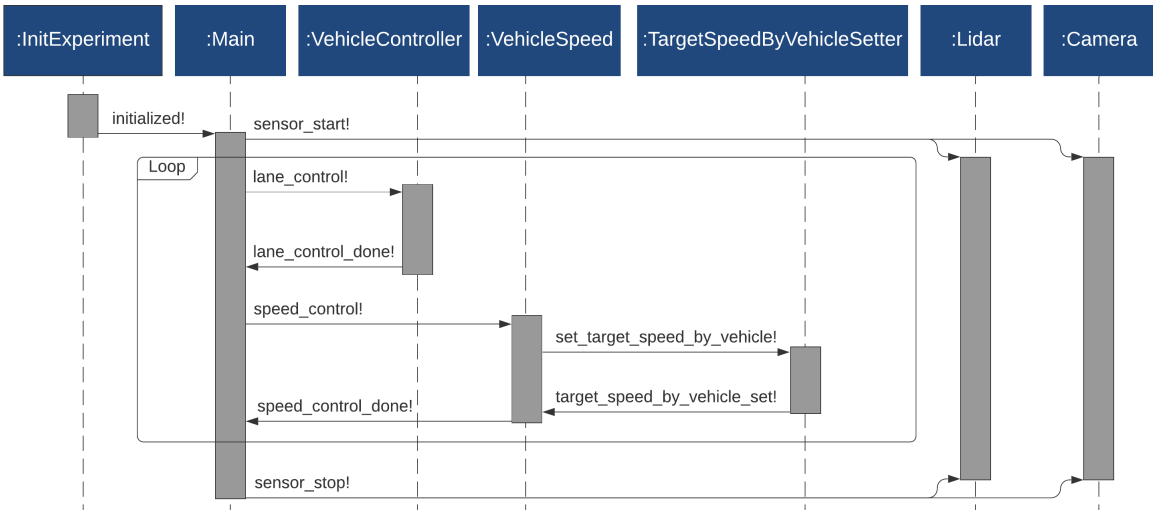
Tato kapitola popisuje implementaci modelu samočinně řízeného vozidla a jeho okolí.

Na implementaci byl zvolen nástroj UPPAAL (popsaný v sekci 3.1.1), konkrétně jeho verze Stratego umožňující statistický model checking a možnost zapojení strategií.

Následující sekce obsahují popisy jednotlivých šablon, jejich automatů a logiky na pozadí. Funkce a proměnné jednotlivých šablon jsou popsány v příloze C

## 4.1 Základní prvky

Jádrem modelu je centrální proces, který vznikne instancí šablony `Main` a obsahuje hlavní cyklus modelu. Tento proces nejdříve čeká na převzetí řízení od procesu šablony `InitExperiment` a poté, těsně před začátkem hlavního cyklu, spouští senzory implementované v šablonách `Lidar` a `Camera`, které pracují až do konce experimentu. Dále každou iterací cyklu dává řízení procesům ze šablon `VehicleController` a `VehicleSpeed`, které rozhodují o akcích samočinně řízeného vozidla, konkrétně o změnách jízdních pruhů a rychlosti. Proces šablony `VehicleSpeed` dále spouští pomocný proces šablony `TargetSpeedByVehicleSetter`, který slouží pro určení cílové rychlosti dané vozidlem, viz 3.2.5. Logika ostatních vozidel je řešena programově v centrálním procesu. Obrázek 4.1 obsahuje sekvenční diagram, který přibližuje synchronizaci mezi jednotlivými procesy modelu.




Obrázek 4.1: Synchronizace procesů modelu

### Vozidla a dopravní značky

Samočinně řízené vozidlo, ostatní vozidla a dopravní značky jsou uloženy ve stejném datovém typu `traffic_entity`. Struktura tvořící tento typ je popsána v tabulce 4.1. Strukturu uchovávající pozici entity zobrazuje tabulka 4.2.

| `struct traffic_entity` | | | |
|---|---|---|---|
| datový typ | identifikátor | význam (vozidlo) | význam (značka) |
| `int` | `id` | unikátní identifikátor entity daného typu | |
| `entity_position` | `position` | pozice | |
| `int` | `speed` | aktuální rychlost vozidla | omezující rychlost |
| `int` | `target_speed` | cílová rychlost vozidla | |
| `int` | `max_speed` | maximální rychlost vozidla[1] | |
| `int` | `last_speed_change` | hodnota poslední změny rychlosti vozidla | |
| `int[0,3]` | `entity_type` | typ entity (prázdná/samočinně řízené vozidlo/ostatní vozidla/značka) | |

Tabulka 4.1: Struktura `traffic_entity`

| `struct entity_position` | | | |
|---|---|---|---|
| datový typ | identifikátor | význam (vozidlo) | význam (značka) |
| `int` | `lane` | jízdní pruh | |
| `int` | `meters` | pozice v metrech | |
| `int` | `centimeters` | doplněk pozice v centimetrech | |

Tabulka 4.2: Struktura `position`

Na jednu stranu by se možná raději hodilo mít dopravní značky reprezentovány speciálním datovým typem, protože takto je několik položek struktury nevyužitých, na druhou stranu tento způsob implementace umožňuje používat stejné pomocné funkce nehledě na tom, jestli je daná věc, se kterou se manipuluje, vozidlo, či značka.

samočinně řízené vozidlo reprezentuje proměnná `main_vehicle`. Ostatní vozidla a značky jsou uspořádány do příslušných kolekcí v proměnných `vehicles`, resp. `speed_limit_signs`.

[1] U samočinně řízeného vozidla je tato hodnota vždy nastavena na maximální povolenou rychlost. U ostatních vozidel se pomocí této hodnoty provádí náhodná změna rychlosti.

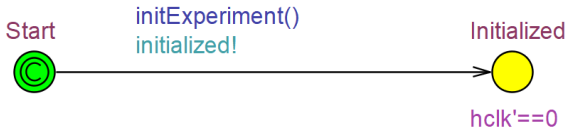
## 4.2 Popis jednotlivých šablon

Tato sekce popisuje implementaci jednotlivých šablon, konkrétně `InitExperiment`(4.2.1), `Main` (4.2.2), `Lidar` a `Camera` (4.2.3), `VehicleController` (4.2.4), `VehicleSpeed` (4.2.5) a `TargetSpeedByVehicleSetter` (4.2.6).

### 4.2.1 Inicializace experimentu

Šablona `InitExperiment` se stará o inicializaci prováděného experimentu. Při vytváření instance vyžaduje šablona čtyři parametry. Parametr `length_of_road` stanovuje délku úseku, na kterém se bude experiment provádět, `number_of_lanes` určuje počet jízdních pruhů na vozovce, `exp_no` je číslo experimentu, podle kterého se vybere určitý scénář, který upravuje výchozí pozice a rychlosti ostatních vozidel a také umístění značek úpravy rychlosti. Posledním je `random_speed_change`, který povoluje či zakazuje ostatním vozidlům, aby náhodně měnila svoji rychlost během experimentu.

Obrázek 4.2 zobrazuje automat šablony `InitExperiment`, který při přechodu ze stavu `Start` do stavu `Initialized` inicializuje výchozí stav samočinně řízeného vozidla a poté inicializuje celý experiment. Pomocí synchronizačního kanálu `initialized` předává řízení hlavnímu procesu, který vznikne instancí šablony `Main`. Dále se nastavují hybridní hodiny `hclk`, které jsou od verze 4.1.20.5 nutné pro korektní vykreslování grafů [14].



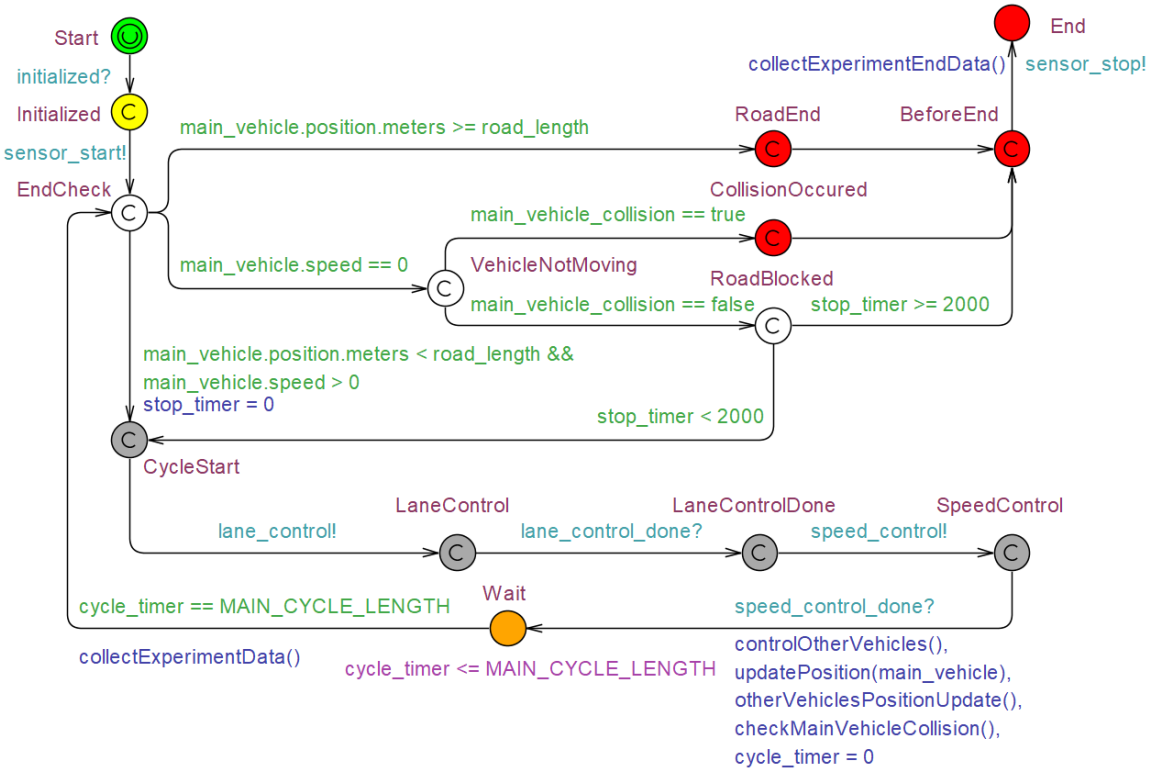

Obrázek 4.2: Automat šablony `InitExperiment`, který se stará o inicializaci modelu a spuštění hlavního cyklu simulace.

### 4.2.2 Hlavní cyklus

Šablona `Main` obsahuje automat zobrazený na obrázku 4.3, který tvoří hlavní programový cyklus modelu. Tento cyklus je nejdůležitějším prvkem celé simulace, jelikož předává řízení ostatním procesům, které se starají o kontrolu nad samočinně řízeným vozidlem.

Automat nejprve čeká na signál od inicializačního procesu, popsaného v sekci 4.2.1. Po jeho přijetí vysílá pomocí synchronizačního kanálu `sensor_start` signál procesům, které se starají o detekci informací o okolí samočinně řízeného vozidla (popsané v sekci 4.2.3). Poté začne probíhat hlavní cyklus modelu, kde se jako první vynulují hodiny `stop_timer`. Tyto hodiny uchovávají čas, po který vozidlo stojí na jednom místě. Poté se vyšle signál synchronizačním kanálem `lane_control` procesu, který obstarává přejezdy samočinně řízeného vozidla mezi pruhy vozovky a jakmile je daná obsluha dokončená, pošle se signál synchronizačním kanálem `speed_control` procesu, který upravuje rychlost samočinně řízeného vozidla. Další přechod aktualizuje ostatní vnitřní stavy, o které se nestará žádný samostatný proces. Funkce `controlOtherVehicles()` se postará o ovládání ostatních vozidel a funkce `updatePosition(main_vehicle)` provede posun samočinně řízeného vozidla na základě rychlosti. Posun ostatních vozidel se poté zajistí funkcí `otherVehiclesPositionUpdate()`. Následně funkce `checkMainVehicleCollision()` zjišťuje, zdali je samočinně řízené vozidlo v kolizi s některým z vozidel a nastavuje podle toho proměnnou `main_vehicle_collision`.

Nakonec se provede vynulování hodinové proměnné `cycle_timer`. Ve stavu `Wait` proces čeká po délku 100 milisekund, které nese konstanta `MAIN_CYCLE_LENGTH`. Přechodem zpět do stavu `EndCheck` se pomocí funkce `collectExperimentData()` posbírají různé statistické informace, které se poté dají vykreslovat do grafů.




Obrázek 4.3: Hlavní automat, který řídí celý průběh simulace a spouští ostatní procesy. Legenda: **zelená** – začátek simulace, **žlutá** – inicializováno, **šedá** – průběh hlavního cyklu, **oranžová** – čekání v cyklu, **bílá** – potenciální konec simulace, **červená** – konec simulace.

Před začátkem nové iterace cyklu se ve stavu `EndCheck` vždy testuje, zdali se nemá simulace ukončit a děje se tak při splnění jedné z podmínek popsaných v sekci 3.2.8. Jak bylo v zmíněné sekci řečeno, pokud je rychlost vozidla 0, nemusí to znamenat, že bude simulace hned končit. Simulace by v tomto případě končila, pokud vozidlo stojí na místě alespoň dvě sekundy (určené hodinovou proměnnou `stop_timer`). Opačně se vrací proces zpět do hlavního cyklu. Další možností ukončení simulace by bylo překročení délky úseku, což značí podmínka `main_vehicle.position.meters >= road_length` předcházející stav `RoadEnd`. Poslední možností je, že vznikla kolize samočinně řízeného vozidla s nějakým jiným vozidlem. Tuto informaci uchovává stavová proměnná `main_vehicle_collision`.

Pokud bylo opravdu rozhodnuto o konci simulace, přechodem ze stavu `BeforeEnd` do `End` se vyšle signál pro ukončení činnosti senzorů a funkcí `collectExperimentEndData` se provede sběr koncových statistik, jako například výpočet průměrné rychlosti vozidla.

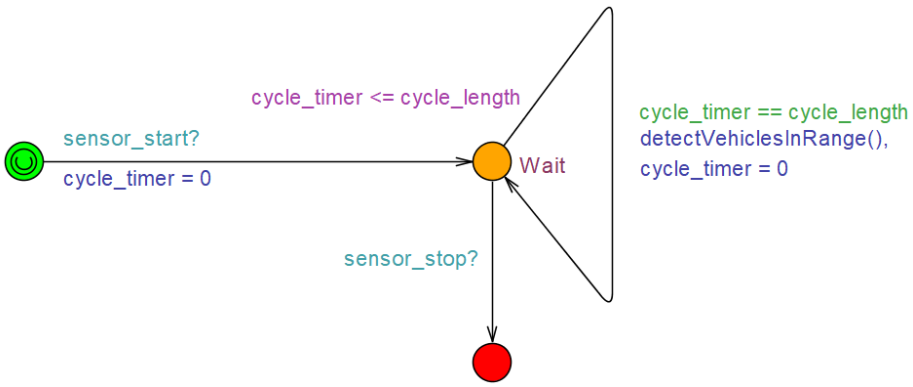
### 4.2.3 Senzory samočinně řízeného vozidla

Jak již bylo zmíněno v sekci 3.2.3, model samočinně řízeného vozidla obsahuje dva typy senzorů, pomocí kterých vozidlo získává informace o svém okolí.

### Lidar

Prvním je Lidar, který je implementován ve stejnojmenné šabloně. Vytvoření instance šablony je prováděno čtyřikrát – jednou pro každý směr. Výsledkem tak jsou čtyři procesy, které pracují nezávisle na sobě. Při instanci šablony jsou požadovány tři parametry. Parametr `direction` značící směr umístění, `range` určující vzdálenost od vozidla, kterou bude senzor pokrývat a parametr `cycle_length`, který nastavuje latenci senzoru.

Obrázek 4.4 ukazuje automat šablony `Lidar`, který nejprve čeká na signál značící na začátek simulace. Poté cyklí v časovém intervalu daným argumentem šablony `cycle_length`, kdy v každé iteraci cyklu volá funkci `detectVehiclesInRange()`. Tato funkce postupně prochází strukturu obsahující všechna ostatní vozidla `vehicles` a ověřuje, zdali jejich pozice zapadá do plochy, kterou senzor pokrývá. Detekovaná vozidla jsou kopírována do struktury dané argumentem `direction`, která se nachází v poli struktur `vehicles_in_range`.




Obrázek 4.4: Automat šablony `Lidar`, který se stará o detekci vozidel v daném směru.

### Kamera

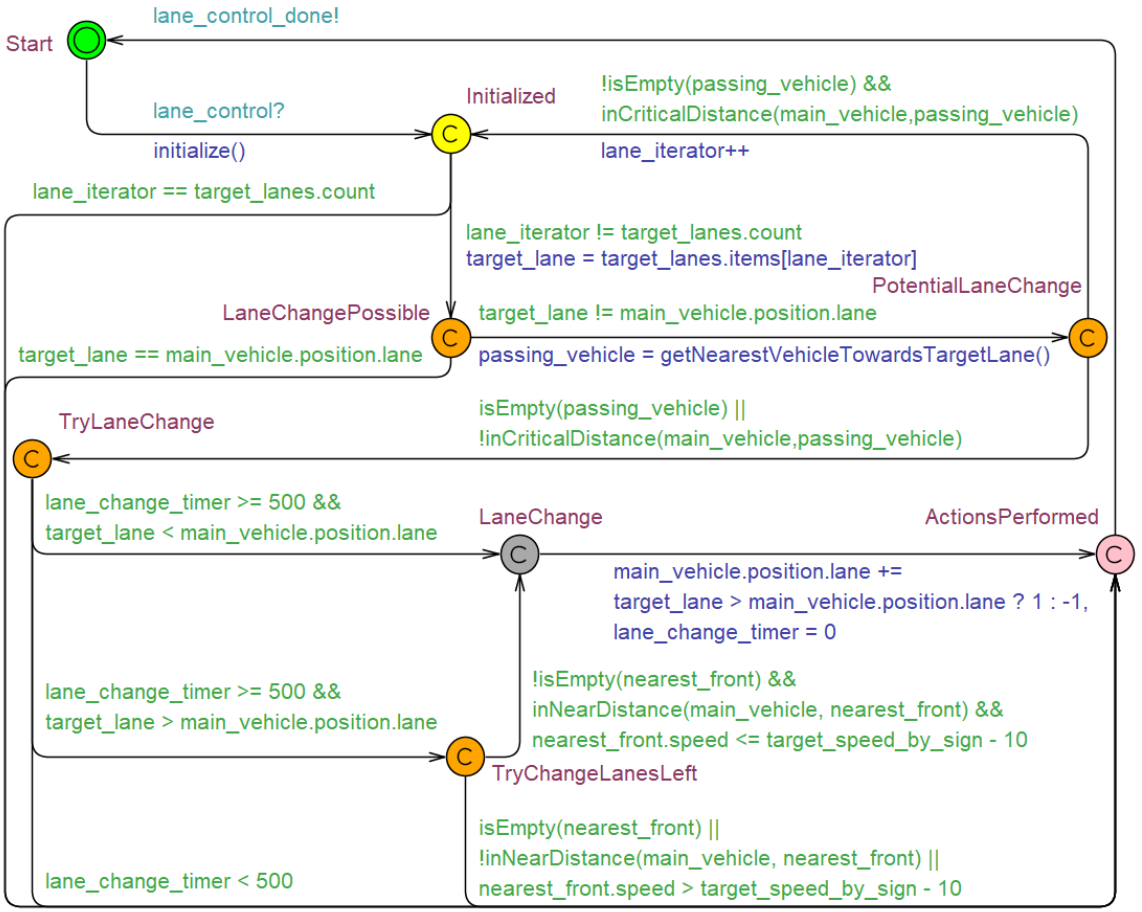
Dalším senzorem, kterým samočinně řízené vozidlo disponuje je kamera, která se snaží detekovat dopravní značení určující maximální povolenou rychlost. Šablona `Camera` řeší implementaci tohoto senzoru. Argumenty šablony jsou kromě směru stejné jako má šablona Lidaru – `range` a `cycle_length`. I automat je téměř identický tomu v šabloně Lidaru, co se liší je funkce volaná každou iterací, v tomto případě funkce `speedSignDetect()` detekuje dopravní značky v maximální vzdálenosti dané argumentem `range` a kopíruje je do struktury `speed_limit_signs_in_range`.

## 4.2.4 Přejezd samočinně řízeného vozidla mezi jízdními pruhy

O přejíždění samočinně řízeného vozidla mezi jízdními pruhy se stará šablona `VehicleController`. Tato šablona implementuje algoritmus popsaný v sekci 3.2.3.

Automat, který je možné vidět na obrázku 4.5 nejprve provádí inicializaci stavových proměnných a analyzuje shluk vozidel. Poté podle hodnoty proměnné `lane_iterator` buď přejde do stavu `ActionsPerformed` – to v případě, že byly vyčerpány všechny možnosti potenciálních cílových pruhů vozovky, nebo analogicky do stavu `LaneChangePossible`. Přechodem do zmíněného stavu se ze struktury potenciálních cílových pruhů vozovky `target_lanes` vybere cílový jízdní pruh a uloží do proměnné `target_lane`, do kterého se v dalších stavech bude vozidlo pokoušet přejet, pokud již v něm není – v tom případě přechází proces do stavu `ActionsPerformed`. Pokud vozidlo není v cílovém jízdním pruhu, přejde proces

do stavu `PotentialLaneChange` a do proměnné `passing_vehicle` vkládá vozidlo, které je nejblíže v sousedním jízdním pruhu ve směru k cílovému jízdnímu pruhu.




Obrázek 4.5: Automat šablony `VehicleController`, který obstarává přejezdy samočinně řízeného vozidla mezi pruhy.
Legenda: **zelená** – začátek procesu, **žlutá** – inicializováno, **šedá** – změna pruhu, **oranžová** – rozhodování o změně pruhu, **růžová** – akce provedeny.

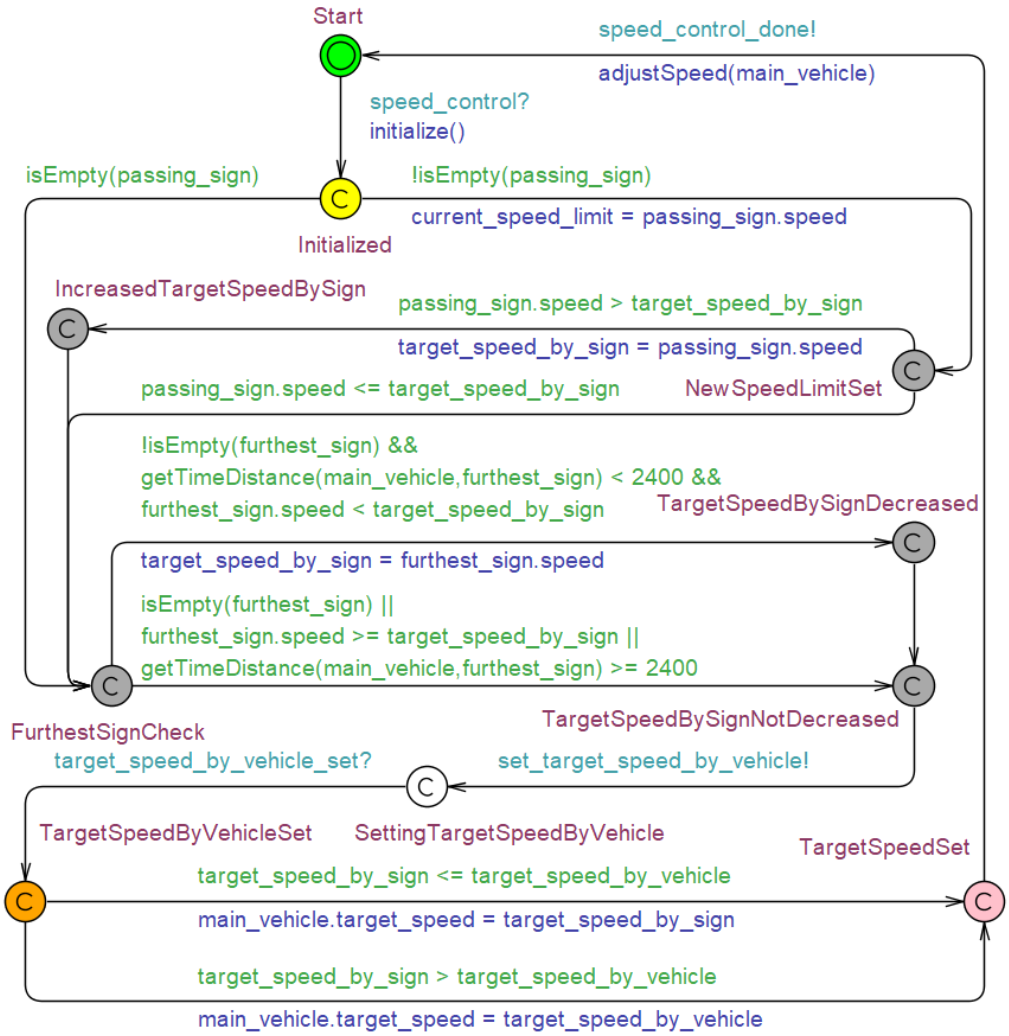
Pokud vozidlo v daném jízdním pruhu existuje, testuje se, zdali není v kritické vzdálenosti[2] – v tom případě by nebylo možné do jízdního pruhu přejet a vybíral by se další jízdní pruh ze struktury `target_lanes`. V případě, že vozidlo není v kritické vzdálenosti, přechází proces do stavu `TryLaneChange`. Ve stavu `TryLaneChange` se testuje hodnota hodinové proměnné `lane_change_timer`, která udává čas v milisekundách od poslední změny jízdního pruhu. Pokud je tato hodnota menší než 500, znamená to, že změna jízdního pruhu v poslední půl sekundě proběhla, tudíž další v tomto časovém intervalu již proběhnout nemůže a proces se přesouvá do stavu `ActionsPerformed`. V druhém případě se testuje, jestli se provádí změna jízdního pruhu doprava či doleva, nýbrž v druhé z těchto možností je navíc nutné znát informaci, zdali vozidlo, které se samočinně řízené vozidlo snaží předjet, jede rychlostí alespoň o 10 km/h nižší, než stanovuje aktuální nebo nadcházející dopravní značení. Pokud jede rychlostí, při které je povoleno předjíždění, proces přechází do stavu `LaneChange`, do kterého přechází i v případě, že vozidlo provádí změnu jízdního pruhu doprava. Z tohoto

[2]Kritická vzdálenost – časová vzdálenost menší než 1,9 sekundy, viz 3.2.7

stavu se provádí přechod do stavu `ActionsPerformed`, při kterém se změní hodnota jízdního pruhu `position.lane` ve struktuře samočinně řízeného vozidla `main_vehicle` a nuluje se hodnota hodinové proměnné `lane_change_timer`. Poté už následuje jenom přechod zpět do počátečního stavu.

### 4.2.5 Ovládání rychlosti samočinně řízeného vozidla

Jak již bylo vysvětleno v sekci 3.2.5, vozidlo upravuje svoji rychlost na základě atributu „cílová rychlost“, který se určí podle vozidla jedoucího v jízdním pruhu před samočinně řízeným vozidlem a aktuálním či budoucím rychlostním omezením. Právě o určení cílové rychlosti se stará šablona `VehicleSpeed`, jejíž automat je možné vidět na obrázku 4.6.




Obrázek 4.6: Automat šablony `VehicleSpeed`, který se stará o nastavení cílové rychlosti samočinně řízeného vozidla.
Legenda: **zelená** – začátek procesu, **žlutá** – inicializováno, **šedá** – nastavení cílové rychlosti dané značkou, **bílá** – nastavení cílové rychlosti dané vozidlem, **oranžová** – nastavení konečné cílové rychlosti, **růžová** – cílová rychlost nastavena.

Tento automat po obdržení signálu synchronizačním kanálem `speed_control` inicializuje potřebné proměnné a nastaví je na správné hodnoty. Jmenovitě se jedná o proměnnou `passing_sign`, která obsahuje strukturu reprezentující dopravní značku, kterou v daném momentě samočinně řízené vozidlo míjí, a dále proměnnou `furthest_sign`, která obsahuje strukturu značky, která je nejdále od vozidla.

Pokud vozidlo zrovna žádnou značku nemíjí, přesouvá se proces do stavu `FurthestSignCheck`, jinak se nastavuje proměnná aktuálního rychlostního limitu `current_speed_limit` na hodnotu rychlosti, kterou tato značka nese a proces jde do stavu `NewSpeedLimitSet`. V tomto stavu se určuje, zdali je rychlost, kterou značka určuje větší než dosavadní cílová rychlost určená značkou. Pokud tomu tak je, proměnná nesoucí cílovou rychlost určenou značkou `target_speed_by_sign` je v následujícím přechodu nastavena právě na tuto hodnotu. Ve stavu `FurthestSignCheck` se provádí test, zdali značka, která je nejdále od vozidla nese nižší rychlost než dosavadní hodnota proměnné cílové rychlosti určené značkou a dále musí také být značka vzdálena do 2,4 sekund časové vzdálenosti[3] od vozidla. Pokud tomu tak je, stane se tak novou hodnotou zmíněné proměnné. Tento krok zajišťuje, že vozidlo bude zpomalovat ještě před značkou, pokud nese nižší rychlost, než dovolovaly dosavadní limity.

Přechodem ze stavu `TargetSpeedBySignNotDecreased` do `SettingTargetSpeedByVehicle` se vysílá signál procesu, jenž je popsán v sekci 4.2.6, který se stará o nastavení hodnoty proměnné `target_speed_by_vehicle`, reprezentující cílovou rychlost danou nejbližším vozidlem, umístěným v jízdním pruhu před samočinně řízeným vozidlem. Po nastavení této hodnoty, je nastavena hodnota hlavní cílové rychlosti `main_vehicle.target_speed` buď na hodnotu cílové rychlosti danou vozidlem, nebo na hodnotu cílové rychlosti danou značkou, což jsou proměnné `target_speed_by_vehicle`, resp. `target_speed_by_sign`. Na přiřazení se vybere menší hodnota z těchto dvou hodnot. Přechodem zpět do počátečního stavu je volána funkce `adjustSpeed(main_vehicle)`, která implementuje algoritmus vysvětlený v sekci 3.2.6, jenž upraví aktuální rychlost na základě cílové rychlosti, která byla v tomto procesu nastavena.

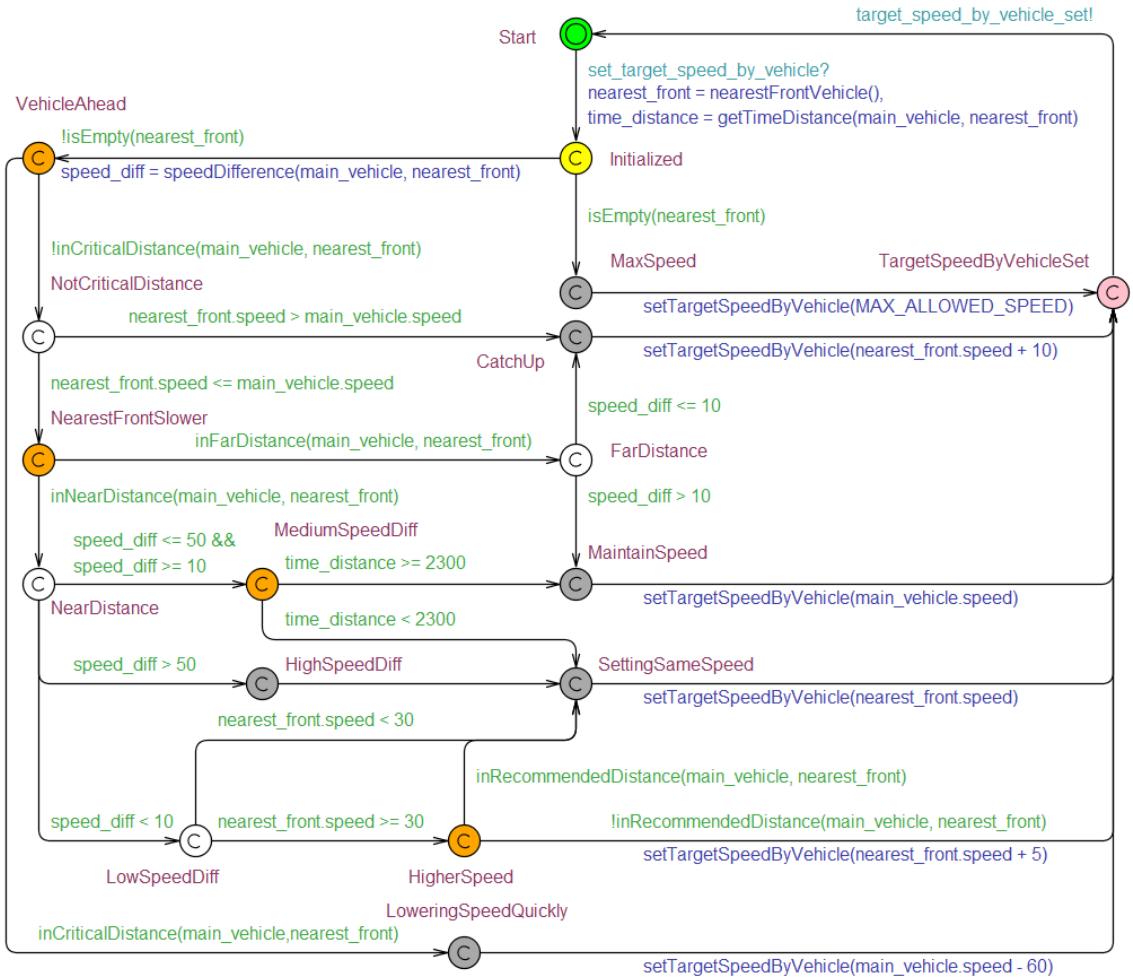
### 4.2.6 Nastavení cílové rychlosti dané vozidlem

O nastavení cílové rychlosti dané vozidlem (dále v této sekci jen „cílová rychlost“) se stará šablona `TargetSpeedByVehicleSetter`. Tato šablona úzce souvisí se šablonou popsanou v předchozí sekci, a to z toho důvodu, že proces vzniklý instancí šablony `TargetSpeedByVehicleSetter` funguje jako takový podproces procesu šablony `VehicleSpeed`. Hlavním cílem tohoto procesu je, aby se samočinně řízené vozidlo drželo v doporučené vzdálenosti[4] za vozidlem, které následuje.

Automat na obrázku 4.7 nejprve načítá do proměnné `nearest_front` nejbližší vozidlo před a zároveň ve stejném jízdním pruhu, jako je samočinně řízené vozidlo (dále v této sekci jen „nejbližší vozidlo“). Nastavena je také hodnota proměnné `time_distance`, která reprezentuje časovou vzdálenost mezi vozidly v milisekundách. Čistě na základě nejbližšího vozidla koná proces svoje další kroky. Pokud takové vozidlo není, proces přejde do stavu `MaxSpeed`, jinak přechází do stavu `VehicleAhead` a nastavuje hodnotu proměnné `speed_diff` na rozdíl rychlostí samočinně řízeného vozidla a nejbližšího vozidla. Dále se automat větví podle vzdálenosti nejbližšího vozidla. Pokud je samočinně řízené vozidlo v kri-

[3] Časová vzdálenost – čas, za který vozidlo vzadu ujede při zachování rychlosti aktuální vzdálenost mezi entitami, viz 3.2.7

[4] Doporučená vzdálenost – časová vzdálenost mezi 1,9 a 2,1 sekundami, viz 3.2.7




Obrázek 4.7: Automat šablony `TargetSpeedByVehicleSetter`, který se stará o nastavení cílové rychlosti dané nejbližším vozidlem.
Legenda: **zelená** – začátek procesu, **žlutá** – inicializováno, **oranžová** – rozhodování na základě vzdálenosti, **bílá** – rozhodování na základě rychlosti, **šedá** – nastavení cílové rychlosti dané vozidlem, **růžová** – cílová rychlost daná vozidlem nastavena.

tické vzdálenosti vůči nejbližšímu vozidlu, zvolí se přechod do stavu `LoweringSpeedQuickly`, kde je cílová rychlost samočinně řízeného vozidla výrazně snížena. Cílová rychlost je zde nastavena o 60 km/h menší, než aktuální rychlost z důvodu zajištění maximální decelerace (viz 3.2.6). Pokud nejbližší vozidlo není v kritické vzdálenosti, přejde proces do stavu `NotCriticalDistance`. Pokud je rychlost nejbližšího vozidla větší než rychlost samočinně řízeného vozidla, nastavuje se cílová rychlost o 10 km/h větší, než je rychlost nejbližšího vozidla, přechodem ze stavu `CatchUp` do stavu `TargetSpeedByVehicleSet`. V druhém případě pokračuje proces do stavu `NearestFrontSlower`.

Ze stavu `NearestFrontSlower` se v případě daleké vzdálenosti[5] nejbližšího vozidla přechází do stavu `FarDistance`, kde je větvení na základě rozdílu rychlostí vozidel. Pokud je rozdíl rychlostí větší než 10, je rychlost vozidla udržována. Pokud není, přejde se do stavu `CatchUp`. V případě, že se nejbližší vozidlo nachází blízké vzdálenosti[6], přejde proces ze

[5]Daleká vzdálenost – časová vzdálenost větší než 2,6 sekundy, viz 3.2.7
[6]Blízká vzdálenost – časová vzdálenost mezi 1,9 a 2,6 sekundami, viz 3.2.7

stavu `NearestFrontSlower` do stavu `NearDistance`, kde se průběh větví do tří různých přechodů na základě rychlostního rozdílu mezi oběma vozidly. V případě, že je rozdíl rychlostí větší než 50 km/h, nastavuje se cílová rychlost na rychlost nejbližšího vozidla. Pro případ, že je rychlostní rozdíl mezi padesáti a deseti kilometrů za hodinu se dále hledí na časovou vzdálenost mezi vozidly. Pokud je časová vzdálenost menší než 2,3 sekundy, cílová rychlost nastavena na rychlost nejbližšího vozidla. Jinak je cílová rychlost nastavena aktuální rychlost samočinně řízeného vozidla.

V případě, že je rozdíl rychlostí menší než 10 km/h, zohledňuje se rychlost nejbližšího vozidla. Pokud je vyšší nebo rovna 30 km/h, přejde se do stavu `HigherSpeed`. V případě, že je ale nižší, než 30 km/h, přechází se do stavu `SettingSameSpeed`, kde se nastaví cílová rychlost na rychlost nejbližšího vozidla.

Ve stavu `HigherSpeed` už jsou očekávány pouze malé změny rychlosti. Pokud je vozidlo v doporučené vzdálenosti, je cílová rychlost nastavena na rychlost nejbližšího vozidla. Pokud vozidlo není v doporučené vzdálenosti, nastavuje se hodnota cílové rychlosti o 5 km/h větší, než je aktuální rychlost samočinně řízeného vozidla, což umožňuje samočinně řízenému vozidlu dojet vozidlo, které následuje a dostat se tak do doporučené vzdálenosti.

Tyto malé změny rychlostí se neprovádí při rychlostech nižších než 30 km/h z důvodu, že i pouhá inkrementace rychlosti o 1 km/h dokáže výrazně změnit hodnotu časové vzdálenosti a tím pádem by mohla vzniknout situace, kdy by se rychlost nikdy neustálila a neustále přeskakovala mezi dvěma hodnotami. Toto ale zapříčiní, že vozidlo ustálí svoji rychlost v časové vzdálenosti 1,9 – 2,6 sekund, místo obvyklých 1,9 – 2,1 sekund.

Jelikož software UPPAAL nepodporuje v aktuální verzi proměnné s plovoucí desetinnou čárkou ve strukturách, jednou z možností, jak se zbavit tohoto problému by bylo použít další položku struktury, která by uchovávala doplněk rychlosti, podobně jako je použit doplněk metrů ve struktuře `position`.

# Kapitola 5

# Ověření vlastností modelu

Tato kapitola popisuje a vyhodnocuje experimenty, které byly provedeny za účelem ověřením funkčnosti a očekávaných vlastností modelu. Pokud není jinak uvedeno, samočinně řízené vozidlo vždy začíná simulaci na pozici 0 metrů v prvním jízdním pruhu, jede rychlostí 130 km/h a vozovka obsahuje 4 jízdní pruhy. Proměnná `hclk`, která je na konci každého `simulate` dotazu, je součástí z důvodu správného vykreslování grafů, v grafu ji ale není nutné zobrazovat.

## 5.1 Experimenty s konkrétními scénáři

V této části jsou popsány experimenty s scénáři, kde jsou ostatní vozidla umístěna na předem danou pozici a jedou předem danou rychlostí. Náhodná změna rychlosti ostatních vozidel je v tomto případě zakázána. Toto nám umožní ověřit reakce samočinně řízeného vozidla na konkrétní situace.

### 5.1.1 Předjíždění vozidel

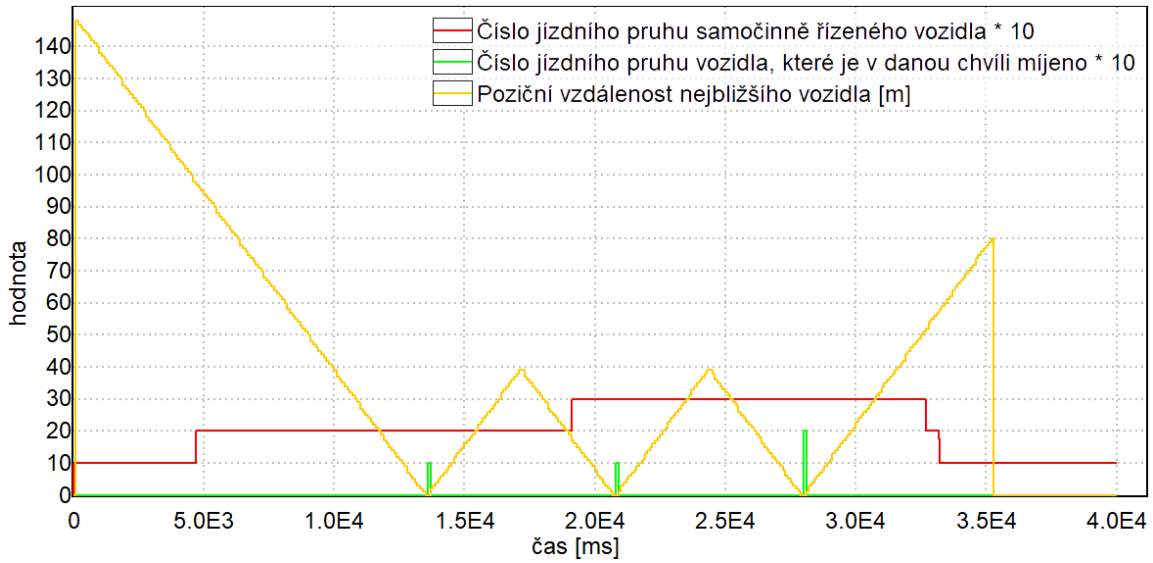
Následující experiment ukazuje, jak se samočinně řízené vozidlo zachová v situaci, kdy dojede čtyři vozidla jedoucí nižší rychlostí (90 km/h) a je možné je předjet.

První dvě vozidla jsou umístěna sériově – jedou tedy za sebou v pruhu č. 1. Zbylé dva vozy jsou paralelně – jedou podél sebe v sousedních pruzích č.1 a 2. Přesné pozice vozidel na začátku simulace lze znázornit množinou souřadnic $\{(150, 1), (230, 1), (310, 1), (310, 2)\}$, kde souřadnice $x$ značí pozici na vozovce v metrech a $y$ označení jízdního pruhu. Samočinně řízené vozidlo je tedy nuceno po přejetí do druhého pruhu změnit pruh ještě jednou. Toto lze ověřit pomocí dotazu 5.1, který spustí simulaci, jejíž výsledky mohou být vidět na obrázku 5.1.

Jak je vidno z grafu na obrázku 5.1, v momentě, kdy se samočinně řízené vozidlo dostane do blízké vzdálenosti[1] prvního vozidla, což je v rychlosti 130 km/h vzdálenost kolem 93 metrů, přejíždí do druhého jízdního pruhu. Jakmile se dostane do stejné vzdálenosti k vozidlu v druhém pruhu, přejíždí do třetího. Jakmile se dostane před poslední vozidlo a není již v jeho kritické vzdálenosti[2], vrací se zpět do prvního pruhu.

[1] Blízká vzdálenost – časová vzdálenost mezi 1,9 a 2,6 sekundami, viz 3.2.7

[2] Kritická vzdálenost - časová vzdálenost menší než 1,9 sekundy, viz 3.2.7




Obrázek 5.1: Graf, který zobrazuje situaci, kdy samočinně řízené vozidlo předjíždí čtyři vozidla, dvě umístěna sériově, dvě paralelně.

```
simulate 1 [<=40000] {(main_vehicle.position.lane+1)*10,
                      nearest_vehicle_distance, hclk,
 (passing_vehicle*(nearest_vehicle.position.lane+1))*10}
```
(5.1)

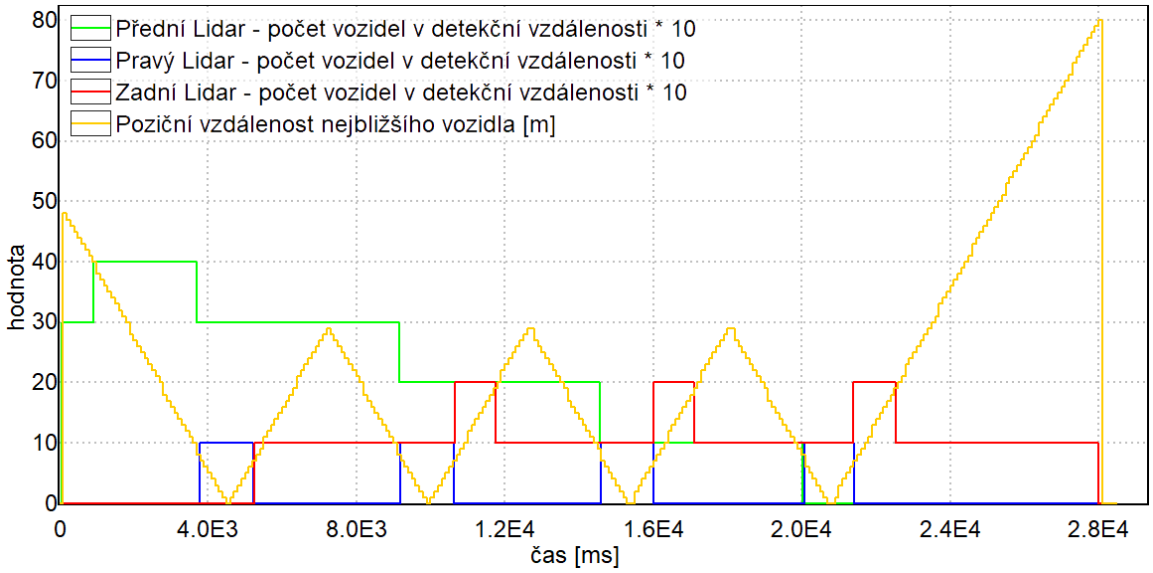
### 5.1.2 Lidar senzory

Uvažujme situaci, kdy samočinně řízené vozidlo jede v druhém pruhu vozovky. V prvním jedou čtyři vozidla za sebou rychlostí 90 km/h s rozestupem 60 metrů. Samočinně řízené vozidlo tedy v tomto případě předjíždí tento zástup vozidel.

Pomocí dotazu 5.2 lze generovat graf na obrázku 5.2. Graf ukazuje, kolik vozidel je v daném čase detekováno jednotlivými Lidar senzory. Plochy, které jednotlivé Lidar senzory pokrývají se nepřekrývají, což je viditelné i z grafu. Jakmile vozidlo přestane být detekováno předním Lidarem, začíná je okamžitě detekovat pravý Lidar, přesně toho chování může být viděno v čase kolem 4000 milisekund. Po opuštění dosahu pravého Lidaru v čase 5000 milisekund začíná být zaznamenáváno zadním Lidarem. Podle grafu je poziční vzdálenost[3] vozidla, v momentě, kdy je na hranici pokrytí mezi dvěma Lidar senzory, 7–8 metrů. Tuto informaci lze ověřit matematicky. Použitím vztahu 3.4 jako parametr $a$ rovnice přímky vznikne rovnice $\Delta y = \Delta x \cdot tan(\frac{\alpha}{2})$, kde $\Delta y$ je vzdálenost vozidel v rámci pruhů, $\alpha$ zorný úhel předního Lidaru a $\Delta x$ poziční vzdálenost, kterou hledáme. Po dosazení příslušných hodnot nám vyjde poziční vzdálenost $7,23$ metrů, což odpovídá hodnotám v grafu.

$$\begin{aligned} \Delta y &= \Delta x \cdot tan(\frac{\alpha}{2}) \\ 3 &= \Delta x \cdot tan(\frac{45}{2}) \\ 3 &= 0,414 \cdot \Delta x \\ \rightarrow \quad \Delta x &\doteq 7,23 \end{aligned}$$

[3] Poziční vzdálenost – rozdíl pozic entit, viz 3.2.7




Obrázek 5.2: Graf, který znázorňuje kolik vozidel je v daném čase detekováno jednotlivými Lidar senzory.

Z grafu lze také ověřit maximální dosah senzorů. V čase kolem 1000 milisekund, kdy přední Lidar začíná detekovat čtvrté (nejvzdálenější) vozidlo je samočinně řízené vozidlo vzdáleno 40 metrů od prvního (nejbližšího). Jelikož jsou jednotlivá vozidla od sebe vzdálena 60 metrů, sečtením vzdáleností mezi jednotlivými vozidly a vzdálenosti k nejbližšímu vozidlu nám vyjde 220 metrů, což je přesně maximální vzdálenost, kterou přední Lidar pokrývá.

```
simulate 1 [<=28500] {vehicles_in_range[FRONT].count*10,
                      vehicles_in_range[RIGHT].count*10,
                       vehicles_in_range[BACK].count*10,
                         nearest_vehicle_distance, hclk}
```
(5.2)

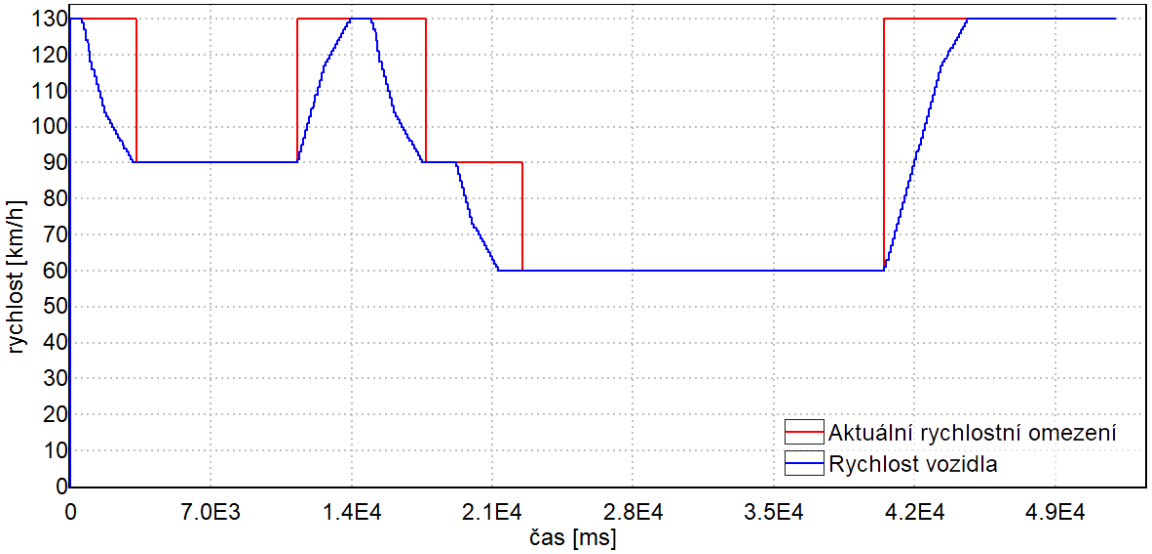
### 5.1.3 Upravení rychlosti podle rychlostního značení

Tento experiment ověřuje chování vozidla v situaci, kdy jsou podél vozovky umístěny značky upravující maximální povolenou rychlost vozidel. Umístění jednotlivých značek shrnuje následující tabulka:

| pozice [m] | rychlost [km/h] |
|---|---|
| 100 | 90 |
| 300 | X |
| 500 | 90 |
| 600 | 60 |
| 900 | X |

Tabulka 5.1: Vlastnosti dopravního značení upravující rychlost

Graf na obrázku 5.3 zobrazuje reakci samočinně řízeného vozidla na tyto značky. Rychlost vozidla (modrá barva) by nikdy neměla překročit aktuální rychlostní omezení (červená barva), což se samočinně řízenému vozidlu v tomto experimentu daří. Tento graf lze získat spuštěním simulace dotazem 5.3.




Obrázek 5.3: Graf, který zobrazuje reakci samočinně řízeného vozidla na dopravní značení upravující rychlost.

```
simulate 1 [<=52000] {current_speed_limit, main_vehicle.speed, hclk}
```
(5.3)

### 5.1.4 Reakce na kolonu vozidel

Tento experiment obsahuje situaci, kdy je od samočinně řízeného vozidla očekáváno upravení rychlosti kvůli neprůjezdné koloně vozidel. Experiment se dále dělí na dvě části.

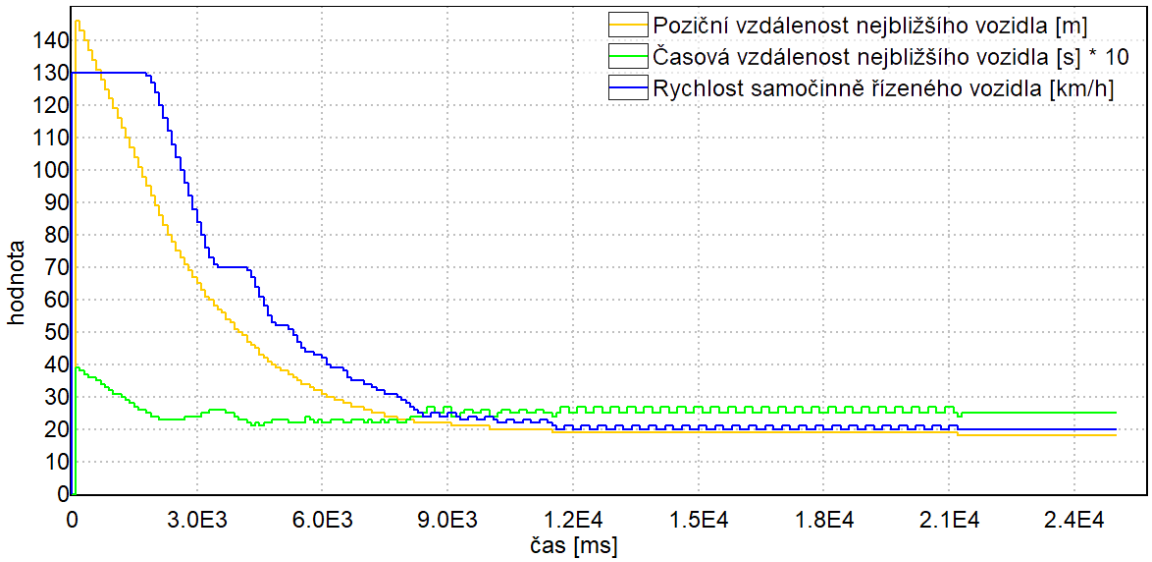
#### Reakce na kolonu jedoucí předem danou rychlostí

Tato část experimentu počítá s kolonou vozidel, která se pohybuje rychlostí 20 km/h a simulaci začíná na 100. metru vozovky.

Jak může být vidět z grafu na obrázku 5.4, jakmile se vozidlo dostane do blízké vzdálenosti nejbližšího vozidla kolony, začíná zpomalovat. Hodnota časové vzdálenosti[4] (světle zelená) neklesne pod doporučenou vzdálenost dvou sekund. Důvod proč rychlost vozidla mezi 12 a 21 sekundami simulace skáče mezi 20 a 21 km/h je, protože reálná hodnota této proměnně by byla někde v intervalu 20 a 21 a v aktuální verzi softwaru UPPAAL není možné používat proměnné s plovoucí čárkou ve strukturách. Časová vzdálenost se nakonec ustálí na hodnotě 2,6 sekund, jelikož nejbližší vozidlo jede rychlostí menší než 30 km/h[5].

[4] Časová vzdálenost – čas, za který vozidlo vzadu ujede při zachování rychlosti aktuální vzdálenost mezi entitami, viz 3.2.7

[5] Přesný důvod, proč se hodnota u rychlostí menších než je 30 km/h neustálí na doporučené vzdálenosti, byl popsán v sekci 4.2.6




Obrázek 5.4: Graf, který zobrazuje reakci samočinně řízeného vozidla na kolonu vozidel jedoucí 20 km/h.

Graf na obrázku 5.4 byl získán pomocí následujícího dotazu:

```
simulate 1 [<=25000] {nearest_vehicle_distance, main_vehicle.speed,
                      TargetSpeedByVehicleSetter.time_distance/100, hclk}
```
(5.4)

### Reakce na kolonu jedoucí náhodnou rychlostí

Jak mohlo být vidět v první části experimentu, časová vzdálenost u kolony jedoucí 20 km/h se vždy držela minimálně v doporučených hodnotách. Stejný výsledek ale nemusí platit v případech jiných rychlostí kolony. Místo toho, abychom manuálně zkoušeli jednu rychlost po druhé, tato část experimentu spouští simulací vícekrát s náhodnými hodnotami rychlostí kolony v intervalu 0 – 50 km/h a hledá pravděpodobnost, že časová vzdálenost je menší než doporučená. Výsledkem 11458 běhů simulace je pravděpodobnost v intervalu 0,74 – 0,84 %. Znamená to tedy, že v 0,79 % ± 0.05 běhů simulace klesne hodnota časové vzdálenosti pod 1,9 sekundy, což se označuje jako kritická vzdálenost. Tento výsledek byl získán pomocí dotazu 5.5. V příloze D je obrázek D.1, který zachycuje graf rozdělení pravděpodobnosti tohoto jevu.

```
Pr [<=30000] (<> TargetSpeedByVehicleSetter.time_distance < 1900 &&
                 TargetSpeedByVehicleSetter.time_distance > 0)
```
(5.5)

Fakt, že v cca 0,79 % případů se dostane vozidlo do kritické vzdálenosti, není žádoucí. Na druhou stranu, dokud se hodnota časové vzdálenosti nepřiblíží nule, lze tento test označit za úspěšný.

Chceme-li tuto skutečnost ověřit, spustíme dotaz modifikovaný tak, aby se tázal na pravděpodobnost klesnutí hodnoty časové vzdálenosti o něco méně než v posledním dotazu. Zvolíme tedy například 1,5 sekundy. Po spuštění dotazu 5.6 dostáváme jako výsledek pravděpodobnost v intervalu 0 – 0.09 % se standardně nastavenou mírou neurčitosti na $\varepsilon = 0.005$. Pokud dotaz spustíme znovu, tentokrát ale s mírou neurčitosti $\varepsilon = 0.0005$, dostaneme se výsledku v intervalu 0 – 0.009 %. Pokud je spodní hranice pravděpodobnosti 0,

jako bylo v posledních dvou případech, znamená to, že verifikátor nenašel v žádném běhu pravdivou odpověď na podmínku v dotazu (hodnota časové vzdálenosti v 299, ani v 2995 bězích simulace nikdy neklesla pod 1,5 sekundy).

```
Pr [<=30000] (<> TargetSpeedByVehicleSetter.time_distance < 1500 &&
                 TargetSpeedByVehicleSetter.time_distance > 0)
```
(5.6)

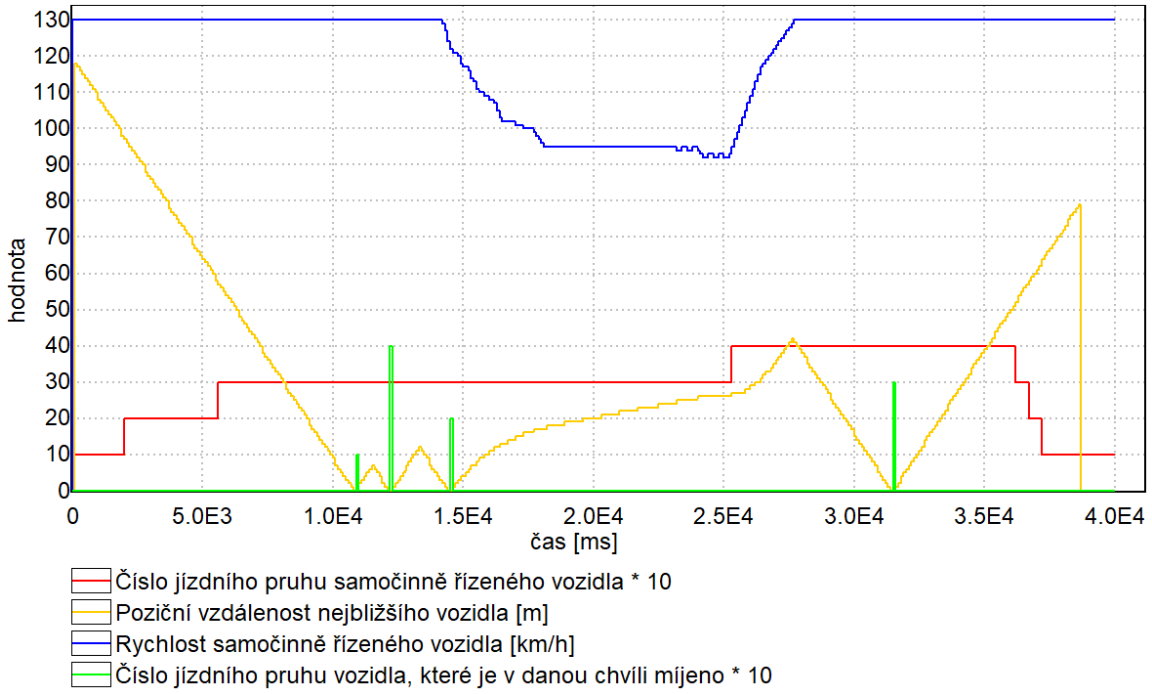
Výsledkem experimentu je tedy zjištění, že i když se vozidlo dostane do kritické vzdálenosti vozidla v koloně, časová vzdálenost mezi vozidly nebude s největší pravděpodobností nikdy nižší než 1,5 sekundy.

### 5.1.5 Průjezd shlukem vozidel

Tento experiment dává přímo do testu a algoritmus popsaný v sekci 3.2.3. Na vozovce jsou v této situaci rozmístěna čtyři vozidla tak, že každý pruh je obsazený. Mezi jednou dvojicí vozidel v sousedních pruzích je ale nechána mezera 110 metrů, což je dostatečná vzdálenost na projetí shluku. Pozice a rychlost vozidel v tomto experimentu shrnuje následující tabulka:

| označení pruhu | pozice [m] | rychlost [km/h] |
|---|---|---|
| 1 | 120 | 90 |
| 2 | 160 | 90 |
| 3 | 245 | 90 |
| 4 | 135 | 90 |

Tabulka 5.2: Pozice a rychlost vozidel na vozovce v tomto experimentu




Obrázek 5.5: Graf, který ukazuje průjezd samočinně řízeného vozidla shlukem vozidel.

Obrázek 5.5 zobrazuje graf, který ukazuje průjezd samočinně řízeného vozidla shlukem vozidel. Shluk vozidel byl vyhodnocen jako průjezdný, jelikož mezi vozidlem v 3. a 4. jízdním pruhu je dostatečná mezera na projetí. Vozidlo tedy přejíždí do třetího pruhu a poté zpomaluje, jelikož ještě nemůže přejet do čtvrtého jízdního pruhu, nýbrž by se tím dostalo do kritické vzdálenosti daného vozidla. Jakmile sníží svoji rychlost natolik, aby se dostalo do bližší poziční vzdálenosti k vozidlu v 3. pruhu, přejetím do čtvrtého pruhu by už nebylo v kritické vzdálenosti vozidla v tomto pruhu. Vozidlo tedy v této situaci přejíždí. Následně nabírá na rychlosti, jelikož se před ním už žádné vozidlo nenachází a poté se vrací zpět do prvního pruhu. Graf byl získán pomocí dotazu 5.7.

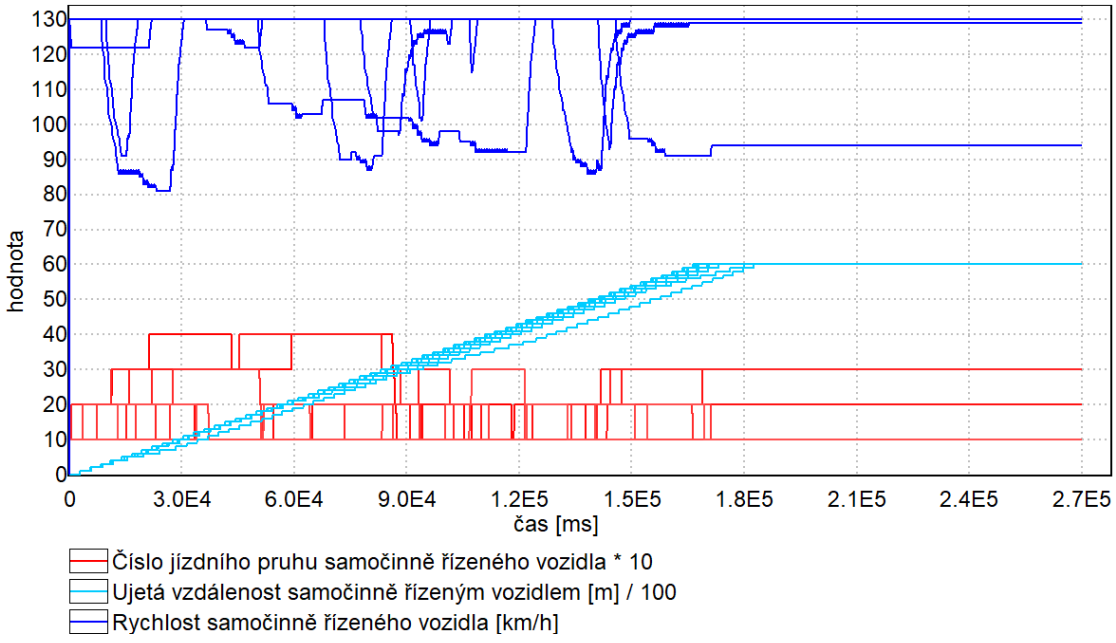
```
simulate 1 [<=40000] {(main_vehicle.position.lane+1)*10,
                nearest_vehicle_distance, main_vehicle.speed,
(passing_vehicle*(nearest_vehicle.position.lane+1))*10, hclk}
```
(5.7)

## 5.2 Experimenty s náhodnými scénáři

V těchto experimentech je sledován průjezd samočinně řízeného vozidla celým úsekem, který má délku 6000 metrů a obsahuje 4 jízdní pruhy. Na úseku je dále rozmístěno 30 ostatních vozidel. Pozice ostatních vozidel jsou dány náhodně s tím, že všechna jsou v první polovině úseku a je povolena náhodná změna rychlosti. Pruhy s nižším označením obsahují obecně více vozidel a vozidla v nich jedou v průměru nižší rychlostí než v pruzích s vyšším označením.

### 5.2.1 Rychlost a pohyb vozidla

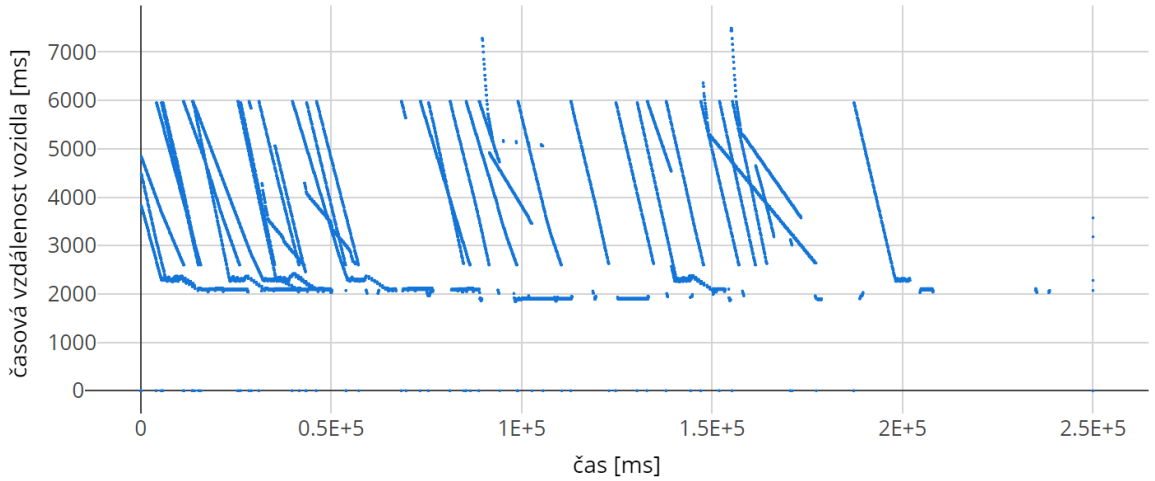
Tento experiment sleduje pohyb a rychlost vozidla na celém úseku. Graf na obrázku 5.6 získaný pomocí dotazu 5.8 zobrazuje 10 simulací průjezdu samočinně řízeného vozidla úsekem s náhodně rozmístěnými vozidly.




Obrázek 5.6: Vizualizace pohybu vozidla na úseku s náhodně rozmístěnými vozidly.

Dle červených linek znázorňující označení pruhu násobená deseti můžeme říci, že se vozidlo nejčastěji pohybuje v prvním, nebo druhém pruhu. Do čtvrtého pruhu přejíždí pouze výjimečně. Jeho rychlost (tmavě modrá linka) občas klesá k hodnotě 80 km/h, pravděpodobně z důvodu, že následuje pomalu jedoucí vůz a pruh, do kterého chce přejet blokuje jiné vozidlo. Světle modrá linka značí ujetou vzdálenost samočinně řízeného vozidla dělenou stem a můžeme podle ní říci, že čas, za který vozidlo nejčastěji projede celým úsekem je do tří minut. Průměrná rychlost vozidla během těchto deseti simulací činila 126,3 km/h. K této hodnotě se došlo za pomocí váženého průměru hodnot proměnné `average_speed`, která je nastavena vždy na konci simulace. Váhou byla v tomto případě časová délka dané simulace.

Dalším výstupem simulací pomocí dotazu 5.8 je graf na obrázku 5.7 zobrazující časovou vzdálenost k nejbližšímu vozidlu, které je před samočinně řízeným vozidlem ve stejném jízdním pruhu. Z grafu lze vidět, že pokud se vozidlo dostane v blízké vzdálenosti, linka buď mizí (samočinně řízené vozidlo přejelo do jiného pruhu), nebo se snižuje poměr jejího klesání v případě přizpůsobování rychlosti. Klesání se zastaví, v momentě, kdy se vozidlo dostane do doporučené vzdálenosti[6] a hodnota časové vzdálenosti neklesne nikdy neklesne pod 1,9 sekundy.




Obrázek 5.7: Graf časové vzdálenosti nejbližšího vozidla, které je před samočinně řízeným vozidlem ve stejném jízdním pruhu.

```
simulate 10 [<=270000] {(main_vehicle.position.lane+1)*10,
    main_vehicle.speed, main_vehicle.position.meters/100,
TargetSpeedByVehicleSetter.time_distance/10, average_speed, hclk}
```
(5.8)

### 5.2.2 Míra nehod

Hlavním ověřením bezpečnosti modelu samočinně řízeného vozidla je míra nehod, které v simulacích nastanou. Dotaz 5.9 po spuštění s koeficientem míry neurčitosti nastaveným na hodnotu 0.00025 vrátil pravděpodobnost v intervalu 0–0,0049 % – během 5990 simulací nenastala ani jedna havárie samočinně řízeného vozidla. Takovýto počet simulací nelze označit

[6] Doporučená vzdálenost – časová vzdálenost mezi 1,9 a 2,1 sekundami, viz 3.2.7

za příliš vysoký, zvláště pro ověření bezpečnosti vozidla, ale z důvodu časové náročnosti se již netestovalo s vyšší přesností.

```
Pr [<=270000] (<> Main.CollisionOccured)
```
(5.9)

### 5.2.3 Udržování vzdálenosti

V minulé sekci bylo prakticky ověřeno, že pravděpodobnost havárie vozidla je velmi nízká – téměř nulová, aktuální sekce na tuto informaci navazuje a zjišťuje, jak blízko se samočinně řízené vozidlo dostane k jinému vozidlu. Dotaz 5.10 zjistil, že v 46,1 – 51,1 % simulací ($\varepsilon = 0.025$) klesla hodnota časové vzdálenosti nejbližšího vozidla v pruhu před samočinně řízeným vozidlem pod 1,9 sekundy. Součástí přílohy D je obrázek D.2, který zobrazuje graf rozdělení pravděpodobnosti tohoto jevu.

```
Pr [<=270000] (<> TargetSpeedByVehicleSetter.time_distance < 1900 &&
                  TargetSpeedByVehicleSetter.time_distance > 0)
```
(5.10)

Jelikož se zjištěná pravděpodobnost neblíží nule, je nutno provést dotaz, zdali se časová vzdálenost dostane pod hodnotu 1,5. Na to slouží dotaz 5.11, který vrací pravděpodobnost v intervalu 0–0,99 % ($\varepsilon = 0.005$).

```
Pr [<=270000] (<> TargetSpeedByVehicleSetter.time_distance < 1500 &&
                  TargetSpeedByVehicleSetter.time_distance > 0)
```
(5.11)

## 5.3 Časová náročnost

Tento experiment má podobnou konfiguraci jako předchozí – náhodné rozmístění ostatních vozidel, které mohou náhodně měnit rychlost. Délka úseku ($s$) a počet vozidel na vozovce ($N$) se zde mění a zkoumá se, jaký mají tyto parametry vliv na časovou náročnost simulace. Změřené výsledky v sekundách zachycuje tabulka 5.3.

| $N$ \ $s[m]$ | 4000 | 6000 | 8000 | 10000 | 12000 |
|---|---|---|---|---|---|
| 20 | 0,71 | 1,02 | 1,34 | 1,63 | 2,01 |
| 30 | 1,11 | 1,55 | 2 | 2,52 | 3,09 |
| 40 | 1,53 | 2,21 | 3,01 | 3,7 | 4,37 |
| 50 | 2,02 | 2,98 | 4,01 | 4,89 | 6,03 |
| 60 | 2,66 | 3,82 | 5,34 | 6,39 | 7,77 |

Tabulka 5.3: Časová náročnost jedné simulace v sekundách v závislosti na délce úseku $s$ a počtu ostatních vozidel $N$.

Jelikož se v některých měřeních počítá až s 60 vozidly, byl počet jízdních pruhů navýšen na 6. Zvolená metoda testování byla v tomto případě skrze dotaz typu `simulate`, který byl pro každý typ měření nastaven, aby proběhl 10krát. Časová hodnota, která omezuje délku jednotlivých simulací byla nastavována v závislosti na délce úseku tak, aby každá simulace

v rámci stejné délky úseku měla stejný počet iterací hlavního cyklu. Časové omezení se tedy nastavilo vždy na dobu, za kterou samočinně řízené vozidlo projede celý úsek maximální možnou rychlostí. Délka těchto simulací byla pokaždé podělena deseti k dosažení průměrné doby trvání jedné simulace. Z naměřených hodnot je možné vyčíst, že pokud trojnásobně zvětšíme počet vozidel na stejném úseku, doba trvání simulace se zvětší v průměru 3,85krát. Pokud naopak ztrojnásobíme délku úseku při zachování počtu vozidel, zvýší se doba simulace v průměru 2,87krát. Z toho lze udělat závěr, že zvyšující se počet vozidel na vozovce má vyšší vliv na časovou náročnost simulace než úměrně se zvyšující délka úseku. Simulace byly provedeny na 6-vláknovém CPU Intel Core i5-8400 @ 2.8 GHz.

## 5.4 Shrnutí výsledků

Vlastnosti samočinně řízeného vozidla byly statisticky ověřeny v předchozích sekcích. Vozidlo bezpečně předjíždí ostatní vozy na vozovce, jak bylo otestováno v sekcích 5.1.1 a 5.1.5. Senzory automobilu v podobě Lidaru a kamery detekují dané objekty podle představ (5.1.2, resp. 5.1.3). Vozidlo také dokáže upravit rychlost a následovat kolonu vozidel jedoucí nízkou rychlostí, toto bylo ověřeno v sekci 5.1.4.

V sekci 5.2 bylo provedeno několik testů zabývající se průjezdem vozidla úsekem s náhodnými pozicemi ostatních vozidel, ze kterých vyšla pravděpodobnost havárie vozidla menší než 0,0049 %. Také zde bylo zjištěno, že se může samočinně řízené vozidlo přiblížit k jinému vozidlu na kratší vzdálenost, než je doporučená vzdálenost dvou sekund. Na druhou stranu, nikdy se nestalo, že by se vozidlo dostalo do vzdálenosti menší než 1,5 sekundy.

Poslední experiment (5.3) se zabýval zjišťováním časové náročnosti simulace v závislosti na délce úseku a počtu vozidel, kde bylo zjištěno, že delší úsek nepřidá tolik na strojovém času, jako větší počet vozidel.

# Kapitola 6

# Závěr

Cílem této práce bylo provést detailní rešerši v oblastech samočinně řízených vozidel a modelování systémů, na základě získaných znalostí navrhnout a implementovat model samočinně řízeného vozidla a jeho okolí a vlastnosti modelu poté ověřit v několika vhodných situacích.

Výsledkem praktické části práce je model samočinně řízeného vozidla s jeho okolím, vytvořen pomocí paralelně pracujících časovaných automatů v modelovacím, validačním a verifikačním nástroji UPPAAL, konkrétně jeho verzi Stratego. Okolí vozidla je v našem případě omezeno na úsek dálnice s libovolným počtem jízdních pruhů, po kterém se samočinně řízené vozidlo spolu s ostatními vozidly pohybuje. Model samočinně řízeného vozu dokáže adekvátně reagovat na běžné situace vzniklé ostatními vozidly provozu. Samočinně řízené vozidlo přejíždí mezi jízdními pruhy za účelem předjetí okolních vozidel a přizpůsobuje rychlost v závislosti na vozidlu které následuje, nebo na dopravním značením upravujícím rychlost. Ostatní vozidla provozu pouze mění svoji rychlost – buď náhodně, nebo za účelem zamezení kolize.

Zmíněné vlastnosti modelu byly ověřeny s pomocí verifikačního jazyka, který je součástí zvoleného nástroje. Pomocí těchto dotazů byl model otestován jak nad konkrétními scénáři, tak nad náhodnými, kde bylo samočinně řízené vozidlo zasazeno do úseku s náhodně rozmístěnými okolními vozidly, přičemž byl od něj očekáván bezpečný průjezd tímto úsekem. V těchto experimentech bylo zjištěno, že pravděpodobnost nehody samočinně řízeného vozidla při průjezdu daným úsekem je maximálně 0,0049 %.

Jelikož okolí vozidla představuje pouze omezený úsek dálnice, směr, kterým by tato práce mohla potenciálně pokračovat, by mohlo být posunutí těchto omezení a rozšíření okolí o klasické jednoproudové silnice, kde by samočinně řízené vozidlo muselo také počítat s vozidly jedoucími v protisměru. Další možností by mohla být pokročilejší implementace pohybu vozidla na základě fyzikálních jevů, a dále by se mohlo například předpokládat se zvýšenou kluzností vozovky při dešti nebo mrazu.

# Literatura


[1] *BeiDou Navigation Satellite System* [online]. [cit. 4. května 2020]. Dostupné z: http://en.beidou.gov.cn/.

[2] *Bezpečná vzdálenost*. BESIP [cit. 21. dubna 2020]. Dostupné z: https://www.ibesip.cz/tematicke-stranky/zasady-bezpecne-jizdy-v-aute/bezpecna-vzdalenost.

[3] *CODE AND MODEL EXPORT*. Dassault Systèmes [cit. 10. ledna 2020]. Dostupné z: https://www.3ds.com/products-services/catia/products/dymola/code-and-model-export/.

[4] *GPS Accuracy* [online]. Official U.S. government information about the Global Positioning System (GPS) and related topics [cit. 4. května 2020]. Dostupné z: https://www.gps.gov/systems/gps/performance/accuracy/.

[5] *GPS: The Global Positioning System* [online]. Official U.S. government information about the Global Positioning System (GPS) and related topics [cit. 4. května 2020]. Dostupné z: https://www.gps.gov/.

[6] *Simulink. Simulace a Model-Based Design.* HUMUSOFT [cit. 9. ledna 2020]. Dostupné z: https://www.humusoft.cz/matlab/simulink/.

[7] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Standard J3016. SAE International, červen 2018.

[8] *Police-reported motor vehicle traffic crashes in 2018 (Traffic Safety Facts Research Note)*. DOT HS 812 860. National Highway Traffic Safety Administration – National Center for Statistics and Analysis, listopad 2019. Dostupné z: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812860.

[9] *Waymo* [online]. Waymo LLC, 2019 [cit. 5. května 2020]. Dostupné z: https://waymo.com/.

[10] Amadeo, R. *Google's Waymo invests in LIDAR technology, cuts costs by 90 percent* [online]. Ars Technica, leden 2017 [cit. 24. listopadu 2019]. Dostupné z: https://arstechnica.com/cars/2017/01/googles-waymo-invests-in-lidar-technology-cuts-costs-by-90-percent/.

[11] David, A., Jensen, P. G., Larsen, K. G., Mikučionis, M. a Taankvist, J. H. Uppaal Stratego. In: Baier, C. a Tinelli, C., ed. *Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, s. 206–211. ISBN 978-3-662-46681-0.

[12] David, A. a Larsen, K. A Tutorial on Uppaal 4.0. Department of Computer Science, Aalborg University, Denmark: [b.n.]. Leden 2006.

[13] David, A., Larsen, K. G., Legay, A., Mikučionis, M. a Poulsen, D. B. Uppaal SMC tutorial. *International Journal on Software Tools for Technology Transfer*. Aug 2015, roč. 17, č. 4, s. 397–415. Dostupné z: https://doi.org/10.1007/s10009-014-0361-y. ISSN 1433-2787.

[14] Gardáš, M. *Výpočetní model a analýza samočinně řízeného vozidla*. Brno, 2018. Diplomová práce. Vysoké učení technické v Brně. Fakulta informačních technologií. Vedoucí práce Josef STRNADEL.

[15] Gerd Behrmann, A. D. E. F. K. G. L. D. L. *Uppaal Tiga User-manual*.

[16] Gert Rudolph, U. V. *Three Sensor Types Drive Autonomous Vehicles* [online]. FierceElectronics, listopad 2017 [cit. 29. listopadu 2019]. Dostupné z: https://www.fierceelectronics.com/components/three-sensor-types-drive-autonomous-vehicles.

[17] HADJ BACHIR, M. a De Souza, P. *LIDAR sensor simulation in adverse weather condition for driving assistance development*. leden 2019. Working paper or preprint. Dostupné z: https://hal.archives-ouvertes.fr/hal-01998668.

[18] Hall, S. *Elon Musk says that the LIDAR Google uses in its self-driving car 'doesn't make sense in a car context'* [online]. 9to5Google, říjen 2016 [cit. 28. listopadu 2019]. Dostupné z: https://ww.9to5google.com/2015/10/16/elon-musk-says-that-the-lidar-google-uses-in-its-self-driving-car-doesnt-make-sense-in-a-car-context/.

[19] Horton, M. *7 Reasons Your Life Depends on an Accurate IMU (Inertial Measurement Unit) in a Self-Driving Car* [online]. Medium, červen 2018 [cit. 28. listopadu 2019]. Dostupné z: https://medium.com/@mikehorton/7-reasons-your-life-depends-on-an-accurate-imu-inertial-measurement-unit-in-a-self-driving-car-75298d5cff9e.

[20] IAC. *IAC INFORMATION AND ANALYSIS CENTER FOR POSITIONING, NAVIGATION AND TIMING* [online]. 2005 [cit. 4. května 2020]. Dostupné z: https://www.glonass-iac.ru/en/index.php.

[21] Iovescu, C. a Rao, S. The fundamentals of millimeter wave sensors. *Texas Instruments, SPYY005*. [online]. 2017. Dostupné z: http://www.ti.com/lit/wp/spyy005/spyy005.pdf.

[22] Jarvis, A. *Guide to LiDAR Wavelengths* [online]. Velodyne Lidar, listopad 2018 [cit. 24. listopadu 2019]. Dostupné z: https://velodynelidar.com/newsroom/guide-to-lidar-wavelengths/.

[23] Jeyachandran, S. *Introducing the 5th-generation Waymo Driver: Informed by experience, designed for scale, engineered to tackle more environments* [online]. Waymo LLC, březen 2020 [cit. 5. května 2020]. Dostupné z: https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html.

[24] Lambert, F. *Tesla claims to have 'world's most advanced computer for autonomous driving' with Autopilot 3.0 update coming next year* [online]. Electrek, srpen 2018 [cit. 6. května 2020]. Dostupné z: https://electrek.co/2018/08/01/tesla-chip-most-advanced-computer-autonomous-driving-autopilot-hardware-3-update/.

[25] Lambert, S. *5 top Autonomous Vehicle Companies to watch in 2020* [online]. MES Insights, březen 2020 [cit. 7. května 2020]. Dostupné z: https://www.mes-insights.com/5-top-autonomous-vehicle-companies-to-watch-in-2020-a-910825/.

[26] Lang, B. *An Introduction to Positional Tracking and Degrees of Freedom (DOF)* [online]. Road to VR, únor 2013 [cit. 28. listopadu 2019]. Dostupné z: https://www.roadtovr.com/introduction-positional-tracking-degrees-freedom-dof/.

[27] Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D. et al. Towards fully autonomous driving: Systems and algorithms. In: IEEE. *2011 IEEE Intelligent Vehicles Symposium (IV)*. 2011, s. 163–168.

[28] Litman, T. *Autonomous Vehicle Implementation Predictions: Implications for Transport Planning*. Victoria Transport Policy Institute, 2013. DesLibris: Documents collection. Dostupné z: https://books.google.cz/books?id=G3FKnwEACAAJ.

[29] Marshall, A. *Elon Musk Promises a Really Truly Self-Driving Tesla in 2020* [online]. WIRED, únor 2019 [cit. 6. května 2020]. Dostupné z: https://www.wired.com/story/elon-musk-tesla-full-self-driving-2019-2020-promise/.

[30] Peringer, P. *Popis simulační knihovny SIMLIB* [online]. 1997 [cit. 9. ledna 2020]. Dostupné z: https://www.fit.vutbr.cz/~peringer/SIMLIB/doc/html-cz/.

[31] Peringer, P. *Modelování a simulace. Studijní opora.* VUT, 2012.

[32] Porter, J. *Tesla just made its one millionth car* [online]. The Verge, březen 2020 [cit. 6. května 2020]. Dostupné z: https://www.theverge.com/2020/3/10/21172895/tesla-one-million-cars-production-model-y.

[33] Rosique, F., Navarro Lorente, P., Fernandez, C. a Padilla, A. A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research. *Sensors*. Únor 2019, roč. 19, s. 648.

[34] Singh, S. *Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey (Traffic Safety Facts)*. DOT HS 812 115. National Highway Traffic Safety Administration, únor 2015. Dostupné z: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115.

[35] Walz, E. *Baidu Announces 'Apollo Lite' a Camera-based Autonomous Driving Technology* [online]. FutureCar, červen 2019 [cit. 7. května 2020]. Dostupné z: https://www.futurecar.com/3297/Baidu-Announces-Apollo-Lite-a-Camera-based-Autonomous-Driving-Technology.

[36] Wikipedia. *Lidar — Wikipedia, The Free Encyclopedia* [online]. 2019 [cit. 24. listopadu 2019]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Lidar&oldid=925009442.

[37] Wikipedia. *Radar — Wikipedia, The Free Encyclopedia* [online]. 2019 [cit. 26. listopadu 2019]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Radar&oldid=927497591.

[38] Wikipedia. *Velodyne LiDAR — Wikipedia, The Free Encyclopedia* [online]. 2019 [cit. 24. listopadu 2019]. Dostupné z: http://en.wikipedia.org/w/index.php?title=Velodyne%20LiDAR&oldid=925368556.

# Příloha A

# Obsah paměťového média

Paměťové médium přiložené k této práci má následující adresářovou strukturu:

- `./doc/` – zdrojové soubory technické zprávy
- `./tool/` – nástroj UPPAAL Stratego verze 4.1.20-7
- `./src/` – adresář obsahující zdrojový soubor modelu ve formátu .xml
- `./xcurda02-Samocinne-rizeni-vozidel.pdf` – dokument technické zprávy

# Příloha B

# Návod ke spuštění experimentů

Součástí modelu je 15 předem připravených experimentů. Některé z nich byly představeny v kapitole 5.

V záložce verifikátoru jsou připraveny všechny dotazy použité v kapitole 5. V komentáři u jednotlivých dotazů je zmíněné číslo dotazu, tak jak je uvedeno v dokumentu. Dále je tam zmíněno číslo experimentu, které je nutné zadat jako třetí argument instance šablony `InitExperiment` v systémové deklaraci. Ostatní tři argumenty je nutné správně vyplnit podle daného experimentu (délka úseku, počet jízdních pruhů, náhodná změna rychlosti ostatních vozidel).

Kromě dotazů použitých v kapitole 5, je ve verifikátoru připraven obecný dotaz vhodný pro všechny experimenty.

# Příloha C

# Popis významných funkcí a proměnných

### Globální deklarace

#### Konstanty

`MAX_LANE_COUNT` maximální možný počet jízdních pruhů

`MAX_VEHICLE_COUNT` maximální možný počet ostatních vozidel

`MAX_SPEED_LIMIT_SIGNS` maximální možný počet značek upravujících rychlost

`MAX_ALLOWED_SPEED, MIX_ALLOWED_SPEED` maximální/minimální povolená rychlost

`MAIN_CYCLE_LENGTH` časová délka hlavního cyklu v milisekundách

`MAX_ACC, MIN_ACC` maximální/minimální hodnota změny rychlosti vozidla

`FRONT, LEFT, RIGHT, BACK` směrové konstanty

`OTHER_VEHICLE_VISION_RANGE` dosah vize ostatních vozidel

`EMPTY, SELF_DRIVING_VEHICLE, OTHER_VEHICLE, SPEED_LIMIT_SIGN` typy entit (prázdná, samočinně řízené vozidlo, ostatní vozidla, dopravní značka)

`NEAR_LOWER_BOUND` modifikátor doporučené vzdálenosti – spodní hranice blízké vzdálenosti

`RECOMMENDED_UPPER_BOUND` modifikátor doporučené vzdálenosti – horní hranice doporučené vzdálenosti

`NEAR_UPPER_BOUND` modifikátor doporučené vzdálenosti – horní hranice blízké vzdálenosti

`PASSABLE_LOWER_BOUND` modifikátor doporučené vzdálenosti – spodní hranice průjezdné vzdálenosti

#### Proměnné

`main_vehicle` samočinně řízené vozidlo

`vehicles` kolekce ostatních vozidel

`speed_limit_signs` kolekce dopravních značek

`vehicles_in_range` pole kolekcí ostatních vozidel v dosahu vize samočinně řízeného vozidla

`speed_limit_signs_in_range` kolekce dopravních značek v dosahu vize samočinně řízeného vozidla

`lane_count` počet jízdních pruhů

`road_length` délka úseku

`other_vehicles_random_speed_change` stav povolené náhodné změny rychlosti ostatních vozidel

`target_speed_by_sign` cílová rychlost daná značkou – pomocná proměnná pro samočinně řízené vozidlo

`target_speed_by_vehicle` cílová rychlost daná vozidlem – pomocná proměnná pro samočinně řízené vozidlo

`current_speed_limit` aktuální rychlostní omezení – pomocná proměnná pro samočinně řízené vozidlo

`passing_vehicle` proměnná udávající, zdali je zrovna předjížděno vozidlo – stavová proměnná experimentu

`last_passing_vehicle` poslední vozidlo, které bylo přejížděno – stavová proměnná experimentu

`average_speed` průměrná rychlost samočinně řízeného vozidla – stavová proměnná experimentu

`nearest_vehicle` nejbližší vozidlo – stavová proměnná experimentu

`nearest_vehicle_distance` poziční vzdálenost nejbližšího vozidla nehledě na jízdním pruhu – stavová proměnná experimentu

`main_vehicle_old, vehicles_old, nearest_vehicle_old` proměnné uchovávající stav jejich protějšků bez přípony „_old“ v minulé iteraci hlavního cyklu

### Hodinové proměnné

`hclk` hybridní hodiny, které pomáhají při vykreslování grafů

`cumulative_time` časová proměnná uchovávající aktuální délku simulace

### Synchronizační kanály

`initialized` konec inicializace experimentu

`sensor_start` spuštění senzorů samočinně řízeného vozidla

`sensor_stop` zastavení senzorů samočinně řízeného vozidla

`speed_control` začátek změny rychlosti samočinně řízeného vozidla

`speed_control_done` dokončení změny rychlosti samočinně řízeného vozidla

`lane_control` začátek změny pruhů samočinně řízeného vozidla

`lane_control_done` dokončení změny pruhů samočinně řízeného vozidla

`set_target_speed_by_vehicle` začátek určení změny rychlosti dané vozidlem

`target_speed_by_vehicle_set` dokončení určení změny rychlosti dané vozidlem

### Funkce

`fPos(e)` výpočet pozice entity `e` v datovém typu `double`

`inFrontOf(e1,e2)` rozhodnutí, zdali je entita `e2` fyzicky umístěna před entitou `e1`

`clearEntity(e)` převod entity `e` na prázdnou entitu

`isEmpty(e)` vyhodnocení, zdali je entita `e` prázdná

`adjustSpeed(e)` poupravení rychlosti vozidla `e` na základě jeho cílové rychlosti

`distance(e1, e2)` výpočet a navrácení poziční vzdálenosti mezi entitami `e1` a `e2`

`bumperToBumperDistance(e1, e2)` výpočet a navrácení vzdálenosti mezi entitami `e1` a `e2` od nárazníku k nárazníku

`getTimeDistance(e1, e2)` výpočet času, za který vozidlo vzadu dojede na aktuální pozici vozidla ve předu

`getDistanceByTime(time_sec, e1, e2)` výpočet vzdálenosti v metrech, kterou vozidlo vzadu ujede při zachování aktuální rychlosti za čas daný parametrem `time_sec`

`getRecommendedDistance(e1, e2)` výpočet bezpečné vzdálenosti vozidla vzadu při zachování aktuální rychlosti; hodnota nikdy neklesne pod 3 metry

`inCriticalDistance(e1, e2)` rozhodnutí, zdali jsou entity `e1` a `e2` v kritické vzdálenosti

`inNearDistance(e1, e2)` rozhodnutí, zdali jsou entity `e1` a `e2` v blízké vzdálenosti

`inRecommendedDistance(e1, e2)` rozhodnutí, zdali jsou entity `e1` a `e2` v doporučené vzdálenosti

`inFarDistance(e1, e2)` rozhodnutí, zdali jsou entity `e1` a `e2` v daleké vzdálenosti

`inPassableDistance(e1, e2)` rozhodnutí, zdali jsou entity `e1` a `e2` v průjezdné vzdálenosti

`nearestFrontVehicle(direction)` výpočet nejbližšího vozidla vůči samočinně řízenému vozidlu, které je ve stejném jízdním pruhu a před vozidlem

`inCollision(e1,e2)` rozhodnutí, zdali jsou entity `e1` a `e2` mezi sebou v kolizi

`isColliding(e)` zkouška, zdali je entita `e` v kolizi s některým z ostatních vozidel

## Šablona InitExperiment

### Funkce

`normalDistGenerator(mean, sd, a, b)` generátor celočíselných hodnot v normální distribuci omezené intervalem (`a`,`b`)

`inCriticalDistanceInLaneAny(e)` vyhodnocení, zdali je entita `e` v kritické vzdálenosti jiného vozidla ve stejném pruhu

`randomVehicleSpawn(count)` náhodné rozmístění ostatních vozidel na vozovku, počet dán parametrem `count`

`spawnNewVehicle(lane, pos, initial_speed, max_speed)` vytvoření a umístění nového vozidla

`spawnNewSign(pos, speed)` vytvoření a umístění nové značky upravující rychlost

`initMainVehicle(lane, pos, initial_speed)` vytvoření a umístěné samočinně řízeného vozidla

`initExperiment()` inicializace experimentu

### Šablona Main

#### Proměnné

`main_vehicle_collision` proměnná uchovávající informaci, zdali samočinně řízené vozidlo havarovalo

#### Hodinové proměnné

`cycle_timer` časovač délky hlavního cyklu

`stop_timer` proměnná uchovávající čas, po který samočinně řízené vozidlo stojí na místě

#### Funkce

`otherGetLowestSpeedLimitInRange(e)` zjištění nejnižší hodnoty rychlostního limitu v dosahu vize vozidla `e` (určeno pro ostatní vozidla)

`otherGetCurrentSpeedLimit(e)` zjištění aktuální hodnoty rychlostního limitu pro vozidlo v proměnné `e` (určeno pro ostatní vozidla)

`otherGetTargetSpeedBySign(e)` výpočet cílové rychlosti dané značkou pro vozidlo v proměnné `e` (určeno pro ostatní vozidla)

`otherGetNearestFrontVehicle(e)` vyhodnocení nejbližšího vozidla ve stejném pruhu a před vozidlem v proměnné `e` (určeno pro ostatní vozidla)

`otherSetVehicleTargetSpeed(e)` nastavení cílové rychlosti vozidla v proměnné `e` (určeno pro ostatní vozidla)

`otherRandomAdjustMaxSpeed(e)` náhodná změna rychlosti vozidla `e` (určeno pro ostatní vozidla)

`updatePosition(e)` aktualizace hodnot pozice vozidla `e` na základě aktuální rychlosti

`collectPassingVehicleData()` sběr informací o vozidle, které samočinně řízené vozidlo aktuálně míjí

`getNearestVehicle()` zjištění a navrácení vozidla, které je pozičně nejblíže samočinně řízenému vozidlu, nehledě na pruhu

`checkMainVehicleCollision()` vyhodnocení, zdali samočinně řízené vozidlo havarovalo

`otherVehiclesPositionUpdate()` cyklus, který aktualizuje hodnoty pozice všech ostatních vozidel

`controlOtherVehicles()` ovládání ostatních vozidel

`collectExperimentEndData()` sběr dat na konci experimentu

`collectExperimentData()` sběr dat experimentu (volá se každou iterací hlavního cyklu)

### Šablona Lidar

#### Konstanty

`M_PI` konstanta $\pi$

#### Hodinové proměnné

`cycle_timer` časovač délky cyklu

#### Funkce

`detectVehiclesInRange()` detekce vozidel v dosahu Lidaru samočinně řízeného vozidla

### Šablona Camera

#### Hodinové proměnné

`cycle_timer` časovač délky cyklu

#### Funkce

`sortSignsInRange()` uspořádání kolekce vozidel v dosahu podle vzdálenosti (insertion sort)

`speedPenalty()` výpočet jak se dosah senzoru sníží na základě rychlosti samočinně řízeného vozidla

`speedSignDetect()` detekce značek upravující rychlost v dosahu samočinně řízeného vozidla

### Šablona VehicleController

#### Proměnné

`passable_lanes` kolekce průjezdných jízdních pruhů

`target_lanes` kolekce cílových jízdních pruhů

`nearest_front` nejbližší vozidlo ve stejném pruhu a před samočinně řízeným vozidlem

`nearest_left` nejbližší vozidlo v levém vedlejším pruhu

`nearest_right` nejbližší vozidlo v pravém vedlejším pruhu

`passing_vehicle` nejbližší vozidlo ve vedlejším pruhu, který je směrem k cílovému pruhu

`vehicle_cluster` kolekce obsahující shluk vozidel před samočinně řízeným vozidlem

`vehicles_in_critical_distance` kolekce obsahující vozidla v kritické vzdálenosti vůči samočinně řízenému vozidlu

`lane_iterator` iterátor skrze kolekci cílových pruhů `target_lanes`

`target_lane` aktuální cílový pruh

`target_lanes_old, vehicle_cluster_old, vehicles_in_critical_distance_old` proměnné uchovávající poslední stav jejich protějšků bez přípony „_old“

#### Hodinové proměnné

`lane_change_timer` časovač, uchovávající čas od poslední změny pruhu

#### Funkce

`nearestSidelaneVehicle(direction)` funkce vracející nejbližší vozidlo, které je ve vedlejším pruhu

`getNearestVehicleFromSpecificLane(ref_vehicle, lane, in_front)` funkce vracející nejbližší vozidlo vůči referenčnímu vozidlu `ref_vehicle` v pruhu daném parametrem `lane` s možností omezení na vozidla před referenčním vozidlem pomocí přepínače `in_front`

`getNearestFrontSideVehicle()` funkce vracející nejbližší vozidlo, které je pozičně před samočinně řízeným vozidlem, nehledě na pruhu

`clearVehicleCluster()` vymazání prvků z kolekce `vehicle_cluster`

`clearPassableLanes()` vymazání prvků z kolekce `passable_lanes`

`getNearestVehicleTowardsTargetLane()` funkce vracející nejbližší vozidlo, které je ve vedlejším pruhu ve směrem k cílovému pruhu

`vehicleClusterChanged()` vyhodnocení, zdali se kolekce `vehicle_cluster` změnila oproti poslednímu záznamu

`targetLanesChanged()` vyhodnocení, zdali se kolekce `target_lanes` změnila oproti poslednímu záznamu

`vehiclesInCriticalDistanceContains(e)` vyhodnocení, zdali kolekce `vehicles_in_critical_distance` obsahuje vozidlo `e`

`nearLaneVehicleLeftCriticalDistance()` vyhodnocení, zdali nějaké vozidlo ve vedlejším pruhu opustilo kolekci `vehicles_in_critical_distance`

`checkVehiclesInCriticalDistance()` kolekce `vehicles_in_critical_distance` je zde naplněna vozidly v kritické vzdálenosti

`analyzeVehicleCluster()` analýza shluku vozidel před samočinně řízeným vozidlem a na základě toho naplnění kolekce `target_lanes`

`initialize()` inicializace hodnot potřebných pro správný průběh procesu

### Šablona VehicleSpeed

#### Proměnné

`passing_sign` značka, která je aktuálně míjena samočinně řízeným vozidlem

`nearest_sign` nejméně vzdálená značka od samočinně řízeného vozidla

`furthest_sign` nejvíce vzdálená značka od samočinně řízeného vozidla

#### Funkce

`initialize()` inicializace proměnných uchovávající značky, se kterými proces pracuje

### Šablona TargetSpeedByVehicleSetter

#### Proměnné

`nearest_front` nejbližší vozidlo ve stejném pruhu a před samočinně řízeným vozidlem

`speed_diff` rozdíl rychlostí samočinně řízeného a nejbližšího vozidla

`time_distance` časová vzdálenost samočinně řízeného a nejbližšího vozidla

**Funkce**

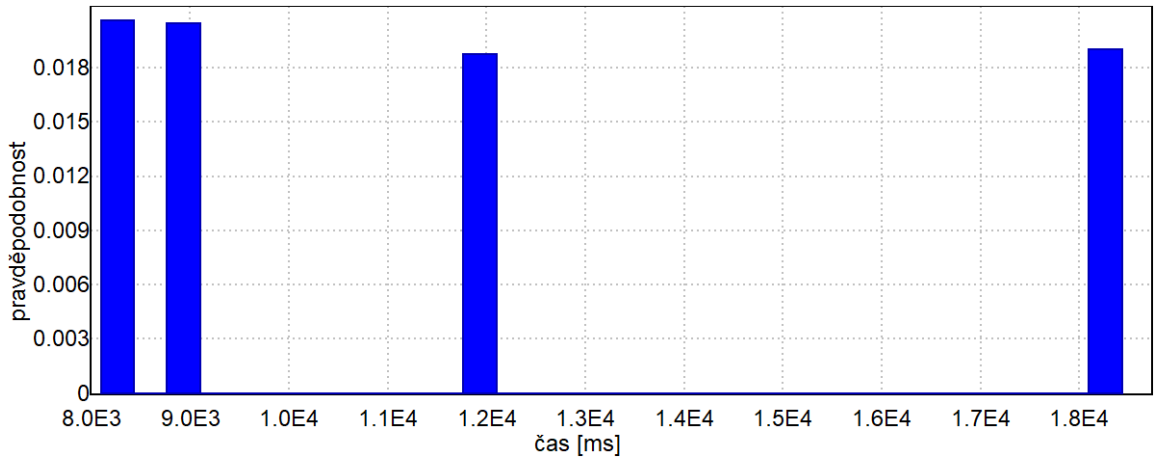
`speedDifference(e1,e2)` výpočet rozdílu rychlostí vozidel `e1` a `e2`

`setTargetSpeedByVehicle(speed)` proměnná `target_speed_by_vehicle` je zde nastavena podle parametru `speed`

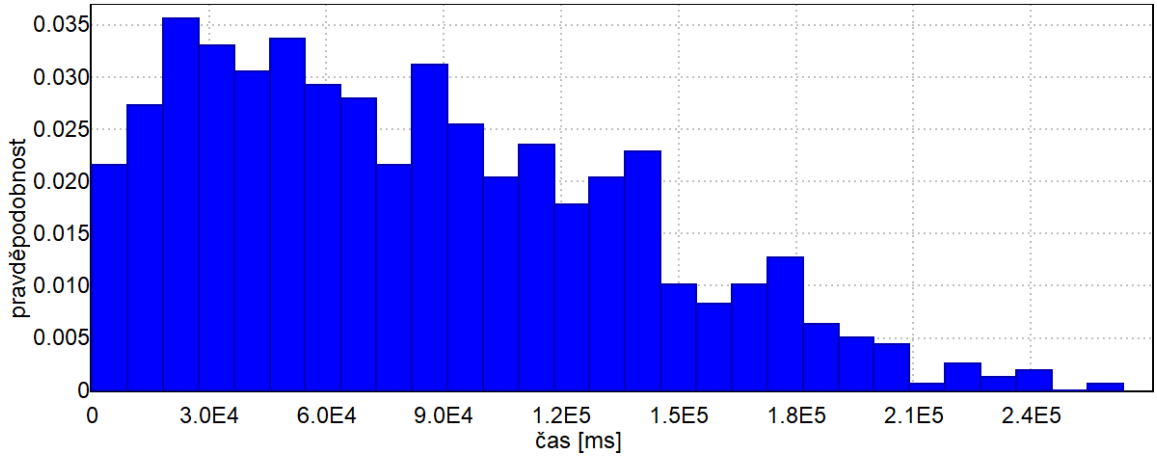
# Příloha D

# Grafy rozdělení pravděpodobnosti

Tato příloha obsahuje grafy rozdělení pravděpodobnosti pravděpodobnostních dotazů z kapitoly 5.




Obrázek D.1: Graf rozdělení pravděpodobnosti pravděpodobnostního dotazu 5.5.




Obrázek D.2: Graf rozdělení pravděpodobnosti pravděpodobnostního dotazu 5.10.